ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ & ΣΥΣΤΗΜΑΤΩΝ
ΠΛΗΡΟΦΟΡΙΚΗΣ

Ανάπτυξη εφαρμογής για την συσχέτηση XML δομών δεδομένων με πηγές δεδομένων και άλλων XML δομών δεδομένων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Σπύρος Μ. Δουλγερίδης

Επιβλέπων : Γεώργιος Ι. Στασινόπουλος
Καθηγητής ΕΜΠ

Αθήνα, Μάρτιος 2005

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ & ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ανάπτυξη εφαρμογής για την συσχέτηση XML δομών δεδομένων με πηγές δεδομένων και άλλων XML δομών δεδομένων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Σπύρος Μ. Δουλγερίδης

Επιβλέπων : Γεώργιος Ι. Στασινόπουλος
Καθηγητής ΕΜΠ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 1η Μαρτίου 2005.

.....	.....	.....
Γεώργιος Στασινόπουλος Καθηγητής ΕΜΠ	Μιχαήλ Θεολόγου Καθηγητής ΕΜΠ	Μιχαήλ Λούμος Αναπληρωτής Καθηγητής ΕΜΠ

Αθήνα, Μάρτιος 2005

.....
Σπύρος Μ. Δουλγερίδης

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

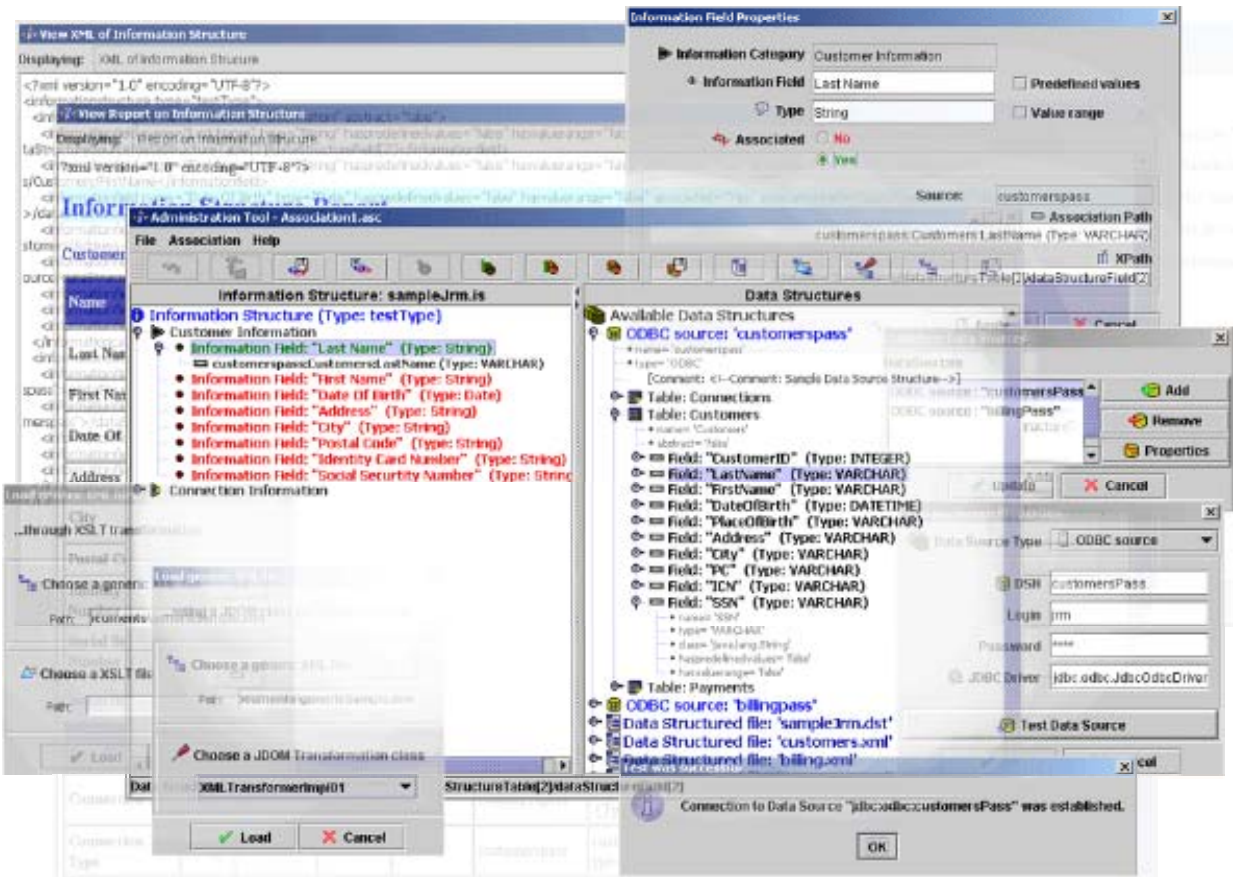
Περίληψη

Το αντικείμενο της παρούσης διπλωματικής εργασίας είναι η δημιουργία μιας εφαρμογής για την συσχέτιση XML δομών δεδομένων με πηγές δεδομένων και άλλες XML δομές δεδομένων.

Το πρόβλημα που έρχεται να δώσει λύση η εφαρμογή που αναπτύχθηκε είναι το εξής. Έστω ότι έχουμε μια βάση δεδομένων, της οποίας την δομή την έχουμε αντιστοιχίσει σε ένα XML αρχείο, και θέλουμε να την γεμίσουμε με δεδομένα απο διαφορετικές πηγές δεδομένων ή απο διαφορετικά αρχεία που αναπαριστούν μια δομή πληροφορίας. Κάθε πεδίο της βάσης μας θέλουμε να το συσχετίσουμε με πεδίο από διαφορετικές πηγές ή αρχεία. Έτσι ώστε η βάση να γεμισει με πληροφορίες που προέρχονται απο πολλές και διαφορές πηγές. Η εφαρμογή αναπτύχθηκε με το σκοπό να κάνει δυνατό το συσχετισμό αυτό.

Η εφαρμογή αναπτύχθηκε με άξονα την λειτουργικότητα και ευκολία χρήσης. Για την επίτευξη του σκοπού αυτού χρειάστηκε να μπορεί να αναπαριστά XML κείμενα σε μορφή δένδρου μέσω γραφικού περιβάλλοντος για την άμεση κατανόηση απο τον χρήστη της δομής τους, και να παρέχει την δυνατότηατα εύκολου συσχετισμού τους μέσω της λειτουργίας Drag and Drop. Επίσης, δίνει στο χρήστη τη δυνατότητα να μπορεί να διαβάσει και να αντιστοιχίσει τη δομή απο πηγές δεδομένων, ώστε να μπορεί στην συνέχεια να την συσχετίσει, παρέχοντας ένα ολοκληρωμένο περιβάλλον για την διαχείριση πηγών δεδομένων. Εκτός απο πηγές δεδομένων ο χρήστης μπορεί να συσχετίσει και αρχεια XML που φέρουν πληροφορία είτε μετασχηματίζοντας τα είτε με την αρχική τους μορφή.

Λέξεις κλειδιά: Java, Swing, Drag n Drop, JTree, XML, XPath, JDOM, JDBC, XSLT



Abstract

The scope of this project is the development of an application for the association XML data structures with Data Sources and other XML data structures.

The problem that this application is willing to solve is the following. Consider that there is a database, that its structure is mapped into an XML document, and we want to fill it with data from different data sources or different files that represent a data structure, Every database field might be associated with a field from different data sources or files, in order for the database to be filled with information from different and various sources. The aim of this application is to make the above association possible.

The application was developed aiming to provide functionality and ease of use. To meet these needs, the application had to be able to display XML documents in GUI tree structure for better user understanding, and to provide the capability of easy association through Drag and Drop. Moreover, it gives the user the capability to read and map the structure of data sources, so as to be able to associate them, by providing an integrated environment for the manipulation of data sources. Besides data sources, user can associate XML files bearing information, either by transforming them or with its initial form.

Keywords: Java, Swing, Drag n Drop, JTree, XML, XPath, JDOM, JDBC, XSLT

Προλογος

Το παρακάτω έγγραφο προσπαθεί να τεκμηριώσει την εφαρμογη Administration Tool που αναπτύχθηκε στα πλαίσια της διπλωματικής. Η τεκμηρίωση προσπαθεί να ακολουθήσει όπου είναι δυνατον και αναγκαίο της αρχές της Αντικειμενοστραφής ανάπτυξης λογισμικών συστημάτων. Όπου είναι δυνατον διοτι ήταν αδύνατον χρονικα να τηρηθούν πλήρως οι αρχές αυτές.

Η δομή της διπλωματικής

Το 1^{ο} κεφάλαιο έχει μια περιγραφή των τεχνολογιών που χρησιμοποιήθηκαν.

Στο 2^{ο} κεφάλαιο ακολουθεί ο προσδιορισμός των απαιτήσεων της εφαρμογής, δηλαδή τι πρέπει να προσφέρει η εφαρμογή στο χρήστη για να θεωρείται ότι είναι επιτυχημένη, η προδιαγραφή των απαιτήσεων, δηλαδή η ανάλυση των απαιτήσεων και τέλος η τεκμηρίωση τους, δηλαδή μια περιγραφή της λειτουργικότητας της εφαρμογής σε λεπτομερές επίπεδο.

Το 3^{ο} κεφάλαιο αφορά το σχεδιασμό της εφαρμογής σε αρχιτεκτονικό επίπεδο. Περιλαμβάνει διαγράμματα κλάσεων για την καλύτερη κατανόηση της δομής της εφαρμογής.

Το 4^{ο} κεφάλαιο είναι το API της εφαρμογής, που ουσιαστικά είναι ο λεπτομερής σχεδιασμός της εφαρμογής.

Το 5^{ο} κεφάλαιο αποτελειται απο ένα σενάριο χρήσης της εφαρμογής για την καλύτερη κατανόηση της λειτουργικότητας της.

Τέλος, ακολουθεί ο κώδικας της εφαρμογής.

Ευχαριστίες

Αρχικά θα ήθελα να ευχαριστήσω την Δρ Γ.Ι. Στασσινόπουλο και τον επιβλέποντα μου Δ. Μ. Κυριαζάνο για το άψογο κλίμα συνεργασίας το οποίο συνέβαλε σημαντικά στην ευκολότερη ανάπτυξη της εφαρμογής.

Θέλω να ευχαριστήσω όλους αυτούς που ήταν κοντά μου στην ανάπτυξη της εφαρμογης και με στήριξαν ο καθένας με το δικό του ξεχωριστό τρόπο.

Τέλος, θα ήθελα να αφιερώσω τη διπλωματική αυτή στον πατέρα μου που πίστεψε σε μένα και με στήριξε για να ολοκληρώσω τις σπουδές μου στο ΕΜΠ. Χωρίς την βοηθεια του θα ήταν πραγματικά αδύνατον να φτάσω στο σημείο να γράφω αυτές τις γραμμές.

Περιεχόμενα

Τεχνολογίες που χρησιμοποιήθηκαν

Java

Η Java είναι μία γλώσσα προγραμματισμού υψηλού επιπέδου. Αναπτύχθηκε από την εταιρεία Sun microsystems και γρήγορα έλαβε μεγάλη δημοσιότητα λόγω της ανεπανάληπτης χρησιμότητάς της στην ανάπτυξη (κυρίως Web) εφαρμογών. Έχει τα εξής χαρακτηριστικά: είναι απλή στην χρήση. Θεωρείται γλώσσα πολύ υψηλού επιπεδου (γλώσσα που παρέχει πολλές κλάσεις έτοιμες και κρύβει απο το χρήστη την περισσότερη απο την πολυπλοκότητα που έχει η αναπτυξή λογισμικού). Η απλότητα της στηρίζεται και στο αμέσως επόμενο χαρακτηριστικό της, ειναι:

- αντικειμενοστραφής (object-oriented). Αυτό τις προσφέρει όλες τα πλεονεκτήματα που έχει μια αντικειμενοστραφής γλώσσα όπως: ευκολη επαναχρησιμοποίηση κώδικα (reuse), πολυμορφισμό, ευκολία στην διαχείριση (κατανόηση - τροποποίηση) μιας μεγάλης εφαρμογής και στην επέκταση της κα
- εύρωστη (robust). Έχει μια πολύ καλά σχεδιασμένη δομή
- μεταφέρσιμη (portable). Ειναι η σημαντικότερη γλώσσα δημιουργίας platform-independent εφαρμογών.
- κατανεμημένη (distributed),
- υψηλής απόδοσης (high performance),
- πολυνηματώδης (multithreaded),
- ασφαλής (secure) κ.ά.

Μια περαιτέρω περιγραφή της Java θεωρείται έξω απο το σκοπό αυτό του κειμένου αφού μπορεί να βρεθεί σε οποιοδήποτε σύγραμμα.. Παρόλα αυτά αξίζει να αναφέρουμε της ιστορία αυτής της τεχνολογίας που άλλα το κοσμο της πληροφορικής.

Η ιστορία της Java
Ίσως η πιο σημαντική συνεισφορά μέχρι τις μέρες μας που είχε η επανάσταση των μικροεπεξεργαστών είναι ότι έκανε δυνατή την δημιουργία των προσωπικών υπολογιστών, οι οποίοι τώρα αριθμούν κάποιες εκατοντάδες εκατομμύρια παγκοσμίως. Οι προσωπικοί υπολογιστές είχαν ένα προφανές αντίκτυπο στην ζωή των ανθρώπων και στους τρόπους που οι οργανισμοί λειτουργούν.
Πολλοί άνθρωποι πιστεύουν ότι η επόμενη τεράστια περιοχή που οι μικροεπεξεργαστές θα έχουν αντίκτυπο είναι οι έξυπνες ηλεκτρονικές συσκευές κατανάλωσης (intelligent consumer-electrinics). Η Sun Microsystems αναγνωρίζοντας αυτή την πιθανή προοπτική, χρηματοδότησε το 1991, ένα εσωτερικό ερευνητικό πρόγραμμα με κωδικό όνομα Green Το πρόγραμμα οδήγησε στην δημιουργία μιας γλώσσας προγραμματισμού που βασιζόταν στην C++, όπου ο δημιουργός της James Gosling, ονόμασε Oak, λόγω μιας βαλανιδιάς (oak) που υπήρχε έξω απο το παράθυρο του στην Sun.Αργότερα ανακαλύφτηκε ότι υπήρχε ήδη μια άλλη γλώσσα προγραμματισμού με το όνομα Oak. Όταν μια ομαδα απο ανθρώπους της Sun επισκέφτηκαν το τοπική καφετέρια, το όνομα Java προτάθηκε και τελικά παρέμεινε.
Το πρόγραμμα Green είχε κάποιες δυσκολίες. Η αγορά έξυπνων ηλεκτρονικων συσκευών κατανάλωσης δεν αναπτυσσόταν όσο γρήγορα όσο είχε υπολογίσει η Sun. Ακόμα χειρότερα, ένα μεγάλο συμβόλαιο το οποίο είχε διεκδικήσει η Sun, ανατέθηκε σε άλλη εταιρία. Υπήρχε κίνδυνος το πρόγραμμα να τερματιστεί. Για καλή του τύχη όμως, το 1993 έγινε η έκρηξη του WWW, και οι άνθρωποι της Sun είδαν την άμεση δυνατότητα της χρησιμοποίησης της Java για να προσθέσουν δυναμικό περιεχόμενο και κίνηση στις σελίδες του διαδικτύου.

Η Sun ανακοίνωσε επισημα την Java σε ένα συνέδριο το Μάιος του 1995. Συνήθως, ένα γεγονός σαν και αυτό δεν θα είχε δημιουργήσει τόση προσοχή. Παρόλα αυτά, η Java δημιούργησε άμεσο ενδιαφέρον στο κύκλο των εταιριών λόγω του φαινομενικο ενδιαφέρον που υπήρχε για το WWW.

XML - eXtensible Markup Language.

Η XML (Extended Mark-up Language) είναι μια δηλωτική γλώσσα πού ήρθε μετά την ASN1. Κύρια διαφορά είναι ότι στην XML χρησιμοποιούνται αποκλειστικά χαρακτήρες (character based, έναντι της ASN1 πού είναι bit coded). Επιπλέον, η δομή της είναι πιο απλή και πιο εύκαμπτη. Η XML χρησιμοποιείται για την αποθήκευση δομημένης πληροφορίας. Η δομή της πληροφορίας προσδιορίζεται με τη βοήθεια των tags (ετικετών) μέσα στα οποία περικλείονται συναφή τμήματα πληροφορίας.

Στην πραγματικότητα, η XML είναι markup γλώσσα για έγγραφα που περιέχουν δομημένες πληροφορίες. Markup γλώσσα είναι ένας μηχανισμός που καθορίζει δομές σε ένα έγγραφο. Οι δομημένες πληροφορίες περιλαμβάνουν περιεχόμενο και κάποιες διευκρινίσεις για το ρόλο που παίζει το περιεχόμενο. Τι είναι η XML:

- Η XML είναι markup γλώσσα, η οποία μοιάζει πολύ με την HTML
- Η XML σχεδιάστηκε για να περιγράφει δεδομένα
- Τα XML tags δεν είναι προκαθορισμένα. Πρέπει να ορίσουμε τα δικά μας tags. Η XML σχεδιάστηκε για τη δόμηση, την αποθήκευση και την αποστολή δεδομένων, συνεπώς η ένα XML κείμενο απο μόνο δεν περιγράφει τίποτα
- Η XML χρησιμοποιεί το Document Type Definition (DTD) ή το XML Schema για να περιγράψει τα δεδομένα, δηλαδη να δόση νόημα στα δεδομένα. Για παράδειγμα, ότι το συγκεκριμένο tag, έχει κάποιο δοσμένο νόημα ().
- Ένα XML κείμενο με ένα DTD ή ένα XML Schema σχεδιάστηκε για να περιγράφει επαρκώς τον εαυτό του.

XML and Java

Η XML είναι μια γλώσσα για την δημιουργία κειμένων που είναι vendor-independed, δηλαδή τα συστηματα απο όλους τους κατασκευαστές μπορούν να διαχειριστούν ένα XML κείμενο. Αυτό οφείλεται στα πλεονεκτήματα αναφέρθηκαν στην σχετική παράγραφο, αλλα κυρίως και στο γεγονός ότι είναι ένα W3C standard που οριοθετεί μια δομή που όλοι οι κατασκευαστες συστημάτων σέβονται και ακολουθούν. Το αποτέλεσμα είναι ότι η XML είναι προσπελάσιμοι απο κάθε συστημα.
Η Java, όπως αναφέρθηκε είναι και αυτή platform-independent. Η Java είναι και αυτή ένα de facto standard που ελέγχεται απο μία και μόνο εταιρία την Sun Microsystems. Συνεπώς:

- η XML παρέχει τα platform-independent δεδομένα, ενώ
- η Java παρέχει τον platform-independent τρόπο επεξεργασίας τους

Συνεπώς, μια Java εφαρμογή που διαχειρίζεται XML δεδομένα, είναι μια εφαρμογή που τρέχει σε οποιαδήποτε σχεδον πλατφορμα (αρκεί να αυτή να έχει εγκατεστημένο ένα JVM) και διαβάζει, διαχειρίζεται και αποθηκεύει XML δεδομένα που παλι μπορούν να γίνουν κατανοητά απο οποιαδήποτε άλλη εφαρμογή.

Οι τρείς τρόποι χρησιμοποίησης της XML σε μια Java εφαρμογη.

Με όλες τις XML τεχνολογίες που υπάρχουν σήμερα (XSL, RDF, Namespaces, RSS, XML chema, XSLT...) μπορεί να φαίνεται οτι υπάρχουν πολλοί τρόποι με τους οποίους μια Java εφαρμογη μπορεί να έχει πρόσβαση σε XML δεδομένα. Δυστυχώς, ουσιαστικά υπάρχουν μόνο τρείς τρόποι για πρόσβαση σε XML δεδομένα.

Callbacks

Το Callback ειναι ο τροπος που βασιζεται στα events που προκύπτουν κατα το parsing των xml δεδομένων. Όπως αναλύεται γραματικά (parsing) το xml κείμενο, δημιουργούνται απο τον επεξεργαστη του κειμένου (parser) κατάλληλα events (γεγονότα που εξυπηρετούνται απο συναρτήσεις). Τα events μπορεί να είναι η αρχή του xml εγγράφου ή το text περιεχόμενο σε ένα xml element. Τα events αυτά πυροδοτούν κατάλληλες μεθόδους. Με την κατάλληλη υλοποίηση αυτών των μεθόδων μπορεί να υλοποιηθεί η επιθυμητη λογική απο την εφαρμογή. Ο τρόπος αυτός δεν είναι και ο ποιό εύκολος και συνήθως απαιτείται αρκετός χρόνο εξοικίωσης απο έναν προγραμματιστή. Το βασικό πλεονέκτημα του είναι ότι μπορει να διαχειρίζεται μεγαλα xml κείμενα με γρήγορο τρόπο και κυρίως χωρίς να απαιτεί σημαντικό μερος απο τους πόρους του συστήματος, και αυτό για το λόγο οτι δεν χρειάζεται να έχει όλο το xml κείμενο «φορτωμένο» στην μνήμη. Απο την αλλή το τελευταίο χαρακτηριστικό που του προσδίδει και το προαναφερόμενο πλαεονέκτημα, του προσδίδει και το βασικότερο μειονεκτημα. Αν καθώς επεξεργαζομαστε καποιο σημειο του xml κειμένου θελήσουμε να επεξεργαστούμε κάποιο προηγούμενο σημείο, τότε πρέπει να αποθηκευσουμε το σημειο που βρισκόμαστε και να αρχίσουμε απο την αρχή του κειμένου μεχρι να βρούμε το επιθυμητο σημειο. Το SAX (Simple API for XML) ειναι το de facto standard για αυτό το τρόπο πρόσβασης σε XML δεδομενα.

Δέντρα

Περισσότερο συνηθισμένα είναι τα API που παίρνουν ένα XML κείμενο και δημιουργούν μια δεντροειδή δομή των δεδομένων του. Το XML κείμενο (XML Document) γίνεται το κεφάλι του δέντρου και λειτουργεί σαν ένα κιβώτιο. Έχει παιδιά, όπως την ρίζα του κειμένου, η οποία έχει και αυτή παιδιά κοκ., ώσπου δημιουργείται ένας γράφος απο XML δεδομένα. Επειδή οι περισσότεροι απο εμας έχουνε έρθει σε επαφη με δομές δέντρων, ο τρόπος αυτος αποτελεί ένα πολύ εύκολο τρόπο αναπαράστασης και επεξεργασίας XML δεδομένων.
Το πιο δημοφιλές API για δενδρική αναπαράσταση ενός XML κειμένου έιναι η W3C εισήγηση, το DOM (Document Object Model). Ένα άλλο πιο νεο API, είναι το JDOM, το οποίο είναι μια προσέγγιση βασισμένη στις δομές δεδομένων που παρέχει η γλώσσα Java. Δηλαδη τα Attribute σε ένα element αποτελούν μια java.util.List, γεγονός το οποίο προσδίδει πολύ μεγαλύτερη απλότητα και ευκολία στον προγραμματισμό εφαρμογών Java για XML δεδομένα. Δυστυχώς, το JDOM όπως είναι προφανές μπορεί να χρησιμοποιθει μονο σε Java εφαρμογές.
Αν και τα API που βασιζονται στη δενδρικη αναπαρασταση του XML ειναι πιο εύκολα στη χρήση απο το SAX που βασίζεται στην επεξεργασία event (event-driven API), δεν είναι πάντα κατάλληλα. Όπως αναφερθηκε, πολύ μεγάλα κείμενα μπορεί να χρειαστουν πολυ μνήμη (κυρίως όταν χρησιμοποιείται το DOM). Όταν γίνονται μετασχηματισμοί σε δενδρικές δομές (XSLT), το σύσημα μπορεί να παγώσει ή να καταρεύσει τελείως. Ακόμα και αν πιο νεα API όπως το JDOM προσπαθούν να λύσουν αυτά τα προβλήματα, όταν λειτοπυργούν πανω σε τεράστια XML κείμενα. Επίσης, αντι να δουλευουν σε δενδρικές δομες, οι προγραμματιστες μερικές φορες θέλουν τα δεδομένα στο XML κείμενο να είναι μοντελοποιημένα σαν ένα ένα Αντικείμενο της γλώσσας Java, με mutators (μέθοδοι που τροποποιούν τα δεδομένα ενός αντικειμένου) και accessors (μέθοδοι που επιστρέφουν την τιμή των δεδομένων ενος αντικειμένου χωρίς να την αλλάζουν) για τα δεδομένα αυτα. Για παράδειγμα, αντι να

πηγαίνουμε σε ένα κόμβο παιδι με όνομα Α και να αλλάζουμε το text περιεχόμενο του κόμβου, ο προγραμματιστής μπορεί απλά να θέλει να καλέσει την μέθοδο setA("valueforA") και να συνεχίσει..

Data binding

Ειναι σχεδιασμένο στο να διευκολύνει το «δεσιμο» ενος Java αντικειμένου με ένα XML κείμενο, το οποίο κάνει εύκολη την μετατροπή του ενος format στο άλλο και αντιστρόφως. Το Binding αναφέρεται σε ενα Java αντικείμενο με accessor και mutator μεθόδους που επιδρούν στο XML κείμενο που αντιστοιχεί στο αντικείμενο και αντιστοιχεί ακριβώς στα ονοματα των elements και attributes του XML κειμένου. Αυτη η πρόσβαση είναι ιδιαίτερα χρήσιμη όταν αποθηκεύεται πληροφορία διαμόρφωσης (configuration information) σε XML κείμενα. Ειναι πολύ πιο εύκολο να έχει κανείς πρόσβαση αμεσα σε παραμέτρους απο το να χρειαζεται να χρησιμοποιησει πολύπλοκες δενδρικες δομές. Αν και αυτου του είδους η πρόσβαση δεν είναι ιδιαίτερα χρήσιμη για μετασχηματισμούς XML κειμένων ή XML μηνυματων που ανταλλάσονται μεταξυ εφαρμογών, είναι ιδιαίτερα πρόσφορος για απλή διαχείριση δεδομένων και κυρίως για serialization Java αντικειμένων.

JDOM

Εισαγωγή

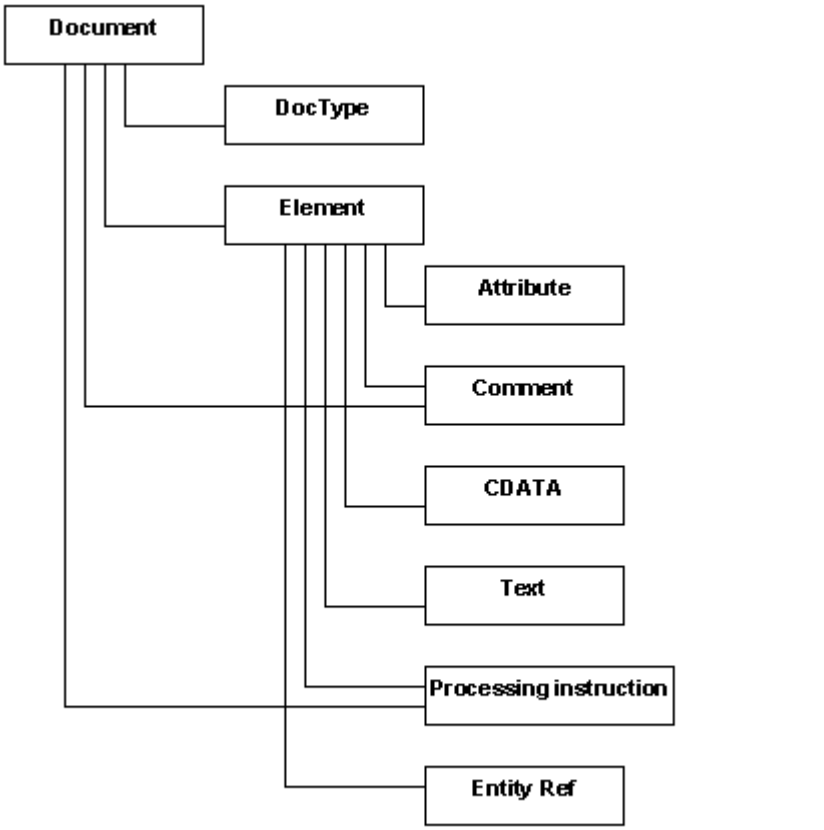
Ο σκοπός του JDOM είναι να παρέχει μια λύση για την χρήση της XML μέσω της Java. Η λύση να είναι τόσο εύκολη στην χρήση όσο η ίδια η Java. Δεν πρέπει αν είναι απαραίτητο ένα Java API για το χειρισμός της XML να είναι πολύπλοκο. Η λύση που προσφέρει το JDOM είναι και υλοποιημένος με βάση την Java και βελτιστοποιημένος για αυτή. Συμπεριφέρεται σαν την Java, χρησιμοποιεί της συλλογές (collections) της Java, είναι ένα «φυσικό» API για τους Java προγραμματιστές, και παρέχει μια εύκολη στην υιοθέτηση της λύση για την χρήση της XML.

Αν και το JDOM συνεργάζεται άψογα με ήδη υπάρχοντα standards όπως το Simple API for XML (SAX) και το Document Object Model (DOM), δεν είναι ένα αφαιρετικό API (δηλαδή δεν «καθεται» πάνω απο αυτά, και χρησιμοποιώντας τα σαν βαση, να κάνει την χρήση τους πιο εύκολη) ή ένα API που επεκτείνει τα δυο αυτά APIs. Αντι για αυτό, ο σκοπός του είναι να παρέχει ένα δομημένο και εύκολο στην κατανόηση και στην γραφή XML δεδομένων τρόπο, χωρίς το πολύπλοκοτητα και την υψηλή απαίτηση σε μνήμη που έχουν τα δυο παραπάνω APIs.

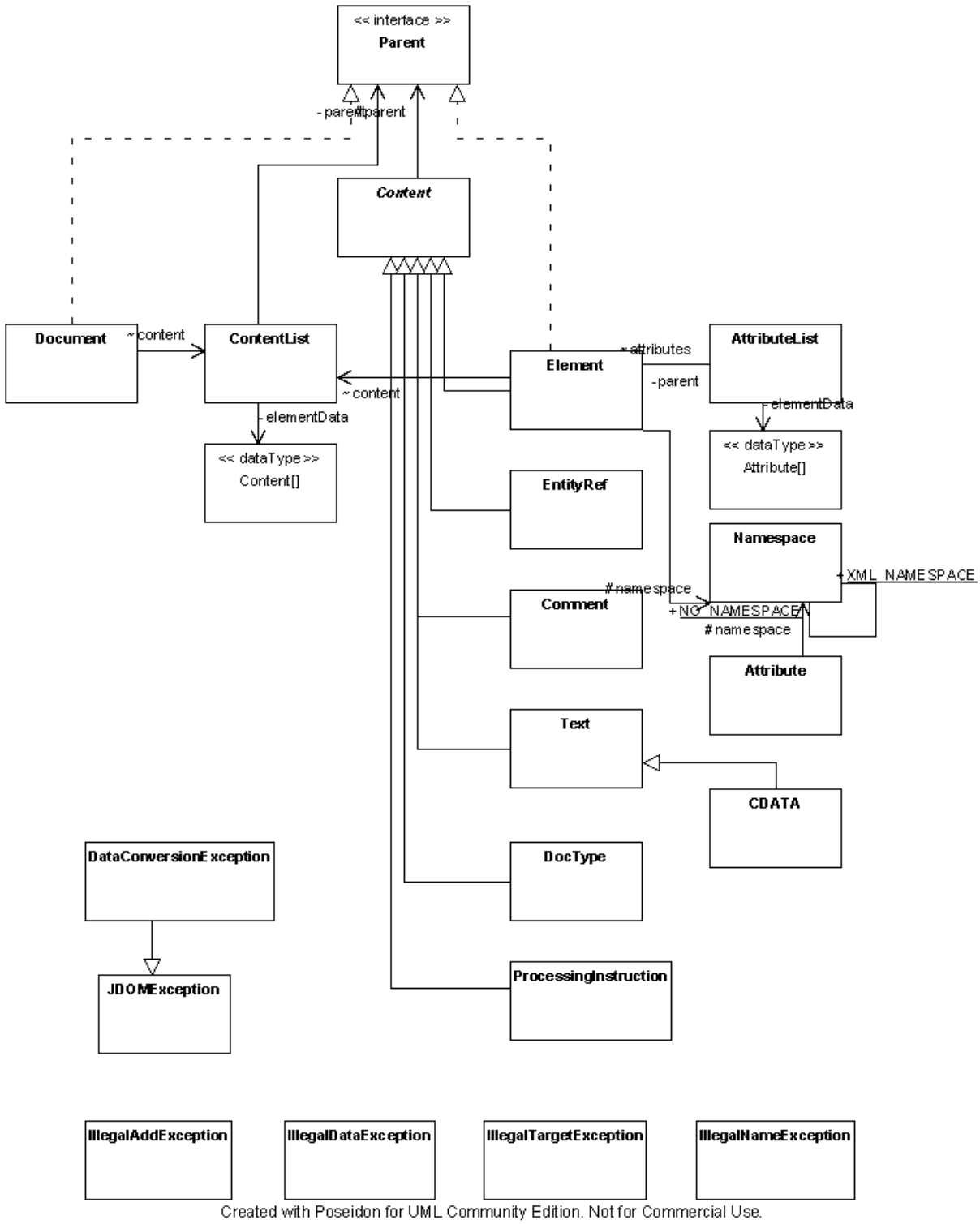
Τι είναι το JDOM.

Το JDOM είναι, πολυ απλά, μια Java αναπαράσταση ενίς XML κειμένου. Παρέχει ενα τρόπο αναπαράστασης αυτό του κειμένου ώστε να είναι εύκολη η ανάγνωση, η διαχείριση και η γραφή του. Έχει ένα ευκολο και άμεσο στην κατανόηση API, έχει χαμηλές απαιτήσεις απο το σύστημα που τρέχει και είναι γρήγορο, και είναι βελτιστοποιημένο για το Java προγραμματιστή. Είναι η εναλλακτική λύση για το DOM και το SAX, αν και συνεργάζεται αψογα και με τα δύο. Τέλος, παρέχει υποστήριξη για όλα τα χαρακτηριστικά της XML.

Ακολουθεί ένα διαγραμμα δομής του JDOM.



Ακολουθεί το διάγραμμα κλάσεων του JDOM.



Περισσότερες λεπτομέρειες για το διάγραμμα κλάσεων στο API του JDOM (http://www.jdom.org/docs/apidocs/index.html).

Τι δεν είναι το JDOM.

Το JDOM δεν είναι ένα wrapper API για το W3C DOM, ή κάποια άλλη έκδοση του DOM. Το JDOM είναι document object μοντέλο για XML που είναι βασισμένο στην Java. Δηλαδή, το JDOM εξυπηρετεί το ίδιο σκοπο με το DOM, αλλά είναι πολύ πιο εύκολο στη χρήση.

Wrapper API είναι ένα API το οποίο παίρνει την λειτουργικότητα ενος άλλου API και την παρέχει στον χρήστη με διοφορετικό τρόπο. Για παράδειγμα το DOM είναι factory based, δηλαδή αν ο προγραμματιστής θέλει να φτιάξει ένα sax builder πρέπει πρώτα να δημιουργήσει ένα factory object και με αυτό να δημιουργήσει το sax builder, αντι να χρησιμοποιήσει constructor (SaxBuilder builder = new SaxBuilder();). Ένα wrapper API θα μπορούσε να "κατσει" πάνω στο DOM και απο factory based να το κανει constructor based.

Το JDOM δεν είναι ένας XML parser, όπως ο Xerces. Είναι ένα document object μοντέλο που χρησιμοποιεί XML parsers για να δημιουργήσει κείμενα. Για παράδειγμα, η κλάση SAXBuilder του JDOM χρησιμοποιεί SAX events που παράγονται απο ένα XML parser για να δημιουργήσει ένα JDOM δέντρο. Το JDOM μπορεί να χρησιμοποιήσει σχεδόν οποιοδήποτε parser.

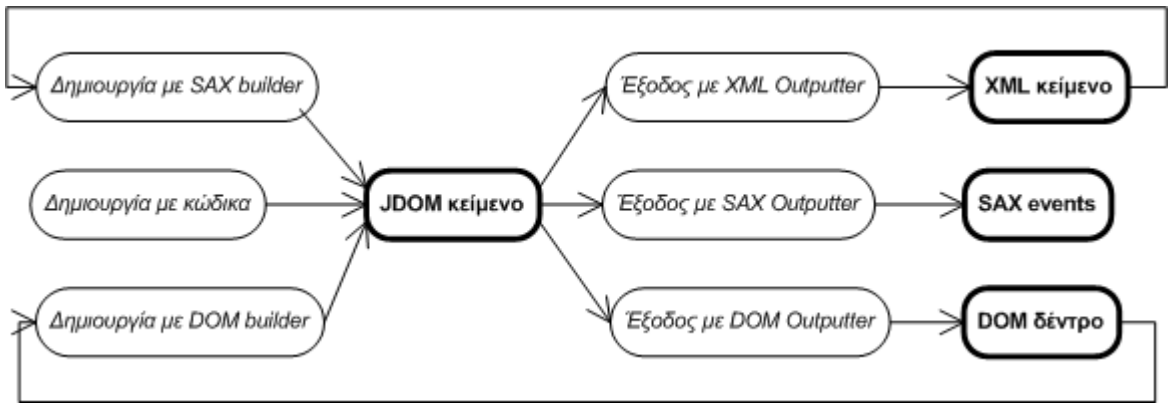
Η φιλοσοφία του JDOM

Το JDOM έχει δημιουργηθεί πανω στην εξής φιλοσοφία:

- πρέπει να είναι εύκολο στην κατανόηση στους Java προγραμματιστές
- πρεπει να παρέχει υποστήριξη για εύκολη και αποτελεσματική τροποποίηση XML κειμένων
- πρέπει να «κρύβει» τις πολυπλοκότητες της XML όποτε είναι δυνατόν, και να παραμένει πιστό στην προδιαγραφή της XML.
- πρέπει να μπορεί να συνεργάζεται άψογα με το SAX και το DOM
- πρέπει να έχει λίγες απαιτήσεις απο το σύστημα που τρέχει και να είναι γρήγορο
- πρέπει να λύνει 80% των προβλημάτων της Java με την XML με 20% λιγότερη προσπάθεια

Συνεργασία του JDOM με το SAX και DOM.

Τα JDOM κείμενα μπορούν να δημιουργηθούν απο XML αρχεία, DOM δέντρα, SAX events, ή οποιαδήποτε άλλη πηγή. Αντιστρόφως, τα JDOM κείμενα μπορούν να μετατραπούν σε XML αρχεία, DOM δέντρα, SAX events, ή οποιοδήποτε άλλο προορισμό. Αυτη η δυνατότητα αποδεικνύεται χρήσιμη όταν, για παράδειγμα ένα πρόγραμμα ειναι σχεδιασμένο να δέχεται SAX events. Ο προγραμματιστής μπορεί να φορτώσει το XML αρχείο με το JDOM, να τροποποιήσει εύκολα και αποτελεσματικά το κείμενο, και μετά να πυροδοτήσει SAX events στο πρόγραμμα που τα περιμένει απευθείας, χωρίς καποια άλλη μετατροπή.



Άλλα χαρακτηριστικά του JDOM

Το JDOM παρέχει υποστήριξη για **XSLT**. Υπάρχουν πολύ τρόποι να μετασχηματίσει με XSLT κανείς όταν χρησιμοποιεί το JDOM. Ο ποιος προφανής τρόπος είναι μεσω της Transformer interface και τις κλάσεις JDOMSource και JDOMResult που υπάρχουν στο πακέτο org.jdom.transform.

Επίσης το **XPath** υποστηρίζεται απο το πακέτο org.jdom.xpath. Η υποστήριξη βασίστηκε στο Jaxen API (jaxen.org).

Τέλος, το JDOM φιλτραρίσματος των περιεχομένου του κειμένου του. Για παράδειγμα παρέχει την δυνατότητα στο προγραμματιστή να φορτώσει ένα XML κείμενο παραλείποντας του Text κόμβους που έχουν ignorable whitespace χωρίς να χρειάζεται να υπάρχει κάποιο αρχείο επικύρωσης (DTD ή XML Schema). Η υποστήριξη βρίσκεται στο πακέτο org.jdom.filter

Απαιτήσεις της εφαρμογής

Προσδιορισμός Απαιτήσεων

Σκοπός
Σκοπός της εφαρμογης ειναι η διαχείρηση XML δομών δεδομένων και η συσχέτιση τους με πηγες δεδομένων που βρίσκονται σε βάσεις δεδομένων καθώς και με άλλες XML δομές δεδομένων.

Δεδομένα
Με βάση τις απαιτήσεις των λειτουργιών που πρέπει να εκτελεί το λογισμοκο έχει τα παρακάτω δεδομενα εισόδου και εξόδου.
Δεδομένα Εισόδου
Τα δεδομένα εισόδου που θα πρέπει να έχει στη διάθεσή του προέρχονται από: τα αρχεια που θα φορτώνει σε αυτο ο χρήστης καθώς και τα δεδομένα που θα διαβάζει το λογισμικο απο πηγές δεδομένων (ODBC). Επισης και οι επιλογες του χρήστη (εντολες που μπορεί να δώσει ο χρήστης) αποτελούν δεδομένα για το λογισμικο.
Δεδομένα Εξόδου
Τα δεδομένα που θα παράγει το λογισμικο είναι αρχικα η απεκόνιση των XML δομών σε δενδρικη δομή για πιο ευκολη διαχειριση τους. Τα κυρίως δεδομένα εξόδου ειναι αρχεία στα οποία θα αποθηκεύονται οι συσχετίσεις των δομών.

Εργασιες
Η εφαρμογη πρεπει να επιτελει τις παρακάτω εργασίες
3.1.Εργασία 1
Να παρέχει δυνατότητα στο χρήση να μπορεί να συσχετίσει τις δομές δεδομένων του με πηγες δεδομένων που παρέχονται απο βάσεις δεδομένων.

Να παρέχει δυνατότητα στο χρήση να μπορεί να συσχετίσει τις δομές δεδομένων του με αρχεια δεδομένων XML μορφής.

Να παρουσιαζει τις δομές δεδομένων των XML κειμέμων και των πηγών δεδομένων σε δενδική μόρφή για ευκολη κατανόηση και καλύτερη αντίληψη απο το χρήστη.

Να παρέχει ενα εύκολο για το χρήστη τρόπο συσχετισμού των δομών δεδομένων που φορτώνονται.

Να μπορεί ο χρήστης να δεί σε μια φόρμα όλες τις ιδιότητες του συσχετισμού που δημιουργησε..

Να μπορεί να αποθηκεύει τους συσχετισμούς που δημιούργησε ο χρήστης σε μορφή αρχείου. Τα αρχεια αυτα να μπορούν να ανοιχτουν μετέπειτα απο την εφαρμογη για προσθεση περισσότερων συσχετίσεων.

Να μπορεί ο χρήστης να δημιουργήσει νέες διαμορφώσεις με βάσεις δεδομένες, να φορτώσει υπάρχουσες, να διαχειριστεί τις υπάρχουσες διαμορφώσεις και τέλος να μπορεί να αποθηκεύσει τις διαμορφώσεις αυτές σε αρχεια ώστε να μην χρειαζεται καθε φορα να δημιουργεί τις ίδιες. Ο

ορος «διαμόρφωση» σημαίνει όλες οι παραμέτρους που απαιτούνται για να μπορεσει η εφαρμογη να συνδεθει με την πηγη δεδομένων.

Να μπορεί ο χρήστης διαχειρίζεται και XML αρχεία με οποιοδήποτε μορφή. Δηλαδή να μην προαπαιτείται απο το σύστημα μια συγκεκριμένη μορφή.

Αν η εφαρμογη έχει μια προκαθορισμένη μορφη XML αρχείων, να μπορει ο χρήστης να μετασχηματίζει με ενα XML αρχειο οποιασδήποτε μορφής στη προκαθορισμένη μορφη.

Να παρέχει την δυνατότητα δημιουργίας Αναφορας των συσχετίσεων που εχουν γίνει αλλα και δυνατότητα να μπορεί άμεσα ο χρήστης να βλέπει καθεαυτό XML αρχειο που δουλευει χωρις να χρειάζεται πρώτα να το αποθηκεύσει.

Η εφαρμογη πρεπει να έχει ενα φιλικο προς στο χρήστη περιβάλλον, πραγμα που συνεπαγεται οτι πρέπει να τον προειδοποιεί οταν υπαρχει περιπτωση απώλειας δεδομένων, αν δηλαδη ο χρήστης δώσει εντολή να κλείσει η εφαρμογή χωρίς να έχει αποθηκεύσει τα δεδομένα του.

Η εφαρμογη να αναπτυχθει με τις αρχές του αντικειμονοστρεφους προγραμματισμού ώστε να εξασφαλιστει η ευκολη επεκτασιμοτητα της με προσθετες λειτουργίες.

Προδιαγραφή Απαιτήσεων

Προσδιορισμός Actors

Οι actor του συστήματος είναι:

- Ο χρήστης της εφαρμογή (User)
- Οι πηγές δεδομένων απο τις οποίες τραβάει η εφαρμογή δεδομένα (Data Sources)

Ενας actor είναι όποιος ή ότι αλληλεπιδρά με μια περίπτωση χρήσης (use case).

Προσδιορισμος Περιπτώσεις χρήσεις (Use cases)

Περίπτωση χρήσης: Create Association file
Απαίτηση: Ο χρήστης δημιουργεί ένα νέο αρχείο συσχετίσεων.
Actor: Χρήστης
Περιγραφή περιπτώσεις χρήσης: Δημιουργία νεου αρχειου συσχέτισης

Περίπτωση χρήσης: Load Association file
Απαίτηση: Ο χρήστης ανοίγει ενα αρχείο συσχέτισεων βλέπει τις συσχετίσεις που αυτό έχει και μπορεί να συνεχίσει να εργαζεται σε αυτο τροποποιώντας το
Actor: Χρήστης
Περιγραφή περιπτώσεις χρήσης: Ανοιγμα προυπαρχονταος αρχειου συσχέτισης

Περίπτωση χρήσης: Save Association file
Απαίτηση: Ο χρήστης μπορει να αποθηκευσει στο αρχειο συσχετίσεων που εργαστηκε τις συσχετίσεις που δημιούργησε
Actor: Χρήστης
Περιγραφή περιπτώσεις χρήσης: Αποθήκευσης Αρχείου συσχετίσεων

Περίπτωση χρήσης: Close Association file
Απαίτηση: Ο χρήστης μπορεί να κλείσει το αρχειο συσχετίσεων που εργαζεται και εφαρμογη να παραμένει ανοιχτή, χωρις απώλεια των συσχετίσεων που δημιουργήθηκαν. Δηλαδή η εφαρμογή να ενημερώνει το χρήστη όταν το αρχειο συσχετίσεων έχει τροποποιηθεί και δεν έχει σωθεί όταν επιχειρείται το κλείσιμο του απο τον χρήστη.
Actor: Χρήστης
Περιγραφή περιπτώσεις χρήσης: Κλείσιμο αρχείο συσχετίσεων

Περίπτωση χρήσης: View Report
Απαίτηση: Ο χρήστης μπορεί να δημιουργήσει αναφορας των συσχετίσεων που εχουν γίνει
Actor: Χρήστης
Περιγραφή περιπτώσεις χρήσης: Εμφάνιση Αναφοράς Συσχετίσεων

Περίπτωση χρήσης: View XML of Information file
Απαίτηση: Ο χρήστης μπορεί να δει το XML αρχείο που αντιστοιχεί στο αρχείο συσχετίσεων που εργάζεται.
Actor: Χρήστης
Περιγραφή περιπτώσεις χρήσης: Εμφανιση XML του Αρχείο Συσχετίσεων

Περίπτωση χρήσης: View XML of selection in Data Structure
Απαίτηση: Ο χρήστης μπορεί να δει το XML αρχείο που αντιστοιχεί στις δομες δεδομένων που έχει φορτώσει.
Actor: Χρήστης
Περιγραφή περιπτώσεις χρήσης: Εμφανιση XML των δομων δεδομένων

Περίπτωση χρήσης: Exit Application
Απαίτηση: Κλείσιμο εφαρμογής, χωρις απώλεια των συσχετίσεων που δημιουργήθηκαν αν εκεινη την στιγμή είναι ανοιχτό ένα αρχείο συσχετίσεων. Δηλαδή η εφαρμογή να ενημερώνει το χρήστη όταν το αρχειο συσχετίσεων έχει τροποποιηθεί και δεν έχει σωθεί όταν επιχειρείται το κλείσιμο του απο τον χρήστη.
Actor: Χρήσης
Περιγραφή περιπτώσεις χρήσης: Κλείσιμο εφαρμογής

Περίπτωση χρήσης: Load Information Structure
Απαίτηση: Ο χρήστης μπορει να φορτώσει στο νεο αρχειο συσχετίσεων μια Δομή πληροφοριών και να δημιουργήσει συσχετίσεις.
Actor: Χρήστης

Περιγραφή περιπτώσεις χρήσης: Ανοιγμα Δομής πληροφοριών

Περίπτωση χρήσης: Load New Data Source Configuration
Απαίτηση: Ο χρήστης μπορεί δημιουργήσει μια Διαμόρφωση Πηγών Δεδομένων και να τις φορτώσεις στην εφαρμογή
Actor: Χρήστης, Πηγές δεδομένων
Περιγραφή περιπτώσεις χρήσης: Δημιουργία Διαμόρφωσης και Φόρτωση Διαμόρφωσης Πηγών δεδομένων

Περίπτωση χρήσης: Save Data Source Configuration
Απαίτηση: Ο χρήστης μπορεί να αποθηκευσει την διαμόρφωση των πηγών δεδομένων που δημιουργησε σε ένα αεχειο, ώστε να μπορεί να την φορτώσει πάλι στο μέλλον.
Actor: Χρήστης
Περιγραφή περιπτώσεις χρήσης: Αποθήκευση Διαμόρφωσης Πηγών δεδομένων

Περίπτωση χρήσης: Load Existing Data Source Configuration
Απαίτηση: Ο χρήστης μπορεί να ανοίξει ένα αρχειο διαμόρφωσης πηγών δεδομένων και να φορτώσει την διαμόρφωση.
Actor: Χρήστης, Πηγές δεδομένων
Περιγραφή περιπτώσεις χρήσης: Ανοιγμα αρχείο διαμόρφωσης πηγών δεδομένων και Φόρτωση Διαμόρφωσης Πηγών δεδομένων

Περίπτωση χρήσης: Configure Data Sources
Απαίτηση: Ο χρήστης μπορεί να διαχειριστεί τις διαμορφώσεις των πηγων δεδομένων. Η δοαχείριση συμπεριλαμβάνει την προσθήκη νέας πηγης δεδομένων στην διαμόρφωση, την κατάργηση πηγης που έχει φορτωθεί και την αλλαγή παραμέτρων σε πηγή που είναι φορτωμένη. Οι αλλαγες στην διαμόρφωση επιδρούν αμεσα στις πηγές δεδομένων που έχει η εφαρμογή.
Actor: Χρήστης, Πηγές δεδομένων
Περιγραφή περιπτώσεις χρήσης: Διαχειριση πηγών δεδομένων.

Περίπτωση χρήσης: Load Data Structure file
Απαίτηση: Ο χρήστης μπορει να φορτώσει ένα αρχείο δομής δεδομένων (Data Structure file), το οποίο περιγράφει μια δομή δεδομένων. Το αρχείο είναι ένα XML αρχείο με συγκεκριμένη προκαθορισμένη μορφή που αναγνωρίζει η εφαρμογή.
Actor: Χρήστης
Περιγραφή περιπτώσεις χρήσης: Ανοιγμα αρχειου δομής δεδομένων.

Περίπτωση χρήσης: Load generic XML through XSLT
Απαίτηση: Ο χρήστης μπορεί να ανοιξει ένα XML αρχείο οποιασδήποτε μορφής, να το μετασχηματίσει σε ενα αρχειο δομής δεδομένων μέσω ενος XSL αρχειου που θα επιλέξει και να το φορτώσει στην εφαρμογή.
Actor: Χρήστης

Περιγραφή περιπτώσεις χρήσης: Ανοιγμα XML αρχειου και μετασχηματισμού του μεσω XSLT.

Περίπτωση χρήσης: Load generic XML through JDOM Transformation
Απαίτηση: Ο χρήστης μπορεί να ανοιξει ένα XML αρχείο οποιασδήποτε μορφής, να το μετασχηματίσει σε ενα αρχειο δομής δεδομένων χρησιμοποιώντας μια κλαση μετασχηματισμου που θα επιλέξει και να το φορτώσει στην εφαρμογή. Η κλαση μετασχηματισμου θα περιέχει το κατάλληλο κώδικα που θα μετασχηματίζει το XML αρχειο στην επιθυμητη δομή και θα μπορεί να παρέχεται απο το χρήστη.
Actor: Χρήστης
Περιγραφή περιπτώσεις χρήσης: Ανοιγμα XML αρχειου και μετασχηματισμού του μεσω XSLT, Φορτωση κατα την εκκίνηση της εφαρμοογή των κλάσεων μετασχηματισμού.

Περίπτωση χρήσης: Load generic XML
Απαίτηση: Ο χρήστης μπορεί να φορτώσει ένα XML αρχείο οποιασδήποτε μορφής στην εφαρμογή.
Actor: Χρήστης
Περιγραφή περιπτώσεις χρήσης: Ανοιγμα XML αρχείου

Περίπτωση χρήσης: Save Data Structure file
Απαίτηση: Ο χρήστης μπορει να αποθήκευσει αρχείο δομης δεδομένων.
Actor: Χρήστης
Περιγραφή περιπτώσεις χρήσης: Αποθήκευση ενος αρχείου δομης δεδομένων.

Περίπτωση χρήσης: Create Association
Απαίτηση: Ο χρήστης μπορει να δημιουργήσει συσχετισμούς μεταξύ ενος αρχειο συσχετίσεων και των πηγών δεδομένων και αρχείων δεδομένων που έχει φορτώσει με ευκολο τρόπο. Ο δομές δεδομένων που θα έχουν φορτωθεί δεν θα επιτρεπεται να τροποποιηθούν (Read Only). Οι συσχετισμοι θα αποθηκεύονται μόνο στα φύλλα της δομής Πληροφοριων.
Actor: Χρήστης
Περιγραφή περιπτώσεις χρήσης: Δημιουργία συσχετιμού.

Περίπτωση χρήσης: Remove Association
Απαίτηση: Ο χρήστης μπορεί να αφαιρέσει ένα συσχετισμό που δημιούργησε.
Actor: Χρήστης
Περιγραφή περιπτώσεις χρήσης: Αφαίρεση συσχετισμού

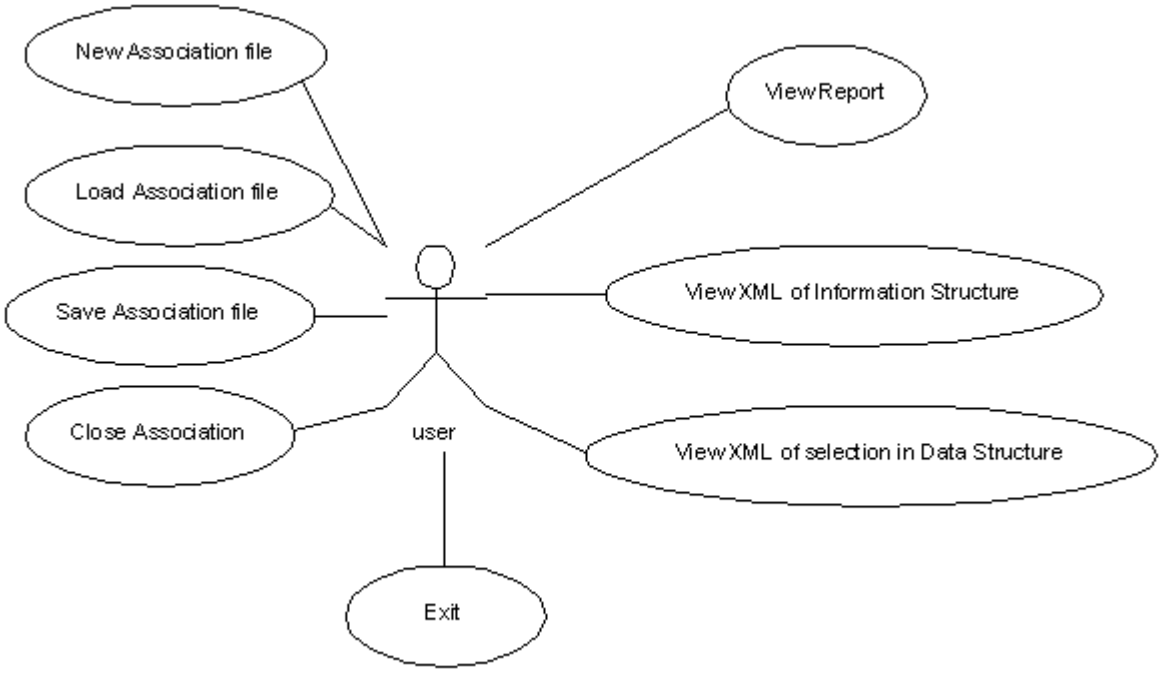
Περίπτωση χρήσης: Association Properties
Απαίτηση: Ο χρήστης χρήστης να δεί σε μια φόρμα όλες τις ιδιότητες του συσχετισμού που δημιουργησε και να τις τροποποιήσει.
Actor: Χρήστης
Περιγραφή περιπτώσεις χρήσης: Εμφανιση και τροποποίηση ιδιοτήτων του συσχετισμού

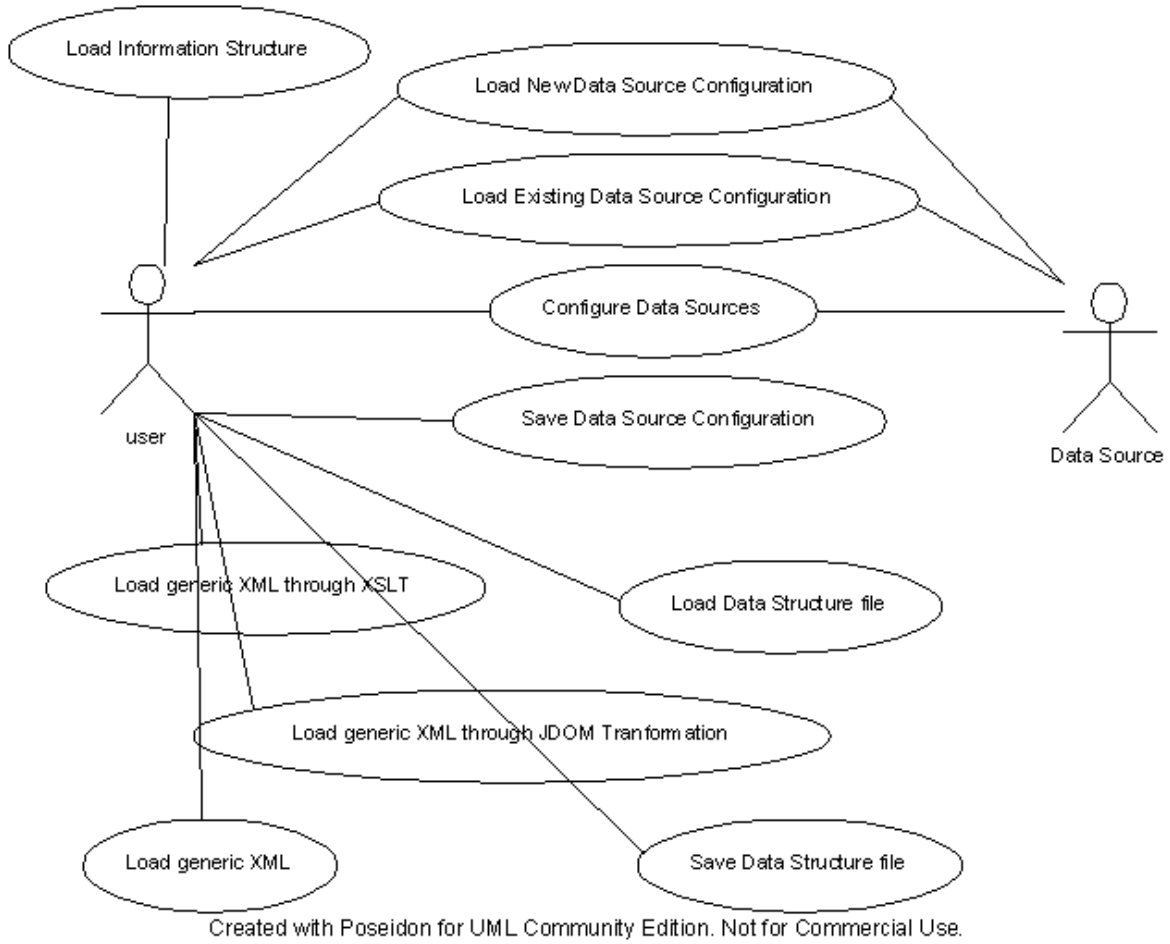
Περίπτωση χρήσης: Remove file

Απαίτηση: Ο χρήστης μπορεί να κλείσει ένα αρχείο δομής δεδομένων που φόρτωσε.
Actor: Χρήστης
Περιγραφή περιπτώσεις χρήσης: Κλείσιμο αρχείου δομής δεδομένων

Περίπτωση χρήσης: Remove all
Απαίτηση: Ο χρήστης μπορεί καθαρίσει τελείως το δενδρο της δομής δεδομένων. Δηλαδή να κλείσει όλα τα αρχεια δομής δεδομένων που έχει φορτώσει στην εφαρμογή καθώς και όλες τις πηγές που έχουν φορτωθεί.
Actor: Χρήστης
Περιγραφή περιπτώσεις χρήσης: Καθαρισμα του δενδρου δομής δεδομένων

Περίπτωση χρήσης: About application
Απαίτηση: Πληροφορίες για την εφαρμογή
Actor: Χρήστης
Περιγραφή περιπτώσεις χρήσης: Πληροφορίες για την εφαρμογή





Τεκμηρίωση Περιπτώσεων χρήσεως (Use cases)

<u>Περίπτωση χρήσης: Create Association file</u>
Σύντομη περιγραφή
Δημιουργία νεου αρχειου συσχέτισης.
Actors
Χρήστης
Προυποθέσεις
Να μην υπαρχει ήδη ανοικτο καποιο αλλο αρχειο συσχετίσεων
Κυρίως ροή
Το use case ξεκινάει όταν ο χρήστης επιλέγει το Create Association. Η εφαρμογή προτρέπει το χρήστη να δώσει ένα όνομα στο αρχειο συσχετίσεων. Ο χρήστης παρέχει ένα όνομα και πατάει OK.
Εναλλακτικές ροές

- Ο χρήστης δίνει κενο «» ονομα για το αρχείο συσχετίσεων. Εμφάνιση μηνύματος λάθους
- Ο χρηστης εγκαταλείπει την δημιουργια νεο αρχειο (παταει Cancel)

Postconditions
Αν το use case τελειώνει επιτυχώς δημιουργείται νέο αρχείο συσχετίσεων. Διαφορετικά, η κατάσταση του συστήματος δεν αλλάζει.

Περίπτωση χρήσης: Load Association file

Σύντομη περιγραφή

Ανοιγμα προυπαρχονταος αρχειου συσχέτισης

Actors

Χρήστης

Προϋποθέσεις

Να μην υπαρχει ήδη ανοικτο καποιο αλλο αρχειο συσχετίσεων

Κυρίως ροή

Το use case ξεκινάει όταν ο χρήστης επιλέγει το Load Association. Η εφαρμογή προτρέπει το χρήστη να επιλέξει ένα αρχειο συσχετίσεων. Ο χρήστης επιλέγει και πατάει OK.

Εναλλακτικές ροές

- Ο χρήστης επιλέγει λάθος αρχείο, δηλαδή αρχειο που δεν είναι αρχειο συσχετισμού. Εμφάνιση μηνύματος λάθους
- Ο χρήστης εγκαταλείπει την επιλογή αρχείο πατώντας Cancel

Postconditions

Αν το use case τελειώνει επιτυχώς φορτώνεται το αρχείο συσχετίσεων. Διαφορετικά, η κατάσταση του συστήματος δεν αλλάζει.

Περίπτωση χρήσης: Save Association file

Σύντομη περιγραφή

Αποθήκευσης Αρχείου συσχετίσεων **Actors**

Χρήστης

Προϋποθέσεις

Υπάρχει φορτωμένο ένα τροποποιημένο αρχειο συσχετίσεων που δεν εχει σωθει.

Κυρίως ροή

Το use case ξεκινάει όταν ο χρήστης επιλέγει Save Association. Είναι πρώτη φορα που σώζεται το αρχείο, και η εφαρμογή προτρέπει το χρήστη να καθορίσει την τοποθεσία που θέλει να σωθεί το αρχειο

Εναλλακτικές ροές

- Αν εχει σωθεί ήδη για πρώτη φορα, δηλαδη έχει καθοριστεί η τοποθεσία, τότε η εφαρμογη σώζει το αρχειο στην προκαθορισμένη τοποθεσία
- Ο χρήστης επιλέγει Cancel οταν ερωτηθει για την τοποθεσία.

Postconditions

Στην κυρίως ροή και στην πρώτη απο τις εναλλακτικές ροές το αρχειο έχει σωθει. Στην Εναλακτική ροή η κατάσταση του συστήματος δεν αλλάζει.

Περίπτωση χρήσης: Close Association file

Σύντομη περιγραφή

Κλείσιμο αρχείο συσχετίσεων.

Actors

Χρήστης

Προυποθέσεις

Να μην είναι ήδη ανοικτο καποιο αλλο αρχειο.

Κυρίως ροή

Το use case ξεκινάει όταν ο χρήστης επιλέγει το Close Association. Η εφαρμογη ελέγχει αν το αρχειο είναι σωσμένο και έχει τις ακόλουθες ροές:

- Αν αρχειο είναι σωσμένο, τοτε κλείνει και σταματάει να το εμφανίζει.

Εναλλακτικές ροές

- Αν αρχειο δεν εχει σωθεί, τότε ρωτάει το χρήστη αν θέλει να το σώσει. Ο χρήστης επιλέγει Ναι και η εφαρμογή σώζει το αρχειο εκτελώντας την περίπτωση χρήσης Save Association file.
- Ο χρήστης επιλέγει Οχι οταν ερωτηθεί και η εφαρμογή δεν σώζει τις τελευταίες τροποποιήσεις που έγιναν απο τότε που το αρχείο σώθηκε για τελευταια φορα. Κλείνει και σταματάει να εμφανίζει το αρχείο.
- Ο χρήστης επιλέγει Cancel οταν ερωτηθεί. Η εφαρμογή δεν κλεινει το αρχειο ούτε κα ιαπιθηκευει τις αλλαγές.

Postconditions
Στην περίπτωση που ο χρήστης πατήσει Cancel, τοτε η εφαρμογή επιστρέφει στην κατάσταση που βρισκόταν πριν την περίπτωση χρήσης. Στις υπόλοιπες τρείς περιπτώσεις επιστρέφει στην αρχική καταστάση που έχει όταν ξεκιναει δηλ χωρις να είναι φορτωμένο κανένα αρχειο συσχετίσεων ή να έχει δημιουργηθεί καποιο.

Περίπτωση χρήσης: View Report
Σύντομη περιγραφή
Εμφάνιση Αναφοράς Συσχετίσεων
Actors
Χρήστης
Προυποθέσεις
Να έχει δημιουργηθεί ενας νέο ή να έχει φορτωθεί ένα αρχειο συσχετίσεων
Κυρίως ροή
Το use case ξεκινάει όταν ο χρήστης επιλέγει το View Report on Information Structure. Η εφαρμογή εμφανίζει μια αναφορα σε μορφή πινάκα με τους συσχετισμούς που έχουν γίνει.
Εναλλακτικές ροές
Καμια
Postconditions
Οταν ο χρήστης κλείσει το παράθυρο της αναφορά το σύστημα επιστρέφει στην κατάσταση που ήταν πριν.

Περίπτωση χρήσης: View XML of Information file
Σύντομη περιγραφή
Εμφανιση XML του Αρχείο Συσχετίσεων
Actors
Χρήστης
Προυποθέσεις
Να έχει δημιουργηθεί ενας νέο ή να έχει φορτωθεί ένα αρχειο συσχετίσεων
Κυρίως ροή
Το use case ξεκινάει όταν ο χρήστης επιλέγει το View XML of Information Structure. Η εφαρμογή εμφανίζει το XML κείμενο που αντιστοιχεί στο δενδρική δομή που εμφανίζει το πρόγραμμα.
Εναλλακτικές ροές
Καμια
Postconditions
Οταν ο χρήστης κλείσει το παράθυρο της αναφορά το σύστημα επιστρέφει στην κατάσταση που ήταν πριν.

Περίπτωση χρήσης: View XML of selection in Data Structure
Σύντομη περιγραφή
Εμφανιση XML των δομων δεδομένων

Actors
Χρήστης
Προυποθέσεις
Να έχει δημιουργηθεί ενας νέο ή να έχει φορτωθεί ένα αρχειο συσχετίσεων
Και ο χρήστης έχει επιλέξει οποιοδήποτε απο τους κόμβου του δενδρου της δομής δεδομένων.
Κυρίως ροή
Το use case ξεκινάει όταν ο χρήστης επιλέξει View XML of selection in Data Structure.
Εναλλακτικές ροές
Καμια
Postconditions
Οταν ο χρήστης κλείσει το παράθυρο της αναφορά το σύστημα επιστρέφει στην κατάσταση που ήταν πριν.

Περίπτωση χρήσης: Exit Application
Σύντομη περιγραφή
Κλείσιμο εφαρμογής
Actors
Χρήστης
Προυποθέσεις
Καμια. Ο χρήστης μπορεί να κλεισει οποιαδήποτε στιγμη την εφαρμογή
Κυρίως ροή
Το use case ξεκινάει όταν ο χρήστης επιλέξει Exit. Η εφαρμογή ελέγχει αν υπαρχει ανοικτό κάποιο αρχειο και οχι, τοτε κλείνει.
Εναλλακτικές ροές
- Αν είναι ανοικτο καποιο αρχειο συσχετίσεων τότε εκτελείται η περίπτωση χρήσης Close Association και στην συνέχεια η εφαρμογη κλείνει

Postconditions
Η εφαρμογή έχει κλείσει.

Περίπτωση χρήσης: Load Information Structure
Σύντομη περιγραφή
Ανοιγμα Δομής πληροφοριών
Actors
Χρήστης
Προυποθέσεις
Να έχει δημιουργηθεί ένα νέο αρχειο συσχετίσεων
Κυρίως ροή
Το use case ξεκινάει όταν ο χρήστης επιλέγει Load Information Structure. Η εφαρμογή προτρέπει το χρήστη να επιλέξει ένα αρχειο Δομής πληροφοριών. Ο χρήστης επιλέγει και πατάει OK.
Εναλλακτικές ροές
- Ο χρήστης επιλέγει λάθος αρχείο. Εμφάνιση μηνύματος λάθους
- Ο χρήστης εγκαταλείπει την επιλογή αρχείο πατώντας Cancel

Postconditions
Αν το use case τελειώνει επιτυχώς φορτώνεται το αρχείο Δομής πληροφοριών. Διαφορετικά, η κατάσταση του συστήματος δεν αλλάζει.

Περίπτωση χρήσης: Load New Data Source Configuration
Σύντομη περιγραφή
Δημιουργία Διαμόρφωσης και Φόρτωση Διαμόρφωσης Πηγών δεδομένων

Actors
Χρήστης
Προυποθέσεις
Καμια. Ο χρήστης μπορει να δημιουργήσει διαμορφώσεις πηγών δεδομένων και να τις φορτώσει ακομα και όταν δεν υπαρχει ανοικτό κάποιο αρχείο συσχετίσεων.
Κυρίως ροή
Το use case ξεκινάει όταν ο χρήστης επιλέξει Load New Data Source Configuration. Η εφαρμογή εμφανίζει στο χρήστης ένα παράθυρο όπου έχει επιλογές να δημιουργήσει μια νέα διαμόρφωση πηγων δεδομένων.
Εναλλακτικές ροές
Καμία
Postconditions
Όταν το use case τελειώνει φορτώνονται οι πηγες της νέας διαμορφωσης στην εφαρμογή

<u>Περίπτωση χρήσης: Save Data Source Configuration</u>
Σύντομη περιγραφή
Αποθήκευση Διαμόρφωσης Πηγών δεδομένων
Actors
Χρήστης
Προυποθέσεις
Καμια.
Κυρίως ροή
Το use case ξεκινάει όταν ο χρήστης επιλέγει Save Data Source Configuration. Η εφαρμογή προτρέπει το χρήστη να δώσει το όνομα και την τοποθεσια του αρχείου. Ο χρήστης τα παρέχει και πατάει OK.
Εναλλακτικές ροές
Ο χρήστης επιλέγει Cancel.
Postconditions
Στην κυρίως ροή το αρχειο έχει σωθει. Στην Εναλακτική ροή η κατάσταση του συστήματος δεν αλλάζει.

<u>Περίπτωση χρήσης: Load Existing Data Source Configuration</u>
Σύντομη περιγραφή
Ανοιγμα αρχείο διαμόρφωσης πηγών δεδομένων και Φόρτωση Διαμόρφωσης Πηγών δεδομένων
Actors
Χρήστης
Προυποθέσεις
Καμια. Ο χρήστης μπορει να φορτώσει διαμορφώσεις πηγών δεδομένων ακομα και όταν δεν υπαρχει ανοικτό κάποιο αρχείο συσχετίσεων.
Κυρίως ροή
Το use case ξεκινάει όταν ο χρήστης επιλέγει Load Existing Data Source Configuration. Η εφαρμογή προτρέπει το χρήστη να επιλέξει ένα αρχειο. Στην συνέχεια φορτώνει τις πηγές δεδομένων που περιγράφονται απο το αρχείο.
Εναλλακτικές ροές

- Ο χρήστης επιλέγει ένα αρχειο που δεν ειναι αρχείο διαμόρφωσης. Η εφαρμογη εμφανίζει μήνυμα λάθους.
- Ο χρήστης επιλέγει Cancel

Postconditions

Αν το use case τελειώνει επιτυχώς (κυρίως ροή) τοτε στην εφαρμογή έχουν φορτωθεί οι πηγές. Διαφορετικά, ηα κατάσταση του συστήματος δεν αλλάζει.

Περίπτωση χρήσης: Configure Data Sources

Σύντομη περιγραφή

Διαχειριση πηγών δεδομένων

Actors

Χρήστης

Προυποθέσεις

Καμια. Ο χρήστης μπορει να διαχειριστει διαμορφώσεις πηγών δεδομένων ακομα και όταν δεν υπαρχει ανοικτό κάποιο αρχείο συσχετίσεων.

Κυρίως ροή

Το use case ξεκινάει όταν ο χρήστης επιλέγει Configure Data Sources. Η εφαρμογή εμφανίζει στο χρήστης ένα παράθυρο όπου μπορει να διαχειριστεί μια διαμόρφωση πηγων δεδομένων.

Εναλλακτικές ροές

Καμία

Postconditions

Όταν το use case τελειώνει φορτώνονται οι πηγες της νέας διαμορφωσης στην εφαρμογή

Περίπτωση χρήσης: Load Data Structure file

Σύντομη περιγραφή

Ανοιγμα αρχειου δομής δεδομένων

Actors

Χρήστης

Προυποθέσεις

Καμία.

Κυρίως ροή

Το use case ξεκινάει όταν ο χρήστης επιλέγει Load Data Structure file. Η εφαρμογή προτρέπει το χρήστη να επιλέξει ένα αρχειο δομής δεδομένων. Ο χρήστης επιλέγει και πατάει OK.

Εναλλακτικές ροές

- Ο χρήστης επιλέγει λάθος αρχείο. Εμφάνιση μηνύματος λάθους
- Ο χρήστης εγκαταλείπει την επιλογή αρχείο πατώντας Cancel

Postconditions

Αν το use case τελειώνει επιτυχώς φορτώνεται το αρχείο. Διαφορετικά, η κατάσταση του συστήματος δεν αλλάζει.

Περίπτωση χρήσης: Load generic XML through XSLT

Σύντομη περιγραφή

Ανοιγμα XML αρχειου και μετασχηματισμού του μεσω XSLT.

Actors

Χρήστης

Προυποθέσεις

Καμια

Κυρίως ροή

Το use case ξεκινάει όταν ο χρήστης επιλέγει Load generic XML through XSLT. Η εφαρμογή προτρέπει το χρήστη να επιλέξει ένα αρχειο XML και το αντίστοιχο XSL αρχείο που θα χρησιμοποιηθεί για το μετασχηματισμό. Ο χρήστης επιλέγει και τα δυο και πατάει OK.

Εναλλακτικές ροές

- Ο χρήστης επιλέγει λάθος XML αρχείο. Εμφάνιση μηνύματος λάθους
- Ο χρήστης επιλέγει λάθος XSL αρχείο. Εμφάνιση μηνύματος λάθους

- Ο χρήστης εγκαταλείπει την επιλογή αρχείο πατώντας Cancel

Postconditions
Αν το use case τελειώνει επιτυχώς φορτώνεται το αρχείο. Διαφορετικά, η κατάσταση του συστήματος δεν αλλάζει.

Περίπτωση χρήσης: Load generic XML through JDOM Transformation
Σύντομη περιγραφή
Ανοιγμα XML αρχειου και μετασχηματισμού του μεσω JDOM κλάσεων, Φορτωση κατα την εκκίνηση της εφαρμογή των κλάσεων μετασχηματισμού.
Actors
Χρήστης
Προυποθέσεις
Καμια
Κυρίως ροή
Το use case ξεκινάει όταν ο χρήστης επιλέγει Load generic XML through JDOM Transformation. Η εφαρμογή προτρέπει το χρήστη να επιλέξει ένα αρχειο XML και την αντίστοιχη κλάση μετασχηματισμού που θα χρησιμοποιηθεί για το μετασχηματισμό. Ο χρήστης επιλέγει και τα δυο και πατάει OK.
Εναλλακτικές ροές

- Ο χρήστης επιλέγει λάθος XML αρχείο. Εμφάνιση μηνύματος λάθους
- Ο χρήστης εγκαταλείπει την επιλογή αρχείο πατώντας Cancel

Postconditions
Αν το use case τελειώνει επιτυχώς φορτώνεται το αρχείο. Διαφορετικά, η κατάσταση του συστήματος δεν αλλάζει.

Περίπτωση χρήσης: Load generic XML
Σύντομη περιγραφή
Ανοιγμα XML αρχείου
Actors
Χρήστης
Προυποθέσεις
Καμια
Κυρίως ροή
Το use case ξεκινάει όταν ο χρήστης επιλέγει Load generic XML. Η εφαρμογή προτρέπει το χρήστη να επιλέξει ένα XML αρχειο. Ο χρήστης επιλέγει και πατάει OK.
Εναλλακτικές ροές

- Ο χρήστης επιλέγει λάθος αρχείο. Εμφάνιση μηνύματος λάθους
- Ο χρήστης εγκαταλείπει την επιλογή αρχείο πατώντας Cancel

Postconditions
Αν το use case τελειώνει επιτυχώς φορτώνεται το αρχείο. Διαφορετικά, η κατάσταση του συστήματος δεν αλλάζει.

Περίπτωση χρήσης: Save Data Structure file
Σύντομη περιγραφή
Αποθήκευση ενος αρχείου δομης δεδομένων
Actors
Χρήστης
Προυποθέσεις
Και ο χρήστης έχει επιλέξει στο δενδρο της δομής δεδομένων ένα αρχείο δομής δεδομένων.
Κυρίως ροή

Το use case ξεκινάει όταν ο χρήστης επιλέγει Save Data Structure file. Η εφαρμογή προτρέπει το χρήστη να δώσει το όνομα και την τοποθεσια του αρχείου. Ο χρήστης τα παρέχει και πατάει OK.

Εναλλακτικές ροές

- Ο χρήστης επιλέγει κομβο που δεν αντιπροσωπεύει αρχειο. Εμφάνιση μηνύματος λάθους
- Ο χρήστης εγκαταλείπει πατώντας Cancel

Postconditions

Στην κυρίως ροή το αρχειο έχει σωθει. Στις Εναλακτικες η κατάσταση του συστήματος δεν αλλάζει.

<u>Περίπτωση χρήσης: Create Association</u>

Σύντομη περιγραφή

Δημιουργία συσχετιμού

Actors

Χρήστης

Προυποθέσεις

Ο χρήστης έχει επιλέξει τον κόμβο στο δένδρο δομής δεδομένων που θέλει να συσχετίσει.

Κυρίως ροή

Το use case ξεκινάει όταν ο χρήστης κάνει Drag το κόμβο που έχει επιλέξει. Ο χρήστης κάνει drop πανω σε ένα φύλλο του δένδρου της δομής πληροφορίας. **Εναλλακτικές ροές**

- Ο χρήστης κάνει drop σε οποιοδήποτε κόμβο του δένδρου της δομής δεδομένων. Εμφάνιση μηνύματος λάθους.
- Ο χρήστης κάνει drop σε οποιοδήποτε κόμβο του δένδρου της δομής πληροφορίας που δεν είναι φύλλο της. Εμφάνιση μηνύματος λάθους.

Postconditions

Αν το use case τελειώνει επιτυχώς δημιουργείται μια συσχέτιση μεταξυ του dragged κόμβου και του drop κόμου. Διαφορετικά η κατάσταση του συστήματος δεν αλλάζει.

<u>Περίπτωση χρήσης: Remove Association</u>

Σύντομη περιγραφή

Αφαίρεση συσχετισμού

Actors

Χρήστης

Προυποθέσεις

Ο χρήστη κάνει δεξί κλικ πάνω απο το δενδρο δομής Πληροφορίας.

Κυρίως ροή

Το use case ξεκινάει όταν ο χρήστης επιλέγει ένα συσχετισμό και μετα επιλέγει Remove Association.

Εναλλακτικές ροές

- Ο χρήστης έχει επιλέξει ενα κόμβο που δεν ειναι συσχετισμό. Εμφάνιση μηνύματος λάθους

Postconditions

Αν το use case τελειώνει επιτυχώς ο συσχετισμό αφαιρείται. Διαφορετικά η κατάσταση του συστήματος δεν αλλάζει

<u>Περίπτωση χρήσης: Association Properties</u>

Σύντομη περιγραφή

Εμφανιση και τροποποίηση ιδιοτήτων του συσχετισμού

Actors
Χρήστης
Προυποθέσεις
Ο χρήστη κάνει δεξί κλικ πάνω απο το δενδρο δομής Πληροφορίας.
Κυρίως ροή
Το use case ξεκινάει όταν ο χρήστης επιλέγει ένα φύλλο του δένδρου δομής Πληροφορίας και μετα επιλέγει Association Properties. Η εφαρμογή εμφανίζει ένα παράθυρο όπου ο χρήστης μπορεί να δει και τροποποιήσει τις ιδιότητες ενός συσχετισμού.
Εναλλακτικές ροές

- Ο χρήστης έχει επιλέξει ενα κόμβο που δεν ειναι φύλλο του δένδρου δομής Πληροφορίας . Εμφάνιση μηνύματος λάθους

Postconditions
Αν το use case τελειώνει επιτυχώς οι ιδιότητες του συσχετισμού εχουν τροποποιηθεί κατ' επιθυμία του χρήστη. Διαφορετικά η κατάσταση του συστήματος δεν αλλάζει

<u>Περίπτωση χρήσης: Remove file</u>
Σύντομη περιγραφή
Κλείσιμο αρχείου δομής δεδομένων
Actors
Χρήστης
Προυποθέσεις
Ο χρήστη κάνει δεξί κλικ πάνω απο το δενδρο δομής Δεδομένων και έχει επιλέξει στο δενδρο της δομής δεδομένων ένα αρχείο δομής δεδομένων
Κυρίως ροή
Το use case ξεκινάει όταν ο χρήστης επιλέγει επιλέγει ένα αρχειο και μετα επιλέγει Remove file.
Εναλλακτικές ροές

- Ο χρήστης επιλέγει μια πηγή δεδομένων. Εμφάνιση μηνύματος λάθους.

Postconditions
Αν το use case τελειώνει επιτυχώς επιλεγμένο αρχείο αφαιρείται. Διαφορετικά η κατάσταση του συστήματος δεν αλλάζει.

<u>Περίπτωση χρήσης: Remove all</u>
Σύντομη περιγραφή
Καθαρισμα του δενδρου δομής δεδομένων
Actors
Χρήστης
Προυποθέσεις
Ο χρήστη κάνει δεξί κλικ πάνω απο το δενδρο δομής Δεδομένων
Κυρίως ροή
Το use case ξεκινάει όταν ο χρήστης επιλέγει Remove all.
Εναλλακτικές ροές
Καμια
Postconditions
Το δένδρο της δομής δεδομένω έχει καθαρίσει απο όλες τις πηγές και τα αρχεία που είχαν φορτωθεί σε αυτό.

<u>Περίπτωση χρήσης: About application</u>
Σύντομη περιγραφή
Πληροφορίες για την εφαρμογή
Actors

Χρήστης
Προυποθέσεις
Καμια
Κυρίως ροή
Το use case ξεκινάει όταν ο χρήστης επιλέγει About. Εμφάνιση πληροφοριων της εφαρμογης
Εναλλακτικές ροές
Καμια
Postconditions
Η κατάσταση του συστήματος δεν αλλάζει.

Σχεδιασμός της εφαρμογής

Περιγράφεται η σχεδιασμός της εφαρμογής σε αρχιτεκτονικό επίπεδο. Το ρόλο της ανάλυσης του λεπτομερούς σχεδιασμού καλύπτει το API της εφαρμογής που βρίσκεται στο επόμενο κεφαάλιο.

Πρέπει να επισημανθεί ότι τα UML *διαγράμματα κλάσεων,* που παρέχονται για την καλύτερη κατανόηση της δομής της εφαρμογής, *δεν είναι πλήρη*. Πολλές φορές καποιές δευτερεύοντες σε σημασία κλάσεις που σχετίζονται με την κυρίως κλάση που περιγράφεται, παραλείπονται. Αυτο είναι απαραίτητο για την σωστή απεικόνιση του διαγράμματος διοτι λόγω του μεγέθους της εφαρμογής τα διαγραμματα τείνουν να γίνουν εξαιτερικά μεγάλα.

Δομές δεδομένων που χρησιμοποιεί η εφαρμογή

Όλες οι δομές δεδομένων που διαχειρίζεται η εφαρμογή είναι XML δομές, οι οποίες είναι:

- Information Structure (Δομη Πληροφοριών)
- Data Structure (Δομή Δεδομένων)
- Data Source Configuration (Διαμόρφωση Πηγών Δεδομένων)

Η παραπάνω δομές έχουν μια καθαρά XML δομή, ετσι ώστε να είναι πολύ εύκολο να φορτώθει στην εφαρμογή μια δομή που ειναι αποθηκευμένη σε ένα xml αρχείο. Δεν πρέπει να συγχέονται παραπάνω δομές με τα αρχείο εισόδου και εξόδου της εφαρμογής. Οι παραπάνω είναι οι δομές, δηλαδή τα μοντέλα δεδομένων, που διαχειρίζεται εσωτερικά της η εφαρμογή.

Information Structure

Είναι η δομή που θα προστεθούν οι συσχετισμοί. Ένα Αρχείο Συσχετισμών αποτελείται από ένα Information Structure στο οποίο έχουν προστεθεί συσχετισμοί.
Ακολουθεί ένα παράδειγμα της XML μορφής μιας Information Structure η οποία δεν έχει καποιο συσχετισμό:

```
<?xml version="1.0" encoding="utf-8"?>
<informationstructure type="testType">
      <informationcategory name="Customer Information" abstract="false">
            <informationfield name="Last Name" type="String"
haspredefinedvalues="false" hasvaluerange="false" associated="No">
        </informationfield>
            <informationfield name="First Name" type="String"
haspredefinedvalues="false" hasvaluerange="false" associated="No">
        </informationfield>
      </informationcategory>
      <informationcategory name="Connection Information" abstract="false">
            <informationfield name="Payment Type" type="String"
hasvaluerange="false" haspredefinedvalues="false" associated="No">
          </informationfield>
      </informationcategory>
</informationstructure>
```

Η εφαρμογη απαιτεί η XML δομή που διαχειρίζεται να έχει την παραπάνω συγκεκριμένη δομή. Οποιοδήποτε xml αρχειο με κατάληξη *.is με την παραπάνω δομή μπορεί να φορτωθεί στην εφαρμογή σαν μια Information Structure.

Data Structure

Στην δομή δεδομένων φορτώνονται πηγές για να συσχετιστούν με την Information Structure. Οι πηγές μπορεί να έιναι:

- Πηγές δεδομένων απο βάσεις δεδομένων, πχ. ODBC πηγες δεδομένων
- DST αρχεία
- XML αρχεία

Πηγές δεδομένων απο βάσεις δεδομένων

Μια απο τις απαιτήσεις της εφαρμογής ήταν να μπορεί ο χρήστης να συσχετίσει την Information Structure του με πεδία μιας βάσης δεδομένων. Για να επιτευχθεί αυτό η δομή της πηγής δεδομένων που προσφέρει η βαση αντιστοιχείται (map) σε μια XML δομή. Ένα παράδειγμα μιας XML δομής στην οποία έχει αντιστοιχηθεί η βάση είναι η ακόλουθή:

```
  <dataStructureSource name="customerspass" type="ODBC">
    <!--Comment: Sample Data Source Structure-->
    <dataStructureTable name="Connections" abstract="false">
     </dataStructureTable>
    <dataStructureTable name="Customers" abstract="false">
      <dataStructureField name="CustomerID" type="INTEGER"
class="java.lang.Integer" haspredefinedvalues="false" hasvaluerange="false"
/>
      <dataStructureField name="LastName" type="VARCHAR"
class="java.lang.String" haspredefinedvalues="false" hasvaluerange="false"
/>
      <dataStructureField name="FirstName" type="VARCHAR"
class="java.lang.String" haspredefinedvalues="false" hasvaluerange="false"
/>
     </dataStructureTable>
    <dataStructureTable name="Payments" abstract="false">
     </dataStructureTable>
  </dataStructureSource>
```

Η παραπάνω μορφή ορίζει την Data Structure file μορφή ενος xml κειμένου που αναφέρθηκε παραπάνω.

DST δομή

Η δομή αυτή έχει την παραπάνω μορφή, δηλαδή την μορφή που έχει η XML αντιστοιχία μιας πηγής δεδομένων. Φορτώνεται στην εφαρμογή μέσω ενός DST αρχείου, το οποίο είναι ένα xml αρχειο που έχει την παραπάνω μορφή και που μπορεί να έχει προκύψει με οποιοδήποτε τρόπο. Το αρχείο αυτό μπορεί να φορτωθεί στην εφαρμογή και να συσχετιστεί.

XML δομή

Μια οποιαδήποτε xml δομή. Φορτώνεται μέσω ένος XML αρχειου στην εφαρμογή και να συσχετιστεί.

Ακολουθεί ένα της Data Structure όταν έχουν φορτωθεί και τα τρία είδη δομών.

```
<dataStructure>
  <dataStructureSource name="customerspass" type="ODBC">
    <!--Comment: Sample Data Source Structure-->
```

```
    <dataStructureTable name="Connections" abstract="false">
    </dataStructureTable>
    <dataStructureTable name="Customers" abstract="false">
      <dataStructureField name="CustomerID" type="INTEGER"
class="java.lang.Integer" haspredefinedvalues="false" hasvaluerange="false"
/>
      <dataStructureField name="LastName" type="VARCHAR"
class="java.lang.String" haspredefinedvalues="false" hasvaluerange="false"
/>
      <dataStructureField name="FirstName" type="VARCHAR"
class="java.lang.String" haspredefinedvalues="false" hasvaluerange="false"
/>
    </dataStructureTable>
    <dataStructureTable name="Payments" abstract="false">
    </dataStructureTable>
  </dataStructureSource>
  <dataStructureFile name="sampleJrm.dst" type="dst">
    <!-- Comment: Sample Data Source Structure -->
    <dataStructureTable name="Customers" abstract="false">
      <dataStructureField name="CustomerID" type="Integer"
haspredefinedvalues="false" hasvaluerange="false" />
      <dataStructureField name="Last Name" type="String"
haspredefinedvalues="false" hasvaluerange="false" />
      <dataStructureField name="First Name" type="String"
haspredefinedvalues="false" hasvaluerange="false" />
      </dataStructureTable>
  </dataStructureFile>
  <root name="genericSample.xml" type="xml">
    <customers>
      <Customers>
        <CustomerID>
          123
          <type haspredefined="true" hasvaluerange="false">Integer</type>
        </CustomerID>
        <LastName>
          Jagger
          <type haspredefined="false" hasvaluerange="false">VARCHAR</type>
        </LastName>
        <FirstName>
          Mick
          <type haspredefined="false" hasvaluerange="false">VARCHAR</type>
        </FirstName>
        </Customers>
    </customers>
  </root>
</dataStructure>
```

Data Source Configuration

Η δομή αυτή διατηρεί τις παραμέτρους όλων των πηγών δεδομένων που είναι φορτωμένες στην εφαρμογή. Ενα παράδειγμα της είναι:

```
    <datasources>
      <datasource type="ODBC" driver="sun.jdbc.odbc.JdbcOdbcDriver">
         <configuration dsn="customersPass" login="jrm" >
         </configuration>
      </datasource>
      <datasource type="ODBC" driver="sun.jdbc.odbc.JdbcOdbcDriver">
         <configuration dsn="billingPass" login="jrm" >
```

```
        </configuration>
      </datasource>
   </datasources>
```
όπου περιγράφονται δύο πηγές δεδομένων η customersPass και η billingPass.

Συσχετισμοί (Associations)

Οι συσχετισμοί (Associations) είναι με σειρά απο παραμέτρους που προστείθονται στα φύλλα μιας Information Structure και τα συσχετίζουν με κόμβους της Data Structure. Οι συσχετισμοί είναι ν προς 1. Δηλαδή ένας Information Field μπορεί να συσχετιστεί μόνο με ένα κόμβο του Data Structure, αλλα ένας κόμβος της Data Structure μπορεί να συσχετίσει πάνω από ένα Information Field.

Ένας συσχετισμός αποτελείται από τα παρακάτω στοιχεία, τα οποία προστείθονται στο Information Field:

- associated attribute: δηλώνει αν είναι ή όχι συσχετισμένο ένα Information Field
- associationPath attribute: έχει το Path μιας ODBC πηγής. Πχ. `customerspass:Customers:LastName,` δηλώνει την στήλη LastName του πίνακα Custoomers της ODBC πηγής customerPass. Στην περίπτωση που ο συσχετισμός αναφέρεται σε αρχείο τότε ταυτίζεται με το XPath.
- Source attribute: έχει το ονομα του αρχείο ή της ODBC πηγής που αναφέρεται ο συσχετισμός.
- Text κόμβος: περιέχει το XPath. Στην περίπτωση των πηγών δεδομένων, το XPath δείχνει το κόμβο της xml δομής που έχει αντιστοιχηθεί η πηγή. Στην περίπτωση των αρχειων δείχνει το κόμβο στο αρχείο.

Ακολουθεί ενα παράδειγμα όπου εχουν προστεθεί δυο συσχετισμοί (με έντονα γράμματα) στο παράδειγμα της Information Structure:

```
<informationstructure type="testType">
  <informationcategory name="Customer Information" abstract="false">
    <informationfield name="Last Name" type="String"
haspredefinedvalues="false" hasvaluerange="false" associated="Yes"
associationPath="customerspass:Customers:LastName (Type: VARCHAR)"
source="customerspass">/dataStructureSource/dataStructureTable[2]/dataStruct
ureField[2]</informationfield>
    <informationfield name="First Name" type="String"
haspredefinedvalues="false" hasvaluerange="false" associated="Yes"
associationPath="customerspass:Customers:FirstName (Type: VARCHAR)"
source="customerspass">/dataStructureSource/dataStructureTable[2]/dataStruct
ureField[3]</informationfield>
  </informationcategory>
  <informationcategory name="Connection Information" abstract="false">
    <informationfield name="Payment Type" type="String"
hasvaluerange="false" haspredefinedvalues="false" associated="No" />
  </informationcategory>
</informationstructure>
```

Αρχεία εισόδου της εφαρμογής

Η εφαρμογή μπορεί να δεχτεί τα παρακάτω αρχεία σαν είσοδο της:

Ονομα αρχείου	Extension	Περιγραφή
Association file	*.asc	Αρχεια Συσχετισμού που φορτώνονται για να τροποποιηθουν. Αποτελούνται απο Information Structure δομές στις οποιες έχουν γίνει συσχετισμοί.
Information file	*.is	Τα αρχεια που περιέχουν την δομή Information Structure, πανω στην οποία θα δημιουργηθούν οι συσχετισμοί
Data Source Configuration file	*.dsc	Αρχειο που περιέχει την δομή μιας διαμόρφωση πηγών δεδομένων
XML file	*.xml	Οποιοδήποτε αρχείο XML μπορεί να φορτωθεί στην εφαρμογή με δυο τρόπους: μέσω μετασχηματισμού και χωρίς
XSLT file	*.xsl	XSL αρχείο για το μετασχηματισμό ενός XML αρχείο σε μια DST δομή
Java class files	*.class	Κλάσεις για το μετασχηματισμό ένος αρχείου XML σε DST δομή
Data Structure file	*.dst	DST αρχεία

Τα αρχεία *.asc,*.is, *.dsc, *.dst είναι ουσιαστικά XML αρχεια. με διαφορετική κατάληξη.

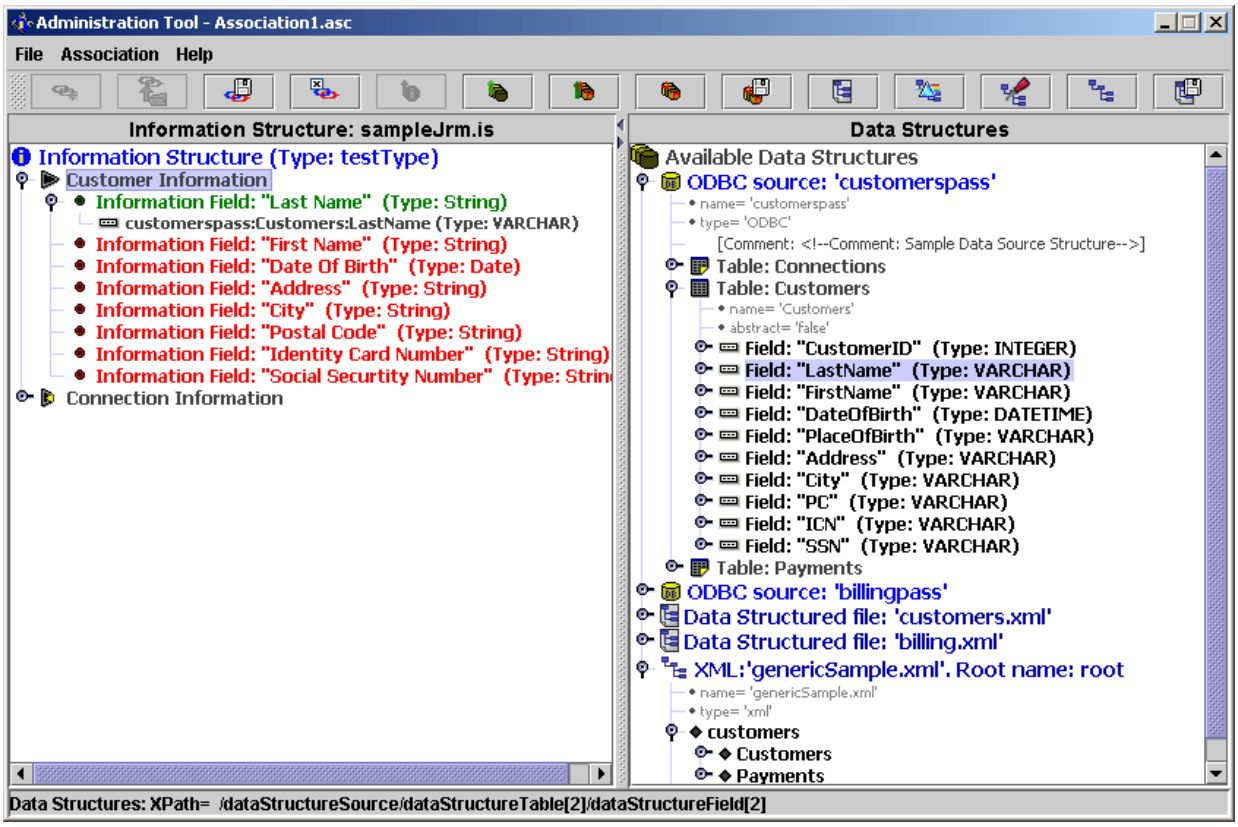
Αρχεία εξόδου της εφαρμογής

Η εφαρμογή παράγει τα ακόλουθα αρχεία:

Ονομα αρχείου	Extension	Περιγραφή
Association file	*.asc	Αρχεια Συσχετισμού. Αποτελούνται απο Information Structure δομές στις οποιες έχουν γίνει συσχετισμοί. Είναι και το κυρίως προιον της εφαρμογής.
Data Source Configuration file	*.dsc	Αρχειο που περιέχει την δομή μιας διαμόρφωση πηγών δεδομένων. Ο χρήστης μπορεί να δημιουργήσει μια διαμόρφωση και να την απθηκευσει ώστε να μην χρειαζεται να την δημιουργήσει πάλι.
Data Structure file	*.dst	XML αρχεία που έχουν μετασχηματιστεί στην DST δομή μπορεί να αποθηκευτούν σε αυτή την δομή ώστε ο χρήστης να μην χρειάζεται να τα μετασχηματίσει πάλι.

Το κυρίως παράθυρο της εφαρμογής

Ακολουθεί ένα στιγμιότυπο του κυρίως παραθύρου της εφαρμογής

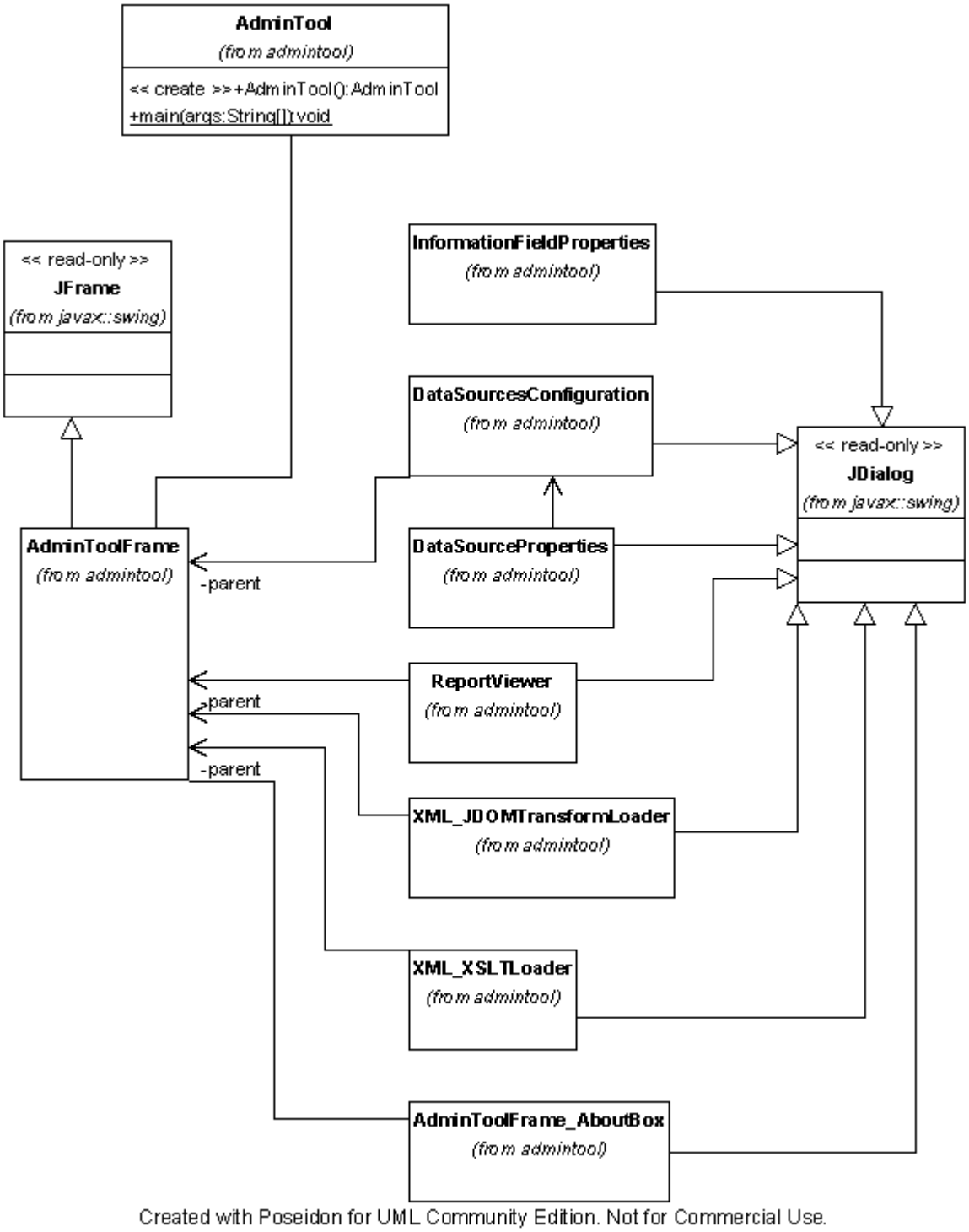


admintool.AdminTool

Είναι η κλάση που έχει την main μεθοδο που ξεκινά την εφαρμογή. Η κλάση αυτή περιορίζεται στο να δημιουργεί ένα AdminToolFrame και να το απεικονίσει σωστά.

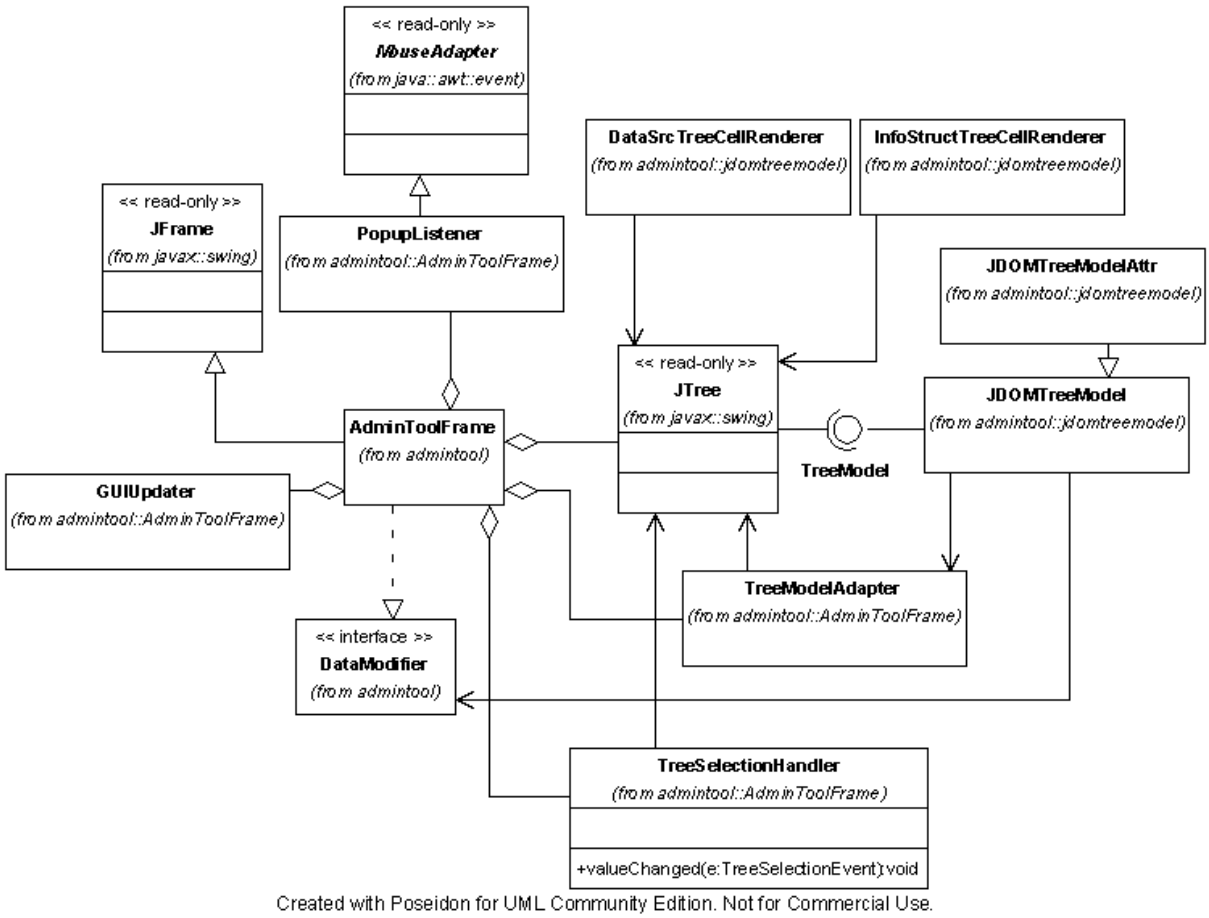
adintool.AdminToolFrame

Αποτελεί την κυρίως κλάση της εφαρμογής. Όπως φαίνεται και απο τα παρακάτω διάγραμματα όλη η λειτουργικότητα της εφαρμογής υλοποιείται γύρω απο αυτή.
Το ακόλουθο διάγραμμα απεικονίζει της κλάσεις που αποτελούν τα παράθυρα της εφαρμογής.



Η κλάση είναι ένα JFrame (επεκτείνει την JFrame) και αποτελεί το μονο και κυρίως παράθυρο της εφαρμογής. Καλέι όλα τα παράθυρα διαλόγου που φαίνονται στο δεξί μέρος το διαγράμματος. Τα παράθυρα διαλόγου είναι οι κλάσεις που επεκτείνουν το JDialog. Το JDialog παρέχεται απο το Swing για την δημιουργία μικρό παραθύρων.

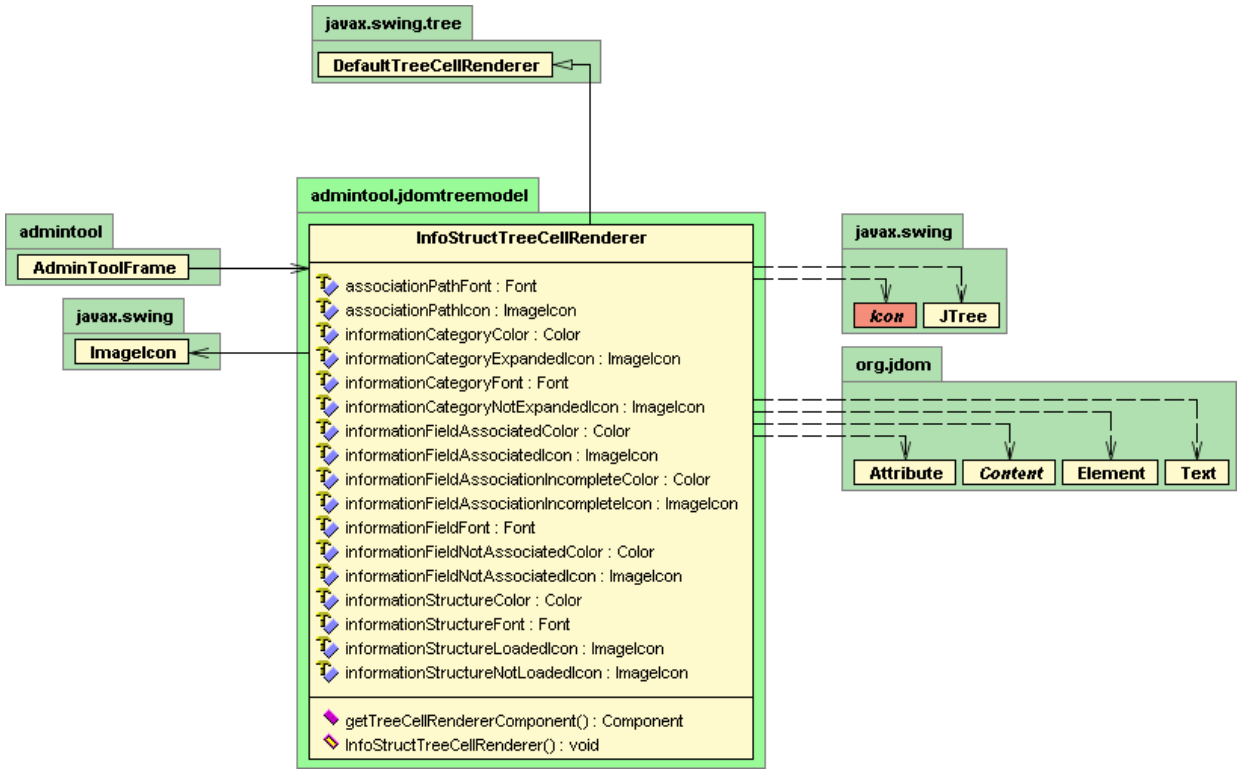
Το παρακάτω διάγραμμα απεικονίζει της κλάσεις που υποστηρίζουν την λειτουργικότητα του frame της εφαρμογής.



Το frame για να απεικονίσει της XML δομές, που περιγράφτηκαν σε προηγούμενη παράγραφο, έχει δυο JTrees. Τα JTrees για να απεικονιστούν στην επιθυμητή μορφή χρησιμοποιούνται δύο TreeCellRenderers.

admintool.jdomtreemodel.InfoStructTreeCellRenderer

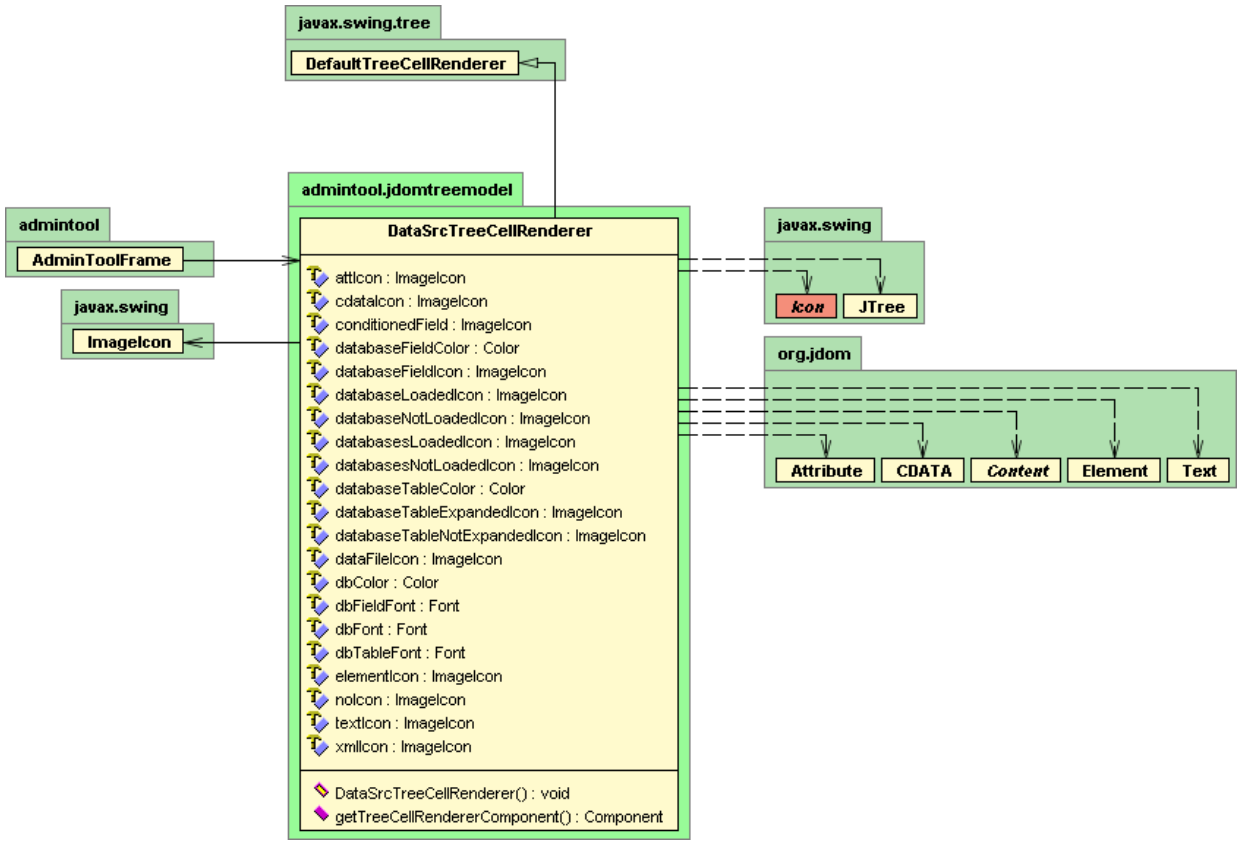
Είναι ο TreeCellRender της Information Structure δομής.



Επεκτείνει τον DefaultTreeCellRenderer για να υποστηρίξει τις απαιτήσεις που έχει η εφαρμογή για το πως φαίνεται η δομή Information Structure. Υποστηρίζει rendering των JDOM αντικειμένων που φαίνονται δεξια του διαγράμματος.

admintool.jdomtreemodel.DataSrcTreeCellRenderer

Είναι ο TreeCellRender της Data Structure δομής.



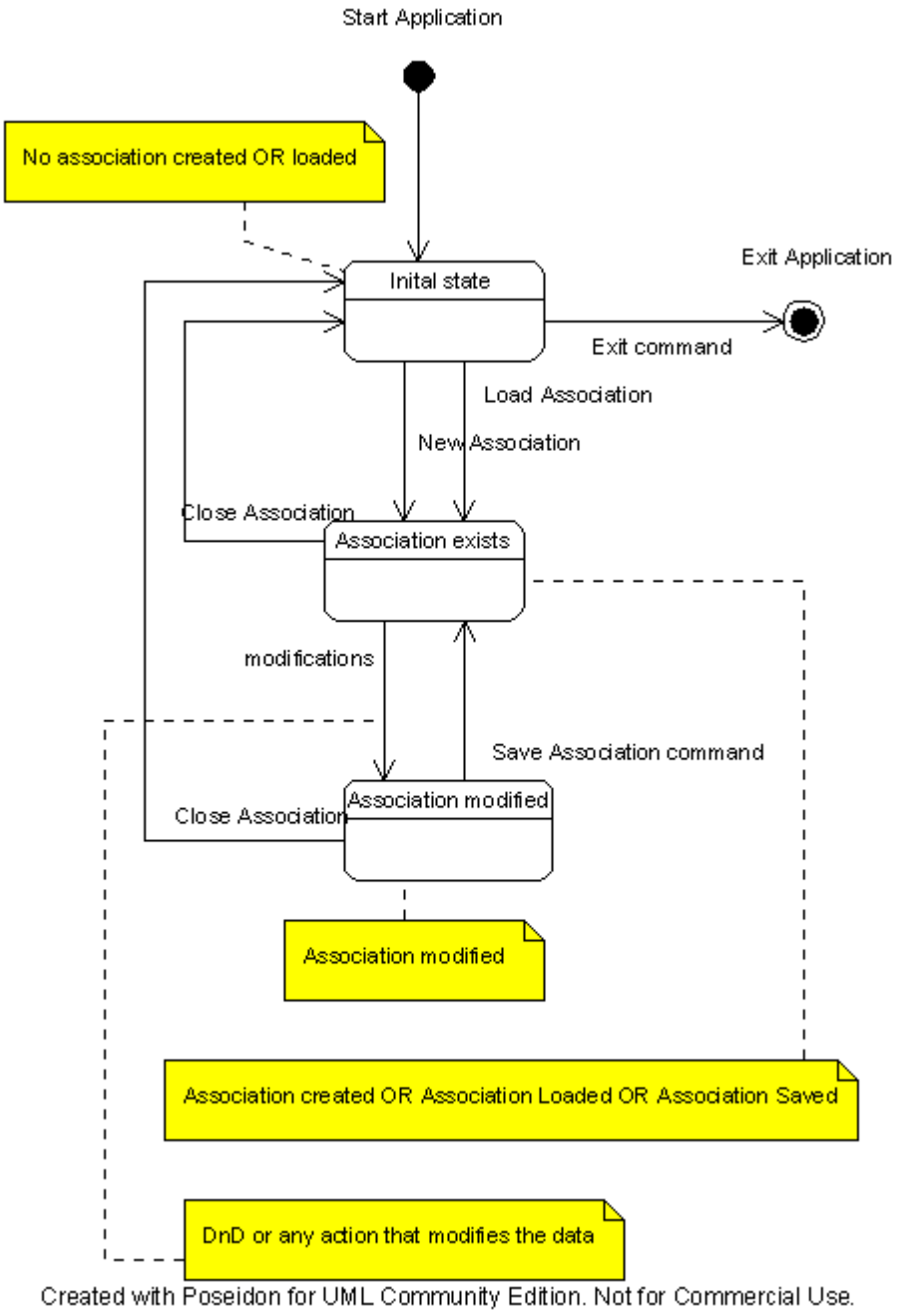
Ομοίως με τον InfoStructTreeCellRenderer.

admintool. AdmintoolFrame.GUIUPdater

Ειναι κλάση η οποία κρατάει την κατάσταση που βρίσκεται το AdminToolFrame και ανάλογα την κατάσταση ενημερώνει (ενεργοποιεί - απενεργοποιεί) τα menu και τα κουμπια του, ώστε να περιορίζονται έτσι οι επιλογές του χρήστης μόνο στις έγκυρες.
Οι καταστάσεις του GUIUpdater μπορεί να είναι τρείς:

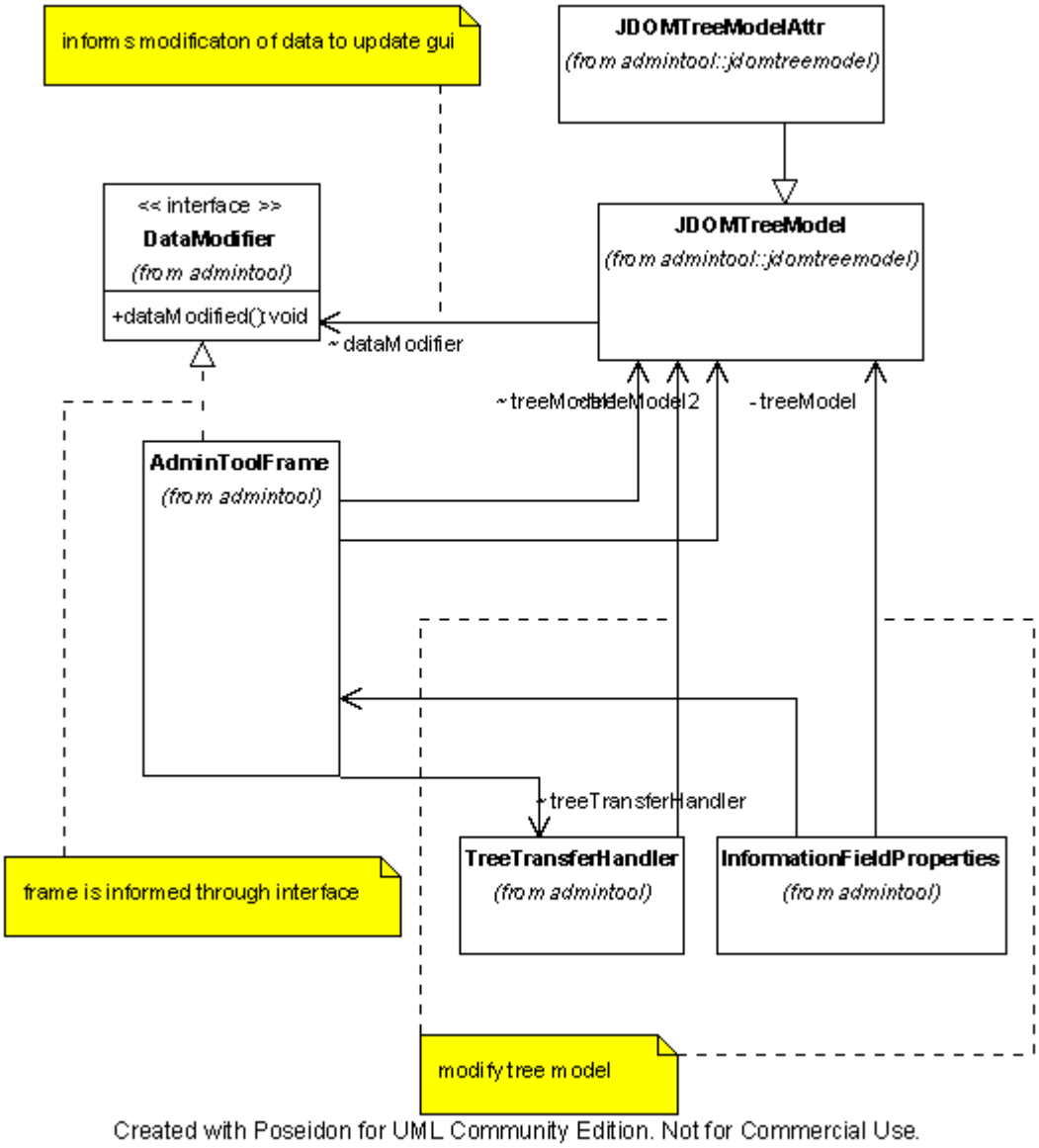
- Initial state: Να μην έχει δημιουργηθεί ή φορτωθεί κανένα αρχειο συσχετήσεων
- Association exists: Να υπάρχει κάποιο αρχειο συσχετίσεων αλλα να μην είναι τροποποιημένο. Αυτό σημαίνει ότι:
    - Έχει δημιουργηθεί αλλα δεν έχει τροποποιηθεί ακόμα
    - Έχει φορτωθεί υπάρχον αλλά δεν έχει τροποποιηθεί ακόμα
    - Είχε τροποποιηθεί αλλά τωρα οι αλλαγές είναι αποθηκευμένες
- Association modified: Να υπάρχει κάποιο αρχειο συσχετίσεων και να είναι τροποποιημένο χωρις να έχουν αποθηκευτεί οι αλλαγές

Επίσης, η εφαρμογή για να προστατεύσει το χρήστη απο απώλεια δεδομένων (ο χρήστης κλείνει το παράθυρο κατα λάθος και τα δεδομένα του χάνονται) το επιτρέπει να την τερματίσει μόνο όταν δεν υπάρχεια κάποιο αρχείο συσχετίσεων (δηλαδή ο χρήστης δεν έχει δημιουργήσει ή φορτώσει καποιο αρχείο συσχετίσεων).
Ακολουθεί το διαγραμμα καταστάσεων του GUIUpdater όπου φαίνονται οι τρείς καταστασεις και οι δυνατές μεταβάσεις μεταξύ των καταστάσεων. Σε κάθε μεταβάση αναγράφεται και η αντίστοιχη εντολη που δίνει ο χρήστης.



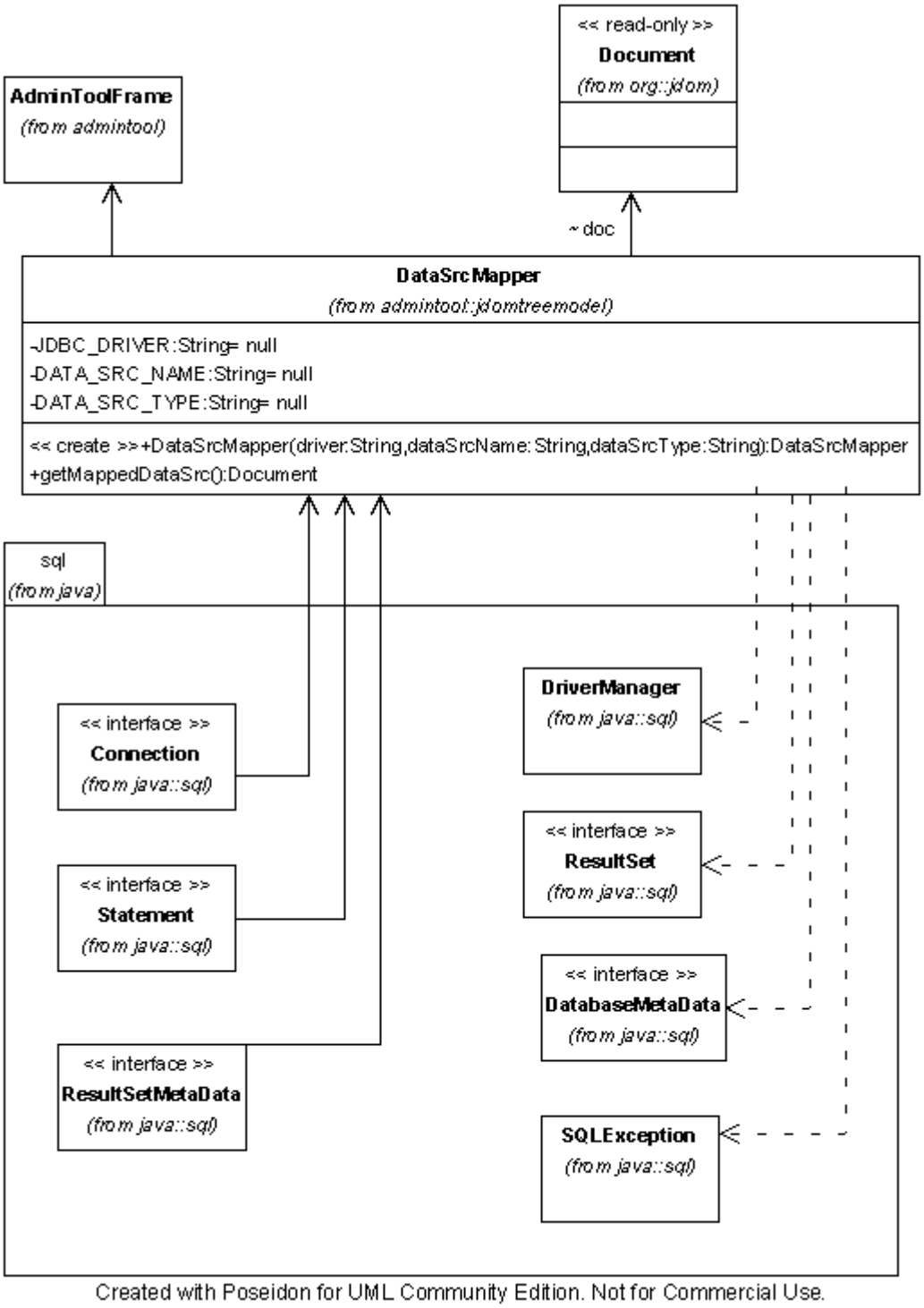
admintool.Datamodifier

Η Interface DataModifier παρέχει την δυνατότητα στο κυρίως frame της εφαρμογής (δηλαδη στο AdminToolFrame) να ενημερώνεται ότι τα δεδομένα της εφαρμογής έχουν τροποποιηθεί. Τα δεδομένα της εφαρμογής θεωρείται το Information Structure που δέχεται τους συσχετισμούς.



Το AdminToolFrame μπορεί να καλέσει μεθόδους του JDOMTreeModel και να αλλάξει τα δεδομένα, δηλαδή την XML δομή του Information Structure που αυτό περιέχει. Για αυτή την περίπτωση δεν είναι ακομα απαραίτητη η DataModifier.
Όμως, το frame έχει και όλα τα αλλα παράθυρα διαλόγου όπως το InformationFieldPropertes καθώς και την λειτουργικότητα του DnD που υποστηρίζεται μέσω της TreeTransferHandler. Όλα αυτα καλούν απευθείας μεθόδους και τροποποιούν XML δομή. Το πρόβλημα αυτό θα μπορούσε να λυθεί και με την κλήση μιας μεθόδου στο AdminToolFrame κάθε φορα που καποια κλάση τροποποιούσε την δεδομένα. Αυτή η λύση δεν προτιμήθηκε γιατι είχε κακή επεκτασιμότητα και θα οδηγούσε σίγουρα σε λάθη.
Η λύση που προτιμήθηκε να περαστεί μια αναφορα DataModifier στο JDOMTreeModel έτσι όποτε γινόταν μια τροποποίηση, η JDOMTreeModel θα αναλάμβανε να ενημέρωσει το frame. Το AdminToolFrame υλοποιεί την interface DataModifier ώστε να περιορίζει την πρόσβαση στο JDOMTreeModel σε μόνο μια μέθοδο. Ότι ισχύει για το JDOMTreeModel ισχύει και για την subclass της JDOMTreeModelAttr.

admintool.jdomtreemodel.DataSrcMapper
Είναι η κλάση η οποία αντιστοιχεί την δομή μιας πηγής δεδομένων σε μια Data Structure δομή.

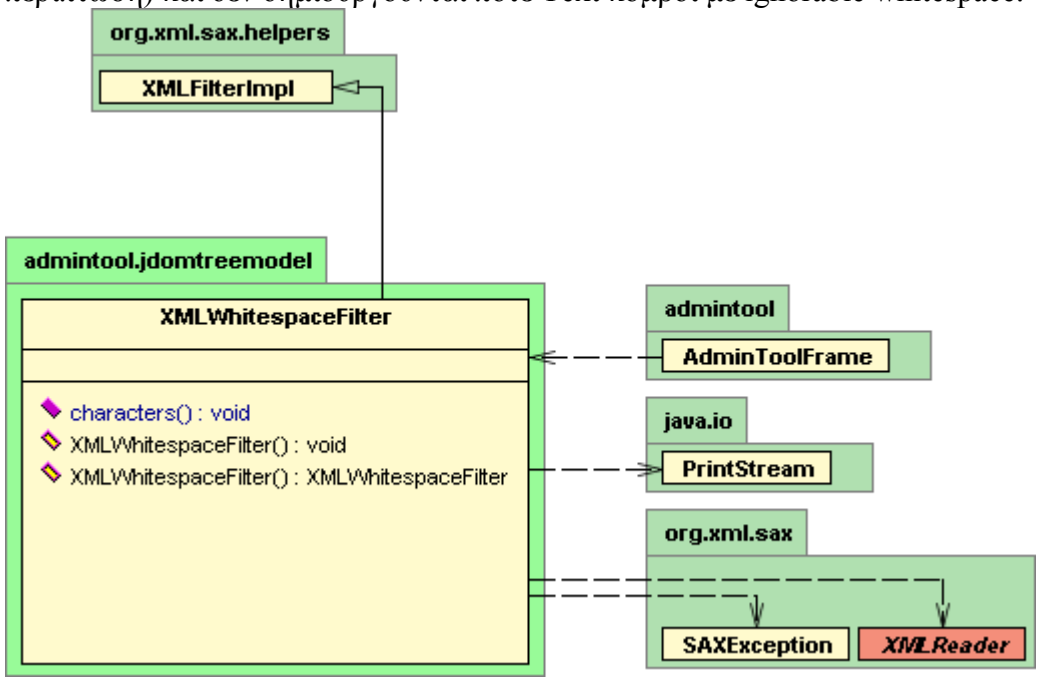


Δέχεται τις παραμέτρους της πηγής δεδομένων, διαβάζει την πηγή και δημιουργεί ένα JDOM κείμενο που αντιστοιχεί σε αυτή την δομή.

admintool.jdomtreemodel.XMLWhitespaceFilter

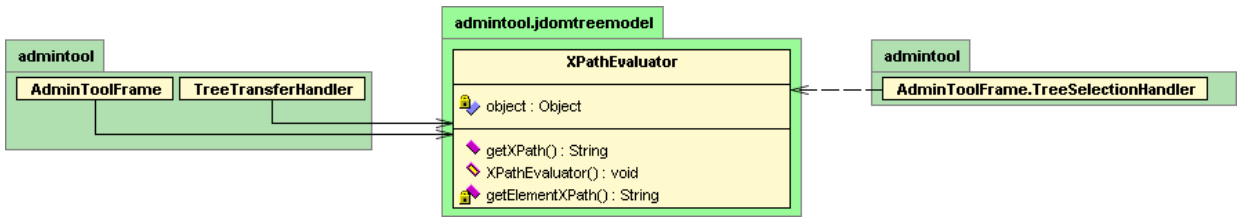
Δίνει την δυνατότητα να φιλτραριστούν τα ignorable whitespace όταν φορτώνεται ένα XML αρχείο με τον SAX builder χωρίς αυτό να έχει το αντιστοιχό του DTD ή XML Schema. Με τον

τρόπο αυτό το φιλτράρισμα γίνεται απευθείας στον parser (SAX builder στην προκειμένη περίπτωση) και δεν δημιουργούνται ποτέ Text κομβοι με ignorable whitespace.



admintool.jdomtreemodel.XPathEvaluator

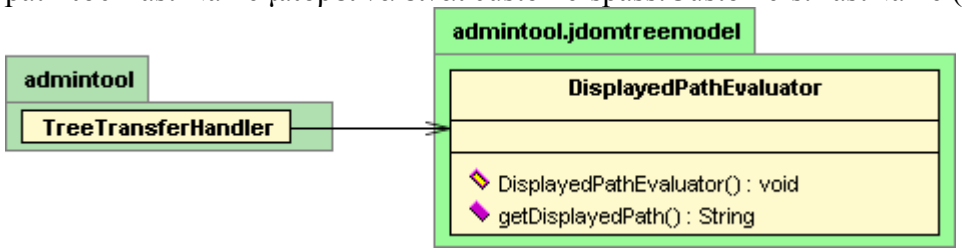
Η κλάση υπολογίζει το XPath ενός κόμβου σε ένα XML κείμενο.



Απο το παραπάνω διάγραμμα φαίνονται οι κλάσεις που την χρησιμοπιούν.

admintool.jdomtreemodel.DisplayedPathEvaluator

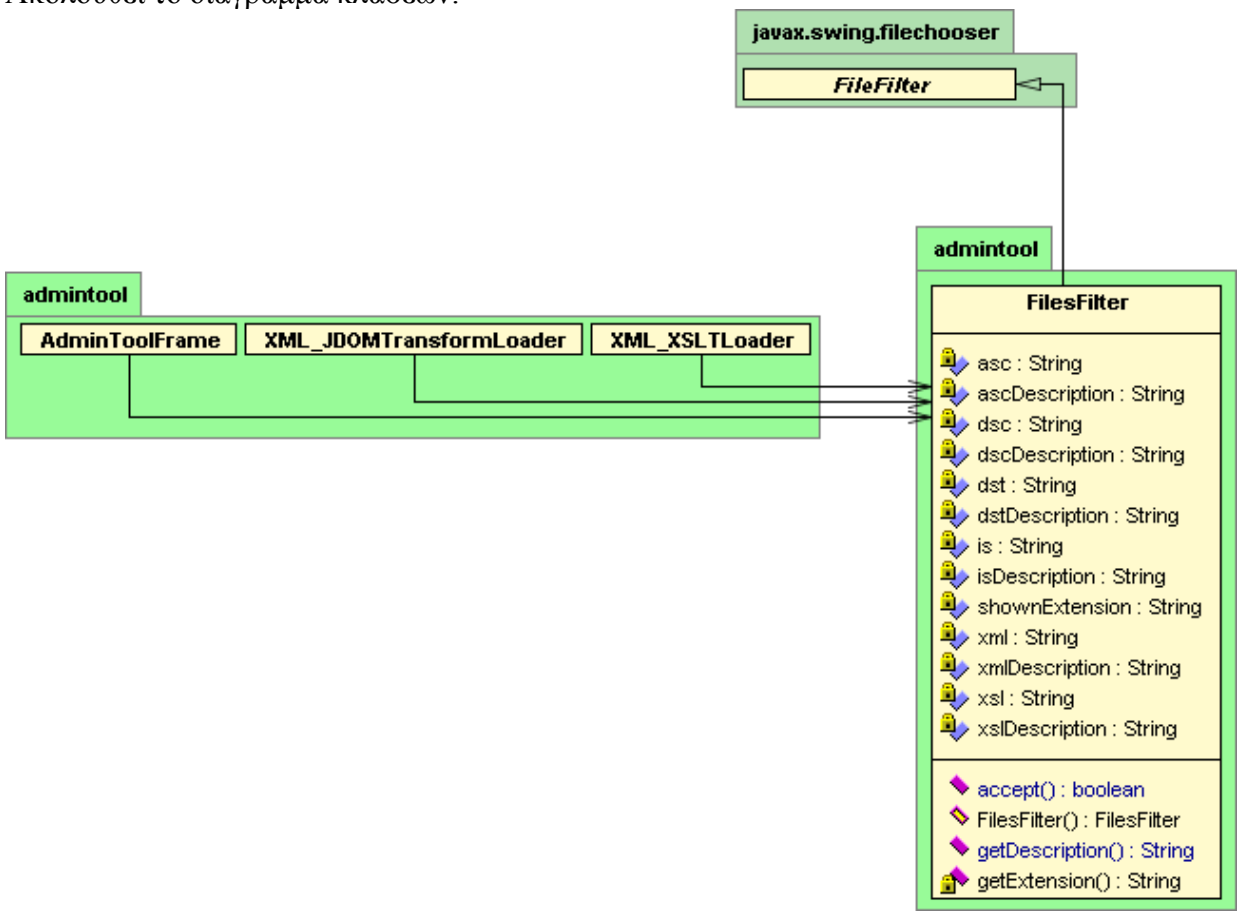
Η κλάση υπολογίζει το path ενός πεδίου στην δομή μιας πηγής δεδομένων. Για παράδειγμα το path του Last Name μπορεί να είναι customerspass:Customers:LastName (Type: VARCHAR).



admintool.FilesFilter

Η εφαρμογή πρέπει να εμποδίζει απο την μια το χρήστη να φορτώσει ένα αρχείο που δεν πρέπει, για παράδειγμα να μην μπορεί ο χρήστης να επιλέξει ένα Information Structure (*.is)

αρχείο όταν του ζητηθεί να επιλέξει ένα Data Source Configurtion (*.dsc) αρχειο. Αυτό το επιτυγχάνει με το να μην του δείχνει αρχεία που δεν μπορεί να φορτώσει στην προκειμένη στιγμή. Επίσης, με το κρύψιμο των ακατάλληλων αρχείων ο χρήστης επιλέγει πιο εύκολα χωρίς να μπερδεύεται απο άχρηστα για αυτόν αρχεία.
Η επιλογή των αρχείων γίνεται μέσω της κλάσης JFileChooser. Η κλάση admintool.FilesFilter παρέχει υποστήριξη για το φιλτράρισμα των αρχείων που φαίνονται στον JFileChooser.
Η κλάση υποστηρίζει φιλτράρισμα για όλα τα αρχεία που μπορεί να ανοίξει η εφαρμογή. Κάθε φορα που χρειάζεται να γίνει φιλτράρισμα, δημιουργείται μια instance της για το αντίστοιχο τύπο αρχείο που θέλουμε να φαίνεται στο JFileChooser. Οι υπόλοιποι τύποι αρχείων δεν θα εμφανίζονται.
Ακολουθεί το διάγραμμα κλάσεων:

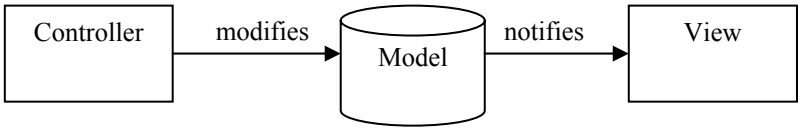


Όπως φαίνεται απο το παραπάνω διάγραμμα η κλάση επεκτείνει την FileFilter ώστε να μπορεί να τοποθετεί στον JFileChooser σαν ένας File filter. Εππίσης, στο αριστερό μερος φαίνονται όλες οι κλάσει χρησιμοποιείται.

Υλοποιώντας την Tree Model Interface

Η αρχιτεκτονικη Model-View-Controller (MVC).

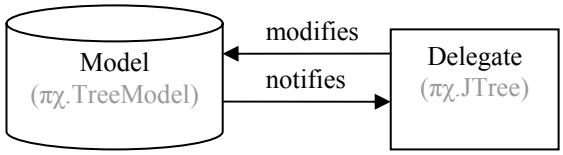
Η αρχιτεκτονικη MVC ξεχωρίζει τα δεδομένα της εφαρμογής (που περιέχονται στο model - *μοντελο*) απο τα components της γραφικής απεκόνισης (η view - *οψη*) και την λογική που επεξεργάζεται τα δεδομένα εισόδου (ο controller - *ελεγκτής*).



Ο ελεγκτής υλοποιεί την λογική που επεξεργάζεται τις εισόδους απο το χρήστη. Το μοντέλο περιέχει τα δεδομένα της εφαρμογής και η όψη δημιουργεί μια αναπαράσταση των δεδομένων που είναι αποθηκευμένα στο μοντέλο. Όταν ο χρήστης παρέχει κάποια είσοδο, ο ελεγκτής αλλάζει το μοντέλο με την είσοδο του χρήστη. Πρεπει να επισημανθεί ότι το μοντέλο περιέχει μόνο τα δεδομένα της εφαρμογής και τιποτα άλλο. Όταν αλλάζει το μοντέλο, ειδοποιεί την όψη για την αλλαγή, έτσι ώστε να μπορέσει η όψη να αλλάξει την αναπαράσταση τις αλλαγές που έγιναν στα δεδομένα. Η αρχιτεκτονικη MVC δεν περιορίζει την εφαρμογή σε μία μόνο όψη ή ελεγκτη. Μια εφαρμογή μπορεί να έχει περισσοτερες απο μια όψεις του μοντέλου της ή περισσότερους απο ενα ελεγκτές.

Η αρχιτεκτονική Delegate-model. Η υλοποίηση της MVC στο Java Swing.

Στο Swing τα components υλοποιούν μια παραλλαγή του MVC που συνδυάζει την όψη και τον ελεγκτή σε ένα αντικείμενο, που ονομάζεται delegate. Το delegate παρέχει και μαι γραφική απεκόνιση του μοντέλου και μια διαπροσωπια (interface) για την τροποποίηση του μοντέλου.



Για παράδειγμα κάθε JTree έχει ένα σχετικό Tree Model για το οποίο το JTree είναι delegate. Το JTree είναι ένα απο τα πιο πολύπλοκα Swing components που υλοποιεί την delegate-model αρχιτεκτονική. Το Tree Model διατηρεί την κατάσταση του αντικειμένου, όπως την δομή του δέντρου, καθώς και μια λίστα απο ActionListeners. Το JTree βαζει τους listeners που έχει στην λίστα αυτή έτσι όταν το Tree Model αλλάξει να πυροδοτήσει κατάλληλα events και να τα περάσει στους listeners. Το JTree που έχει τους listeners μόλις παρει ενα events, ενημερώνεται μεσα απο κατάλληλη Interface που του παρέχει το Tree Model. Ετσι καταφέρνει να ενημερώσει το Tree Model την αλλαγή που έγινε σε αυτό. Απο την άλλη το JTree παρέχει την γραφική απεκόνιση του δένδρου και μπορεί να αλλάξει το Tree Model. Ο όρος Tree Model χρησιμοποιήθηκε για να αποδόσει την κλάση στην οποία διατηρείται η δενδρική δομή, ή δενδρικό μοντέλο που θελουμε να περιγράψουμε με το JTree και δεν πρέπει να συγχέεται με την Interface javax.swing.tree.TreeModel.

Η υλοποίηση της αρχιτεκτονικής delegate-model στο JTree του Swing.

Η Java παρέχει την Interface **javax.swing.tree.TreeModel** η οποια ορίζει της μεθόδους που είναι απαραίτητες για την αναπαράσταση μιας δενδρικής δομής δεδομενων στο JTree. Η TreeModel δεν αποτελεί το μοντέλο, όπως μπορεί εύκολα να παρερμηνευτεί. Στην περίπτωση μας το *μοντέλο* είναι η δενδρική δομή δεδομένων που θέλουμε να απεκονίσουμε με το JTree. Η TreeModel καθορίζει τις μεθόδους που θα καλέσει το JTree για να λειτουργήσει, έτσι ώστε ο προγραμματιστής να υλοποιήσει κατάλληλα τις μεθόδους αυτές ώστε το JTree να απεκονίζει

σωστά το μοντέλο δεδομένων του. Δηλαδή η TreeModel εξασφαλίζει την διαεπάφή της επικοινωνίας μεταξύ μοντέλου και JTree.

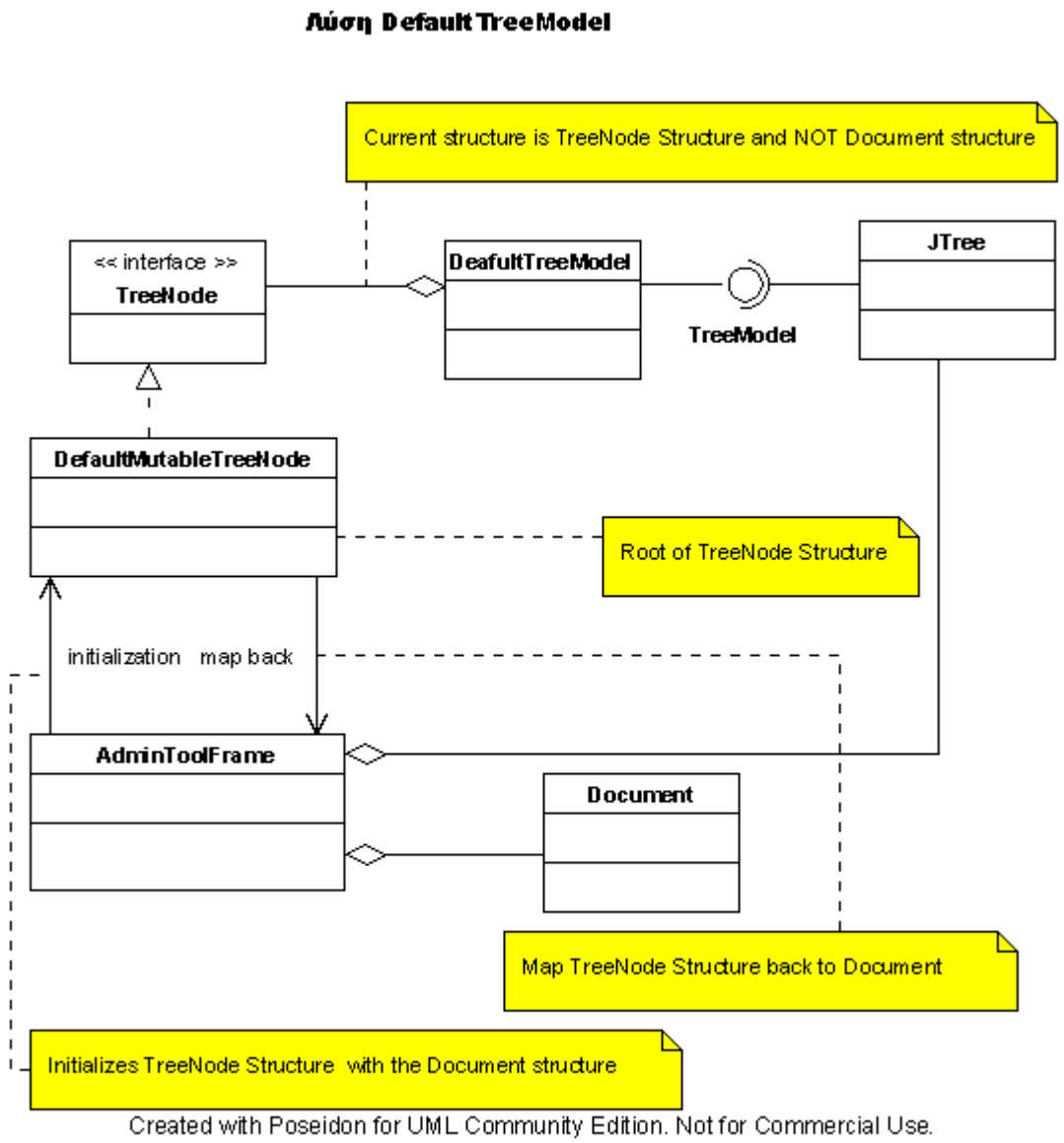
Η Java παρέχει και μια ετοιμη υλοποίηση της TreeModel, την κλάση **javax.swing.tree.DefaultTreeModel**, η οποία όμως μπορεί να αναπαραστίσει δενδρική δομή που αποτελείται απο αντικείμενα που υλοποιούν την Interface TreeNode. Αυτο δεν δημιουργεί πρόβλημα γιατί η Java παρέχει ετοιμη υλοποίηση και της κλαση που υλοποιεί την TreeNode, την DefaultMutableTreeNode. Συνεπώς, ο προγραμματιστής μπορεί δηλαδή να παρει την δενδρική δομή που θέλει να αντιστοιχίσει, δηλαδή το μοντέλο του, και να δημιουργήσει (map) μια αντίστοιχη δομή απο TreeNodes (TreeNode Structure). Με το TreeNode Structure θα αρχικοποιήσει το JTree. Δυστυχώς όμως οι αλλάγές του delegate (JTree) θα εφαρμόζονται στο TreeNode Structure και όχι το πραγματικό μοντέλο. Για εφαρμόσει τις αλλαγές στο μοντέλο του πρέπει να αντιστοιχίσει (map back) απο την αρχή την δομή του TreeNode Structure στο μεντέλο του.

Η αμέσως επόμενη ενναλακτικη λύση είναι η *υλοποίηση* της TreeModel interface, η οποία όμως μπορεί να αναπαραστίσει δενδρική δομή οποιασδήποτε κλάσης. Η λύση αυτή έχει το πλεονέκτημα ότι η δενδρική δομή που απεκονίζεται είναι του ίδιο του μοντέλου και όχι του TreeNode Structure. Συνεπώς δεν χρειάζονται να γίνουν χρονοβορες και προσφορες σε λάθη (error-prone) αντιστοιχίσεις (maps) και το κυριότερο αυτο που βλέπει ο χρήστης είναι η δομή που έχει και όχι μια αντιστοίχηση της. Τέλος,

Η υλοποίηση που υιοθετήθηκε στο Administration Tool

Στην περίπτωση του Administration Tool το μοντέλο, δηλαδή η δενδρική δομη που θέλουμε να αναπαραστίσουμε, είναι ένα XML κείμενο.
Στην προηγούμενη παράγραφο είδαμε τις δυο λύσεις που υπάρχουν. Εκ των προτέρων να αναφερθεί ότι η λύση που υιοθετήθηκε ήταν να υλοποιηθεί η interface TreeModel. Αναλύοντας της δυο λύσεις τεμηριώνεται και λόγος της επιλογής αυτής.

Λύση του DefaultTreeModel.
Το διάγραμμα κλάσης της λύσης είναι:



Τα πλεονεκτήματα της λύσης:

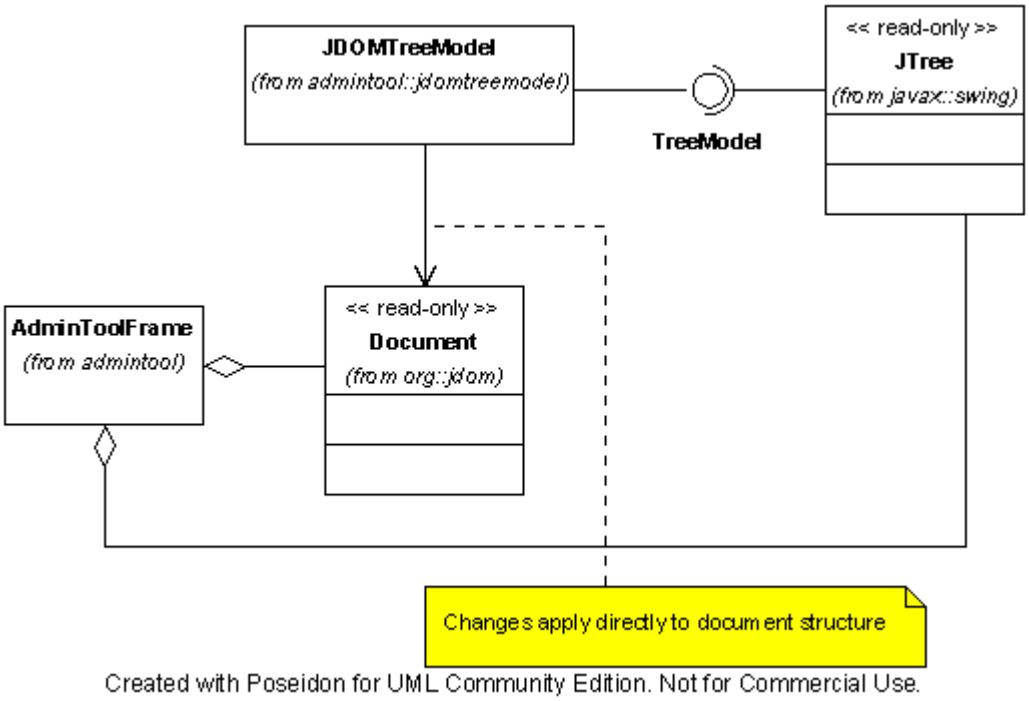
- Δεν χρειάζεται να υλοποιηθεί η interface TreeModel
- Παρέχει έτοιμη υποστήριξη για πρόσθηκη και αφαίρεση κόμβων στο δενδρο.

Τα μειονεκτήματα της λύσης:

- Η δομή που αναπαριστάται στο JTree δεν είναι η δομή του XML αλλά η δομή των TreeNodes. Και αυτο γιατί το DeafultTreeNode διαχειρίζεται μόνο TreeNode κομβους.
- Χρειάζεται να γίνεται αντιστοιχιση του XML με την δομή των TreeNodes στην αρχικοποίηση (map) του δέντρου και κάθε φορα που θα θέλουμε να δούμε το αποτέλεσμα των τροποποιησεων στη πραγματική δομη του XML. Η αντιστοίχηση συνεπάγεται καθυστέρηση.

Λύση της υλοποίησης της TreeModel.

Το διάγραμμα κλάσης της λύσης είναι:



Τα μειονεκτήματα της λύσης:

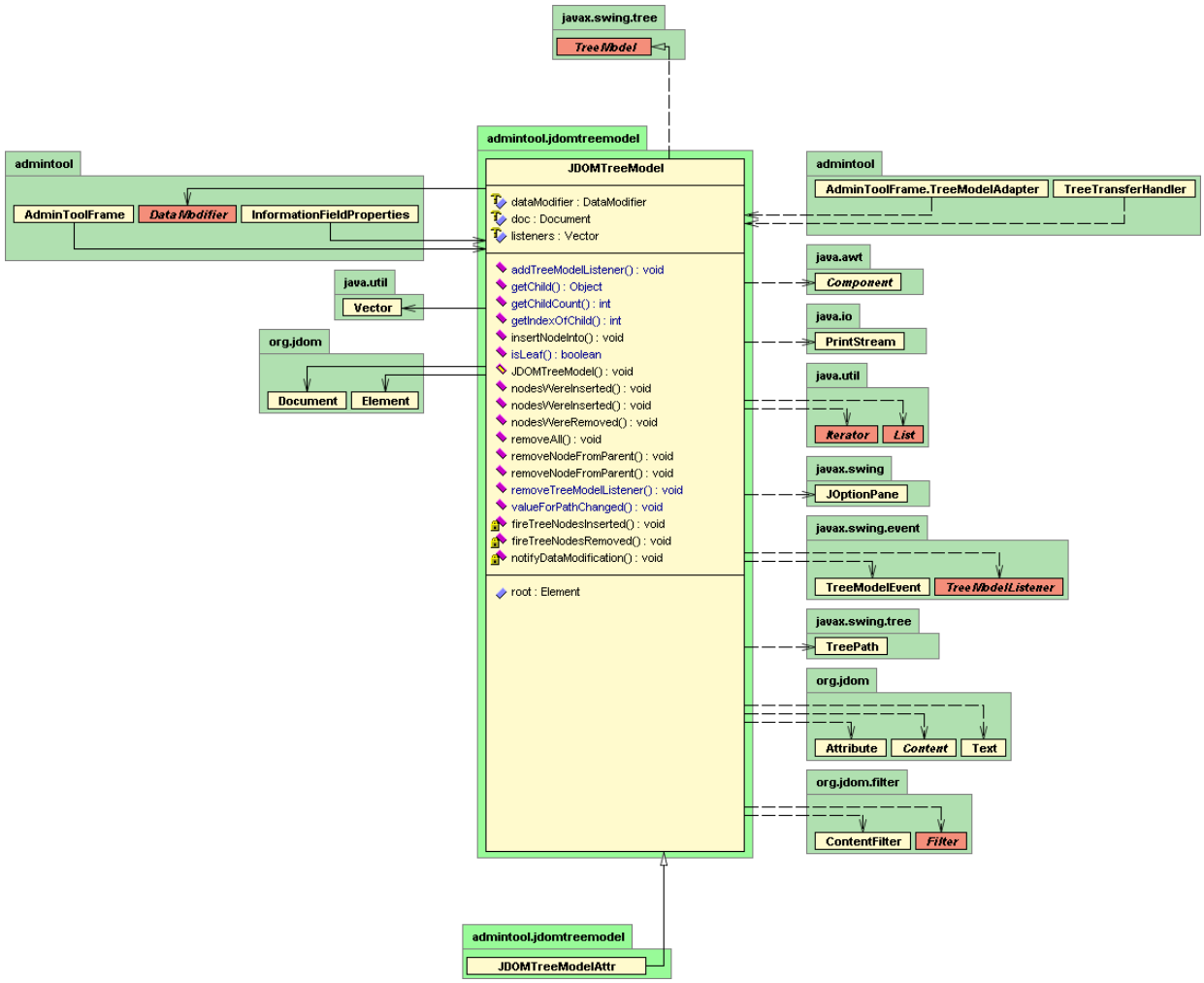
- Χρειάζεται να υλοποιηθεί η interface TreeModel, η οποία είναι πολύπλοκη
- Πρέπει να υλοποιηθει υποστήριξη για πρόσθηκη και αφαίρεση κόμβων στο δενδρο η οποία είναι έτοιμη στη άλλη λύση. Εδώ να επισημανθεί ότι η προσθήκη προσθεση και αφαίρεσης δεν αποτελεί μέρος της TreeModel interace. Παρόλα αυτά είναι πιο δυσκολη στην να υλοποιηση της απο την υλοποιηση της ίδιας της interface.

Τα πλεονεκτήματα της λύσης:

- Η δομή που αναπαριστάται στο JTree είναι ακριβώς η δομή του XML.
- Δεν χρειάζεται να γίνεται οποιαδήποτε αντιστοίχηση του XML με κάποια άλλη δομή.
- Μια επιλογή σε ένα κόμβο του JTree επιστρέφει απευθείας ένα κόμβο του XML κειμένου, πχ. ένα Element και όχι κάποιο άλλη wrapper class, πχ TreeNode κλάση.
- Καλύτερη επεκτασιμότητα.

admintool.treemodel.JDOMTreeModel

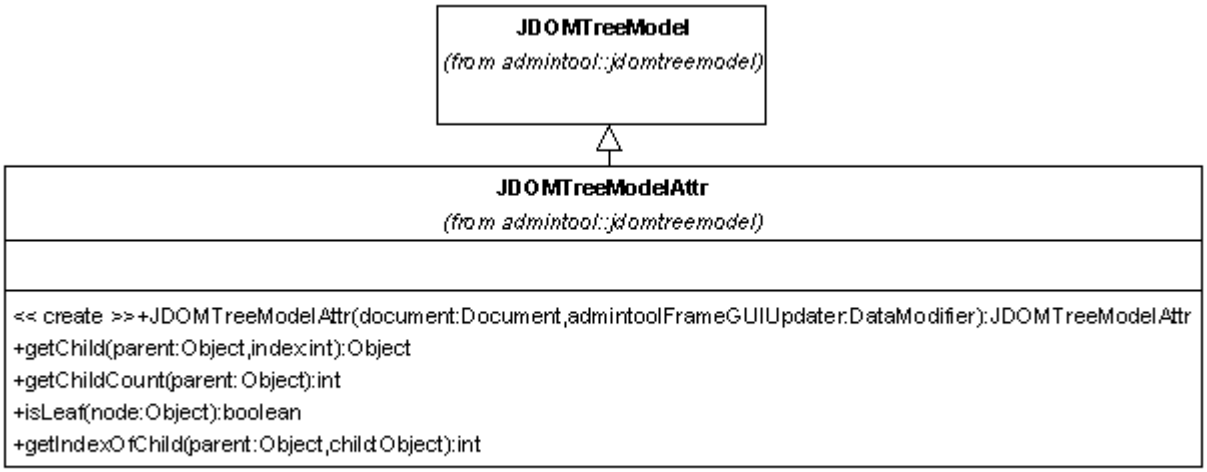
Η Κλάση admintool.treemodel.JDOMTreeModel υλοποιεί την interface TreeModel και παρέχει την υποστήριξη για την προσθαφαιρεση κόμβων στο JTree της εφαρμογής.



Υλοποίηση της TreeModel:
Οι μέθοδοι: `getRoot`, getChild, getChildCount, getIndexOfChild, isLeaf, ειναι οι συναρτήσεις της TreeModel τις οποίες καλει το JTree ώστε να «μάθει» την δομή που πρεπει να απεκονίσει. Η υλοποίηση τους βασίστηκε στις συναρτήσεις που παρεχει το JDOM API αφού όπως αναφέρθηκε με την TreeModel αναπαριστάται το ίδιο το μοντελο.
Η υποστήριξη για εισαγωγη κόμβων υλοποιείται απο την insetNodeInto ενω η αφαίρεση απο την removeNodeFromParent η οποία είναι overloaded για να υποστηρίζει και αφαίρεση Text απο Element και Element απο Element. Ο τρόπος που υλοποιήθηκαν αυτές οι μέθοδοι ακολουθεί το τρόπο υλοποίηση των αντίστοιχων μεθόδων στο DefaultTreeModel. Συνεπώς, για να εσαχθεί ένας κόμβο, καλείται η insertNodeInto, η οποία κανει εισαγωγη του κόμβου στο JDOM Document στο οποίο είναι φορτωμένο το XML αρχείο. Στην συνέχεια η μέθοδος καλεί την nodesWereInserted η οποία αρχικοποιεί τις απαραίτητες παραμέτρους που χρειάζονται και καλει με αυτές την fireTreeNodesInserted. Η fireTreeNodesInserted πυροδοτεί TreeModelEvents δηλαδή events που δηλώνουν ότι η δομή του δέντρου έχει τροποποιηθει και το «περνάει» σε όλους τους listeners για τέτοια events έχει κλάση JDOMTreeModel. Εδω να σημειωθεί ότι με το αρχικοποιείται ένα JTree με το TreeModel του, βάζει τον TreeModelHandler του στην λίστα με τους listeners του TreeModel του, ώστε να ειδοποιείται απο το TreeModel όταν γίνει κάποια αλλαγή στην δομή του δένδρου. Οταν ειδοποιηθεί, το JTree καλεί τις μεθόδους `getRoot`, getChild, getChildCount, getIndexOfChild, isLeaf για να διορθώσει την εμαφανιση του. Αναλυτικά τι κάνει η κάθε μέθοδος περιγράφεται στο API της εφαρμογής.
Αναφορές:

admintool.treemodel.JDOMTreeModelAttr

Η κλάση JDOMTreeModelAttr επεκτείνει την JDOMTreeModel ώστε να παρέχει υποστήριξη για την απεκόνιση των Attributes σαν φύλλα του JTree.
Στο παρακάτω διαγραμμα φαίνονται οι συναρτήσεις που χρειάζεται να γίνουν override.



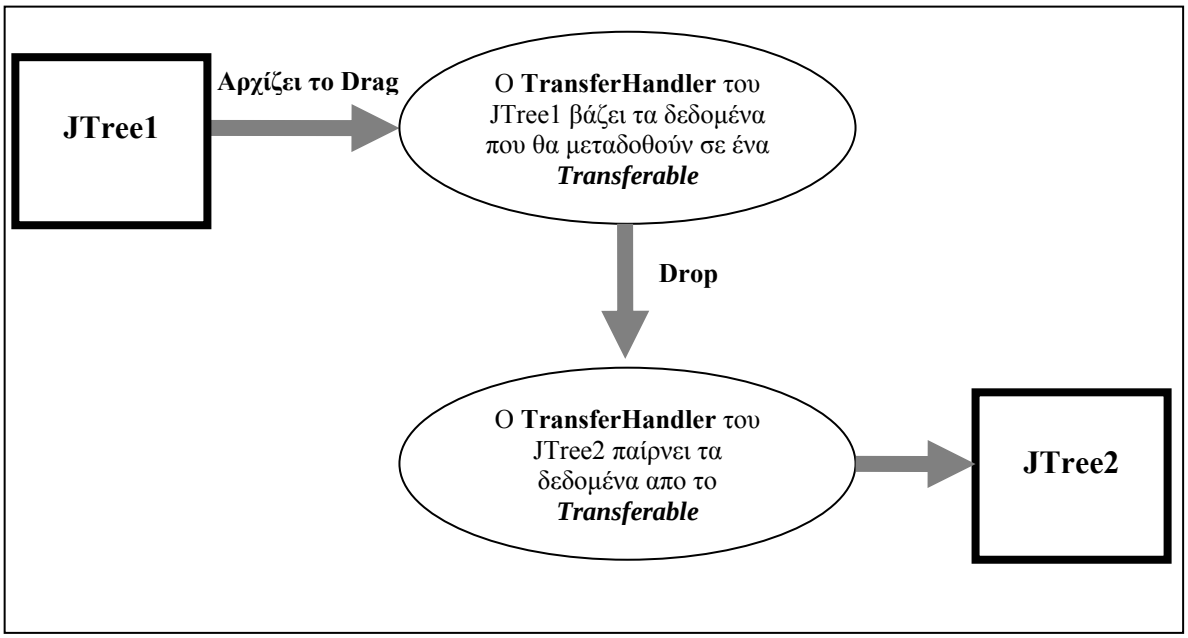
Drag and Drop

Η τεχνολογία του Drag and Drop στην J2SE 1.4.

Απο την έκδοση J2SE 1.4 και μετα, υπάρχει υποστήριξη απο το Swing για μεταφορά δεδομένων (data transfer) μεταξύ java εφαρμογών. Η μεταφορά δεδομένων μπορεί να γίνεται ή μέσω του Clipboard (Cut/Copy - Paste) ή με Drag and Drop (DnD).
Η νεα υποστήριξη που υπάρχει στο J2SE 1.4 δεν δημιουργεί απο το μηδέν ένα νέο υποσύστημα για DnD, αλλά βασίζεται στο προηγούμενο υποσύστημα που υποστηρίζε το DnD, δηλαδή το υποσύστημα που χρησιμοποιούσε μέχρι και η προηγούμενη έκδοση της Java. Το προηγούμενο υποσύστημα αποτελείται απο τα πακέτα (packages) `java.awt.datatransfer` και `java.awt.dnd`. Πριν το 1.4, η υποστήριξη που παρείχε το AWT (Abstract Window Toolkit) της Java ήταν ευέλικτο αλλά πολύπλοκο στην υλοποίηση του. Τώρα, στην νέα υποστήριξη που παρέχεται απο το Swing έχει προστεθεί:

- Ενα νέο μικρό σε έκταση API με το οποίο μπορεί να υλοποιηθεί το DnD με ελάχιστες κλήσεις συναρτήσεων
- Η δυνατότητα της χρήσης του παλιού υποσυστήματος σε περίπτωση που χρειάζεται να προσαρμοστεί η υποστήριξη που παρέχεται απο το Swing

Στο Administration Tool έχει υλοποιηθεί μόνο το Drag and Drop αφού δεν θεωρήθηκε απαραίτητο να υλοποιηθεί και η μεταφορα μέσω Clipboard.
Το παρακάτω διάγραμμα απεικονίζει πως υποστηρίζει το 1.4 το DnD:



Το Swing της J2SE 1.4, παρέχει την δυνατότητα να προσθέσουμε στο JComponent στο οποίο θέλουμε να υποστηρίζει DnD ένα TransferHandler.
Η κλάση **javax.swing.TransferHandler** είναι ουσιαστικά αυτή που μεταφέρει το Transferable και συνεπώς «αυτοματοποιεί» το DnD στην έκδοση 1.4. Αναλαμβάνει αν ακούει τα events του DnD και αναλογα να τα εξυπηρετεί καλώνταστ τις κατάλληλες συνατρήσεις της.
Η Interface **java.awt.datatransfer.Transferable** καθορίζει της μεθόδους που πρέπει να υλοποιεί μια κλάση που θα μεταφέρει τα δεδομένα που θέλουμε. Δηλαδή περικλείουμε τα δεδομένα που θέλουμε σε μια κλάση ώστε να μπορέσουμε να τα μεταφέρουμε. Αυτή η κλάση για να μπορεί να μεταφερθεί απο το TransferHandler, πρέπει να παρέχει σε αυτό τις μεθόδους που προκαθορίζει η Interface Transferable, συνεπώς να υλοποιεί την Interface.
Τέλος, η κλάση **java.awt.datatransfer.DataFlavor** ορίζει το «είδος» των δεδομένων που μεταφέρονται απο το Transferable και που μπορεί να δεχτεί ο TransferHandler κατα το Drop. Συνεπώς, ο TransferHandler μπορεί να ελέγξει αν υποστηρίζει τα δεδομένα που μεταφέρει το Transferable πριν δεχτεί το drop. Έτσι όταν ο χρήστης παει να κάνει drop δεδομένα σε ένα JComponent που δεν τα υποστηρίζει εμφανίζεται το ⃠ για να τον ειδοποιήσει ότι το drop δεν μπορεί να γίνει.

Η υλοποίηση του DnD στο Administration Tool.

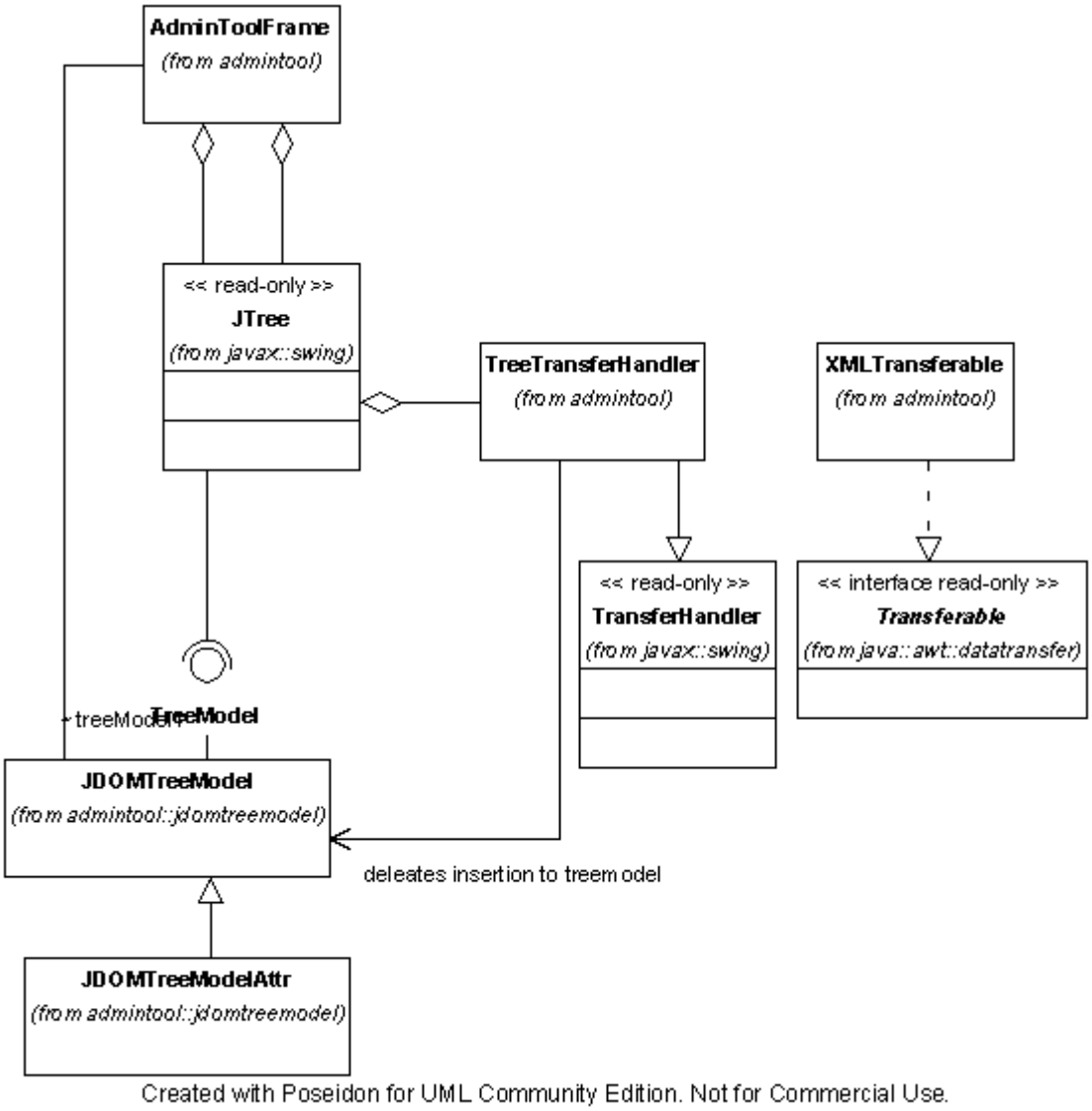
Η κλάση TransferHandler παρέχει κάποια προκαθορισμένη λειτουργικότητα για να εξυπηρετήσει την πλειονότητα των JComponent, η οποία στην περιπτωσή μας δεν μας εξυπηρετεί. Ούτως ή άλλως όσον αφορα το JTree η κλάση TransferHandler δεν παρέχει κάποια λειτουργικότητα κατα το Drop. Αυτο οφείλεται στο γεγονός ότι δεν μπορεί να είναι εκ των προτέρων γνωστο τις αποτελεσμα θα έχει ένα drop σε ένα κόμβο του δέντρου. Ενώ αντιθέτως μπορει να θεωρηθεί τι πρέπει να γίνεται όταν ένα JTextField δεχτεί ένα String drop. Ο κυριότερος λόγος όμως είναι οι απαιτήσεις που είχε η εφαρμογή για το αποτελεσμα που πρεπει να έχει ένα DnD. Συγκεκριμένα κατα το DnD πρέπει:

- Κατα το drag πρέπει ο TransferHandler να υπολογίζει τις τιμές που πρέπει να μεταφερθουν, όπως το XPath και το AssociationPath του Dragged κόμβου, και να δημιουργήσει το Transferable

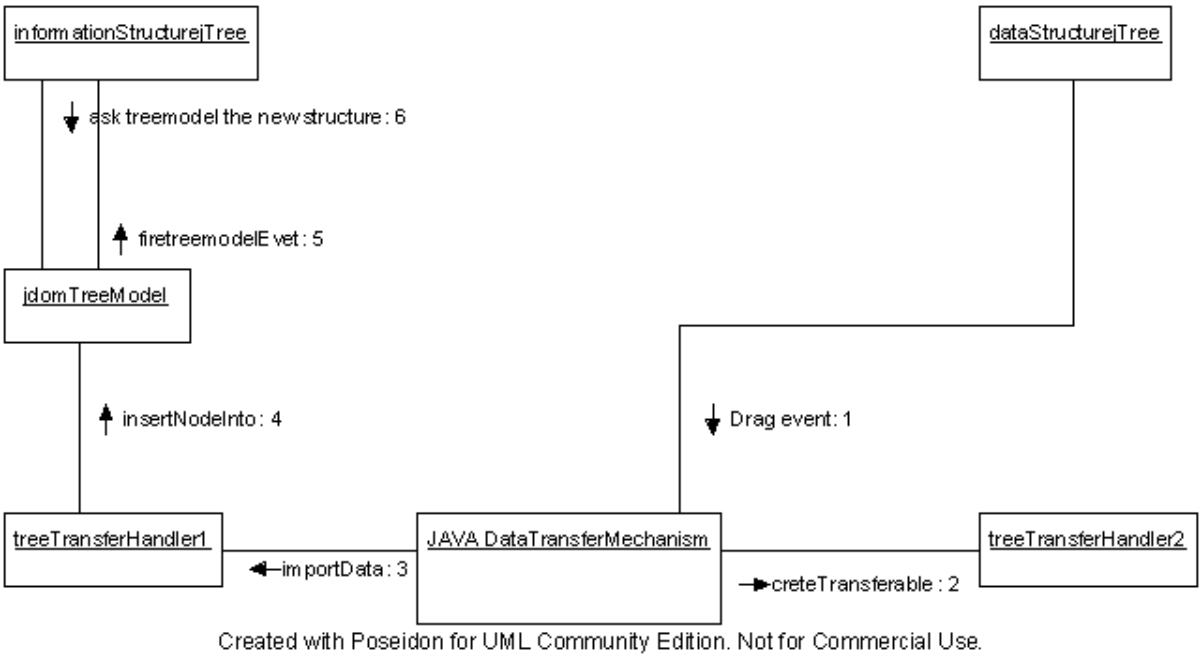
- Κατα το drop, να πάρει τις τιμες απο το Transferable, και να προσθέσει το XPath σαν Text παιδι στο dropped Element καθώς και τις υπόλοιπες τιμές. Εννοείται ότι πρώτα πρέπει να ελέγξει ότι ο dropped κόμβο είναι Element διοτι ως γνωστό στη XML μόνο τα Elements έχουν Text.

Ειναι προφανές ότι οι παραπάνω απαιτήσεις θα ήταν αδύνατον να υποστηρίζονται απο το TransferHandler του J2SE 1.4.

Ακολουθεί το διαγραμμα κλάσεων της υλοποιήσης του DnD στο Administration Tool.

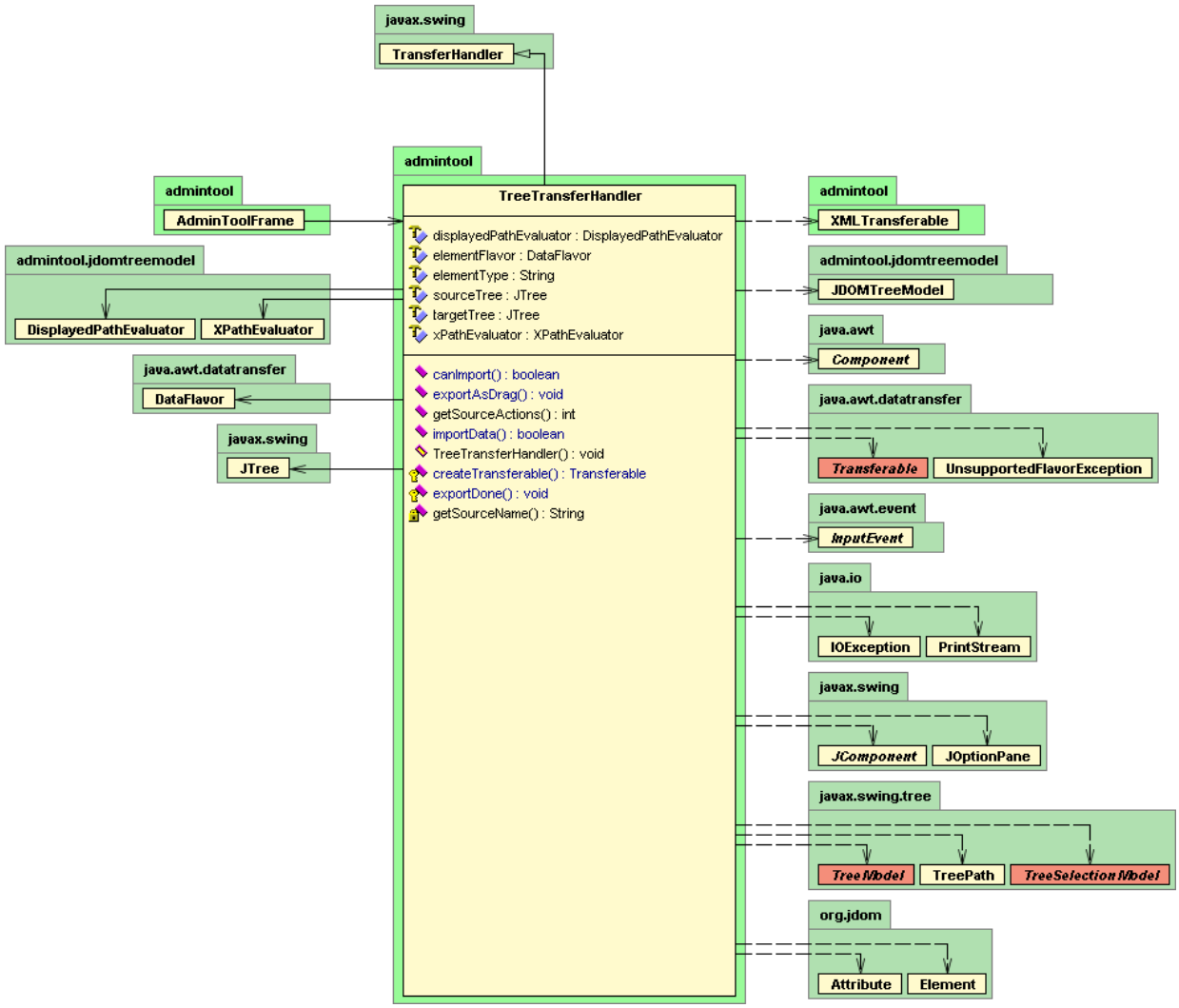


Όπου βλέπουμε πως σχετίζονται μεταξύ τους οι κλάσεις που υλοποιούν το DnD.
Ακολουθεί το διαγραμμα συνεργασίας του DnD.

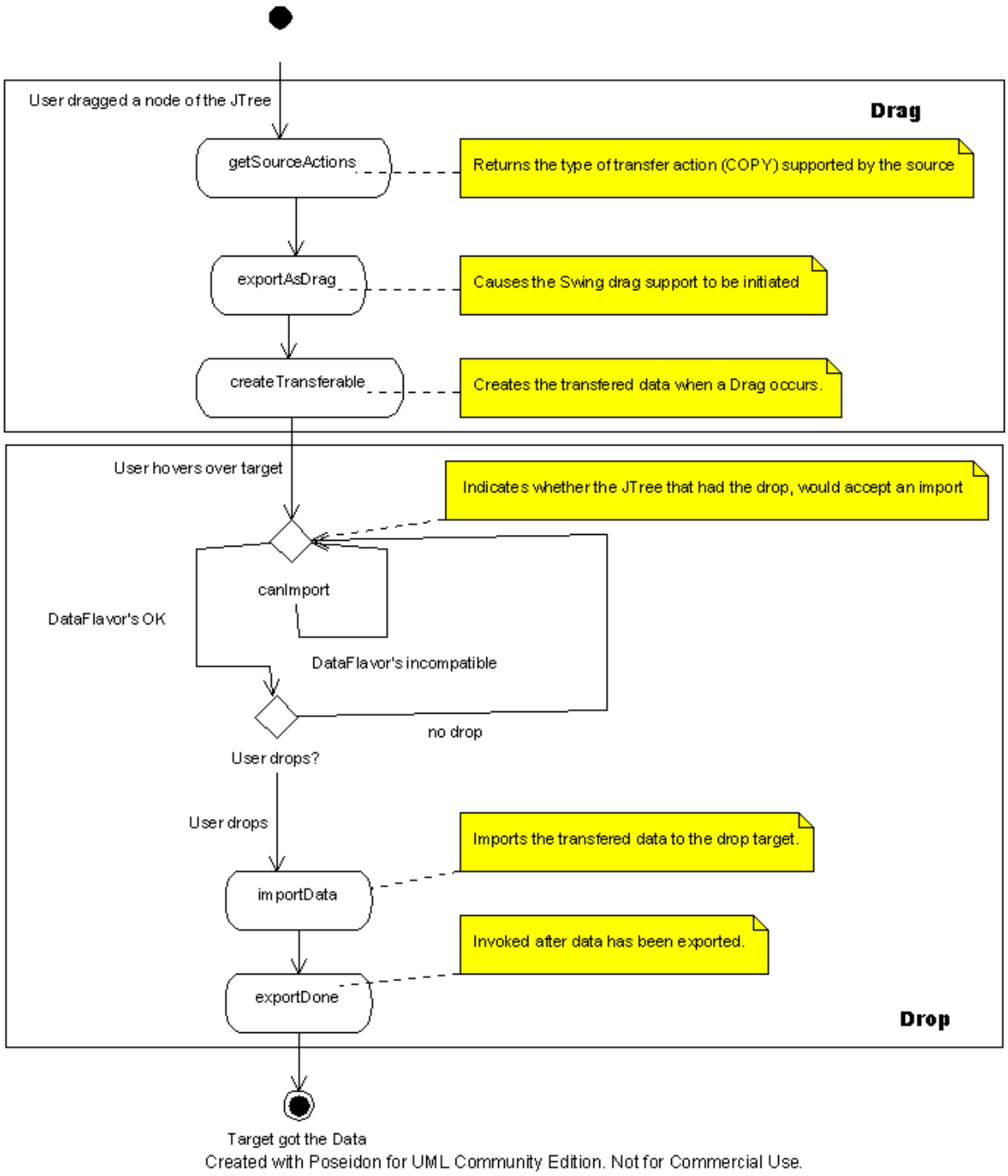


admintool.TreeTransferHandler

Η κλάση admintool.TreeTransferHandler επεκτείνει την TransferHandler και ικανοποιεί τις παραπάνω απαιτήσεις κάνοντας overriding τις απαραίτητες μεθόδους.



Αναλυτικά τι κάνει η κάθε μέθοδος περιγράφεται στο API της εφαρμογής. Γενικά αυτο που θα αναφέρθεί εδώ είναι ότι οι μέθοδοι exportAsDrag και createTransferable υλοποιούν το Drag, ενώ οι canImport, getSourceActions, importData, exportDone το drop. Ακολουθεί το σχετικό διάγγραμμα δραστηριοτήτων:



Οι μέθοδοι που υλοποιούν την βασική λειτουργικότητα είναι:

- createTransferable
- importData

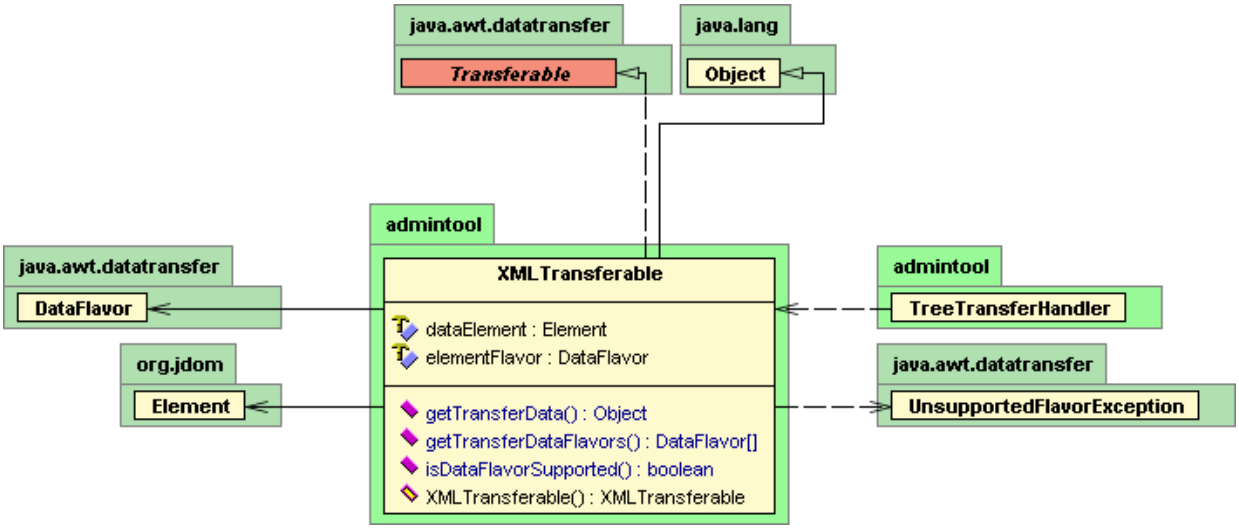
Στη μέθοδο createTransferable γίνεται έλεγχος του αντικειμένου που γινεται dragged ώστε να είναι μέσα στα πλαίσια των απιατήσεων. Όπως αναφέρθηκε όλα τα απαραίτητα δεδομένα εισάγωνται σε ένα νεο Element και με αυτο δημιουργείται ένα XMLTransferable.
Το ElementFlavor είναι ένα ένα DataFlavor για org.jdom.Elements. Αυτο μας εξασφαλίζει ότι το drop μπορει να γίνει μονο στο JTree και όχι σε κάποιο άλλο component της εφαρμογής. Παρόλα αυτά, σύμφωνα και με τις απαιτήσεις της εφαρμογής, μονο τα Information Fields μπορούν να δεχτουν συσχετισμούς. Άρα μόνο τα Information Fields μπορούν να δεχτουν drops. Συνεπως, στην μέθοδο importData γινεται περαιτέρω έλεγχος πάνω στο drop object. Αν το drop

target αντικείμενο μπορεί να δεχτεί το δεδομένα , τότε η εισαγωγή του αναθέτεται (delegate) στο tree model του δέδρου για τους λόγους που περιγράφονται στη σχετική παράγραφο

Κάθε JTree έχει το δικό του TreeTransferHandler. Ο TreeTransferHandler του Data Structure δένδρου ουσιαστικά έκανε μόνο drag, ενώ του Information Structure δένδρού μόνο drop. Παρόλα αυτά, η κλάση TreeTransferHandler υλοποιει και τα δύο ώστε η εφαρμογή να μπορει μελλοντικά να επεκταθεί ευκολότερα.

admintool.XMLTransferable

Η κλάση admintool.XMLTransferable υλοποιεί την interface Transferable ώστε να μπορεί να μεταφέρει ένα JDOM Element. Προτιμήθηκε για λόγους επεκτασιμότητας, η κλάση XMLTransferable να μεταφέρει ένα Element, παρόλο που η βασική μας αναγκη είναι να μεταφέρουμε ένα XPath, δηλαδή ένα String. Δηλαδή στο μέλλον αν χρειαζεται μπορει να κρεμαστεί ολοκληρο υποδένδρο κάτω απο το μεταφερόμενο Element και ετσι να μεταφερθεί και αυτό. Τέλος, για να γίνει δυνατή η μεταφορα των δεδομένων χρησιμοποιήθηκε και ένα DataFlavor για org.jdom.Elements.

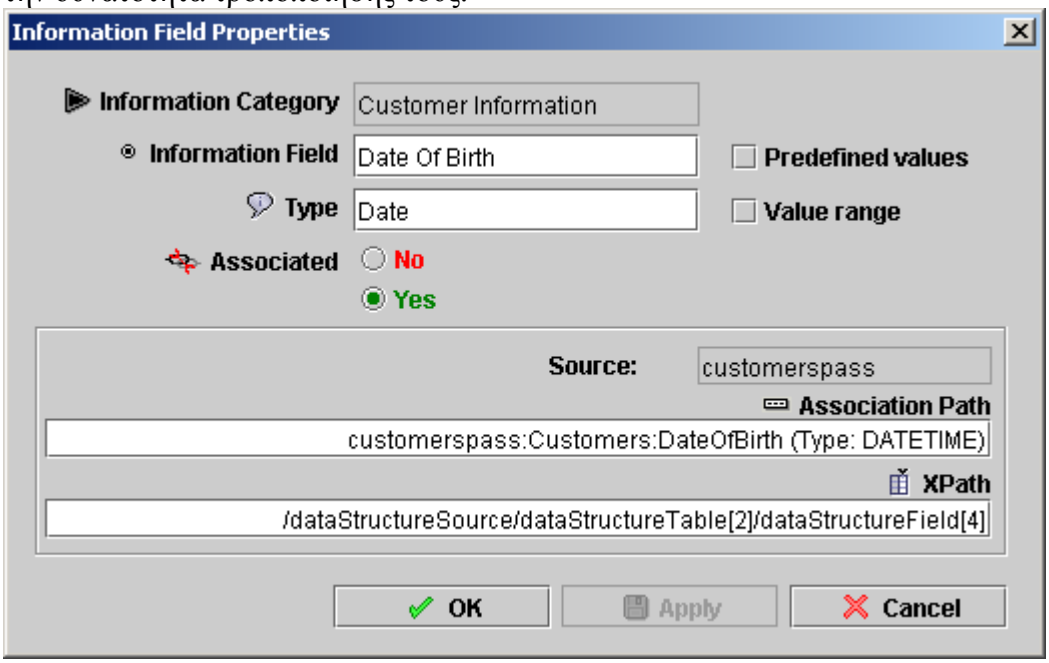


Αναφορές:

- Swing Data Transfer in Java[tm] 2 Platform, Standard Edition 1.4 (Merlin): http://access1.sun.com/tutorials/Swing_Tutorial/Dnd-Merlin-Tutorial/index.html
- Java™ 2 SDK, Standard Edition: Documentation Version 1.4.2: Drag and Drop: Swing Data Transfer
- Trail: Creating a GUI with JFC/Swing: How to Use Drag and Drop and Data Transfer

Το παράθυρο διαλόγου Ιδιοτήτων ενός Information Field

Το παράθυρο αυτό εμφανίζει όλες τις ιδιότητες ενός Infromation Field και ακόμα προσφέρει την δυνατότητα τροποποίησης τους.

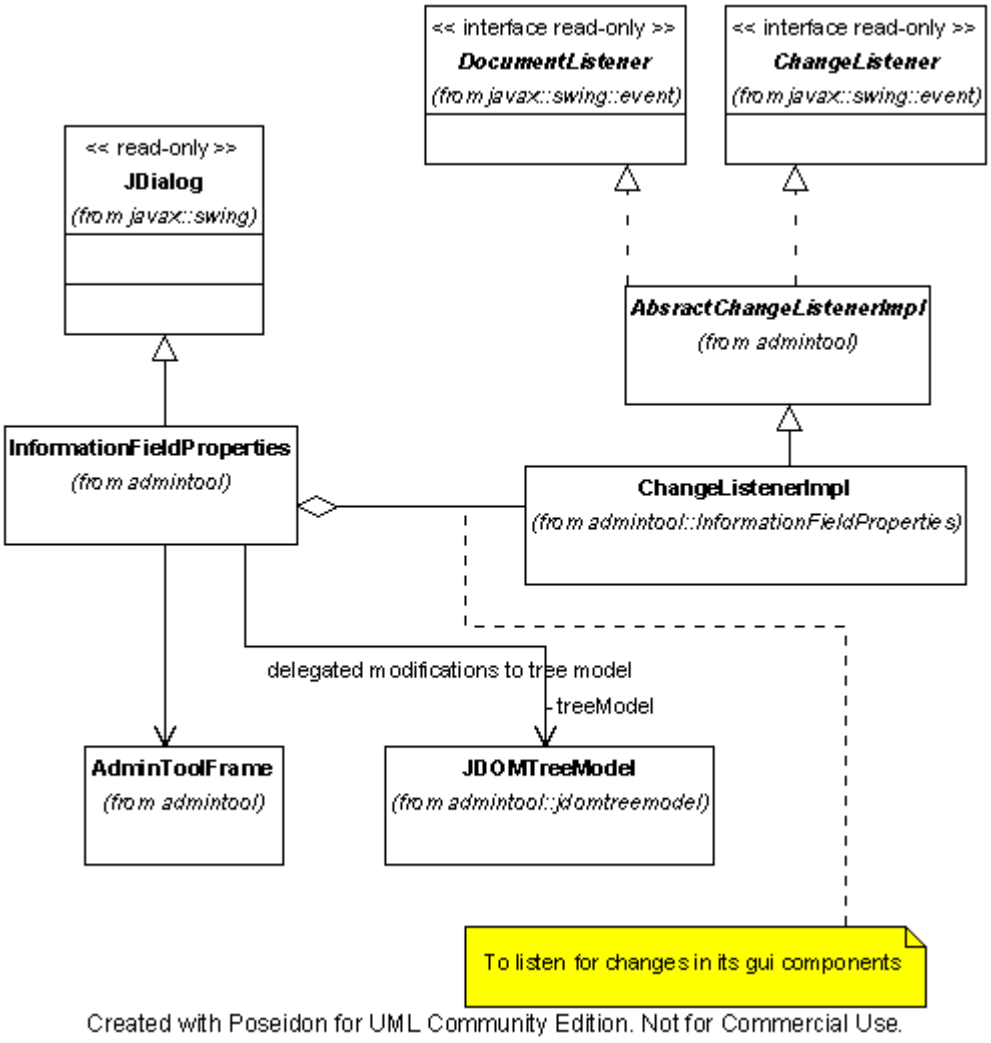


Επιλέγοντας Information Field Properties, εμαφανίζεται τπ παραπάνω παράθυρο όπου παρουσιάζονται όλες οι ιδιότητες του Information Field. Ταυτόχρονα, αυτές μπορούν να τροποποιηθούν άμεσα απο το χρήστη.

admintool.InformationFieldProperties

Ειναι η κυρίως κλάση που υλοποιεί την φόρμα .

Ακολουθεί το διαγραμμα κλάσεων.



Το InformationFieldProperties έχει υλοποιηθεί σαν ένα JDialog. Διατηρεί μια αναφορα στο tree model ώστε να μπορεί να εφαρμόσει απευθείας της τροποποιήσεις που κάνει ο χρήστης σε ένα Information Field.

Η φόρμα έχει την κλάση **ChangeListenerImpl** για να μπορεί να "ακούει" στις αλλαγές των δεδομένων που γίνονται στα πεδια των GUI component (JTextField, JCHeckbox, JRadiobox κτλ), ώστε να προτρέπει το χρήστης να αποθηκεύσει τις αλλαγές.
Η ChangeListenerImpl επεκτείνει την AbstractCHangeListerImpl υλοποιώντας την μέθοδο `changesOccured` ώστε να γίνονται οι απαραίτητες ενέργειες όταν τροποποιηθουν οι ιδιότητες του Information Field. Η κλάση αυτή δηλώνεται σε όλα τα GUI components της φόρμας που απεκονίζουν ιδιότητες του Information Field. Έτσι όταν γίνει μια αλλαγή, το αντίστοιχο component ειδοποιεί την ChangeListenerImpl και αυτή εκτελεί τις ενέργειες που είναι δηλωμένες στην μέθοδο `changesOccured`.

admintool.AbstractCHangeListerImpl
Στην εφαρμογή οι αλλαγές σε ένα JCheckboxes, JRadiobuttons ή σε ένα JTextField αντιμετωπίζονται αντιμετωπίζονται με τον ίδιο τρόπο: οι ιδιότητες του Information Field εχουν τροποποιηθεί και πρέπει να σωθούν.
Συνεπώς, η abstract κλάση AbstractCHangeListerImpl, υλοποιεί της δυο παρακάτω Interfaces:

- **javax.swing.event.DocumentListener**: είναι που πρέπει να δηλώσει ένα αντικείμενο που θέλει να ειδοποιείται για αλλαγές σε text κείμενο. Χρησιμοποιείται στα JTextFields.
- **javax.swing.event.ChangeListener**: Καθορίζει ένα αντικείμενο που ακούει σε ChangeEvents. Χρησιμοποιείται στα JCheckboxes, JRadiobuttons.

Με την υλοποίηση και των δυο interfaces, συγκεντρώνεται η λειτουργικότητα, που χρειάζεται να υλοποιηθεί σε περίπτωση τροποποιησης, σε μια μόνο μέθοδο `changesOccured`, η οποία ειναι αυτη που κανει την κλάση abstract.

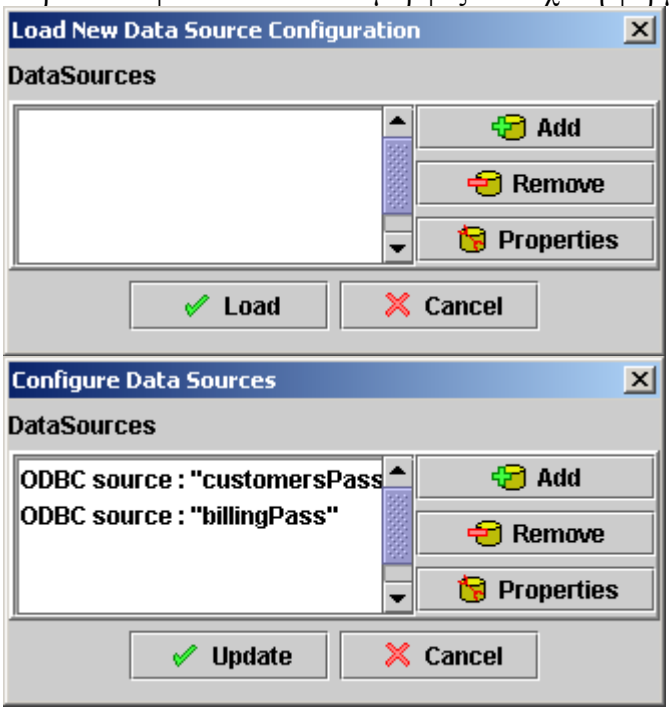
Διαχείριση των πηγών δεδομένων

Συμφωνα με τις απαιτήσεις της εφαρμογής, πρέπει να υπάρχει οι εξής δυνατότητες:

- δημιουργία μιας νέας διαμόρφωσης
- διαχείριση της υπάρχουσας διαμόρφωσης, δηλαδή προσθήκη και αφάιρεση πηγών δεδομένων απο την διαμόρφωση
- δημιουργία μιας νέας πηγής
- Τροποποίηση των ιδιοτήτων μιας πηγής, που ήδη υπάρχει

admintool.DataSourcesConfiguration

Η κλάση αυτή ικανοποιεί τις δυο πρώτες απο τις παραπάνω απαιτήσεις, δηλαδή την δημιοργία και διαχείριση σε επίπεδο διαμόρφωσης. Επειδή υπάρχουν εμφανείς ομοιότητες μεταξυ των δυο απαιτήσεων, χρησιμοποιήθηκε αυτη η κλάση για να ικανοποιήσει και τις δυο. Η κλάση απο τις παραμέτρους που δέχεται στην αρχικοποιηση της (το data source configuration document), πέφτει σε μια απο τις δυο περιπτωσεις.
Παρακάτω φαίνονται οι δυο μορφές που έχει η φόρμα.

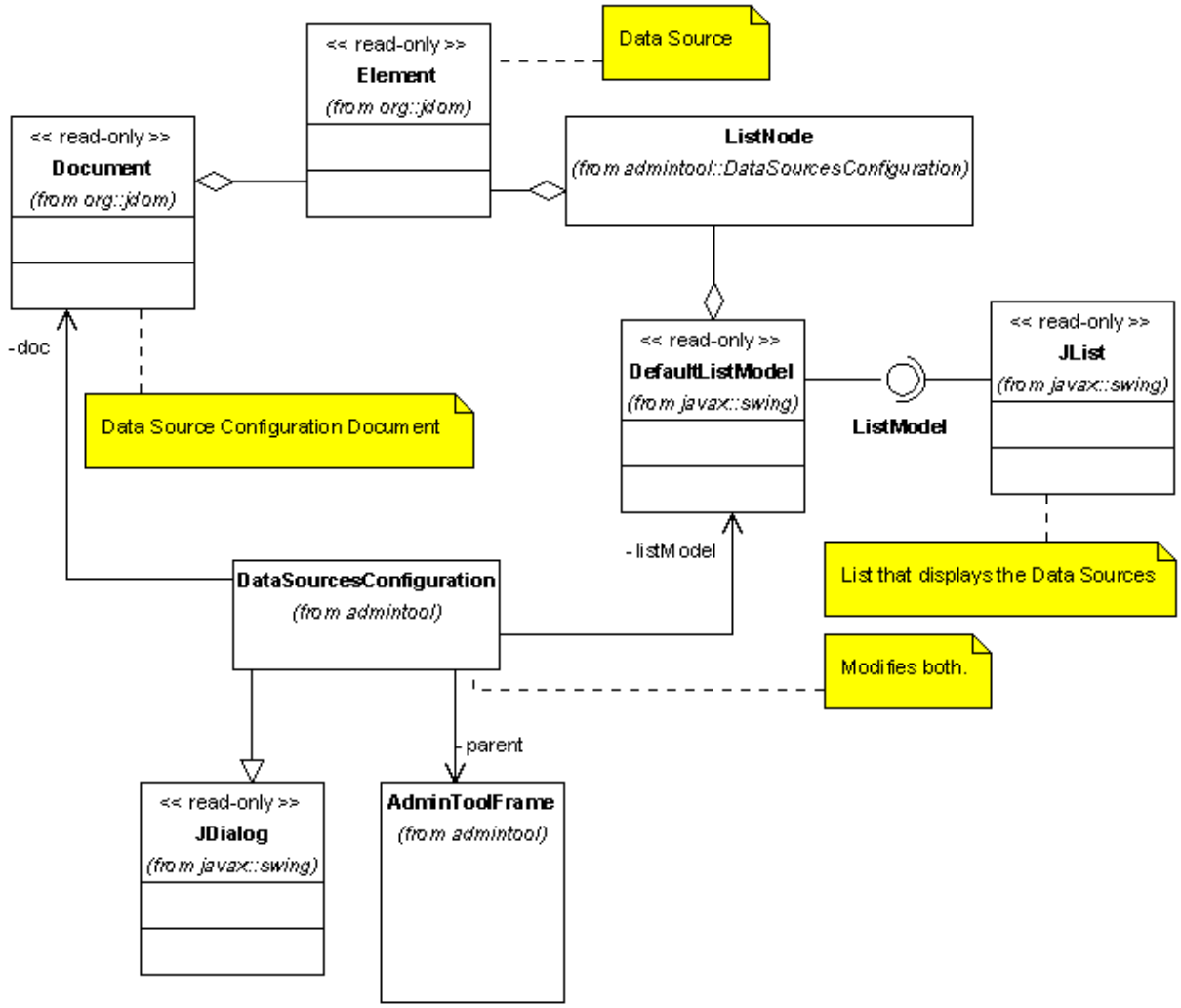


Όταν δεν υπάρχουν πηγές που να έχουν φορτωθεί στην εφαρμογή.

Όταν υπαρχουν ήδη φορτωμένες στην εφαρμογή πηγές, αλλά πρέπει να διαμορφωθούν.

Ουσιαστικά αυτή η κλάση διαχειρίζεται την XML δομή Data Source Configuration, την οποία η εφαρμογή την διατηρεί σε ένα JDOM Document. Οι πηγές που φαίνονται παραπάνω είναι αυτές που αντιστοιχούν στο παράδειγμα της δομής Data Source Configuration, που δόθηκε

στην σχετική παράγραφο. Στο παρακάτω διαγραμμα δίνει το τρόπο που απεκονίζεται κάθε πηγη στηη φορμα.

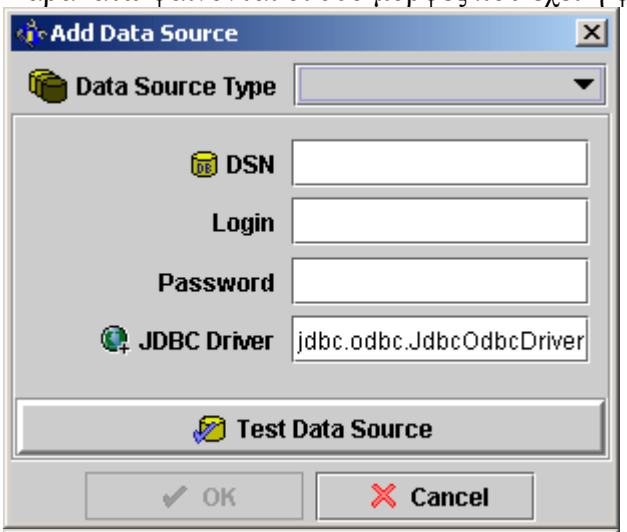


Η φόρμα υλοποιείται επεκτείνοντας ένα JDialog, ενώ σαν πατέρα έχει το κυρίως frame της εφαρμογης.
Καθε πηγη δεδομένων είναι ένα element στο Data Source Configuration document. Το element αυτο γίνεται wrapped (περιτυλίγεται) σε ένα **ListNode** αντικείμενο. Ο βασικός λόγος είναι να μπορέσουμε να κάνουνε override την toString μέθοδο, ώστε το element του Data Source να εμφανίζεται με την επιθυμητη περιγραφή στην JList. Το αντικείμενο αυτο προστίθεται στην DefaultListModel, η οποια με την σειρά της αρχικοποιεί την JList. Όταν ο χρήστης τροποποιεί την δοιαμόρφωση μέσω των επιλογών της εφαρμογής, οι μέθοδοι εξυπηρέτησης των επιλογών υτών (action performed methods) τροποποιούν και το document αλλα και την DefaultListModel. Τέλος, αξιζει να παρατηρηθεί ότι το Element ανήκει και στα δυο αντικείμενα.

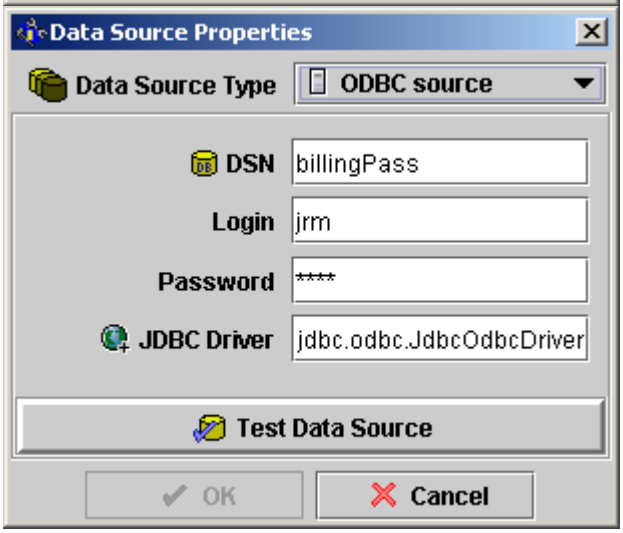
admintool.DataSourcesProperties

Η κλάση αυτή ικανοποιεί τις δυο τελευταίες απο τις παραπάνω απαιτήσεις, δηλαδή την δημιουργία και διαχείριση σε επίπεδο πηγής δεδομένων.
Επειδή υπάρχουν εμφανείς ομοιότητες μεταξυ των δυο απαιτήσεων, χρησιμοποιήθηκε αυτη η κλάση για να ικανοποιήσει και τις δυο.

Όταν ο χρήστης επιλέξει Add ή Properties στην φόρμα DataSourceConfiguration, τότε αρχικοποιείται μια φόρμα DataSourceProperties. Η κλάση απο τις παραμέτρους που δέχεται στην αρχικοποιηση της (το element της πηγής δεδομένων) πέφτει σε μια απο τις δυο περιπτωσεις.
Παρακάτω φαίνονται οι δυο μορφές που έχει η φόρμα.

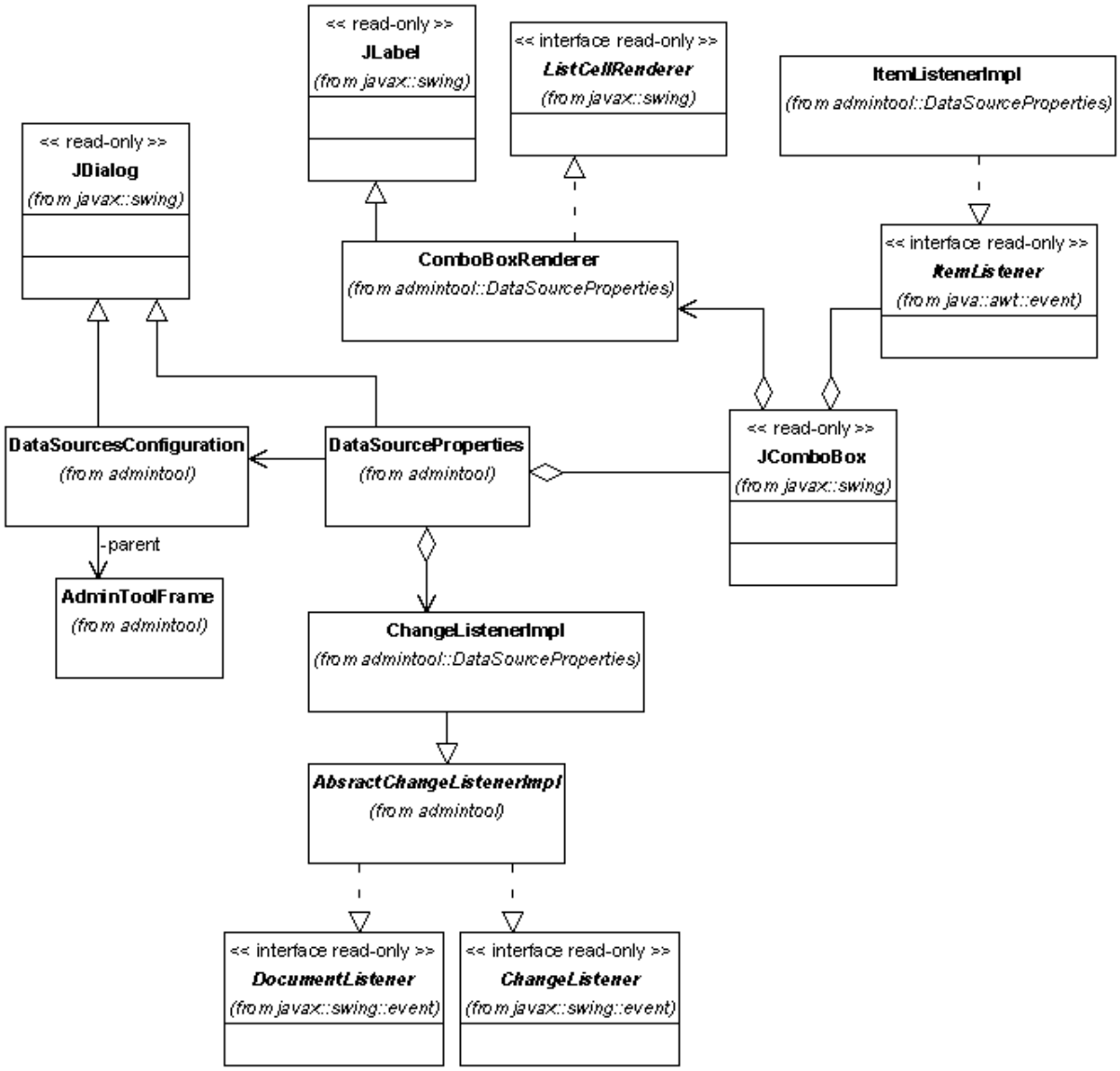


Ο χρήστης επέλεξε Add στην DataSourceConfiguration. Η φόρμα εμφανίζεται κενή.



Ο χρήστης επέλεξε Properties στην DataSourceConfiguration. Η φόρμα περιέχει τις παραμέτρους της πηγής.

Ουσιαστικά αυτή η κλάση διαχειρίζεται το element της πηγής δεδομένων της XML δομή Data Source Configuration. Οι παράμετροι της πηγής δεδομένων που εισάγει ή τροποποιεί ο χρήστης πρέπει πρώτα να περάσουν δοκιμαστούν (Test Data Source button) πριν δοθεί το δικαίωμα να αποθηκευτούν (ενεργοποίηση του OK button).



Η φόρμα έχει ένα combo box με τους τύπους των πηγών δεδομένων που μπορεί να επιλέξει ο χρήστης. Το combo box έχει ένα renderer (ComboBoxRenderer) ο οποίος παρέχει την επιθυμητή περιγραφή κάθε τύπου και το αντίστοιχο εικονίδιο. Επίσης, έχει και ενα ItenListener ο οποίος αντιστοιχεί τον τύπο της πηγής που επολέχτηκε στο combo box με το String που εκφράζεται στην java, πχ. ODBC εκφράζεται με odbc.

Η φόρμα έχει την κλάση **ChangeListenerImpl** για να μπορεί να "ακούει" στις αλλαγές των δεδομένων που γίνονται στα πεδια των GUI component του, ώστε να προτρέπει το χρήστης να αποθηκεύσει τις αλλαγές. Περισσότερα αναφέρονται στην παράγραφο της admintool.InformationFieldProperties φορμας.

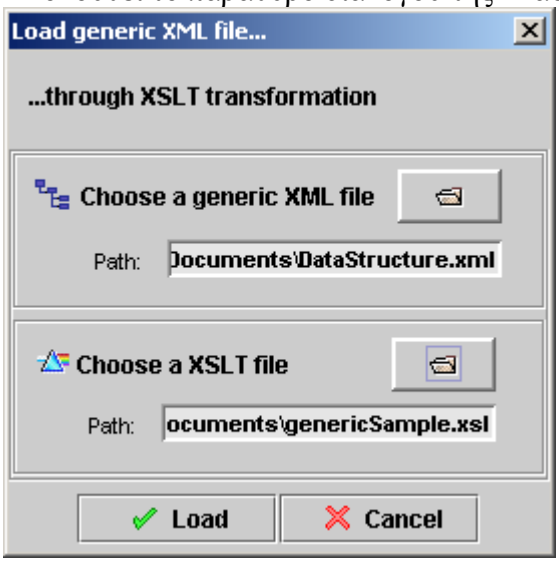
Ανοιγμα Αρχείο XML και μετασχηματισμού του στην Data Structure δομή

Συμφωνα με τις απαιτήσεις της εφαρμογής, πρέπει να υπάρχει οι εξής δυνατότητες:

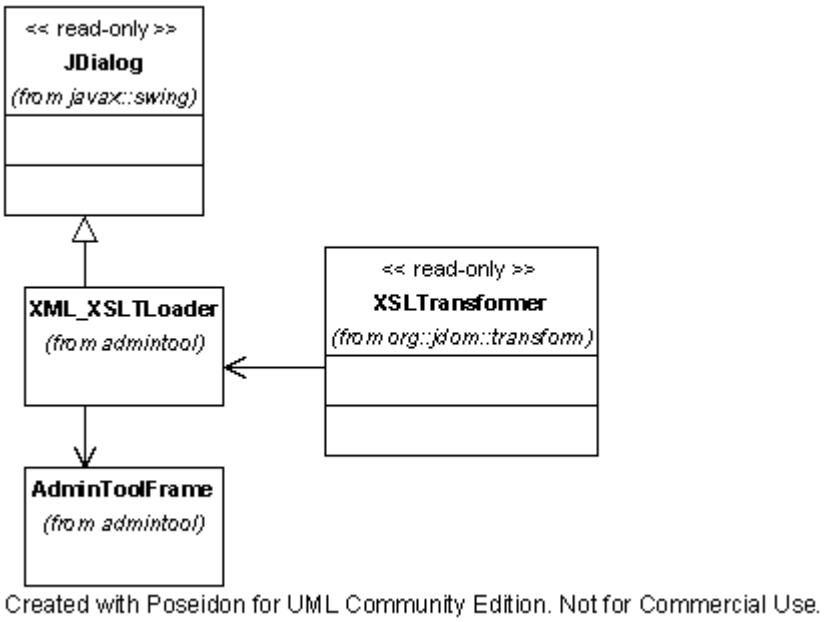
- Ανοιγμα XML αρχειου και μετασχηματισμού του μεσω XSLT
- Ανοιγμα XML αρχειου και μετασχηματισμού του μεσω JDOM κλάσεων, Φορτωση κατα την εκκίνηση της εφαρμογή των κλάσεων μετασχηματισμού.

admintool. XML_XSLTLoader

Η κλάση ικανοποιεί την πρώτη απο τις παραπάνω απιτήσεις. Προτρέπει στο χρήστης να εισάγει το XML αρχείο και το αντιστοιχο XSL με το οποίο θέλει να το μετασχηματίσει.
Ακολουθεί το παράθυρο διαλόγου της κλάσης.



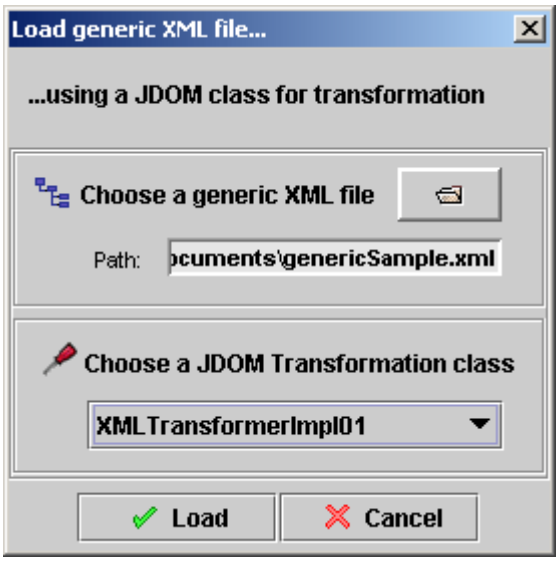
Ακολουθεί το διάγραμμα κλάσεων



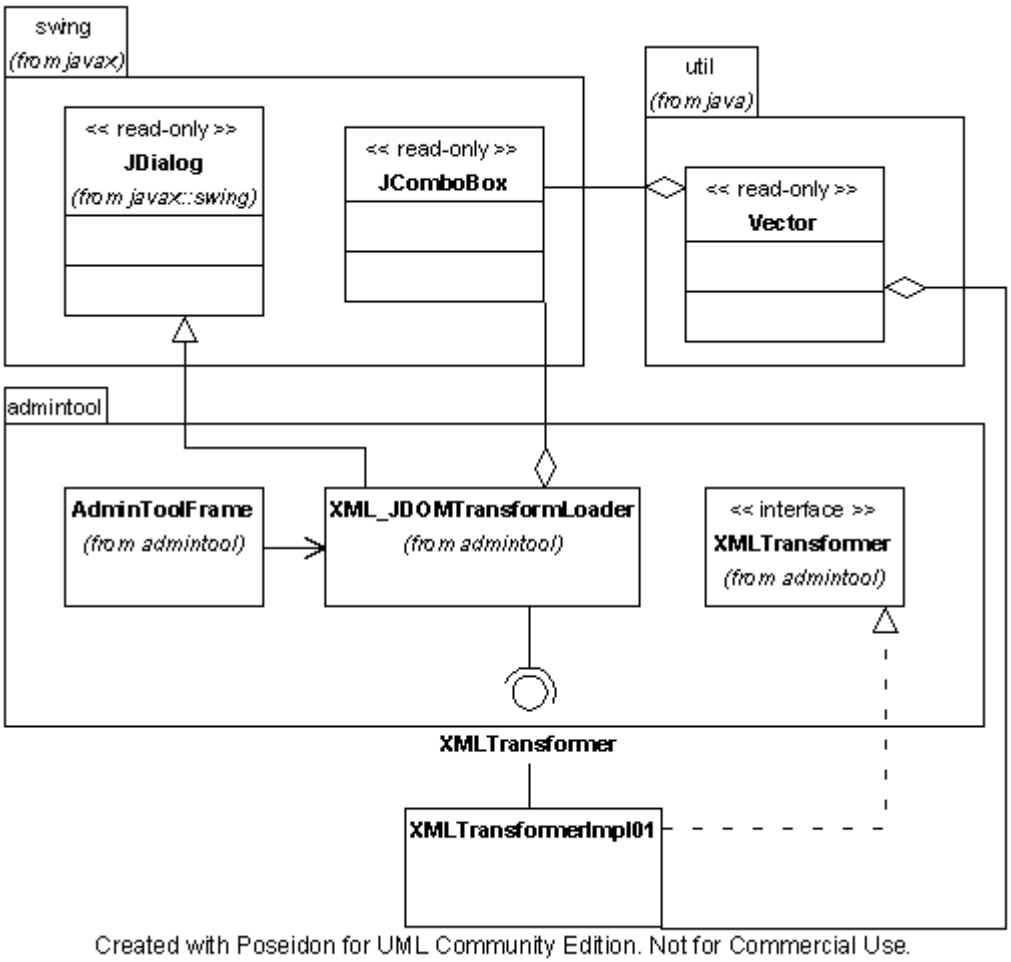
Ο μετασχηματισμός του XML με το XSL αρχειο γίνεται με την κλάση org.jdom.transform.XSLTransformer.

admintool. XML_JDOMTransformLoader

Η κλάση ικανοποιεί την δεύτερη απο τις παραπάνω απιτήσεις. Προτρέπει στο χρήστης να εισάγει το XML αρχείο και να επιλέξει την κλάση μετασχηματισμού που θέλει.
Ακολουθεί το παράθυρο διαλόγου της κλάσης.



Ακολουθεί το διάγραμμα κλάσεων



Η κλάση έχει ένα combo box το οποία δείχνει τις κλάσεις μετασχηματισμου που μπορεί να επιλέξει ο χρήστης. Με την αρχικοποίηση της κλάσης, φορτώνεται απο ένα properties αρχειο ποιες κλάσεις είναι διαθέσιμες. Η εφαρμογή διαβάζει το αρχείο και φορτώνει ενώ εκτελείται (runtime loading) τις κλάσεις μετασχηματισμού. Με αυτές αρχικοποιείται το Vector του combo box.

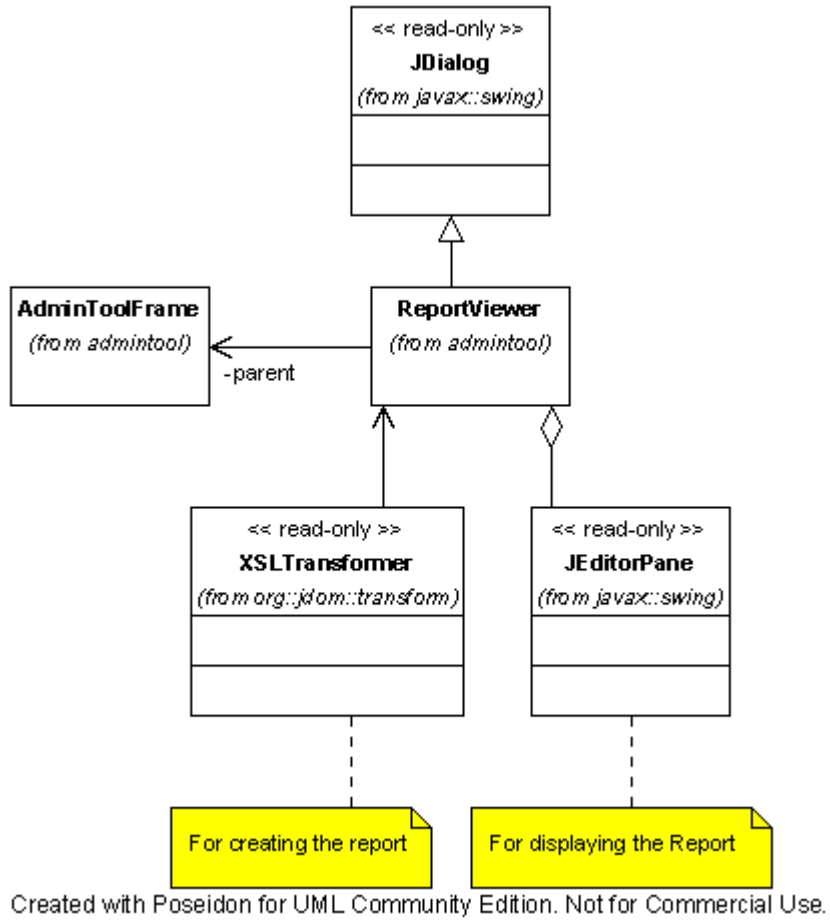
admintool.XMLTransformer
Ο χρήστη μπορεί να παρέχει οποια κλάση θελήσει για να μετσχηματίσει το XML αρχειο που επιθυμεί, αρκεί αυτή να υλοποιεί την interface XMLTransformer. Η εφαρμογή «μιλάει» με τις κλάσεις μετασχηματισμού μέσω αυτής της διεπαφής και συνεπώς είναι αδιαφορο πως έχει υλοποιήσει ο χρήστης την κλάση.

Εμφάνιση αναφορών

admintool.ReportViewer
Υλοποιεί της απαιτήσεις:

- Εμφάνιση Αναφοράς Συσχετίσεων
- Εμφανιση XML του Αρχείο Συσχετίσεων
- Εμφανιση XML των δομων δεδομένων

Το διάγραμμα κλάσεων:



Ουσιαστικά ο ReportViewer είναι ένα παράθυρο διαλόγου (JDialog) που έχει ένα XSLTransformer για την μετατροπή του XML κειμένου σε μια html σελίδα μέσω ενός XSL αρχείου. Το XSL αρχείο είναι το report.xsl και δίνεται στο κεφάλαιο του κώδικα της εφαρμογής. Τέλος, το JEditorPane απεκονίζει την html σελίδα της αναφοράς.
Το ο μετασχηματισμος μέσω XSL λαμβάνει χώρα μόνο στην πρώτη περίπτωση. Στις άλλες δυο περιπτώσεις απλά απεκονίζεται το XML κείμενο, χωρίς μετασχηματισμό.

Πληροφορίες για την εφαρμογή

admintool.AdminToolFrame_AboutBox
Παρέχει πληροφορίες για την εφαρμογή. Είναι ένα JDialog με πληροφορίες για την εφαρμογή χωρίς καποια άλλη λειτουργικότητα που χρειάζεται να αναλυθεί.

Εργαλεία που χρησιμοποιήθηκαν στην ανάπτυξη της εφαρμογής

Για την δημιουργία της εφαρμογής χρησιμοποιήθηκαν τα καόλουθα:

- το ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) της Borland το JBuilderX (http://www.borland.com/jbuilder/). Η χρήση ενός IDE ήταν απαραίτητη για την ανάπτυξη μιας εφαρμογης τόσο μεγαλου μεγέθους (9300 γραμμές κώδικας) .
- Το JDK (Java Development Kit) 1.4.2 (java.sun.com).
- ως parser ο Apache Xerces στην εκδοση 2.6.2 (http://xml.apache.org/xerces2-j/) και
- το JDOM στην έκδοση 1.0. (www.jdom.org)

Το API - Λεπτομερής Σχεδιασμός - της εφαρμογής

Ακολουθεί το API της εφαρμογής όπου αποτελεί ουσιαστικά το λεπτομερή σχεδιασμό της εφαρμογής.
Το API δημιουργήθηκε με το σχετικό εργαλείο που παρέχει η Java, λεγε με javadoc, και είναι σε μορφή html.

Hierarchy For All Packages

Package Hierarchies:
admintool, admintool.jdomtreemodel

Class Hierarchy

- class java.lang.Object
  - class admintool.**AbsractChangeListenerImpl** (implements javax.swing.event.ChangeListener, javax.swing.event.DocumentListener)
    - class admintool.**DataSourceProperties.ChangeListenerImpl**
    - class admintool.**InformationFieldProperties.ChangeListenerImpl**
  - class admintool.**AdminTool**
  - class admintool.**AdminToolFrame.AdminToolFrame_jButtonCloseAssoc_actionAdapter** (implements java.awt.event.ActionListener)
  - class admintool.**AdminToolFrame.AdminToolFrame_jButtonConfigDSC_actionAdapter** (implements java.awt.event.ActionListener)
  - class admintool.**AdminToolFrame.AdminToolFrame_jButtonLoadAssoc_actionAdapter** (implements java.awt.event.ActionListener)
  - class admintool.**AdminToolFrame.AdminToolFrame_jButtonLoadDST_actionAdapter** (implements java.awt.event.ActionListener)
  - class admintool.**AdminToolFrame.AdminToolFrame_jButtonLoadExistDSC_actionAdapter** (implements java.awt.event.ActionListener)
  - class admintool.**AdminToolFrame.AdminToolFrame_jButtonLoadIS_actionAdapter** (implements java.awt.event.ActionListener)

- class admintool.**AdminToolFrame.AdminToolFrame_jButtonLoadNewDSC_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**AdminToolFrame.AdminToolFrame_jButtonLoadXML_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**AdminToolFrame.AdminToolFrame_jButtonLoadXMLJDOM_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**AdminToolFrame.AdminToolFrame_jButtonLoadXMLXSLT_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**AdminToolFrame.AdminToolFrame_jButtonNewAssoc_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**AdminToolFrame.AdminToolFrame_jButtonSaveAssoc_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**AdminToolFrame.AdminToolFrame_jButtonSaveDSC_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**AdminToolFrame.AdminToolFrame_jButtonSaveDST_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**AdminToolFrame.AdminToolFrame_jMenuFileExit_ActionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**AdminToolFrame.AdminToolFrame_jMenuHelpAbout_ActionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**AdminToolFrame.AdminToolFrame_jMenuItemConfigDataSrc_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**AdminToolFrame.AdminToolFrame_jMenuItemFileCloseAssoc_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**AdminToolFrame.AdminToolFrame_jMenuItemFileLoadAssoc_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**AdminToolFrame.AdminToolFrame_jMenuItemFileNewAssoc_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**AdminToolFrame.AdminToolFrame_jMenuItemFileSaveAssoc_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**AdminToolFrame.AdminToolFrame_jMenuItemInfoFieldProps_actionAdapter** (implements java.awt.event.ActionListener)

- class admintool.**AdminToolFrame.AdminToolFrame_jMenuItemInfoStructXML_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**AdminToolFrame.AdminToolFrame_jMenuItemLoadDataStruct_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**AdminToolFrame.AdminToolFrame_jMenuItemLoadExistDataSrcConfig_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**AdminToolFrame.AdminToolFrame_jMenuItemLoadInfoStruct_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**AdminToolFrame.AdminToolFrame_jMenuItemLoadNewDataSrcConfig_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**AdminToolFrame.AdminToolFrame_jMenuItemLoadXml_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**AdminToolFrame.AdminToolFrame_jMenuItemLoadXML_jdomTransform_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**AdminToolFrame.AdminToolFrame_jMenuItemLoadXML_XSLT_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**AdminToolFrame.AdminToolFrame_jMenuItemRemoveAll_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**AdminToolFrame.AdminToolFrame_jMenuItemRemoveAssoc_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**AdminToolFrame.AdminToolFrame_jMenuItemRemoveFile_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**AdminToolFrame.AdminToolFrame_jMenuItemReport_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**AdminToolFrame.AdminToolFrame_jMenuItemSaveDataSrcConfig_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**AdminToolFrame.AdminToolFrame_jMenuItemSaveDataStruct_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**AdminToolFrame.AdminToolFrame_jMenuItemSelectedXML_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**AdminToolFrame.GUIUpdater**
- class admintool.**AdminToolFrame.TreeModelAdapter** (implements javax.swing.event.TreeModelListener)
- class admintool.**AdminToolFrame.TreeSelectionHandler** (implements javax.swing.event.TreeSelectionListener)

- class java.awt.Component (implements java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable)
  - class java.awt.Container
    - class javax.swing.JComponent (implements java.io.Serializable)
      - class javax.swing.JLabel (implements javax.accessibility.Accessible, javax.swing.SwingConstants)
        - class admintool.**DataSourceProperties.ComboBoxRenderer** (implements javax.swing.ListCellRenderer)
        - class javax.swing.tree.DefaultTreeCellRenderer (implements javax.swing.tree.TreeCellRenderer)
          - class admintool.jdomtreemodel.**DataSrcTreeCellRenderer**
          - class admintool.jdomtreemodel.**InfoStructTreeCellRenderer**
      - class java.awt.Window (implements javax.accessibility.Accessible)
        - class java.awt.Dialog
          - class javax.swing.JDialog (implements javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants)
            - class admintool.**AdminToolFrame_AboutBox** (implements java.awt.event.ActionListener)
            - class admintool.**DataSourceProperties**
            - class admintool.**DataSourcesConfiguration**
            - class admintool.**InformationFieldProperties**
            - class admintool.**ReportViewer**
            - class admintool.**XML_JDOMTransformLoader**
            - class admintool.**XML_XSLTLoader**
        - class java.awt.Frame (implements java.awt.MenuContainer)
          - class javax.swing.JFrame (implements javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants)
            - class admintool.**AdminToolFrame** (implements admintool.DataModifier)
- class admintool.**DataSourceProperties.DataSourceProperties_jButtonCancel_actionAdapter** (implements java.awt.event.ActionListener)

- class admintool.**DataSourceProperties.DataSourceProperties_jButtonOK_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**DataSourceProperties.DataSourceProperties_jButtonTest_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**DataSourceProperties.ItemListenerImpl** (implements java.awt.event.ItemListener)
- class admintool.**DataSourcesConfiguration.DataSourcesConfiguration_jButtonAdd_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**DataSourcesConfiguration.DataSourcesConfiguration_jButtonCancel_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**DataSourcesConfiguration.DataSourcesConfiguration_jButtonOK_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**DataSourcesConfiguration.DataSourcesConfiguration_jButtonProps_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**DataSourcesConfiguration.DataSourcesConfiguration_jButtonRemove_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**DataSourcesConfiguration.ListNode**
- class admintool.jdomtreemodel.**DataSrcMapper**
- class admintool.jdomtreemodel.**DisplayedPathEvaluator**
- class javax.swing.filechooser.FileFilter
  - class admintool.**FilesFilter**
- class admintool.**InformationFieldProperties.InformationFieldProperties_jButtonApply_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**InformationFieldProperties.InformationFieldProperties_jButtonCancel_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**InformationFieldProperties.InformationFieldProperties_jButtonOK_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**InformationFieldProperties.InformationFieldProperties_jRadioButtonNo_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**InformationFieldProperties.InformationFieldProperties_jRadioButtonYes_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.jdomtreemodel.**JDOMTreeModel** (implements javax.swing.tree.TreeModel)
  - class admintool.jdomtreemodel.**JDOMTreeModelAttr**
- class java.awt.event.MouseAdapter (implements java.awt.event.MouseListener)
  - class admintool.**AdminToolFrame.PopupListener**
- class admintool.**ReportViewer.ReportViewer_jButtonOK_actionAdapter** (implements java.awt.event.ActionListener)

- class javax.swing.TransferHandler (implements java.io.Serializable)
  - class admintool.**TreeTransferHandler**
- class admintool.**XML_JDOMTransformLoader.XML_JDOMTransformLoader_jButtonCancel_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**XML_JDOMTransformLoader.XML_JDOMTransformLoader_jButtonOK_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**XML_JDOMTransformLoader.XML_JDOMTransformLoader_jButtonXML_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**XML_XSLTLoader.XML_XSLTLoader_jButtonCancel_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**XML_XSLTLoader.XML_XSLTLoader_jButtonOK_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**XML_XSLTLoader.XML_XSLTLoader_jButtonXML_actionAdapter** (implements java.awt.event.ActionListener)
- class admintool.**XML_XSLTLoader.XML_XSLTLoader_jButtonXSLT_actionAdapter** (implements java.awt.event.ActionListener)
- class org.xml.sax.helpers.XMLFilterImpl (implements org.xml.sax.ContentHandler, org.xml.sax.DTDHandler, org.xml.sax.EntityResolver, org.xml.sax.ErrorHandler, org.xml.sax.XMLFilter)
  - class admintool.jdomtreemodel.**XMLWhitespaceFilter**
- class admintool.**XMLTransferable** (implements java.awt.datatransfer.Transferable)
- class admintool.jdomtreemodel.**XPathEvaluator**

Interface Hierarchy

- interface admintool.**DataModifier**
- interface admintool.**XMLTransformer**

Package admintool

Administration Tool main package.

See:
Description

Interface Summary	
DataModifier	Objects can update the status of GUIUpdater that ADminToolFrame class has, through this interface.
XMLTransformer	This is the interface that XML_JDOMTransformLoader class needs a class to implement, in order for that class to be considered a Transformation class.

Class Summary	
AbsractChangeListenerImpl	Class that provides a dummy implementation of the DocumentListener and ChangeListener Interfaces.
AdminTool	Administration Tool Description: Main class.
AdminToolFrame	Title: AdministrationFrame Description: Class the everything else is link into.
AdminToolFrame_AboutBox	The About Dialog.
DataSourceProperties	A Dialog for manipulating Data Source Configuration Properties.
DataSourcesConfiguration	A Dialog for manipulating Data Source Configurations.
FilesFilter	The File Filter is used from JFileChooser and supports xml, dsc, dst, is, xsl, asc files.
InformationFieldProperties	A Dialog for displaying and manipulating an Information Field's Properties.
ReportViewer	A Dialog for dispaying Report on Information Structure, XML of Information Structure and XML of a selection in Data Structure.
TreeTransferHandler	The TransferHanlder that provides the support for associating nodes in the JTree through Drag and Drop (DnD).
XML_JDOMTransformLoader	Load generic XML file through JDOM transformation Dialog.

XML_XSLTLoader	Load generic XML file through XSLT transformation Dialog.
XMLTransferable	Transferable interface implementation that supports the transfer of org.jdom.Element objects.

Package admintool Description

Administration Tool main package.

Contains:

- Main class of application
- Frame class of the application
- Dialog classes for the various dialog windows of the application
- Transfer Handler tha provides the support for transfering data between JTrees through Drag and Drop.
- Various helper class for GUI, like Change Listeners and File filter.
- and, of course, the JDOMTreeModel package which contains the necessary classes for displaying a JDOM document into a JTree

See Also:
`admintool.jdomtreemodel`

---

admintool

Class AbsractChangeListenerImpl

```
java.lang.Object
  └admintool.AbsractChangeListenerImpl
```

All Implemented Interfaces:

javax.swing.event.ChangeListener, javax.swing.event.DocumentListener, java.util.EventListener

Direct Known Subclasses:

DataSourceProperties.ChangeListenerImpl, InformationFieldProperties.ChangeListenerImpl

---

public abstract class **AbsractChangeListenerImpl**
extends java.lang.Object
implements javax.swing.event.DocumentListener, javax.swing.event.ChangeListener

Class that provides a dummy implementation of the DocumentListener and ChangeListener Interfaces. Document Listener is for listening changes on JTestFields and Change Listener is for listening changes on JCheckBoxex and JRadioButtons.

This class is only used in InformationFieldProperties class for listening changes of its GUI components, which is also the reason that this class was written. InformationFieldProperties had GUI conmponents that needed a ChangeListener or a DocumentListener. So, in order to avoid to put so much code in a private helper class of InformationFieldProperties, the interfaces are dummy implemented here, resulting in a abstract class which needs only one method to be implement in order to work.

See Also:

`InformationFieldProperties`

Constructor Summary

`AbsractChangeListenerImpl``()`	Default Constructor.

Method Summary

`void`	**`changedUpdate`**`(javax.swing.event.DocumentEvent e)` Calls `changesOccured` helper method.
`(package private) abstract void`	**`changesOccured`**`()` The actions that need to be made when a change is made.
`void`	**`insertUpdate`**`(javax.swing.event.DocumentEvent e)` Calls `changesOccured` helper method.
`void`	**`removeUpdate`**`(javax.swing.event.DocumentEvent e)` Calls `changesOccured` helper method.
`void`	**`stateChanged`**`(javax.swing.event.ChangeEvent e)` Calls `changesOccured` helper method.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

AbsractChangeListenerImpl
`public AbsractChangeListenerImpl()`

Default Constructor.

Method Detail

changedUpdate
`public void changedUpdate(javax.swing.event.DocumentEvent e)`

Calls `changesOccured` helper method.

Specified by:

`changedUpdate` in interface `javax.swing.event.DocumentListener`

Parameters:

`e` - DocumentEvent

See Also:

`changesOccured()`

---

changesOccured
`abstract void changesOccured()`

The actions that need to be made when a change is made. Implement this method according to the actions need to be made when the data kept on the GUI components that have this class as a Change/Document listener are modified.

---

insertUpdate
`public void insertUpdate(javax.swing.event.DocumentEvent e)`

Calls `changesOccured` helper method.

Specified by:

`insertUpdate` in interface `javax.swing.event.DocumentListener`

Parameters:

`e` - DocumentEvent

See Also:

`changesOccured()`

---

removeUpdate
`public void removeUpdate(javax.swing.event.DocumentEvent e)`

Calls `changesOccured` helper method.

Specified by:

`removeUpdate` in interface `javax.swing.event.DocumentListener`

Parameters:
e - DocumentEvent
See Also:
`changesOccured()`

---

stateChanged

`public void stateChanged(javax.swing.event.ChangeEvent e)`

Calls `changesOccured` helper method.
Specified by:
`stateChanged` in interface `javax.swing.event.ChangeListener`
Parameters:
e - ChangeEvent
See Also:
`changesOccured()`

---

admintool

Class AdminTool

`java.lang.Object`
  `└ admintool.AdminTool`

---

public class **AdminTool**
extends java.lang.Object

Administration Tool

Description: Main class. It only creates an AdminToolFrame by calling its constructor and displays it. The all of the application's functionallity is implemented in AdminToolFrame

See Also:
`AdminToolFrame`

Field Summary

(package private) boolean	**packFrame** Set it true to pack the frame or false to validate it.

Constructor Summary

AdminTool()
Construct the application.

Method Summary

static void	**main**(java.lang.String[] args) Main method.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

packFrame

boolean **packFrame**

Set it true to pack the frame or false to validate it.

Constructor Detail

AdminTool

public **AdminTool**()

Construct the application. Generates the application frame and displayes it. The frame is an AdminToolFrame. It Validate frames that have preset sizes, Pack frames that have useful preferred size info, e.g. from their layout and centers the window

Method Detail

main

public static void **main**(java.lang.String[] args)

Main method. Sets the LookandFeel and calls the constructor.s
Parameters:
`args` - String[] not used

Overview **Package** **Class** **Tree** **Index** **Help**

PREV CLASS NEXT CLASS FRAMES NO FRAMES All Classes
SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

---

admintool

Class AdminToolFrame

```
java.lang.Object
  └java.awt.Component
      └java.awt.Container
          └java.awt.Window
              └java.awt.Frame
                  └javax.swing.JFrame
                      └admintool.AdminToolFrame
```

All Implemented Interfaces:
javax.accessibility.Accessible, DataModifier, java.awt.image.ImageObserver, java.awt.MenuContainer, javax.swing.RootPaneContainer, java.io.Serializable, javax.swing.WindowConstants

---

public class **AdminToolFrame**
extends javax.swing.JFrame
implements DataModifier

Title: AdministrationFrame

Description: Class the everything else is link into. This is actually the main application class, because all the functionality is implemented on this class or in other classes connected to it. AdminTool class is only a caller of this class.

The application has three Documents:

- the document that holds the Information Structure that is loaded.

- the document that holds the Data Structures which are loaded from Data Sources or from files.
- the document that holds which data source configurations that are loaded.

Each Data structure loaded from a Data Source has a corresponding Data Source configuration.

*Quick Reference: The word **document** is refered to a JDOM Document.*
An XML which is loaded with JDOM is also refered as a "document". XML is a file in the disk not the data structure created by JDOM when a XML file is loaded.

See Also:

`AdminTool`, Serialized Form

Nested Class Summary

(package private) class	**`AdminToolFrame.AdminToolFrame_jButtonCloseAssoc_actionAdapter`** Helper inner class created automatically by development tool.
(package private) class	**`AdminToolFrame.AdminToolFrame_jButtonConfigDSC_actionAdapter`** Helper inner class created automatically by development tool.
(package private) class	**`AdminToolFrame.AdminToolFrame_jButtonLoadAssoc_actionAdapter`** Helper inner class created automatically by development tool.
(package private) class	**`AdminToolFrame.AdminToolFrame_jButtonLoadDST_actionAdapter`** Helper inner class created automatically by development tool.
(package private) class	**`AdminToolFrame.AdminToolFrame_jButtonLoadExistDSC_actionAdapter`** Helper inner class created automatically by development tool.
(package private) class	**`AdminToolFrame.AdminToolFrame_jButtonLoadIS_actionAdapter`** Helper inner class created automatically by development tool.
(package private) class	**`AdminToolFrame.AdminToolFrame_jButtonLoadNewDSC_actionAdapter`** Helper inner class created automatically by development tool.
(package private) class	**`AdminToolFrame.AdminToolFrame_jButtonLoadXML_actionAdapter`** Helper inner class created automatically by development tool.
(package private) class	**`AdminToolFrame.AdminToolFrame_jButtonLoadXMLJDOM_actionAdapter`** Helper inner class created automatically by development tool.
(package private) class	**`AdminToolFrame.AdminToolFrame_jButtonLoadXMLXSLT_actionAdapter`** Helper inner class created automatically by development tool.
(package private) class	**`AdminToolFrame.AdminToolFrame_jButtonNewAssoc_actionAdapter`** Helper inner class created automatically by development tool.
(package private) class	**`AdminToolFrame.AdminToolFrame_jButtonSaveAssoc_actionAdapter`** Helper inner class created automatically by development tool.

(package private) class	**AdminToolFrame.AdminToolFrame_jButtonSaveDSC_actionAdapter** Helper inner class created automatically by development tool.
(package private) class	**AdminToolFrame.AdminToolFrame_jButtonSaveDST_actionAdapter** Helper inner class created automatically by development tool.
(package private) class	**AdminToolFrame.AdminToolFrame_jMenuFileExit_ActionAdapter** Helper inner class created automatically by development tool.
(package private) class	**AdminToolFrame.AdminToolFrame_jMenuHelpAbout_ActionAdapter** Helper inner class created automatically by development tool.
(package private) class	**AdminToolFrame.AdminToolFrame_jMenuItemConfigDataSrc_actionAdapter** Helper inner class created automatically by development tool.
(package private) class	**AdminToolFrame.AdminToolFrame_jMenuItemFileCloseAssoc_actionAdapter** Helper inner class created automatically by development tool.
(package private) class	**AdminToolFrame.AdminToolFrame_jMenuItemFileLoadAssoc_actionAdapter** Helper inner class created automatically by development tool.
(package private) class	**AdminToolFrame.AdminToolFrame_jMenuItemFileNewAssoc_actionAdapter** Helper inner class created automatically by development tool.
(package private) class	**AdminToolFrame.AdminToolFrame_jMenuItemFileSaveAssoc_actionAdapter** Helper inner class created automatically by development tool.
(package private) class	**AdminToolFrame.AdminToolFrame_jMenuItemInfoFieldProps_actionAdapter** Helper inner class created automatically by development tool.
(package private) class	**AdminToolFrame.AdminToolFrame_jMenuItemInfoStructXML_actionAdapter** Helper inner class created automatically by development tool.
(package private) class	**AdminToolFrame.AdminToolFrame_jMenuItemLoadDataStruct_actionAdapter** Helper inner class created automatically by development tool.
(package private) class	**AdminToolFrame.AdminToolFrame_jMenuItemLoadExistDataSrcConfig_actionAdapter** Helper inner class created automatically by development tool.
(package private) class	**AdminToolFrame.AdminToolFrame_jMenuItemLoadInfoStruct_actionAdapter** Helper inner class created automatically by development tool.
(package private) class	**AdminToolFrame.AdminToolFrame_jMenuItemLoadNewDataSrcConfig_actionAdapter** Helper inner class created automatically by development tool.
(package private) class	**AdminToolFrame.AdminToolFrame_jMenuItemLoadXml_actionAdapter** Helper inner class created automatically by development tool.
(package private) class	**AdminToolFrame.AdminToolFrame_jMenuItemLoadXML_jdomTransform_actionAdapter** Helper inner class created automatically by development tool.

(package private) class	**AdminToolFrame.AdminToolFrame_jMenuItemLoadXML_XSLT_actionAdapter** Helper inner class created automatically by development tool.
(package private) class	**AdminToolFrame.AdminToolFrame_jMenuItemRemoveAll_actionAdapter** Helper inner class created automatically by development tool.
(package private) class	**AdminToolFrame.AdminToolFrame_jMenuItemRemoveAssoc_actionAdapter** Helper inner class created automatically by development tool.
(package private) class	**AdminToolFrame.AdminToolFrame_jMenuItemRemoveFile_actionAdapter** Helper inner class created automatically by development tool.
(package private) class	**AdminToolFrame.AdminToolFrame_jMenuItemReport_actionAdapter** Helper inner class created automatically by development tool.
(package private) class	**AdminToolFrame.AdminToolFrame_jMenuItemSaveDataSrcConfig_actionAdapter** Helper inner class created automatically by development tool.
(package private) class	**AdminToolFrame.AdminToolFrame_jMenuItemSaveDataStruct_actionAdapter** Helper inner class created automatically by development tool.
(package private) class	**AdminToolFrame.AdminToolFrame_jMenuItemSelectedXML_actionAdapter** Helper inner class created automatically by development tool.
private class	**AdminToolFrame.GUIUpdater** GUIUpdater is s class responsible for udating the menus items and buttons of AdminToolFrame.
private class	**AdminToolFrame.PopupListener** Listens for mouse events and when right click occurs, it shows the popup menu.
(package private) class	**AdminToolFrame.TreeModelAdapter** For expanding automatically the inserted nodes after Drag and Drop.
(package private) class	**AdminToolFrame.TreeSelectionHandler** It calculates the XPath expression of the selected node of the tree.

Nested classes inherited from class javax.swing.JFrame
javax.swing.JFrame.AccessibleJFrame

Nested classes inherited from class java.awt.Frame
java.awt.Frame.AccessibleAWTFrame

Nested classes inherited from class java.awt.Window
java.awt.Window.AccessibleAWTWindow

Nested classes inherited from class java.awt.Container
java.awt.Container.AccessibleAWTContainer

Nested classes inherited from class java.awt.Component
java.awt.Component.AccessibleAWTComponent, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Field Summary

private javax.swing.ImageIcon	**aboutIcon** Icon used in corresponding button or menu item.
(package private) FilesFilter	**ascFileFilter** File filter for associations *.asc used in JFileChooser
(package private) java.lang.String	**associationName** The name of the association.
(package private) java.awt.BorderLayout	**borderLayout1**
(package private) java.awt.BorderLayout	**borderLayout2**
(package private) java.awt.BorderLayout	**borderLayout4**
(package private) java.awt.BorderLayout	**borderLayout5**
private javax.swing.ImageIcon	**closeAssocIcon** Icon used in corresponding button or menu item.
private javax.swing.ImageIcon	**configDSCIcon** Icon used in corresponding button or menu item.
(package private) javax.swing.JPanel	**contentPane**
(package private) org.jdom.Document	**dataSrcConfig** Document that holds the data sources configurations (*.dsc) that are

	loaded.
(package private) DataSrcTreeCellRenderer	**dataSrcTreeCellRenderer** Tree Cell Renderer of the Data Sstructure JTree.
(package private) org.jdom.Document	**dataStructDoc** Document that holds the data structures loaded
(package private) org.jdom.Element	**dataStructRoot** The root of the dataStructDoc
(package private) FilesFilter	**dscFileFilter** File filter for data source configurations files *.dsc used in JFileChooser
(package private) FilesFilter	**dstFileFilter** File filter for data structures files *.dst used in JFileChooser
private javax.swing.ImageIcon	**dstIcon** Icon used in corresponding button or menu item.
private javax.swing.ImageIcon	**exitIcon** Icon used in corresponding button or menu item.
private java.awt.Image	**frameImage** Frame icon.
(package private) java.awt.GridLayout	**gridLayout2**
(package private) AdminToolFrame.GUIUpdater	**guiUpdater** The GUIUpdater.
(package private) org.jdom.Document	**infoStructDoc** Document that holds the information structure loaded
(package private) java.io.File	**infoStructFile** The file for loading an informatin Structure
(package private) org.jdom.Element	**infoStructRoot** The root of the infoStructDoc
(package private) InfoStructTreeCellRenderer	**infoStructTreeCellRenderer** Tree Cell Renderer of the Information Structure JTree.
(package private) FilesFilter	**isFileFilter** File filter for information structures files *.is used in JFileChooser
(package private) javax.swing.JButton	**jButtonCloseAssoc** Toolbar | Close Association button
(package private) javax.swing.JButton	**jButtonConfigDSC**

	Toolbar	Configure Data Sources button
(package private) javax.swing.JButton	**jButtonLoadAssoc** Toolbar	Load Association button
(package private) javax.swing.JButton	**jButtonLoadDST** Toolbar	Load Data Structured File button
(package private) javax.swing.JButton	**jButtonLoadExistDSC** Toolbar	Load Existing Data Source Configuration button
(package private) javax.swing.JButton	**jButtonLoadIS** Toolbar	Load Informaiton Structure button
(package private) javax.swing.JButton	**jButtonLoadNewDSC** Toolbar	Load New Data Source Configuration button
(package private) javax.swing.JButton	**jButtonLoadXML** Toolbar	Generic XML button
(package private) javax.swing.JButton	**jButtonLoadXMLJDOM** Toolbar	Generic XML through JDOM Transformation button
(package private) javax.swing.JButton	**jButtonLoadXMLXSLT** Toolbar	Generic XML throough XSLT button
(package private) javax.swing.JButton	**jButtonNewAssoc** Toolbar	New Association button
(package private) javax.swing.JButton	**jButtonSaveAssoc** Toolbar	Save Association button
(package private) javax.swing.JButton	**jButtonSaveDSC** Toolbar	Save Data Source Configuration button
(package private) javax.swing.JButton	**jButtonSaveDST** Toolbar	Save Data Structured File button
private javax.swing.ImageIcon	**jdomXmlTransformIcon** Icon used in corresponding button or menu item.
(package private) javax.swing.JFileChooser	**jFileChooser1** The file chooser used.
(package private) javax.swing.JLabel	**jLabelDataSrc** Label of Data Structure tree
(package private) javax.swing.JLabel	**jLabelInfoStruct** Label of Information Structure tree
(package private) javax.swing.JScrollPane	**jLefttScrollPane**

(package private) javax.swing.JMenu	**jMenuAssoc** Association menu
(package private) javax.swing.JMenuBar	**jMenuBar1** Main menu bar
(package private) javax.swing.JMenu	**jMenuFile** File menu
(package private) javax.swing.JMenuItem	**jMenuFileExit** File | Exit menu item.
(package private) javax.swing.JMenu	**jMenuHelp** Help menu.
(package private) javax.swing.JMenuItem	**jMenuHelpAbout** Help | About menu item.
(package private) javax.swing.JMenuItem	**jMenuItemConfigDataSrc** Association | Configure Data Sources menu item.
(package private) javax.swing.JMenuItem	**jMenuItemFileCloseAssoc** File | Close Association menu item.
(package private) javax.swing.JMenuItem	**jMenuItemFileLoadAssoc** File | Load Association menu item.
(package private) javax.swing.JMenuItem	**jMenuItemFileNewAssoc** File | New Association menu item.
(package private) javax.swing.JMenuItem	**jMenuItemFileSaveAssoc** File | Save Association menu item.
(package private) javax.swing.JMenuItem	**jMenuItemInfoFieldProps** Pop up | Information Field Prperties menu item
(package private) javax.swing.JMenuItem	**jMenuItemInfoStructXML** File | View XML of Information Structure menu item.
(package private) javax.swing.JMenuItem	**jMenuItemLoadDataStruct** Association | Load Data File | Data Structure File menu item.
(package private) javax.swing.JMenuItem	**jMenuItemLoadExistDataSrcConfig** Association | Load Existing Data Source Configuration menu item.
(package private) javax.swing.JMenuItem	**jMenuItemLoadInfoStruct** Association | Load Information Structure menu item.
(package private) javax.swing.JMenuItem	**jMenuItemLoadNewDataSrcConfig** Association | LOad new Data Source Configuration menu item.
(package private) javax.swing.JMenuItem	**jMenuItemLoadXml**

	Association	Load Data File	Generic XML menu item.
(package private) javax.swing.JMenuItem	**jMenuItemLoadXML_jdomTransform** Association	Load Data File	Generic XML through JDOM Transformation menu item.
(package private) javax.swing.JMenuItem	**jMenuItemLoadXML_XSLT** Association	Load Data File	Generic XML through XSLT menu item.
(package private) javax.swing.JMenuItem	**jMenuItemRemoveAll** Pop up	Remove All menu item
(package private) javax.swing.JMenuItem	**jMenuItemRemoveAssoc** Pop up	Remove Association menu item
(package private) javax.swing.JMenuItem	**jMenuItemRemoveFile** Pop up	Remove File menu item
(package private) javax.swing.JMenuItem	**jMenuItemReport** File	View Report on Information Structure menu item.
(package private) javax.swing.JMenuItem	**jMenuItemSaveDataSrcConfig** Association	Save Data Source Configuration menu item.
(package private) javax.swing.JMenuItem	**jMenuItemSaveDataStruct** Association	Save Data Structure menu item.
(package private) javax.swing.JMenuItem	**jMenuItemSelectedXML** File	View XML of selection of Structure menu item.
(package private) javax.swing.JMenu	**jMenuLoadDataFile** Association	Load Data File menu.
(package private) javax.swing.JPopupMenu	**jPopupMenu1** Pop up menu.
(package private) javax.swing.JScrollPane	**jRightScrollPane**
(package private) javax.swing.JSplitPane	**jSplitPane**
(package private) javax.swing.JToolBar	**jToolBar** The Toolbar of the frame.
(package private) javax.swing.JTree	**jTree1** Data Structure tree
(package private) javax.swing.JTree	**jTree2** Information Structure tree
(package private) javax.swing.JPanel	**LeftPanel**

private javax.swing.ImageIcon	**loadAssocIcon** Icon used in corresponding button or menu item.
private javax.swing.ImageIcon	**loadDataFileIcon** Icon used in corresponding button or menu item.
private javax.swing.ImageIcon	**loadExistDSCIcon** Icon used in corresponding button or menu item.
private javax.swing.ImageIcon	**loadInfoStructIcon** Icon used in corresponding button or menu item.
private javax.swing.ImageIcon	**loadNewDSCIcon** Icon used in corresponding button or menu item.
(package private) javax.swing.JPanel	**mainPanel**
private javax.swing.ImageIcon	**newAssocIcon** Icon used in corresponding button or menu item.
private java.util.Map	**passwordMap** It holds the passwords
(package private) javax.swing.JPanel	**RightPanel**
private javax.swing.ImageIcon	**saveAssocIcon** Icon used in corresponding button or menu item.
private javax.swing.ImageIcon	**saveDSCIcon** Icon used in corresponding button or menu item.
private javax.swing.ImageIcon	**saveDSTIcon** Icon used in corresponding button or menu item.
(package private) java.io.File	**savefile** The file that the association is been saved.
(package private) javax.swing.tree.DefaultTreeSelectionModel	**selectionModel1** Selection model of the Data Sstructure JTree.
(package private) javax.swing.tree.DefaultTreeSelectionModel	**selectionModel2** Selection model of the Information Structure JTree.
(package private) javax.swing.JTextField	**statusBar**
(package private) javax.swing.JPanel	**ToolBarPanel**

(package private) JDOMTreeModel	**treeModel1** Tree model of the Data Structure JTree.
(package private) JDOMTreeModel	**treeModel2** Tree model of the Information Structure tree.
(package private) AdminToolFrame.TreeModelAdapter	**treeModelAdapter** Tree model listener implementation for listening for tree model changes
(package private) AdminToolFrame.TreeSelectionHandler	**treeSelectionHandler** Tree Selection Handler of the Data Sstructure JTree.
(package private) TreeTransferHandler	**treeTransferHandler** TransferHandler that provides support for Drag and Drop functionality between the Jtrees
private javax.swing.ImageIcon	**viewReportIcon** Icon used in corresponding button or menu item.
private javax.swing.ImageIcon	**viewXmlInfoStruct** Icon used in corresponding button or menu item.
private javax.swing.ImageIcon	**viewXmlSelection** Icon used in corresponding button or menu item.
(package private) FilesFilter	**xmlFileFilter** File filter for xml files *.xml used in JFileChooser
private javax.swing.ImageIcon	**xmlIcon** Icon used in corresponding button or menu item.
(package private) XPathEvaluator	**xPathEvaluator** For the evaluation of the XPaths
private javax.swing.ImageIcon	**xsltXmlIcon** Icon used in corresponding button or menu item.

Fields inherited from class javax.swing.JFrame
accessibleContext, EXIT_ON_CLOSE, rootPane, rootPaneCheckingEnabled

Fields inherited from class java.awt.Frame

CROSSHAIR_CURSOR, DEFAULT_CURSOR, E_RESIZE_CURSOR, HAND_CURSOR, ICONIFIED, MAXIMIZED_BOTH, MAXIMIZED_HORIZ, MAXIMIZED_VERT, MOVE_CURSOR, N_RESIZE_CURSOR, NE_RESIZE_CURSOR, NORMAL, NW_RESIZE_CURSOR, S_RESIZE_CURSOR, SE_RESIZE_CURSOR, SW_RESIZE_CURSOR, TEXT_CURSOR, W_RESIZE_CURSOR, WAIT_CURSOR

Fields inherited from class java.awt.Window

Fields inherited from class java.awt.Container

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface javax.swing.WindowConstants
DISPOSE_ON_CLOSE, DO_NOTHING_ON_CLOSE, HIDE_ON_CLOSE

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

AdminToolFrame() Construct the frame.

Method Summary

private void	**clearInfoStruct**() Removes all content of Information Structure tree.

void	**dataModified**() When a class (in our case JDOMTreeModel) modifies the data (in our case the Information structure tree), calls this method to inform that data are modified (and need to be saved before exiting).
void	**fileOutputXML**(org.jdom.Document doc, java.io.File file) Writes the JDOM document to a XML file.
private void	**initializePasswordMap**() Initializes the application with some passwords.
private void	**jbInit**() Set up GUI.
(package private) void	**jButtonCloseAssoc_actionPerformed**(java.awt.event.ActionEvent e) Toolbar | Close Associaion button action performed.
(package private) void	**jButtonConfigDSC_actionPerformed**(java.awt.event.ActionEvent e) Toolbar | Configure Data Sources button action performed.
(package private) void	**jButtonLoadAssoc_actionPerformed**(java.awt.event.ActionEvent e) Toolbar | Load Associaion button action performed.
(package private) void	**jButtonLoadDST_actionPerformed**(java.awt.event.ActionEvent e) Toolbar | Load Data Structure button action performed.
(package private) void	**jButtonLoadExistDSC_actionPerformed**(java.awt.event.ActionEvent e) Toolbar | Load Existing Data Source COnfiguration button action performed.
(package private) void	**jButtonLoadIS_actionPerformed**(java.awt.event.ActionEvent e) Toolbar | Load Information Structure button action performed.
(package private) void	**jButtonLoadNewDSC_actionPerformed**(java.awt.event.ActionEvent e) Toolbar | Load New Data Source Configuration button action performed.
(package private) void	**jButtonLoadXML_actionPerformed**(java.awt.event.ActionEvent e) Toolbar | Load a generic XML file button action performed.
(package private) void	**jButtonLoadXMLJDOM_actionPerformed**(java.awt.event.ActionEvent e) Toolbar | Load XML file through JDOM transformation button action performed.
(package private) void	**jButtonLoadXMLXSLT_actionPerformed**(java.awt.event.ActionEvent e) Toolbar | Load XML file through XSLT transformation button action performed.
(package private) void	**jButtonNewAssoc_actionPerformed**(java.awt.event.ActionEvent e) Toolbar | New Associaion button action performed.
(package private)	**jButtonSaveAssoc_actionPerformed**(java.awt.event.ActionEven

void	t e) Toolbar | Save Associaion button action performed.
(package private) void	**jButtonSaveDSC_actionPerformed**(java.awt.event.ActionEvent e) Toolbar | Save Data Source Configuration button action performed.
(package private) void	**jButtonSaveDST_actionPerformed**(java.awt.event.ActionEvent e) Toolbar | Save Data Structure file button action performed.
void	**jMenuFileExit_actionPerformed**(java.awt.event.ActionEvent e) File | Exit action performed.
void	**jMenuHelpAbout_actionPerformed**(java.awt.event.ActionEvent e) Help | About action performed.
(package private) void	**jMenuItemConfigDataSrc_actionPerformed**(java.awt.event.ActionEvent e) Association | Configure Data Sources action performed.
(package private) void	**jMenuItemFileCloseAssoc_actionPerformed**(java.awt.event.ActionEvent e) File | Close Association action performed.
(package private) void	**jMenuItemFileLoadAssoc_actionPerformed**(java.awt.event.ActionEvent e) File | Load Association action performed.
(package private) void	**jMenuItemFileNewAssoc_actionPerformed**(java.awt.event.ActionEvent e) File | New Association action performed.
(package private) void	**jMenuItemFileSaveAssoc_actionPerformed**(java.awt.event.ActionEvent e) File | Save Association action performed.
(package private) void	**jMenuItemInfoFieldProps_actionPerformed**(java.awt.event.ActionEvent e) Pop up | Information Field Properties action performed.
(package private) void	**jMenuItemInfoStructXML_actionPerformed**(java.awt.event.ActionEvent e) File | View XML of Information Structure action performed.
(package private) void	**jMenuItemLoadDataStruct_actionPerformed**(java.awt.event.ActionEvent e) Association | Load Data File | Data Structure File action performed.
(package private) void	**jMenuItemLoadExistDataSrcConfig_actionPerformed**(java.awt.event.ActionEvent e) Association | Load Existing Data Source Configuration action performed.
(package private) void	**jMenuItemLoadInfoStruct_actionPerformed**(java.awt.event.ActionEvent e) Association | Load Information Structure action performed.
(package private) void	**jMenuItemLoadNewDataSrcConfig_actionPerformed**(java.awt.event.ActionEvent e)

	Association	New Data Source Configuration action performed.
(package private) void	**jMenuItemLoadXml_actionPerformed**(java.awt.event.ActionEvent e) Association	Load Data File	Generic XML action performed.
(package private) void	**jMenuItemLoadXML_jdomTransform_actionPerformed**(java.awt.event.ActionEvent e) Association	Load Data File	Generic XML through JDOM Transformation action performed.
(package private) void	**jMenuItemLoadXML_XSLT_actionPerformed**(java.awt.event.ActionEvent e) Association	Load Data File	Generic XML through XSLT action performed.
(package private) void	**jMenuItemRemoveAll_actionPerformed**(java.awt.event.ActionEvent e) Pop up	Remove All action performed.
(package private) void	**jMenuItemRemoveAssoc_actionPerformed**(java.awt.event.ActionEvent e) Pop up	RemoveAssociation action performed.
(package private) void	**jMenuItemRemoveFile_actionPerformed**(java.awt.event.ActionEvent e) Pop up	Remove File action performed.
(package private) void	**jMenuItemReport_actionPerformed**(java.awt.event.ActionEvent e) File	View Report on Information Structure action performed.
(package private) void	**jMenuItemSaveDataSrcConfig_actionPerformed**(java.awt.event.ActionEvent e) Association	Save Data Source Configuration action performed.
(package private) void	**jMenuItemSaveDataStruct_actionPerformed**(java.awt.event.ActionEvent e) Association	Save Data Structure action performed.
(package private) void	**jMenuItemSelectedXML_actionPerformed**(java.awt.event.ActionEvent e) File	View XML of selection in Data Structure action performed.
private void	**loadDataSourceConfig**(org.jdom.Document dataSrcConfig) Loads a Data Source Cofiguration.
private void	**loadDataStruct**(org.jdom.Document loadedDoc) Loads a document to the Data Structure tree.
(package private) void	**loadDataStruct**(org.jdom.Document loadedDoc, java.io.File file) Loads a generic XML document to the Data Structure.
private void	**loadInfoStruct**(org.jdom.Document doc) Loads the document to the Information structure.
org.jdom.Document	**loadXML**(java.io.File file) Loads an XML from a file to a JDOM Document.
protected void	**processWindowEvent**(java.awt.event.WindowEvent e) Overridden so we can exit when window is closed and when no Association exists.

(package private) void	**setDataSrcConfigDocument**(org.jdom.Document doc) Sets the Data source configuration document dataSrcConfig.
void	**updateDataStructure**(org.jdom.Document doc) Updates the data sources loaded on the Data Structure tree.
private void	**updateJTreeGUI**(javax.swing.JTree jTree) Refreshes the view of the JTree.

Methods inherited from class javax.swing.JFrame
addImpl, createRootPane, frameInit, getAccessibleContext, getContentPane, getDefaultCloseOperation, getGlassPane, getJMenuBar, getLayeredPane, getRootPane, isDefaultLookAndFeelDecorated, isRootPaneCheckingEnabled, paramString, remove, setContentPane, setDefaultCloseOperation, setDefaultLookAndFeelDecorated, setGlassPane, setJMenuBar, setLayeredPane, setLayout, setRootPane, setRootPaneCheckingEnabled, update

Methods inherited from class java.awt.Frame
addNotify, finalize, getCursorType, getExtendedState, getFrames, getIconImage, getMaximizedBounds, getMenuBar, getState, getTitle, isResizable, isUndecorated, remove, removeNotify, setCursor, setExtendedState, setIconImage, setMaximizedBounds, setMenuBar, setResizable, setState, setTitle, setUndecorated

Methods inherited from class java.awt.Window
addPropertyChangeListener, addPropertyChangeListener, addWindowFocusListener, addWindowListener, addWindowStateListener, applyResourceBundle, applyResourceBundle, createBufferStrategy, createBufferStrategy, dispose, getBufferStrategy, getFocusableWindowState, getFocusCycleRootAncestor, getFocusOwner, getFocusTraversalKeys, getGraphicsConfiguration, getInputContext, getListeners, getLocale, getMostRecentFocusOwner, getOwnedWindows, getOwner, getToolkit, getWarningString, getWindowFocusListeners, getWindowListeners, getWindowStateListeners, hide, isActive, isFocusableWindow, isFocusCycleRoot, isFocused, isShowing, pack, postEvent, processEvent, processWindowFocusEvent, processWindowStateEvent, removeWindowFocusListener, removeWindowListener, removeWindowStateListener, setCursor, setFocusableWindowState, setFocusCycleRoot, setLocationRelativeTo, show, toBack, toFront

Methods inherited from class java.awt.Container
add, add, add, add, add, addContainerListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY,

getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getContainerListeners, getFocusTraversalPolicy, getInsets, getLayout, getMaximumSize, getMinimumSize, getPreferredSize, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paint, paintComponents, preferredSize, print, printComponents, processContainerEvent, remove, removeAll, removeContainerListener, setFocusTraversalKeys, setFocusTraversalPolicy, setFont, transferFocusBackward, transferFocusDownCycle, validate, validateTree

Methods inherited from class java.awt.Component

action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, coalesceEvents, contains, contains, createImage, createImage, createVolatileImage, createVolatileImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, getBackground, getBounds, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphics, getHeight, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocation, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isOpaque, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, prepareImage, prepareImage, printAll, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processKeyEvent, processMouseEvent, processMouseMotionEvent, processMouseWheelEvent, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, reshape, resize, resize, setBackground, setBounds, setBounds, setComponentOrientation, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeysEnabled, setForeground, setIgnoreRepaint, setLocale, setLocation, setLocation, setName, setSize, setSize, setVisible, show, size, toString, transferFocus, transferFocusUpCycle

Methods inherited from class java.lang.Object

clone, equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Methods inherited from interface java.awt.MenuContainer
getFont, postEvent

Field Detail

aboutIcon
`private javax.swing.ImageIcon` **`aboutIcon`**

Icon used in corresponding button or menu item.

---

ascFileFilter
`FilesFilter` **`ascFileFilter`**

File filter for associations *.asc used in JFileChooser

---

associationName
`java.lang.String` **`associationName`**

The name of the association.
It holds the name of the association provided when New Association is choosen, so as to use the given name on save.
It is also been used as a flag. When it is null, then a New Association is not yet created or loaded.

---

borderLayout1
`java.awt.BorderLayout` **`borderLayout1`**

---

borderLayout2
`java.awt.BorderLayout` **`borderLayout2`**

---

borderLayout4
`java.awt.BorderLayout` **`borderLayout4`**

---

borderLayout5
`java.awt.BorderLayout` **`borderLayout5`**

---

closeAssocIcon
`private javax.swing.ImageIcon` **`closeAssocIcon`**

Icon used in corresponding button or menu item.

---

configDSCIcon
`private javax.swing.ImageIcon` **`configDSCIcon`**

Icon used in corresponding button or menu item.

---

contentPane
`javax.swing.JPanel` **`contentPane`**

---

dataSrcConfig
`org.jdom.Document` **`dataSrcConfig`**

Document that holds the data sources configurations (*.dsc) that are loaded.
Data source configuration are loaded as data structures in the data structure tree

---

dataSrcTreeCellRenderer
`DataSrcTreeCellRenderer` **`dataSrcTreeCellRenderer`**

Tree Cell Renderer of the Data Sstructure JTree.

---

dataStructDoc
`org.jdom.Document` **`dataStructDoc`**

Document that holds the data structures loaded

---

dataStructRoot
`org.jdom.Element` **`dataStructRoot`**

The root of the dataStructDoc

---

dscFileFilter
`FilesFilter` **`dscFileFilter`**

File filter for data source configurations files *.dsc used in JFileChooser

---

dstFileFilter
`FilesFilter` **`dstFileFilter`**

File filter for data structures files *.dst used in JFileChooser

---

dstIcon
`private javax.swing.ImageIcon` **`dstIcon`**

Icon used in corresponding button or menu item.

---

exitIcon
`private javax.swing.ImageIcon` **`exitIcon`**

Icon used in corresponding button or menu item.

frameImage
`private java.awt.Image` **`frameImage`**

Frame icon.

---

gridLayout2
`java.awt.GridLayout` **`gridLayout2`**

---

guiUpdater
`AdminToolFrame.GUIUpdater` **`guiUpdater`**

The GUIUpdater.

---

infoStructDoc
`org.jdom.Document` **`infoStructDoc`**

Document that holds the information structure loaded

---

infoStructFile
`java.io.File` **`infoStructFile`**

The file for loading an informatin Structure

---

infoStructRoot
`org.jdom.Element` **`infoStructRoot`**

The root of the infoStructDoc

---

infoStructTreeCellRenderer
`InfoStructTreeCellRenderer` **`infoStructTreeCellRenderer`**

Tree Cell Renderer of the Information Structure JTree.

---

isFileFilter
`FilesFilter` **`isFileFilter`**

File filter for information structures files *.is used in JFileChooser

---

jButtonCloseAssoc
`javax.swing.JButton` **`jButtonCloseAssoc`**

Toolbar | Close Association button

---

jButtonConfigDSC
`javax.swing.JButton` **`jButtonConfigDSC`**

Toolbar | Configure Data Sources button

---

jButtonLoadAssoc
`javax.swing.JButton` **`jButtonLoadAssoc`**

Toolbar | Load Association button

---

jButtonLoadDST
`javax.swing.JButton` **`jButtonLoadDST`**

Toolbar | Load Data Structured File button

---

jButtonLoadExistDSC
`javax.swing.JButton` **`jButtonLoadExistDSC`**

Toolbar | Load Existing Data Source Configuration button

---

jButtonLoadIS
`javax.swing.JButton` **`jButtonLoadIS`**

Toolbar | Load Informaiton Structure button

---

jButtonLoadNewDSC
`javax.swing.JButton` **`jButtonLoadNewDSC`**

Toolbar | Load New Data Source Configuration button

---

jButtonLoadXML
`javax.swing.JButton` **`jButtonLoadXML`**

Toolbar | Generic XML button

---

jButtonLoadXMLJDOM
`javax.swing.JButton` **`jButtonLoadXMLJDOM`**

Toolbar | Generic XML through JDOM Transformation button

---

jButtonLoadXMLXSLT
`javax.swing.JButton` **`jButtonLoadXMLXSLT`**

Toolbar | Generic XML throough XSLT button

---

jButtonNewAssoc
`javax.swing.JButton` **`jButtonNewAssoc`**

Toolbar | New Association button

---

jButtonSaveAssoc
`javax.swing.JButton` **`jButtonSaveAssoc`**

Toolbar | Save Association button

---

jButtonSaveDSC
`javax.swing.JButton` **`jButtonSaveDSC`**

Toolbar | Save Data Source Configuration button

---

jButtonSaveDST
`javax.swing.JButton` **`jButtonSaveDST`**

Toolbar | Save Data Structured File button

---

jdomXmlTransformIcon
`private javax.swing.ImageIcon` **`jdomXmlTransformIcon`**

Icon used in corresponding button or menu item.

---

jFileChooser1
`javax.swing.JFileChooser` **`jFileChooser1`**

The file chooser used.
It was consider better to use one filechooser, which is set, used and reset, that to user more than one. *Setting* a filechooser refers to setting title, file filter, selected file etc.

---

jLabelDataSrc
`javax.swing.JLabel` **`jLabelDataSrc`**

Label of Data Structure tree

---

jLabelInfoStruct
`javax.swing.JLabel` **`jLabelInfoStruct`**

Label of Information Structure tree

---

jLefttScrollPane
`javax.swing.JScrollPane` **`jLefttScrollPane`**

---

jMenuAssoc
`javax.swing.JMenu` **`jMenuAssoc`**

Association menu

---

jMenuBar1
`javax.swing.JMenuBar` **`jMenuBar1`**

Main menu bar

---

jMenuFile
`javax.swing.JMenu` **`jMenuFile`**

File menu

---

jMenuFileExit
`javax.swing.JMenuItem` **`jMenuFileExit`**

File | Exit menu item.

---

jMenuHelp
`javax.swing.JMenu` **`jMenuHelp`**

Help menu.

---

jMenuHelpAbout
`javax.swing.JMenuItem` **`jMenuHelpAbout`**

Help | About menu item.

---

jMenuItemConfigDataSrc
`javax.swing.JMenuItem` **`jMenuItemConfigDataSrc`**

Association | Configure Data Sources menu item.

---

jMenuItemFileCloseAssoc
`javax.swing.JMenuItem` **`jMenuItemFileCloseAssoc`**

File | Close Association menu item.

---

jMenuItemFileLoadAssoc
`javax.swing.JMenuItem` **`jMenuItemFileLoadAssoc`**

File | Load Association menu item.

---

jMenuItemFileNewAssoc
`javax.swing.JMenuItem` **`jMenuItemFileNewAssoc`**

File | New Association menu item.

---

jMenuItemFileSaveAssoc
`javax.swing.JMenuItem` **`jMenuItemFileSaveAssoc`**

File | Save Association menu item.

---

jMenuItemInfoFieldProps
`javax.swing.JMenuItem` **`jMenuItemInfoFieldProps`**

Pop up | Information Field Prperties menu item

---

jMenuItemInfoStructXML
`javax.swing.JMenuItem` **`jMenuItemInfoStructXML`**

File | View XML of Information Structure menu item.

---

jMenuItemLoadDataStruct
`javax.swing.JMenuItem` **`jMenuItemLoadDataStruct`**

Association | Load Data File | Data Structure File menu item.

---

jMenuItemLoadExistDataSrcConfig
`javax.swing.JMenuItem` **`jMenuItemLoadExistDataSrcConfig`**

Association | Load Existing Data Source Configuration menu item.

---

jMenuItemLoadInfoStruct
`javax.swing.JMenuItem` **`jMenuItemLoadInfoStruct`**

Association | Load Information Structure menu item.

---

jMenuItemLoadNewDataSrcConfig
`javax.swing.JMenuItem` **`jMenuItemLoadNewDataSrcConfig`**

Association | LOad new Data Source Configuration menu item.

---

jMenuItemLoadXml
`javax.swing.JMenuItem` **`jMenuItemLoadXml`**

Association | Load Data File | Generic XML menu item.

---

jMenuItemLoadXML_jdomTransform
`javax.swing.JMenuItem` **`jMenuItemLoadXML_jdomTransform`**

Association | Load Data File | Generic XML through JDOM Transformation menu item.

---

jMenuItemLoadXML_XSLT
`javax.swing.JMenuItem` **`jMenuItemLoadXML_XSLT`**

Association | Load Data File | Generic XML through XSLT menu item.

---

jMenuItemRemoveAll
`javax.swing.JMenuItem` **`jMenuItemRemoveAll`**

Pop up | Remove All menu item

jMenuItemRemoveAssoc
`javax.swing.JMenuItem` **`jMenuItemRemoveAssoc`**

Pop up | Remove Association menu item

---

jMenuItemRemoveFile
`javax.swing.JMenuItem` **`jMenuItemRemoveFile`**

Pop up | Remove File menu item

---

jMenuItemReport
`javax.swing.JMenuItem` **`jMenuItemReport`**

File | View Report on Information Structure menu item.

---

jMenuItemSaveDataSrcConfig
`javax.swing.JMenuItem` **`jMenuItemSaveDataSrcConfig`**

Association | Save Data Source Configuration menu item.

---

jMenuItemSaveDataStruct
`javax.swing.JMenuItem` **`jMenuItemSaveDataStruct`**

Association | Save Data Structure menu item.

---

jMenuItemSelectedXML
`javax.swing.JMenuItem` **`jMenuItemSelectedXML`**

File | View XML of selection of Structure menu item.

---

jMenuLoadDataFile
`javax.swing.JMenu` **`jMenuLoadDataFile`**

Association | Load Data File menu.

---

jPopupMenu1
`javax.swing.JPopupMenu` **`jPopupMenu1`**

Pop up menu.

---

jRightScrollPane
`javax.swing.JScrollPane` **`jRightScrollPane`**

---

jSplitPane
`javax.swing.JSplitPane` **`jSplitPane`**

jToolBar
`javax.swing.JToolBar` **`jToolBar`**

The Toolbar of the frame.

---

jTree1
`javax.swing.JTree` **`jTree1`**

Data Structure tree

---

jTree2
`javax.swing.JTree` **`jTree2`**

Information Structure tree

LeftPanel
`javax.swing.JPanel` **`LeftPanel`**

---

loadAssocIcon
`private javax.swing.ImageIcon` **`loadAssocIcon`**

Icon used in corresponding button or menu item.

---

loadDataFileIcon
`private javax.swing.ImageIcon` **`loadDataFileIcon`**

Icon used in corresponding button or menu item.

---

loadExistDSCIcon
`private javax.swing.ImageIcon` **`loadExistDSCIcon`**

Icon used in corresponding button or menu item.

---

loadInfoStructIcon
`private javax.swing.ImageIcon` **`loadInfoStructIcon`**

Icon used in corresponding button or menu item.

---

loadNewDSCIcon
`private javax.swing.ImageIcon` **`loadNewDSCIcon`**

Icon used in corresponding button or menu item.

---

mainPanel
`javax.swing.JPanel` **`mainPanel`**

newAssocIcon
`private javax.swing.ImageIcon` **`newAssocIcon`**

Icon used in corresponding button or menu item.

---

passwordMap
`private java.util.Map` **`passwordMap`**

It holds the passwords

RightPanel
`javax.swing.JPanel` **`RightPanel`**

---

saveAssocIcon
`private javax.swing.ImageIcon` **`saveAssocIcon`**

Icon used in corresponding button or menu item.

---

saveDSCIcon
`private javax.swing.ImageIcon` **`saveDSCIcon`**

Icon used in corresponding button or menu item.

---

saveDSTIcon
`private javax.swing.ImageIcon` **`saveDSTIcon`**

Icon used in corresponding button or menu item.

---

savefile
`java.io.File` **`savefile`**

The file that the association is been saved.
It is also been used as a flag. When it is null, then the association is not yet yet been saved. The next time time that is going to be saved, it will be the first time.
See Also:
`jMenuItemFileSaveAssoc_actionPerformed(ActionEvent)`

---

selectionModel1
`javax.swing.tree.DefaultTreeSelectionModel` **`selectionModel1`**

Selection model of the Data Sstructure JTree.

---

selectionModel2
`javax.swing.tree.DefaultTreeSelectionModel` **`selectionModel2`**

Selection model of the Information Structure JTree.

---

statusBar
`javax.swing.JTextField` **`statusBar`**

ToolBarPanel
`javax.swing.JPanel` **`ToolBarPanel`**

---

treeModel1
`JDOMTreeModel` **`treeModel1`**

Tree model of the Data Structure JTree.

---

treeModel2
`JDOMTreeModel` **`treeModel2`**

Tree model of the Information Structure tree.

---

treeModelAdapter
`AdminToolFrame.TreeModelAdapter` **`treeModelAdapter`**

Tree model listener implementation for listening for tree model changes

---

treeSelectionHandler
`AdminToolFrame.TreeSelectionHandler` **`treeSelectionHandler`**

Tree Selection Handler of the Data Sstructure JTree.

---

treeTransferHandler
`TreeTransferHandler` **`treeTransferHandler`**

TransferHandler that provides support for Drag and Drop functionality between the Jtrees

---

viewReportIcon
`private javax.swing.ImageIcon` **`viewReportIcon`**

Icon used in corresponding button or menu item.

---

viewXmlInfoStruct
`private javax.swing.ImageIcon` **`viewXmlInfoStruct`**

Icon used in corresponding button or menu item.

---

viewXmlSelection
`private javax.swing.ImageIcon` **`viewXmlSelection`**

Icon used in corresponding button or menu item.

---

xmlFileFilter
`FilesFilter` **`xmlFileFilter`**

File filter for xml files *.xml used in JFileChooser

---

xmlIcon
`private javax.swing.ImageIcon` **`xmlIcon`**

Icon used in corresponding button or menu item.

---

xPathEvaluator
`XPathEvaluator` **`xPathEvaluator`**

For the evaluation of the XPaths

---

xsltXmlIcon
`private javax.swing.ImageIcon` **`xsltXmlIcon`**

Icon used in corresponding button or menu item.

Constructor Detail

AdminToolFrame
`public` **`AdminToolFrame`**`()`

Construct the frame. It initialize variables, loads the images for the gui, calls jbInit to set up the gui and initializes the guiupdater. If the image files are not found an exception will be thrown and the application will not start. The way images are loaded (e.g. * new ImageIcon(AdminToolFrame.class.getResource("xmlJdom.gif"));) guarantees that if the application starts all the images are found and loaded.

Method Detail

clearInfoStruct
`private void` **`clearInfoStruct`**`()`

Removes all content of Information Structure tree.

- Remove everithing from Information structure tree, by delegating the the remove action to its tree model. That means calling JDOMTreeModel's removeAll method.
- Remove type attribute

See Also:
`JDOMTreeModel.removeAll(String)`

---

dataModified
`public final void` **`dataModified`**`()`

When a class (in our case JDOMTreeModel) modifies the data (in our case the Information structure tree), calls this method to inform that data are modified (and need to be saved before exiting).

The method just delegates the update of the status of data to the GUIUpdater. Through the implementation of this method the DataModifier interface is implemented.

Specified by:

`dataModified` in interface `DataModifier`

See Also:

`DataModifier`, `AdminToolFrame.GUIUpdater`

---

fileOutputXML

```
public void fileOutputXML(org.jdom.Document doc,
                          java.io.File file)
```

Writes the JDOM document to a XML file.

- Creatse a XMLOutputter which uses a PrettyFormat.
- Outputs document to the file

  This is the only method of the application that outputs XML documents to file. It is declared public because it is also used by ReportViewer class.

Parameters:

`doc` - - the document to be written to file.

`file` - - the file in which the document will be written.

See Also:

`org.jdom.output.Format.getPrettyFormat()`

---

initializePasswordMap

```
private void initializePasswordMap()
```

Initializes the application with some passwords. key=login, value=password

---

jbInit

```
private void jbInit()
             throws java.lang.Exception
```

Set up GUI.

Sets the components' properties and lays them out.

Most of this code is automatically generated by JBuilder. Especially the laying out of the components.

This is the JBuilders method for setting up the GUI. It is necessary to use this method it order to be able to modify the GUI from JBuilder's Design view.

Throws:

`java.lang.Exception`

---

jButtonCloseAssoc_actionPerformed

```
void jButtonCloseAssoc_actionPerformed(java.awt.event.ActionEvent e)
```

Toolbar | Close Associaion button action performed.

Delegates the call to the corresponding menu item's action performed method.

Parameters:

`e` - ActionEvent

See Also:
jMenuItemFileCloseAssoc_actionPerformed(ActionEvent)

---

jButtonConfigDSC_actionPerformed

`void jButtonConfigDSC_actionPerformed(java.awt.event.ActionEvent e)`

Toolbar | Configure Data Sources button action performed.
Delegates the call to the corresponding menu item's action performed method.

Parameters:
e - ActionEvent

See Also:
jMenuItemConfigDataSrc_actionPerformed(ActionEvent)

---

jButtonLoadAssoc_actionPerformed

`void jButtonLoadAssoc_actionPerformed(java.awt.event.ActionEvent e)`

Toolbar | Load Associaion button action performed.
Delegates the call to the corresponding menu item's action performed method.

Parameters:
e - ActionEvent

See Also:
jMenuItemFileLoadAssoc_actionPerformed(ActionEvent)

---

jButtonLoadDST_actionPerformed

`void jButtonLoadDST_actionPerformed(java.awt.event.ActionEvent e)`

Toolbar | Load Data Structure button action performed.
Delegates the call to the corresponding menu item's action performed method.

Parameters:
e - ActionEvent

See Also:
jMenuItemLoadDataStruct_actionPerformed(ActionEvent)

---

jButtonLoadExistDSC_actionPerformed

`void jButtonLoadExistDSC_actionPerformed(java.awt.event.ActionEvent e)`

Toolbar | Load Existing Data Source COnfiguration button action performed.
Delegates the call to the corresponding menu item's action performed method.

Parameters:
e - ActionEvent

See Also:
jMenuItemLoadExistDataSrcConfig_actionPerformed(ActionEvent e)

---

jButtonLoadIS_actionPerformed

`void jButtonLoadIS_actionPerformed(java.awt.event.ActionEvent e)`

Toolbar | Load Information Structure button action performed.
Delegates the call to the corresponding menu item's action performed method.

Parameters:
e - ActionEvent

See Also:
jMenuItemLoadInfoStruct_actionPerformed(ActionEvent)

---

jButtonLoadNewDSC_actionPerformed

`void jButtonLoadNewDSC_actionPerformed(java.awt.event.ActionEvent e)`

Toolbar | Load New Data Source Configuration button action performed.
Delegates the call to the corresponding menu item's action performed method.

Parameters:
e - ActionEvent

See Also:
jMenuItemLoadNewDataSrcConfig_actionPerformed(ActionEvent)

---

jButtonLoadXML_actionPerformed

`void jButtonLoadXML_actionPerformed(java.awt.event.ActionEvent e)`

Toolbar | Load a generic XML file button action performed.
Delegates the call to the corresponding menu item's action performed method.

Parameters:
e - ActionEvent

See Also:
jMenuItemLoadXml_actionPerformed(ActionEvent)

---

jButtonLoadXMLJDOM_actionPerformed

`void jButtonLoadXMLJDOM_actionPerformed(java.awt.event.ActionEvent e)`

Toolbar | Load XML file through JDOM transformation button action performed.
Delegates the call to the corresponding menu item's action performed method.

Parameters:
e - ActionEvent

See Also:
jMenuItemLoadXML_jdomTransform_actionPerformed(ActionEvent)

---

jButtonLoadXMLXSLT_actionPerformed

`void jButtonLoadXMLXSLT_actionPerformed(java.awt.event.ActionEvent e)`

Toolbar | Load XML file through XSLT transformation button action performed.
Delegates the call to the corresponding menu item's action performed method.

Parameters:
e - ActionEvent

See Also:
jMenuItemLoadXML_XSLT_actionPerformed(ActionEvent)

---

jButtonNewAssoc_actionPerformed

`void jButtonNewAssoc_actionPerformed(java.awt.event.ActionEvent e)`

Toolbar | New Associaion button action performed.
Delegates the call to the corresponding menu item's action performed method.

Parameters:
e - ActionEvent

See Also:
jMenuItemFileNewAssoc_actionPerformed(ActionEvent)

---

jButtonSaveAssoc_actionPerformed

```
void jButtonSaveAssoc_actionPerformed(java.awt.event.ActionEvent e)
```

Toolbar | Save Associaion button action performed.
Delegates the call to the corresponding menu item's action performed method.

Parameters:
e - ActionEvent

See Also:
jMenuItemFileSaveAssoc_actionPerformed(ActionEvent)

---

jButtonSaveDSC_actionPerformed

```
void jButtonSaveDSC_actionPerformed(java.awt.event.ActionEvent e)
```

Toolbar | Save Data Source Configuration button action performed.
Delegates the call to the corresponding menu item's action performed method.

Parameters:
e - ActionEvent

See Also:
jMenuItemSaveDataSrcConfig_actionPerformed(ActionEvent)

---

jButtonSaveDST_actionPerformed

```
void jButtonSaveDST_actionPerformed(java.awt.event.ActionEvent e)
```

Toolbar | Save Data Structure file button action performed.
Delegates the call to the corresponding menu item's action performed method.

Parameters:
e - ActionEvent

See Also:
jMenuItemSaveDataStruct_actionPerformed(ActionEvent)

---

jMenuFileExit_actionPerformed

```
public void jMenuFileExit_actionPerformed(java.awt.event.ActionEvent e)
```

File | Exit action performed. Exits the application.
First checks to see if an assocoation exists (To protect user from losing data). If it exists shows warning message and cancels exiting. If no assocoation exists, then exits.

---

jMenuHelpAbout_actionPerformed

```
public void jMenuHelpAbout_actionPerformed(java.awt.event.ActionEvent e)
```

Help | About action performed. DIsplays the About Box.

---

jMenuItemConfigDataSrc_actionPerformed

```
void jMenuItemConfigDataSrc_actionPerformed(java.awt.event.ActionEvent e)
```

Association | Configure Data Sources action performed.
Calls the DataSourcesConfiguration Dialog.

Parameters:
e - ActionEvent
See Also:
DataSourcesConfiguration

---

jMenuItemFileCloseAssoc_actionPerformed

```
void jMenuItemFileCloseAssoc_actionPerformed(java.awt.event.ActionEvent e)
```

File | Close Association action performed.

Removes all content from information structure tree. But before removing, first checks to see if data is modified, if save is needed etc.
Removing all content means: clear information structure, update GUIUpdater and update JTree GUI. That means clear information structure from data (the actual removing of data), notify GUIUpdater taht data is cleared and refresh JTree appearance. Here the term *data* refers to the XML document heldd in Information structure tree.

- Data is modified, so prompt user to save data:
  - if cancel: Do nothing and exit method.
  - if no: Remove all content, re-intialize file variables
  - if yes: save, by delegating save action to File | Save Association action performed. Check again because user might hit yes or cancel on filechooser.
    - if user hit yes in file chooser: Remove all content, re-intialize file variables
    - if user hit no in file chooser: do nothing and exit
- Dara is not modified:
  - Remove all content. re-intialize file variables

File variables represented the state of the data.
Parameters:
e - ActionEvent
See Also:
jMenuItemFileSaveAssoc_actionPerformed(ActionEvent), clearInfoStruct(), AdminToolFrame.GUIUpdater, updateJTreeGUI(JTree)

---

jMenuItemFileLoadAssoc_actionPerformed

```
void jMenuItemFileLoadAssoc_actionPerformed(java.awt.event.ActionEvent e)
```

File | Load Association action performed.

Loads a association *.asc file.

- Use the OPEN version of the file chooser, to get file. Set up file chooser.
- Get file
- Set association name from file name
- Load XML file into a JDOM document, by calling loadXML method.
- Load document in Information structure, bya calling loadInfoStruct method.
- Update JTree, guiupdater and status bar
- Reset file chooser.

Parameters:
e - ActionEvent
See Also:
loadXML(File), loadInfoStruct(Document)

---

jMenuItemFileNewAssoc_actionPerformed

```
void jMenuItemFileNewAssoc_actionPerformed(java.awt.event.ActionEvent e)
```

File | New Association action performed.

Creates a new associaion. Asks user for the name of the new associon and with the input initializes the necessary variable.

- Ask user for association name.
- if user enter a appropriate name: initialize association name variable and update guiupdater
- if user enters an inappropriate name, that is an empty string, show error message.
- if user hits escape key, show error message.

An *appropriate* name is any string except "".
Parameters:
e - ActionEvent

---

jMenuItemFileSaveAssoc_actionPerformed

```
void jMenuItemFileSaveAssoc_actionPerformed(java.awt.event.ActionEvent e)
```

File | Save Association action performed.

Save cases:

- It is the first time that it is saved:
  - create a file with name = associationName and set filechooser with it. When user creates an new association, he/she is prompted to provide an Association name. That name is stored so as to be used here. But in order for the filechooser to prompt the user with the predifed association name, a file with that name must be created to set as selected file in the filechooser.
  - Use a fileChooser to choose the file name and location. Set up filechooser
  - < fileOutputXML calling by file as it>
  - update the guiupdater
  - if cancel on filechooser then restore savefile initial status. Because savefile == null indicates that the file is never saved, and thus the next time will be the first time that it will be saved.
  -
- it is been saved before so the file is already created
  - < fileOutputXML calling by file as it>
  - update the guiupdater

Parameters:

e - ActionEvent
See Also:
fileOutputXML(Document, File)

---

jMenuItemInfoFieldProps_actionPerformed

```
void jMenuItemInfoFieldProps_actionPerformed(java.awt.event.ActionEvent e)
```

Pop up | Information Field Properties action performed.

- Gets the selected object.
- Cases of object:
  - an information field element:
    - Get path of selection
    - Calls the InformationFieldProperties Dialog
  - not an information field: display relative message
  - not an Element: display relative message

Parameters:
e - ActionEvent

---

jMenuItemInfoStructXML_actionPerformed

```
void jMenuItemInfoStructXML_actionPerformed(java.awt.event.ActionEvent e)
```

File | View XML of Information Structure action performed.
Calls the ReportViewer Dialog.
Parameters:
e - ActionEvent
See Also:
ReportViewer

---

jMenuItemLoadDataStruct_actionPerformed

```
void jMenuItemLoadDataStruct_actionPerformed(java.awt.event.ActionEvent e)
```

Association | Load Data File | Data Structure File action performed.

Calls the JFileChooser for user to pick a file, and then calls the necessary methods to load the file as the information structure.
The steps taken are:

- Sets up the jFileChooser (set title anf file filter) and calls it for the user to choose a file.
- Loads the file into a JDOM Document with loadXML method.
- Loads the loaded Document to the data structure by calling method loadDatasStruct.
- updates the gui: updates related Jtree, statusbar, notifies GUIUpdater of the action that was performed
- removes the file filter used from jfilechooser

Parameters:
e - ActionEvent

See Also:
`loadXML(File)`, `loadDataStruct(Document)`

---

jMenuItemLoadExistDataSrcConfig_actionPerformed

```
void
jMenuItemLoadExistDataSrcConfig_actionPerformed(java.awt.event.ActionEvent e
)
```

Association | Load Existing Data Source Configuration action performed.

- Sets up the jFileChooser (set title anf file filter) and calls it for the user to choose a *.dsc file.
- Loads the file into a JDOM Document with loadXML method.
- Loads the data sources defined in the loaded Document to the data structure by calling method loadDataSourceConfig.
- Updates the data source configuration document `dataSrcConfig` of the data sources loaded
- Updates related Jtree
- Removes the file filter used from jfilechooser

Parameters:
`e` - ActionEvent

---

jMenuItemLoadInfoStruct_actionPerformed

```
void jMenuItemLoadInfoStruct_actionPerformed(java.awt.event.ActionEvent e)
```

Association | Load Information Structure action performed.

Calls the JFileChooser for user to pick a file, and then calls the necessary methods to load the file as the information structure.
The steps taken are:

- Sets up the jFileChooser (set title anf file filter) and calls it for the user to choose a file.
- Loads the file into a JDOM Document with loadXML method.
- Loads the loaded Document to the information structure by calling method loadInfoStruct.
- updates the gui: updates related Jtree, statusbar, notifies GUIUpdater of the action that was performed
- removes the file filter used from jfilechooser

Parameters:
`e` - ActionEvent

---

jMenuItemLoadNewDataSrcConfig_actionPerformed

```
void
jMenuItemLoadNewDataSrcConfig_actionPerformed(java.awt.event.ActionEvent e)
```

Association | New Data Source Configuration action performed.
Calls the DataSourcesConfiguration Dialog.

Parameters:
e - ActionEvent
See Also:
DataSourcesConfiguration

---

jMenuItemLoadXml_actionPerformed

```
void jMenuItemLoadXml_actionPerformed(java.awt.event.ActionEvent e)
```

Association | Load Data File | Generic XML action performed.

- Sets up the jFileChooser (set title anf file filter) and calls it for the user to choose a file.
- Loads the file into a JDOM Document with loadXML method.
- Loads the loaded Document to the data structure by calling method loadDatasStruct(Document, File), so as to put name and type to generic xml.
- updates the gui: updates related Jtree, statusbar, notifies GUIUpdater of the action that was performed
- removes the file filter used from jfilechooser

Parameters:
e - ActionEvent

---

jMenuItemLoadXML_jdomTransform_actionPerformed

```
void
jMenuItemLoadXML_jdomTransform_actionPerformed(java.awt.event.ActionEvent e)
```

Association | Load Data File | Generic XML through JDOM Transformation action performed.
Calls the XML_JDOMTransformLoader Dialog, loads the transformed document and update Jtree GUI.
Parameters:
e - ActionEvent
See Also:
XML_JDOMTransformLoader

---

jMenuItemLoadXML_XSLT_actionPerformed

```
void jMenuItemLoadXML_XSLT_actionPerformed(java.awt.event.ActionEvent e)
```

Association | Load Data File | Generic XML through XSLT action performed.
Calls the XML_XSLTLoader Dialog, loads the transformed document and update Jtree GUI.
Parameters:
e - ActionEvent
See Also:
XML_XSLTLoader

---

jMenuItemRemoveAll_actionPerformed

```
void jMenuItemRemoveAll_actionPerformed(java.awt.event.ActionEvent e)
```

Pop up | Remove All action performed. Removes all content from Data Structure tree.

- Clears data structure tree, by delegating the calls to tree model's removeAll method.
- Clears data source configuration document. It must be cleared to show that no data sources are loaded, because data structure tree is cleared from its loaded data sources too.

The reason for delegating the removal of a node to the tree model is the same as in in loadInfoStruct and loadDataStruct methods.
Parameters:
e - ActionEvent
See Also:
`JDOMTreeModel`, `loadInfoStruct(Document)`, `loadDataStruct(Document)`

---

jMenuItemRemoveAssoc_actionPerformed

```
void jMenuItemRemoveAssoc_actionPerformed(java.awt.event.ActionEvent e)
```

Pop up | RemoveAssociation action performed.

- Gets selection path required for callinf removeNodeFromParent method.
- Gets selected object.
- Checks to see if it is an association. Association are Text nodes in the JDOM document of Information structure.
- Delegate remove action to tree model removeFromParent method.
- Error cases: Not an association

The reason for delegating the removal of a node to the tree model is the same as in in loadInfoStruct and loadDataStruct methods.
Parameters:
e - ActionEvent
See Also:
`JDOMTreeModel`, `loadInfoStruct(Document)`, `loadDataStruct(Document)`

---

jMenuItemRemoveFile_actionPerformed

```
void jMenuItemRemoveFile_actionPerformed(java.awt.event.ActionEvent e)
```

Pop up | Remove File action performed.

- Gets the path of the selection used in `removeNodeFromParent` method.
- Gets selected object
- Cases of selected object:
  - is a dataStructureFile: Delegate removal to `JDOMTreeModel.removeNodeFromParent`.
  - Generic XML that means taht the selected Item has dataStructRoot as its Parent AND it is not a data source: Delegate removal to `JDOMTreeModel.removeNodeFromParent`.
  - Any other case, which at this version is a data source: show an error message.

The reason for delegating the removal of a node to the tree model is the same as in in loadInfoStruct and loadDataStruct methods.

Parameters:
e - ActionEvent
See Also:
JDOMTreeModel, loadInfoStruct(Document), loadDataStruct(Document)

---

jMenuItemReport_actionPerformed

```
void jMenuItemReport_actionPerformed(java.awt.event.ActionEvent e)
```

File | View Report on Information Structure action performed.
Calls ReportViewer Dialog
Parameters:
e - ActionEvent
See Also:
ReportViewer

---

jMenuItemSaveDataSrcConfig_actionPerformed

```
void
jMenuItemSaveDataSrcConfig_actionPerformed(java.awt.event.ActionEvent e)
```

Association | Save Data Source Configuration action performed.

- Set up file chooser.
- Get file from file chooser.
- Save data source configuration document to file by calling fileOutoutXML method.
- Reset file chooser.

Parameters:
e - ActionEvent
See Also:
fileOutputXML(Document, File)

---

jMenuItemSaveDataStruct_actionPerformed

```
void jMenuItemSaveDataStruct_actionPerformed(java.awt.event.ActionEvent e)
```

Association | Save Data Structure action performed.

Saves only data structured files loaded. Data sources displayed in the tree can not be saved as files.

- Gets the selection path and the selected object.
- Check the object to be an element. If it is:
  - Checks to see if parent of the selection has a name. If not user has probably selected the node "No Data Structures Loaded", so display message that no data structures are loaded and exit method.
  - Checks if it is a data source. If yes, show relative error message and exit method. Because data sources cannot be saved.
  - Check if the element is a direct child of the root of the data structure tree. If it is not, this element and its content does not represent a data structure but a subtree of the data structure. That is the element is somewhere

inside the content of the data structure and does not represent the root od the data structure. A subtree of the data structure cannot be saved. So, show error message and exit method.
    - If nothing of the above cases occured then it is a data structure:
        - Clone the content because the element cannot have two parents.
        - Put it in a dummy doc because fileOutputXML is called with a file and a document.
        - Set up file chooser to find the file that the document will be saved.
        - Get file from file chooser
        - Save document to file by calling fileOutputXML method
        - Reset file chooser
- Error cases: No Element is selected or No Selection

Parameters:
e - ActionEvent
See Also:
`fileOutputXML(Document, File)`

---

jMenuItemSelectedXML_actionPerformed

```
void jMenuItemSelectedXML_actionPerformed(java.awt.event.ActionEvent e)
```

File | View XML of selection in Data Structure action performed.

- Gets selection path.
- Gets selected object.
- If selection object is an Element.
    - Clone the selected element
    - Creates a dummy document with clone as its root.
    - Calls ReportViewer passing to it the dummy document. The *dummy document* is created because ReportViewer manipulates only documents and not elements.
- Error cases: when selected object is not an element or user has not selected anything.

Parameters:
e - ActionEvent
See Also:
`ReportViewer`

---

loadDataSourceConfig

```
private void loadDataSourceConfig(org.jdom.Document dataSrcConfig)
```

Loads a Data Source Cofiguration.
Takes a data source configuration Document, that is a *.dsc file that it is already loaded as a JDOM Document and loads it to the Data Structure tree.

More precisely the actions taken are:

- Get content of cofiguration document and acquire Data Source parameters from each configuraion node
- Retrieve Data Sources with the use of `DataSrcMapper`.
- Load data sources to data structure tree by calling `loadDataStruct` method.

Parameters:
`dataSrcConfig` - the data source configuration document.
See Also:
`DataSrcMapper`, `loadDataStruct(Document)`

---

loadDataStruct

```
private void loadDataStruct(org.jdom.Document loadedDoc)
```

Loads a document to the Data Structure tree.

- Builds the root path necessary for calling the JDOMTreeModel.insertNodeInto method
- Clones the content of the document
- Delegate insertion to JDOMTreeModel by calling insertNodeInto method

Loading a document into the Data Source tree is not so trivial. The loading must be performed by JDOMTreeModel in order to have coherence between the tree model and the JTree. But by delegating the insertion to JDOMTreeModel while looping the document here results in a cocurrent modification exception thrown by JDOM. So the content is cloned before insertion.

See Also:
`loadInfoStruct(Document)`

---

loadDataStruct

```
void loadDataStruct(org.jdom.Document loadedDoc,
                    java.io.File file)
```

Loads a generic XML document to the Data Structure.

When a generic XML is loaded, some information (like the name of the \ file and its type) must be added to its root in order to be able to be demostrated correctly from JTree. This is the use case of this method.

- Get file name
- Clone loaded document's root
- Set Attributes
- Call `loadDataStruct(Document)` to do the loading

Parameters:
`loadedDoc` - - loaded document
`file` - - the XML file

---

loadInfoStruct

`private void loadInfoStruct(org.jdom.Document doc)`

Loads the document to the Information structure.

- Builds the root path necessary for calling the JDOMTreeModel.insertNodeInto method
- Clones the content of the document
- Delegate insertion to JDOMTreeModel by calling insertNodeInto method
- Set Information Structure root's attribute

Insert must be delegated to JDOMTreeModel in order to have coherence between the Data hold in the tree model and what JTree shows. Otherwise, if we add the Content by using a addContent method of JDOM Element, then the content will be added but JTree will not know about the modification and so it will not show it.
But by delegating the insertion to JDOMTreeModel while looping the document here results in a cocurrent modification exception thrown by JDOM. So the content is cloned before insertion.

Parameters:
`doc` - Document - loaded document
See Also:
`loadDataStruct(Document)`

---

loadXML

`public org.jdom.Document loadXML(java.io.File file)`

Loads an XML from a file to a JDOM Document. This is the only method throughout the application that loads XML into a Document.
The XML filter is used in order to avoid creating ignorable whitespace `Text` nodes. See XMLWhitespaceFilter for more.

- Create builder
- Set builder's XML filter.
- Load document catching Exceptions

It is declared public because it is used be other classes too.
Parameters:
`file` - - the XML file to be loaded
Returns:
Document - the loaded XML
See Also:
`XMLWhitespaceFilter`

---

processWindowEvent

`protected void processWindowEvent(java.awt.event.WindowEvent e)`

Overridden so we can exit when window is closed and when no Association exists.
If window (that is the AdminToolFrame) is closed, then it delegates the closing action to File | Exit action performed mnethod.
See Also:

jMenuFileExit_actionPerformed(ActionEvent)

---

setDataSrcConfigDocument

`void setDataSrcConfigDocument(org.jdom.Document doc)`

Sets the Data source configuration document `dataSrcConfig`. Used by DataSourceConfiguration Diaog to restore the document to its inital state and so discard any modifications.

See Also:
DataSourcesConfiguration

---

updateDataStructure

`public void updateDataStructure(org.jdom.Document doc)`

Updates the data sources loaded on the Data Structure tree.

Called from DataSourceConfiguration Dialog. Update is performed by erasing all the datasources and write the data sources that the document passed by DataSourcesConfiguration has. This might mean to rewrite the old ones. That is:

- Remove all data sources from data Structure tree
- Load new the Data Src configuration
- Update Jtree GUI

Parameters:
`doc` - - the Data source configuration document passed by calling object.
See Also:
DataSourcesConfiguration

---

updateJTreeGUI

`private void updateJTreeGUI(javax.swing.JTree jTree)`

Refreshes the view of the JTree.

Refresh is needed because modifications in the tree model may not be shown be the Jtree yet. The update is done by collapsing and expanding the root of the JTree.

Parameters:
`jTree` - JTree - The tree to be updated

---

admintool

Class AdminToolFrame_AboutBox

```
java.lang.Object
  └java.awt.Component
      └java.awt.Container
          └java.awt.Window
              └java.awt.Dialog
                  └javax.swing.JDialog
                      └admintool.AdminToolFrame_AboutBox
```

All Implemented Interfaces:

javax.accessibility.Accessible, java.awt.event.ActionListener, java.util.EventListener, java.awt.image.ImageObserver, java.awt.MenuContainer, javax.swing.RootPaneContainer, java.io.Serializable, javax.swing.WindowConstants

---

public class **AdminToolFrame_AboutBox**
extends javax.swing.JDialog
implements java.awt.event.ActionListener

The About Dialog.

See Also:

Serialized Form

Nested Class Summary

Nested classes inherited from class javax.swing.JDialog
`javax.swing.JDialog.AccessibleJDialog`

Nested classes inherited from class java.awt.Dialog
`java.awt.Dialog.AccessibleAWTDialog`

Nested classes inherited from class java.awt.Window

java.awt.Window.AccessibleAWTWindow

Nested classes inherited from class java.awt.Container
java.awt.Container.AccessibleAWTContainer

Nested classes inherited from class java.awt.Component
java.awt.Component.AccessibleAWTComponent, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Field Summary

(package private) java.awt.BorderLayout	**borderLayout1**
(package private) java.awt.BorderLayout	**borderLayout2**
(package private) java.awt.BorderLayout	**borderLayout3**
(package private) java.awt.BorderLayout	**borderLayout4**
(package private) java.awt.BorderLayout	**borderLayout5**
(package private) javax.swing.JButton	**button1**
(package private) java.lang.String	**comments**
(package private) java.lang.String	**copyright**
(package private) javax.swing.ImageIcon	**devIcon**
(package private) javax.swing.ImageIcon	**empIcon**
(package private) javax.swing.ImageIcon	**image1**
(package private) javax.swing.JPanel	**insetsPanel1**
(package private) javax.swing.JPanel	**insetsPanel3**
(package private) javax.swing.JLabel	**jLabel1**

(package private) javax.swing.JLabel	**jLabel10**
(package private) javax.swing.JLabel	**jLabel2**
(package private) javax.swing.JLabel	**jLabel3**
(package private) javax.swing.JLabel	**jLabel4**
(package private) javax.swing.JLabel	**jLabel5**
(package private) javax.swing.JLabel	**jLabel6**
(package private) javax.swing.JLabel	**jLabel7**
(package private) javax.swing.JLabel	**jLabel8**
(package private) javax.swing.JLabel	**jLabel9**
(package private) javax.swing.JLabel	**jLabelAboutDev**
(package private) javax.swing.JLabel	**jLabelEmp**
(package private) javax.swing.JLabel	**jLabeliImage**
(package private) javax.swing.JPanel	**jPanel1**
(package private) javax.swing.JPanel	**jPanel2**
(package private) javax.swing.JPanel	**jPanel3**
(package private) javax.swing.JPanel	**jPanelDeveloper**
(package private) javax.swing.JPanel	**jPanelProgram**
(package private) javax.swing.JPanel	**jPanelSupervisors**
(package private) javax.swing.JTabbedPane	**jTabbedPane1**
(package private) javax.swing.JLabel	**label1**
(package private) javax.swing.JLabel	**label2**
(package private) javax.swing.JLabel	**label4**

(package private) javax.swing.JLayeredPane	**layeredPane**
(package private) javax.swing.JPanel	**panel1**
(package private) java.lang.String	**product**
(package private) java.lang.String	**version**
(package private) com.borland.jbcl.layout.XYLayout	**xYLayout1**
(package private) com.borland.jbcl.layout.XYLayout	**xYLayout2**
(package private) com.borland.jbcl.layout.XYLayout	**xYLayout3**
(package private) com.borland.jbcl.layout.XYLayout	**xYLayout4**
(package private) com.borland.jbcl.layout.XYLayout	**xYLayout5**

Fields inherited from class javax.swing.JDialog
accessibleContext, rootPane, rootPaneCheckingEnabled

Fields inherited from class java.awt.Dialog

Fields inherited from class java.awt.Window

Fields inherited from class java.awt.Container

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT,

TOP_ALIGNMENT

Fields inherited from interface javax.swing.WindowConstants
DISPOSE_ON_CLOSE, DO_NOTHING_ON_CLOSE, EXIT_ON_CLOSE, HIDE_ON_CLOSE

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

(package private)	**AdminToolFrame_AboutBox**() Default Constructor.
	AdminToolFrame_AboutBox(java.awt.Frame parent) Constructor.

Method Summary

void	**actionPerformed**(java.awt.event.ActionEvent e) OK button action performed.
(package private) void	**cancel**() Close the dialog
private void	**jbInit**() Set up GUI.
protected void	**processWindowEvent**(java.awt.event.WindowEvent e) Overridden so we can exit when window is closed.

Methods inherited from class javax.swing.JDialog
addImpl, createRootPane, dialogInit, getAccessibleContext, getContentPane, getDefaultCloseOperation, getGlassPane, getJMenuBar, getLayeredPane, getRootPane, isDefaultLookAndFeelDecorated, isRootPaneCheckingEnabled, paramString, remove, setContentPane, setDefaultCloseOperation, setDefaultLookAndFeelDecorated, setGlassPane, setJMenuBar, setLayeredPane, setLayout, setRootPane, setRootPaneCheckingEnabled, update

Methods inherited from class java.awt.Dialog
addNotify, dispose, getTitle, hide, isModal, isResizable, isUndecorated, setModal, setResizable, setTitle, setUndecorated, show

Methods inherited from class java.awt.Window
addPropertyChangeListener, addPropertyChangeListener, addWindowFocusListener, addWindowListener, addWindowStateListener, applyResourceBundle, applyResourceBundle, createBufferStrategy, createBufferStrategy, finalize, getBufferStrategy, getFocusableWindowState, getFocusCycleRootAncestor, getFocusOwner, getFocusTraversalKeys, getGraphicsConfiguration, getInputContext, getListeners, getLocale, getMostRecentFocusOwner, getOwnedWindows, getOwner, getToolkit, getWarningString, getWindowFocusListeners, getWindowListeners, getWindowStateListeners, isActive, isFocusableWindow, isFocusCycleRoot, isFocused, isShowing, pack, postEvent, processEvent, processWindowFocusEvent, processWindowStateEvent, removeWindowFocusListener, removeWindowListener, removeWindowStateListener, setCursor, setFocusableWindowState, setFocusCycleRoot, setLocationRelativeTo, toBack, toFront

Methods inherited from class java.awt.Container
add, add, add, add, add, addContainerListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getContainerListeners, getFocusTraversalPolicy, getInsets, getLayout, getMaximumSize, getMinimumSize, getPreferredSize, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paint, paintComponents, preferredSize, print, printComponents, processContainerEvent, remove, removeAll, removeContainerListener, removeNotify, setFocusTraversalKeys, setFocusTraversalPolicy, setFont, transferFocusBackward, transferFocusDownCycle, validate, validateTree

Methods inherited from class java.awt.Component
action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, coalesceEvents, contains, contains, createImage, createImage, createVolatileImage, createVolatileImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, getBackground, getBounds, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphics, getHeight, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocation, getLocation,

getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isOpaque, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, prepareImage, prepareImage, printAll, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processKeyEvent, processMouseEvent, processMouseMotionEvent, processMouseWheelEvent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, reshape, resize, resize, setBackground, setBounds, setBounds, setComponentOrientation, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeysEnabled, setForeground, setIgnoreRepaint, setLocale, setLocation, setLocation, setName, setSize, setSize, setVisible, show, size, toString, transferFocus, transferFocusUpCycle

Methods inherited from class java.lang.Object
clone, equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

borderLayout1
java.awt.BorderLayout **borderLayout1**

---

borderLayout2
java.awt.BorderLayout **borderLayout2**

---

borderLayout3
java.awt.BorderLayout **borderLayout3**

---

borderLayout4
java.awt.BorderLayout **borderLayout4**

---

borderLayout5
java.awt.BorderLayout **borderLayout5**

button1
javax.swing.JButton **button1**

comments
java.lang.String **comments**

copyright
java.lang.String **copyright**

devIcon
javax.swing.ImageIcon **devIcon**

empIcon
javax.swing.ImageIcon **empIcon**

image1
javax.swing.ImageIcon **image1**

insetsPanel1
javax.swing.JPanel **insetsPanel1**

insetsPanel3
javax.swing.JPanel **insetsPanel3**

jLabel1
javax.swing.JLabel **jLabel1**

jLabel10
javax.swing.JLabel **jLabel10**

jLabel2
javax.swing.JLabel **jLabel2**

jLabel3
javax.swing.JLabel **jLabel3**

jLabel4
javax.swing.JLabel **jLabel4**

---

jLabel5
javax.swing.JLabel **jLabel5**

---

jLabel6
javax.swing.JLabel **jLabel6**

---

jLabel7
javax.swing.JLabel **jLabel7**

---

jLabel8
javax.swing.JLabel **jLabel8**

---

jLabel9
javax.swing.JLabel **jLabel9**

---

jLabelAboutDev
javax.swing.JLabel **jLabelAboutDev**

---

jLabelEmp
javax.swing.JLabel **jLabelEmp**

---

jLabeliImage
javax.swing.JLabel **jLabeliImage**

---

jPanel1
javax.swing.JPanel **jPanel1**

---

jPanel2
javax.swing.JPanel **jPanel2**

---

jPanel3
javax.swing.JPanel **jPanel3**

jPanelDeveloper
javax.swing.JPanel **jPanelDeveloper**

jPanelProgram
javax.swing.JPanel **jPanelProgram**

jPanelSupervisors
javax.swing.JPanel **jPanelSupervisors**

jTabbedPane1
javax.swing.JTabbedPane **jTabbedPane1**

label1
javax.swing.JLabel **label1**

label2
javax.swing.JLabel **label2**

label4
javax.swing.JLabel **label4**

layeredPane
javax.swing.JLayeredPane **layeredPane**

panel1
javax.swing.JPanel **panel1**

product
java.lang.String **product**

version
java.lang.String **version**

xYLayout1
com.borland.jbcl.layout.XYLayout **xYLayout1**

xYLayout2
`com.borland.jbcl.layout.XYLayout` **`xYLayout2`**

---

xYLayout3
`com.borland.jbcl.layout.XYLayout` **`xYLayout3`**

---

xYLayout4
`com.borland.jbcl.layout.XYLayout` **`xYLayout4`**

---

xYLayout5
`com.borland.jbcl.layout.XYLayout` **`xYLayout5`**

Constructor Detail

AdminToolFrame_AboutBox
`AdminToolFrame_AboutBox``()`

Default Constructor. Overriden to call the other constructor.

AdminToolFrame_AboutBox
`public` **`AdminToolFrame_AboutBox`**`(java.awt.Frame parent)`

Constructor. Does an initialization and then calls the jbInit method to set up the GUI.

Parameters:

`parent` - parent frame

Method Detail

actionPerformed
`public void` **`actionPerformed`**`(java.awt.event.ActionEvent e)`

OK button action performed. Close the dialog on a button event.

Specified by:

`actionPerformed` in interface `java.awt.event.ActionListener`

Parameters:

`e` - ActionEvent

---

cancel
`void` **`cancel`**`()`

Close the dialog

---

jbInit
`private void` **`jbInit`**`()`

```
        throws java.lang.Exception
```
Set up GUI.
Sets the components' properties and lays them out.
Most of this code is automatically generated by JBuilder. Especially the laying out of the components.
This is the JBuilders method for setting up the GUI. It is necessary to use this method it order to be able to modify the GUI from JBuilder's Design view.
Throws:
`java.lang.Exception`

---

processWindowEvent

```
protected void processWindowEvent(java.awt.event.WindowEvent e)
```
Overridden so we can exit when window is closed.

Overview Package Class Tree Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES All Classes
SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

---

admintool

Interface DataModifier

All Known Implementing Classes:
AdminToolFrame

---

public interface **DataModifier**

Objects can update the status of GUIUpdater that ADminToolFrame class has, through this interface. Used by JDOMTreeModel objects to notify GUIUpdater that tree structure is modified.
AdminToolFrame implements this interface, and so a reference to AdminToolFrame is passed to JDOMTreeModel constructor. But JDOMTreeModel constructor casts the reference it gets to be DataModifier. So as to hide the rest of AdminToolFrame's methods.

Method Summary

void	**dataModified**() Notifies that data is modified.

Method Detail

dataModified

`public void dataModified()`

Notifies that data is modified. This is the only way to inform AdminToolFrame that its data (Information Structure) is modified. Only AdminToolFrame implements this method. See the implementation of the method in AdminToolFrame for more.

See Also:

`AdminToolFrame.dataModified()`

Overview Package Class Tree Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES All Classes

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

---

admintool

Class DataSourceProperties

```
java.lang.Object
  └java.awt.Component
      └java.awt.Container
          └java.awt.Window
              └java.awt.Dialog
                  └javax.swing.JDialog
                      └admintool.DataSourceProperties
```

All Implemented Interfaces:

javax.accessibility.Accessible, java.awt.image.ImageObserver, java.awt.MenuContainer, javax.swing.RootPaneContainer, java.io.Serializable, javax.swing.WindowConstants

---

public class **DataSourceProperties**
extends javax.swing.JDialog

A Dialog for manipulating Data Source Configuration Properties. Is called from DataSourceConfiguration Dialog.

It is called both on "Add" and on "Properties" commands. When it is called, it defines which of the two cases is called for, from the element that is passed to the constructor and modifies some of its GUI components.

In the case of "Properties" Dialog, it puts on the text fields the values of the data source configuration element passed to the constructor.

See Also:
Serialized Form

Nested Class Summary

private class	**DataSourceProperties.ChangeListenerImpl** An implementation of abstract class AbsractChangeListenerImpl.
(package private) class	**DataSourceProperties.ComboBoxRenderer** The renderer that modifies the appearance of the Conbo box.
(package private) class	**DataSourceProperties.DataSourceProperties_jButtonCancel_actionAdapter** Helper inner class created automatically by development tool.
(package private) class	**DataSourceProperties.DataSourceProperties_jButtonOK_actionAdapter** Helper inner class created automatically by development tool.
(package private) class	**DataSourceProperties.DataSourceProperties_jButtonTest_actionAdapter** Helper inner class created automatically by development tool.
(package private) class	**DataSourceProperties.ItemListenerImpl** Maps the selection of the ComboBox to the appropriate Data source type.

Nested classes inherited from class javax.swing.JDialog
javax.swing.JDialog.AccessibleJDialog

Nested classes inherited from class java.awt.Dialog
java.awt.Dialog.AccessibleAWTDialog

Nested classes inherited from class java.awt.Window
java.awt.Window.AccessibleAWTWindow

Nested classes inherited from class java.awt.Container
java.awt.Container.AccessibleAWTContainer

Nested classes inherited from class java.awt.Component
java.awt.Component.AccessibleAWTComponent, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Field Summary

(package private) java.awt.BorderLayout	**borderLayout1**
(package private) java.awt.BorderLayout	**borderLayout2**
(package private) java.awt.BorderLayout	**borderLayout3**
(package private) java.awt.BorderLayout	**borderLayout4**
(package private) javax.swing.ImageIcon	**cancelIon** Icon used.
private DataSourceProperties.ChangeListenerImpl	**changeListener** To isten for changes.
(package private) DataSourceProperties.ComboBoxRenderer	**comboBoxRenderer** Renderer used in combo box that displays available data source types.
private org.jdom.Element	**dataSrc** Data Source which is being manipulated or created.
(package private) javax.swing.ImageIcon	**dataSrcTypeIcon** Icon used.
private java.lang.String	**driver** Driver used to access te Data Source.
private java.lang.String	**dsn** DSN of the Data Source.
(package private) javax.swing.ImageIcon	**dsnIcon** Icon used.
(package private) java.awt.FlowLayout	**flowLayout1**

private java.lang.String	**inputDriver** Driver used to access te Data Source entered by user.
private java.lang.String	**inputDsn** DSN of the Data Source entered by user.
private java.lang.String	**inputLogin** Login required to access the Data Source entered by user.
private java.lang.String	**inputPassword** Password required to access the Data Source entered by user.
private java.lang.String	**inputType** Type of the Data Source entered by user.
private DataSourceProperties.ItemListenerImpl	**itemListener** Used in combo box.
(package private) javax.swing.JButton	**jButtonCancel** Cancel button.
(package private) javax.swing.JButton	**jButtonOK** OK button.
(package private) javax.swing.JButton	**jButtonTest** Test button.
(package private) javax.swing.JComboBox	**jComboBox1** Available data source types.
(package private) javax.swing.ImageIcon	**jdbcDriverIcon** Icon used.
(package private) javax.swing.JLabel	**jLabel1**
(package private) javax.swing.JLabel	**jLabel2**
(package private) javax.swing.JLabel	**jLabel3**
(package private) javax.swing.JLabel	**jLabel4**
(package private) javax.swing.JLabel	**jLabelType**
(package private) javax.swing.JPanel	**jPanel1**
(package private) javax.swing.JPanel	**jPanelButtons**
(package private) javax.swing.JPanel	**jPanelChooseSource**
(package private) javax.swing.JPanel	**jPanelFields**

(package private) javax.swing.JPanel	**jPanelODBC**
(package private) javax.swing.JPanel	**jPanelTestButton**
(package private) javax.swing.JPasswordField	**jPasswordField** Password.
(package private) javax.swing.JTextField	**jTextFieldDSN** DSN text field.
(package private) javax.swing.JTextField	**jTextFieldJDBC** JDBC Driver text field.
(package private) javax.swing.JTextField	**jTextFieldLogin** Login text field.
private java.lang.String	**login** Login required to access the Data Source.
private boolean	**modified** Indicates whether the data source properties are modified or not.
private java.lang.String	**MY_SQL** MySQL data source used in Combo box.
private java.lang.String	**ODBC** ODBC data source used in Combo box.
(package private) javax.swing.ImageIcon	**okIcon** Icon used.
private java.lang.String	**ORACLE** Oracle data source used in Combo box.
(package private) javax.swing.JPanel	**panel1**
private java.lang.String	**password** Password required to access the Data Source.
private java.util.Map	**passwordMap** Passwords.
private java.lang.String	**POSTGRESQL** Postgresql data source used in Combo box.
(package private) javax.swing.ImageIcon	**serverIcon** Icon used.
private java.lang.String[]	**sources** Array of the data source used in combo box.

private java.lang.String	**SQL_SERVER** SQL server data source used in Combo box.
(package private) javax.swing.ImageIcon	**testIcon** Icon used.
private java.lang.String	**type** Type of the Data Source.
(package private) com.borland.jbcl.layout.XYLayout	**xYLayout1**
(package private) com.borland.jbcl.layout.XYLayout	**xYLayout3**

Fields inherited from class javax.swing.JDialog
accessibleContext, rootPane, rootPaneCheckingEnabled

Fields inherited from class java.awt.Dialog

Fields inherited from class java.awt.Window

Fields inherited from class java.awt.Container

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface javax.swing.WindowConstants
DISPOSE_ON_CLOSE, DO_NOTHING_ON_CLOSE, EXIT_ON_CLOSE, HIDE_ON_CLOSE

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

DataSourceProperties(java.awt.Dialog dialog, boolean modal, org.jdom.Element ds, java.util.Map passwordMap) Constructor.

Method Summary

org.jdom.Element	**getDataSrc**() Gets the data source configuration element.
private void	**jbInit**() Set up GUI.
(package private) void	**jButtonCancel_actionPerformed**(java.awt.event.ActionEvent e) Cancel button action performed.
(package private) void	**jButtonOK_actionPerformed**(java.awt.event.ActionEvent e) OK button action peformed.
(package private) void	**jButtonTest_actionPerformed**(java.awt.event.ActionEvent e) Test button action performed.
private void	**readDataSource**() Reads Data Source Configuration Properties.
private void	**readInputValues**() Reads values that the user entered forthe data source configuration.
private java.lang.String	**testODBCSource**() Creats an JDBC connection string and tests it.

Methods inherited from class javax.swing.JDialog
addImpl, createRootPane, dialogInit, getAccessibleContext, getContentPane, getDefaultCloseOperation, getGlassPane, getJMenuBar, getLayeredPane, getRootPane, isDefaultLookAndFeelDecorated, isRootPaneCheckingEnabled, paramString, processWindowEvent, remove, setContentPane, setDefaultCloseOperation, setDefaultLookAndFeelDecorated, setGlassPane, setJMenuBar, setLayeredPane, setLayout, setRootPane, setRootPaneCheckingEnabled, update

Methods inherited from class java.awt.Dialog
addNotify, dispose, getTitle, hide, isModal, isResizable, isUndecorated,

setModal, setResizable, setTitle, setUndecorated, show

Methods inherited from class java.awt.Window
addPropertyChangeListener, addPropertyChangeListener, addWindowFocusListener, addWindowListener, addWindowStateListener, applyResourceBundle, applyResourceBundle, createBufferStrategy, createBufferStrategy, finalize, getBufferStrategy, getFocusableWindowState, getFocusCycleRootAncestor, getFocusOwner, getFocusTraversalKeys, getGraphicsConfiguration, getInputContext, getListeners, getLocale, getMostRecentFocusOwner, getOwnedWindows, getOwner, getToolkit, getWarningString, getWindowFocusListeners, getWindowListeners, getWindowStateListeners, isActive, isFocusableWindow, isFocusCycleRoot, isFocused, isShowing, pack, postEvent, processEvent, processWindowFocusEvent, processWindowStateEvent, removeWindowFocusListener, removeWindowListener, removeWindowStateListener, setCursor, setFocusableWindowState, setFocusCycleRoot, setLocationRelativeTo, toBack, toFront

Methods inherited from class java.awt.Container
add, add, add, add, add, addContainerListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getContainerListeners, getFocusTraversalPolicy, getInsets, getLayout, getMaximumSize, getMinimumSize, getPreferredSize, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paint, paintComponents, preferredSize, print, printComponents, processContainerEvent, remove, removeAll, removeContainerListener, removeNotify, setFocusTraversalKeys, setFocusTraversalPolicy, setFont, transferFocusBackward, transferFocusDownCycle, validate, validateTree

Methods inherited from class java.awt.Component
action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, coalesceEvents, contains, contains, createImage, createImage, createVolatileImage, createVolatileImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, getBackground, getBounds, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphics, getHeight, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocation, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMouseWheelListeners, getName, getParent, getPeer,

getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isOpaque, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, prepareImage, prepareImage, printAll, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processKeyEvent, processMouseEvent, processMouseMotionEvent, processMouseWheelEvent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, reshape, resize, resize, setBackground, setBounds, setBounds, setComponentOrientation, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeysEnabled, setForeground, setIgnoreRepaint, setLocale, setLocation, setLocation, setName, setSize, setSize, setVisible, show, size, toString, transferFocus, transferFocusUpCycle

Methods inherited from class java.lang.Object
clone, equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

borderLayout1
java.awt.BorderLayout **borderLayout1**

---

borderLayout2
java.awt.BorderLayout **borderLayout2**

---

borderLayout3
java.awt.BorderLayout **borderLayout3**

---

borderLayout4
java.awt.BorderLayout **borderLayout4**

---

cancelIon
javax.swing.ImageIcon **cancelIon**

Icon used.

---

changeListener
`private DataSourceProperties.ChangeListenerImpl changeListener`

To isten for changes.

---

comboBoxRenderer
`DataSourceProperties.ComboBoxRenderer comboBoxRenderer`

Renderer used in combo box that displays available data source types.

---

dataSrc
`private org.jdom.Element dataSrc`

Data Source which is being manipulated or created.

---

dataSrcTypeIcon
`javax.swing.ImageIcon dataSrcTypeIcon`

Icon used.

---

driver
`private java.lang.String driver`

Driver used to access te Data Source.

---

dsn
`private java.lang.String dsn`

DSN of the Data Source.

---

dsnIcon
`javax.swing.ImageIcon dsnIcon`

Icon used.

---

flowLayout1
`java.awt.FlowLayout flowLayout1`

---

inputDriver
`private java.lang.String inputDriver`

Driver used to access te Data Source entered by user.

---

inputDsn
`private java.lang.String inputDsn`

DSN of the Data Source entered by user.

---

inputLogin
`private java.lang.String` **`inputLogin`**

Login required to access the Data Source entered by user.

---

inputPassword
`private java.lang.String` **`inputPassword`**

Password required to access the Data Source entered by user.

---

inputType
`private java.lang.String` **`inputType`**

Type of the Data Source entered by user.

---

itemListener
`private` `DataSourceProperties.ItemListenerImpl` **`itemListener`**

Used in combo box.

See Also:

`DataSourceProperties.ItemListenerImpl`

---

jButtonCancel
`javax.swing.JButton` **`jButtonCancel`**

Cancel button.

---

jButtonOK
`javax.swing.JButton` **`jButtonOK`**

OK button.

---

jButtonTest
`javax.swing.JButton` **`jButtonTest`**

Test button.

---

jComboBox1
`javax.swing.JComboBox` **`jComboBox1`**

Available data source types.

---

jdbcDriverIcon
`javax.swing.ImageIcon` **`jdbcDriverIcon`**

Icon used.

---

jLabel1
`javax.swing.JLabel` **`jLabel1`**

jLabel2
`javax.swing.JLabel` **`jLabel2`**

jLabel3
`javax.swing.JLabel` **`jLabel3`**

jLabel4
`javax.swing.JLabel` **`jLabel4`**

jLabelType
`javax.swing.JLabel` **`jLabelType`**

jPanel1
`javax.swing.JPanel` **`jPanel1`**

jPanelButtons
`javax.swing.JPanel` **`jPanelButtons`**

jPanelChooseSource
`javax.swing.JPanel` **`jPanelChooseSource`**

jPanelFields
`javax.swing.JPanel` **`jPanelFields`**

jPanelODBC
`javax.swing.JPanel` **`jPanelODBC`**

jPanelTestButton
`javax.swing.JPanel` **`jPanelTestButton`**

jPasswordField
`javax.swing.JPasswordField` **`jPasswordField`**

Password.

jTextFieldDSN

`javax.swing.JTextField` **`jTextFieldDSN`**

DSN text field.

---

jTextFieldJDBC

`javax.swing.JTextField` **`jTextFieldJDBC`**

JDBC Driver text field.

---

jTextFieldLogin

`javax.swing.JTextField` **`jTextFieldLogin`**

Login text field.

---

login

`private java.lang.String` **`login`**

Login required to access the Data Source.

---

modified

`private boolean` **`modified`**

Indicates whether the data source properties are modified or not.

MY_SQL

`private final java.lang.String` **`MY_SQL`**

MySQL data source used in Combo box.

See Also:

Constant Field Values

ODBC

`private final java.lang.String` **`ODBC`**

ODBC data source used in Combo box.

See Also:

Constant Field Values

---

okIcon

`javax.swing.ImageIcon` **`okIcon`**

Icon used.

ORACLE

`private final java.lang.String` **`ORACLE`**

Oracle data source used in Combo box.

See Also:

Constant Field Values

---

panel1
`javax.swing.JPanel` **`panel1`**

---

password
`private java.lang.String` **`password`**

Password required to access the Data Source.

---

passwordMap
`private java.util.Map` **`passwordMap`**

Passwords.

POSTGRESQL
`private final java.lang.String` **`POSTGRESQL`**

Postgresql data source used in Combo box.

See Also:

Constant Field Values

---

serverIcon
`javax.swing.ImageIcon` **`serverIcon`**

Icon used.

---

sources
`private java.lang.String[]` **`sources`**

Array of the data source used in combo box.

SQL_SERVER
`private final java.lang.String` **`SQL_SERVER`**

SQL server data source used in Combo box.

See Also:

Constant Field Values

---

testIcon
`javax.swing.ImageIcon` **`testIcon`**

Icon used.

---

type
`private java.lang.String` **`type`**

Type of the Data Source.

---

xYLayout1
```
com.borland.jbcl.layout.XYLayout xYLayout1
```

---

xYLayout3
```
com.borland.jbcl.layout.XYLayout xYLayout3
```

Constructor Detail

DataSourceProperties
```
public DataSourceProperties(java.awt.Dialog dialog,
                            boolean modal,
                            org.jdom.Element ds,
                            java.util.Map passwordMap)
```
Constructor. Does an initialization and then calls the jbInit method to set up the GUI.

Parameters:

`dialog` - the parent dialog from which this dialog was called.

`modal` - modal

`ds` - data source configuration which is going to be manipulated.

`passwordMap` - Passwords

Method Detail

getDataSrc
```
public org.jdom.Element getDataSrc()
```
Gets the data source configuration element. Used by DataSourceConfiguration Dialog to access the data source configuration.

Returns:

Element

---

jbInit
```
private void jbInit()
             throws java.lang.Exception
```
Set up GUI.

Sets the components' properties and lays them out.

In the case of the Properties Dialog, the text fields are initialized with the data variables. Also OK button's initial ststus is disable, so user must first test the data source before exiting.

Most of this code is automatically generated by JBuilder. Especially the laying out of the components.

This is the JBuilders method for setting up the GUI. It is necessary to use this method it order to be able to modify the GUI from JBuilder's Design view.

Throws:

`java.lang.Exception`

---

jButtonCancel_actionPerformed
```
void jButtonCancel_actionPerformed(java.awt.event.ActionEvent e)
```

Cancel button action performed.
Exits the Dialog.
Parameters:
e - ActionEvent

---

jButtonOK_actionPerformed

```
void jButtonOK_actionPerformed(java.awt.event.ActionEvent e)
```

OK button action peformed.
Saves the data source configuration element and exits dialog.

- if it is an Add Dialog, create a new data source configuration element
- if it is a Properties Dialog and data source configuration is modified, then update the existing element
- exit

Parameters:
e - ActionEvent

---

jButtonTest_actionPerformed

```
void jButtonTest_actionPerformed(java.awt.event.ActionEvent e)
```

Test button action performed.
Reads values entered by user for data source configuration and test them. Only ODBC sources are support so far. The test is delegated to testODBCSource method. If test is passed the OK button is enabled.
Parameters:
e - ActionEvent
See Also:
testODBCSource()

---

readDataSource

```
private void readDataSource()
```

Reads Data Source Configuration Properties.
It decides whether it is an Add Data Source Dialog or a Data Source Properties Dialog. In each case it set proper title. In the cases of Properties Dialog it reads the valeus from the element which was passed to the Constructor. The values are stored in its data variables.

---

readInputValues

```
private void readInputValues()
```

Reads values that the user entered forthe data source configuration.

---

testODBCSource

```
private java.lang.String testODBCSource()
                                 throws java.lang.ClassNotFoundException,
                                        java.sql.SQLException
```

Creats an JDBC connection string and tests it.
Takes the type and the name of the ODBC source creats the connection string and tests if connects using the Default ODBC driver provided by Sun. Other drivers are not supported yet.

Returns:

jdbc connection string, eg. jdbc:odbc:customers

Throws:

`java.lang.ClassNotFoundException`
`java.sql.SQLException`

Overview **Package** **Class** **Tree** **Index** **Help**

PREV CLASS NEXT CLASS FRAMES NO FRAMES All Classes
SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

---

admintool

Class DataSourcesConfiguration

```
java.lang.Object
  └java.awt.Component
      └java.awt.Container
          └java.awt.Window
              └java.awt.Dialog
                  └javax.swing.JDialog
                      └admintool.DataSourcesConfiguration
```

All Implemented Interfaces:

javax.accessibility.Accessible, java.awt.image.ImageObserver, java.awt.MenuContainer, javax.swing.RootPaneContainer, java.io.Serializable, javax.swing.WindowConstants

---

public class **DataSourcesConfiguration**
extends javax.swing.JDialog

A Dialog for manipulating Data Source Configurations.

It is called both on "New Data Source Configuration" and on "Configure Data Sources" commands. When it is called it defines which of the two cases is called for, from the document that is passed to the constructor and modifies some of its GUI components.

In this class, the term *document* is refered to the Document that holds the data sources configurations (*.dsc) that are loaded on the application, and not to the documents of the Information Structure tree and Data Structure tree.

See Also:

Serialized Form

Nested Class Summary	
(package private) class	**DataSourcesConfiguration.DataSourcesConfiguration_jButtonAdd_actionAdapter** Helper inner class created automatically by development tool.
(package private) class	**DataSourcesConfiguration.DataSourcesConfiguration_jButtonCancel_actionAdapter** Helper inner class created automatically by development tool.
(package private) class	**DataSourcesConfiguration.DataSourcesConfiguration_jButtonOK_actionAdapter** Helper inner class created automatically by development tool.
(package private) class	**DataSourcesConfiguration.DataSourcesConfiguration_jButtonProps_actionAdapter** Helper inner class created automatically by development tool.
(package private) class	**DataSourcesConfiguration.DataSourcesConfiguration_jButtonRemove_actionAdapter** Helper inner class created automatically by development tool.
private class	**DataSourcesConfiguration.ListNode** The nodes of the list model that the JList use.

Nested classes inherited from class javax.swing.JDialog
javax.swing.JDialog.AccessibleJDialog

Nested classes inherited from class java.awt.Dialog
java.awt.Dialog.AccessibleAWTDialog

Nested classes inherited from class java.awt.Window
java.awt.Window.AccessibleAWTWindow

Nested classes inherited from class java.awt.Container
java.awt.Container.AccessibleAWTContainer

Nested classes inherited from class java.awt.Component
java.awt.Component.AccessibleAWTComponent, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Field Summary

(package private) javax.swing.ImageIcon	**addDataSrcIcon** Icon used.
(package private) java.awt.BorderLayout	**borderLayout1**
(package private) java.awt.BorderLayout	**borderLayout2**
(package private) java.awt.BorderLayout	**borderLayout3**
(package private) java.awt.BorderLayout	**borderLayout4**
(package private) javax.swing.ImageIcon	**cancelIcon** Icon used.
(package private) javax.swing.ImageIcon	**dataSrcPropsIcon** Icon used.
private org.jdom.Document	**doc** Data source configuration document which contains the Data sources currently loaded.
(package private) javax.swing.ImageIcon	**doneIcon** Icon used.
(package private) java.awt.FlowLayout	**flowLayout1**
(package private) java.awt.GridLayout	**gridLayout1**
private org.jdom.Document	**initialDoc** The initial Data source configuration document that was passed to constructor.
(package private) javax.swing.JButton	**jButtonAdd** Add button.
(package private) javax.swing.JButton	**jButtonCancel** Cancel button
(package private) javax.swing.JButton	**jButtonOK** OK (displayed as Load or Update) button.
(package private) javax.swing.JButton	**jButtonProps** Properties button.
(package private)	**jButtonRemove**

javax.swing.JButton	Remove button.
(package private) javax.swing.JLabel	**jLabelDataSrc**
(package private) javax.swing.JList	**jListDataSrc** The list that the data sources are displayed.
(package private) javax.swing.JPanel	**jPanel1**
(package private) javax.swing.JPanel	**jPanel2**
(package private) javax.swing.JPanel	**jPanel3**
(package private) javax.swing.JPanel	**jPanel4**
(package private) javax.swing.JPanel	**jPanel5**
(package private) javax.swing.JScrollPane	**jScrollPane1**
private javax.swing.DefaultListModel	**listModel** List model used in the list that displays the data sources.
private boolean	**modified** Indicates whether the data source configurations are modified or not.
private java.lang.String	**okButtonName** OK button text.
(package private) javax.swing.JPanel	**panel1**
private AdminToolFrame	**parent** The parent frame.
private java.util.Map	**passwordMap** Passwords.
(package private) javax.swing.ImageIcon	**removeDataSrcIcon** Icon used.
private org.jdom.Element	**root** The root of the doc document.
(package private) javax.swing.border.TitledBorder	**titledBorder1**

Fields inherited from class javax.swing.JDialog
accessibleContext, rootPane, rootPaneCheckingEnabled

Fields inherited from class java.awt.Dialog

Fields inherited from class java.awt.Window

Fields inherited from class java.awt.Container

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface javax.swing.WindowConstants
DISPOSE_ON_CLOSE, DO_NOTHING_ON_CLOSE, EXIT_ON_CLOSE, HIDE_ON_CLOSE

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

DataSourcesConfiguration(java.awt.Frame frame, boolean modal, org.jdom.Document dsc, java.util.Map passwordMap)
Constructor.

Method Summary

private void	**initializeList**() Maps data Sources of loaded document to the List model.
private	**jbInit**()

void	Set up GUI.
(package private) void	**jButtonAdd_actionPerformed**(java.awt.event.ActionEvent e) Add button action performed.
(package private) void	**jButtonCancel_actionPerformed**(java.awt.event.ActionEvent e) Cancel button action performed.
(package private) void	**jButtonOK_actionPerformed**(java.awt.event.ActionEvent e) OK button action performed.
(package private) void	**jButtonProps_actionPerformed**(java.awt.event.ActionEvent e) Properties button action performed.
(package private) void	**jButtonRemove_actionPerformed**(java.awt.event.ActionEvent e) Remove button action performed.
private void	**readDataSrcConfig**() Reads data source configurations.

Methods inherited from class javax.swing.JDialog
addImpl, createRootPane, dialogInit, getAccessibleContext, getContentPane, getDefaultCloseOperation, getGlassPane, getJMenuBar, getLayeredPane, getRootPane, isDefaultLookAndFeelDecorated, isRootPaneCheckingEnabled, paramString, processWindowEvent, remove, setContentPane, setDefaultCloseOperation, setDefaultLookAndFeelDecorated, setGlassPane, setJMenuBar, setLayeredPane, setLayout, setRootPane, setRootPaneCheckingEnabled, update

Methods inherited from class java.awt.Dialog
addNotify, dispose, getTitle, hide, isModal, isResizable, isUndecorated, setModal, setResizable, setTitle, setUndecorated, show

Methods inherited from class java.awt.Window
addPropertyChangeListener, addPropertyChangeListener, addWindowFocusListener, addWindowListener, addWindowStateListener, applyResourceBundle, applyResourceBundle, createBufferStrategy, createBufferStrategy, finalize, getBufferStrategy, getFocusableWindowState, getFocusCycleRootAncestor, getFocusOwner, getFocusTraversalKeys, getGraphicsConfiguration, getInputContext, getListeners, getLocale, getMostRecentFocusOwner, getOwnedWindows, getOwner, getToolkit, getWarningString, getWindowFocusListeners, getWindowListeners, getWindowStateListeners, isActive, isFocusableWindow, isFocusCycleRoot, isFocused, isShowing, pack, postEvent, processEvent, processWindowFocusEvent, processWindowStateEvent, removeWindowFocusListener, removeWindowListener, removeWindowStateListener, setCursor,

setFocusableWindowState, setFocusCycleRoot, setLocationRelativeTo, toBack, toFront

Methods inherited from class java.awt.Container
add, add, add, add, add, addContainerListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getContainerListeners, getFocusTraversalPolicy, getInsets, getLayout, getMaximumSize, getMinimumSize, getPreferredSize, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paint, paintComponents, preferredSize, print, printComponents, processContainerEvent, remove, removeAll, removeContainerListener, removeNotify, setFocusTraversalKeys, setFocusTraversalPolicy, setFont, transferFocusBackward, transferFocusDownCycle, validate, validateTree

Methods inherited from class java.awt.Component
action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, coalesceEvents, contains, contains, createImage, createImage, createVolatileImage, createVolatileImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, getBackground, getBounds, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphics, getHeight, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocation, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isOpaque, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, prepareImage, prepareImage, printAll, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processKeyEvent, processMouseEvent, processMouseMotionEvent, processMouseWheelEvent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, reshape, resize, resize, setBackground, setBounds, setBounds, setComponentOrientation, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeysEnabled, setForeground, setIgnoreRepaint, setLocale,

setLocation, setLocation, setName, setSize, setSize, setVisible, show, size, toString, transferFocus, transferFocusUpCycle

Methods inherited from class java.lang.Object
clone, equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

addDataSrcIcon
`javax.swing.ImageIcon` **`addDataSrcIcon`**

Icon used.

---

borderLayout1
`java.awt.BorderLayout` **`borderLayout1`**

---

borderLayout2
`java.awt.BorderLayout` **`borderLayout2`**

---

borderLayout3
`java.awt.BorderLayout` **`borderLayout3`**

---

borderLayout4
`java.awt.BorderLayout` **`borderLayout4`**

---

cancelIcon
`javax.swing.ImageIcon` **`cancelIcon`**

Icon used.

---

dataSrcPropsIcon
`javax.swing.ImageIcon` **`dataSrcPropsIcon`**

Icon used.

---

doc
`private org.jdom.Document` **`doc`**

Data source configuration document which contains the Data sources currently loaded. The modification are applied directly on it.

---

doneIcon
`javax.swing.ImageIcon` **`doneIcon`**

Icon used.

---

flowLayout1
`java.awt.FlowLayout` **`flowLayout1`**

---

gridLayout1
`java.awt.GridLayout` **`gridLayout1`**

---

initialDoc
`private org.jdom.Document` **`initialDoc`**

The initial Data source configuration document that was passed to constructor. Used to store the document to the constructor as a back up. If user hits cancel on exit, it is used to restore the initial state of the document.

---

jButtonAdd
`javax.swing.JButton` **`jButtonAdd`**

Add button.

---

jButtonCancel
`javax.swing.JButton` **`jButtonCancel`**

Cancel button

---

jButtonOK
`javax.swing.JButton` **`jButtonOK`**

OK (displayed as Load or Update) button. When command "New Data Source Configuration" is entered then the button text is "Load". When "Configure Data Sources" is entered then button text is "Update".

---

jButtonProps
`javax.swing.JButton` **`jButtonProps`**

Properties button.

---

jButtonRemove
`javax.swing.JButton` **`jButtonRemove`**

Remove button.

---

jLabelDataSrc
`javax.swing.JLabel` **`jLabelDataSrc`**

---

jListDataSrc
`javax.swing.JList` **`jListDataSrc`**

The list that the data sources are displayed.

---

jPanel1
`javax.swing.JPanel` **`jPanel1`**

---

jPanel2
`javax.swing.JPanel` **`jPanel2`**

---

jPanel3
`javax.swing.JPanel` **`jPanel3`**

---

jPanel4
`javax.swing.JPanel` **`jPanel4`**

---

jPanel5
`javax.swing.JPanel` **`jPanel5`**

---

jScrollPane1
`javax.swing.JScrollPane` **`jScrollPane1`**

---

listModel
`private javax.swing.DefaultListModel` **`listModel`**

List model used in the list that displays the data sources.

---

modified
`private boolean` **`modified`**

Indicates whether the data source configurations are modified or not.

---

okButtonName
`private java.lang.String` **`okButtonName`**

OK button text. OK button might say "Load" or "Update".

---

panel1
`javax.swing.JPanel` **`panel1`**

---

parent
`private` `AdminToolFrame` **`parent`**

The parent frame.

---

passwordMap
`private java.util.Map` **`passwordMap`**

Passwords.

---

removeDataSrcIcon
`javax.swing.ImageIcon` **`removeDataSrcIcon`**

Icon used.

---

root
`private org.jdom.Element` **`root`**

The root of the doc document.

---

titledBorder1
`javax.swing.border.TitledBorder` **`titledBorder1`**

Constructor Detail

DataSourcesConfiguration
```
public DataSourcesConfiguration(java.awt.Frame frame,
                                boolean modal,
                                org.jdom.Document dsc,
                                java.util.Map passwordMap)
```

Constructor. Does an initialization and then calls the jbInit method to set up the GUI.

Parameters:

`frame` - the parent frame

`modal` -

`dsc` - the data source configuration document.

`passwordMap` - the passwords.

Method Detail

initializeList
```
private void initializeList()
```

Maps data Sources of loaded document to the List model.
The elements of the document are the data sources that are going to be configured.
These elements are not detached from the root of the document.

---

jbInit

```
private void jbInit()
             throws java.lang.Exception
```

Set up GUI.
Sets the components' properties and lays them out.
Most of this code is automatically generated by JBuilder. Especially the laying out of the components.
This is the JBuilders method for setting up the GUI. It is necessary to use this method it order to be able to modify the GUI from JBuilder's Design view.
Throws:
`java.lang.Exception`

---

jButtonAdd_actionPerformed

```
void jButtonAdd_actionPerformed(java.awt.event.ActionEvent e)
```

Add button action performed.
Adds a new Data Source configuration.
First it changes the status of configuration into modified. Calls DataSourceProperties Dialog so user can create a new Data Source configuration. Gets the new data source configuration from the called Dialog and adds it to the document and to the JList.
Parameters:
`e` - ActionEvent
See Also:
`DataSourceProperties`

---

jButtonCancel_actionPerformed

```
void jButtonCancel_actionPerformed(java.awt.event.ActionEvent e)
```

Cancel button action performed.
It restores the document to its inital state using the back up and so discards any modifications.
Parameters:
`e` - ActionEvent

---

jButtonOK_actionPerformed

```
void jButtonOK_actionPerformed(java.awt.event.ActionEvent e)
```

OK button action performed.
If

---

jButtonProps_actionPerformed

```
void jButtonProps_actionPerformed(java.awt.event.ActionEvent e)
```

Properties button action performed.
Displays and manipulates the Data Source configuration Properties.

First it changes the status of configuration into modified. Check if user did make a selection. If user did make a selection, it takes the data source element and calls the DataSourceProperties Dialog with it so user can modify it. The modifications take effect directly to the dataSource element, because elements are not detached from the root of he document. Finally, it updates the JList.

Parameters:

e - ActionEvent

See Also:

`DataSourceProperties`

---

jButtonRemove_actionPerformed

```
void jButtonRemove_actionPerformed(java.awt.event.ActionEvent e)
```

Remove button action performed.

Removes a Data Source configuration.

First it changes the status of configuration into modified. Gets the selected item of the JList. If user did not make a selection then displays an error message. If there is a selection, retrieves the element and removes it from the document. It also removes the selection from the list model of the JList.

Parameters:

e - ActionEvent

---

readDataSrcConfig

```
private void readDataSrcConfig()
```

Reads data source configurations.

It decides whether it is a New Data Source Configuration Dialog or a Configure Data Sources Dialog. In each case it set proper title and OK button name. Also in the case of Configure Data Sources Dialog it initilizes the list of data sources. In both cases it backs up the document that passed to the constructor in order to be able to discard changes if user hits cancel on exit.

See Also:

`initializeList()`

Overview **Package** **Class** **Tree** **Index** **Help**

PREV CLASS NEXT CLASS FRAMES NO FRAMES All Classes

admintool

Class FilesFilter

```
java.lang.Object
  └javax.swing.filechooser.FileFilter
      └admintool.FilesFilter
```

public class **FilesFilter**
extends javax.swing.filechooser.FileFilter

The File Filter is used from JFileChooser and supports xml, dsc, dst, is, xsl, asc files.
Dispite of implementing a seperate class for each file type extension, this class supports all the extensions that are needed by the application. Depending on the parameter passed on initialization the corresponding file filter type extension is created, eg. a "new FilesFilter("xml")" is an xml file filter.

It is necessary to remove the file filter from jfilechooser after use, because file filters are accomulated, that is the addition of a new file filter to jfilechooser does not remove the previous one, but jfilechooser now displays both type of files.

Tip: For more see The Java Tutorial: Creating a GUI with JFC/Swing: How to Use File Choosers

Field Summary

private java.lang.String	**asc** Association file type extension.
private java.lang.String	**ascDescription** Association file description.
private java.lang.String	**dsc** Data Source Configuration file type extension.
private java.lang.String	**dscDescription** Data Source Configuration file description.
private java.lang.String	**dst** Data Structure file type extension.
private java.lang.String	**dstDescription** Data Structure file description.
private java.lang.String	**is** Information Structure file type extension.
private java.lang.String	**isDescription** Information Structure file description.
private java.lang.String	**shownExtension** The extension that the File Filter is initialezed, and thus the file type that the object is filtering.

private java.lang.String	**xml** XML file type extension.
private java.lang.String	**xmlDescription** XML file description.
private java.lang.String	**xsl** XSL file type extension.
private java.lang.String	**xslDescription** XSL file description.

Constructor Summary

FilesFilter(java.lang.String ex) Constructor, which initializes the extension of the file type that is going to be shown on the file chooser.

Method Summary

boolean	**accept**(java.io.File f) Whether the given file is accepted by this filter.
java.lang.String	**getDescription**() The description of this filter.
private java.lang.String	**getExtension**(java.io.File f) Gets the extension of a file.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

asc

private final java.lang.String **asc**

Association file type extension.

See Also:

Constant Field Values

---

ascDescription

`private final java.lang.String` **`ascDescription`**

Association file description.

See Also:

Constant Field Values

---

dsc

`private final java.lang.String` **`dsc`**

Data Source Configuration file type extension.

See Also:

Constant Field Values

---

dscDescription

`private final java.lang.String` **`dscDescription`**

Data Source Configuration file description.

See Also:

Constant Field Values

---

dst

`private final java.lang.String` **`dst`**

Data Structure file type extension.

See Also:

Constant Field Values

---

dstDescription

`private final java.lang.String` **`dstDescription`**

Data Structure file description.

See Also:

Constant Field Values

---

is

`private final java.lang.String` **`is`**

Information Structure file type extension.

See Also:

Constant Field Values

---

isDescription

`private final java.lang.String` **`isDescription`**

Information Structure file description.

See Also:

Constant Field Values

---

shownExtension

`private java.lang.String` **`shownExtension`**

The extension that the File Filter is initialezed, and thus the file type that the object is filtering.

---

xml

`private final java.lang.String` **`xml`**

XML file type extension.

See Also:

Constant Field Values

---

xmlDescription

`private final java.lang.String` **`xmlDescription`**

XML file description.

See Also:

Constant Field Values

---

xsl

`private final java.lang.String` **`xsl`**

XSL file type extension.

See Also:

Constant Field Values

---

xslDescription

`private final java.lang.String` **`xslDescription`**

XSL file description.

See Also:

Constant Field Values

Constructor Detail

FilesFilter

`public` **`FilesFilter`**`(java.lang.String ex)`

Constructor, which initializes the extension of the file type that is going to be shown on the file chooser.

Parameters:

`ex` - String

Method Detail

accept

`public boolean` **`accept`**`(java.io.File f)`

Whether the given file is accepted by this filter.

Parameters:

`f` - the file that is tested if it is accepted.

Returns:

Yes if it is accepted

---

getDescription

`public java.lang.String` **`getDescription`**`()`

The description of this filter. For example: "JInformation Structure files"

Returns:

String

---

getExtension

`private java.lang.String` **`getExtension`**`(java.io.File f)`

Gets the extension of a file.

Parameters:

`f` - the file

Returns:

its extension

Overview **Package** **Class** **Tree** **Index** **Help**

PREV CLASS NEXT CLASS FRAMES NO FRAMES All Classes

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

---

admintool

Class InformationFieldProperties

```
java.lang.Object
  └java.awt.Component
      └java.awt.Container
          └java.awt.Window
              └java.awt.Dialog
                  └javax.swing.JDialog
                      └admintool.InformationFieldProperties
```

All Implemented Interfaces:

javax.accessibility.Accessible, java.awt.image.ImageObserver, java.awt.MenuContainer, javax.swing.RootPaneContainer, java.io.Serializable, javax.swing.WindowConstants

---

public class **InformationFieldProperties**
extends javax.swing.JDialog

A Dialog for displaying and manipulating an Information Field's Properties.

See Also:
Serialized Form

Nested Class Summary

private class	**InformationFieldProperties.ChangeListenerImpl** An implementation of abstract class AbsractChangeListenerImpl.
(package private) class	**InformationFieldProperties.InformationFieldProperties_jButtonApply_actionAdapter** Helper inner class created automatically by development tool.
(package private) class	**InformationFieldProperties.InformationFieldProperties_jButtonCancel_actionAdapter** Helper inner class created automatically by development tool.
(package private) class	**InformationFieldProperties.InformationFieldProperties_jButtonOK_actionAdapter** Helper inner class created automatically by development tool.
(package private) class	**InformationFieldProperties.InformationFieldProperties_jRadioButtonNo_actionAdapter** Helper inner class created automatically by development tool.
(package private) class	**InformationFieldProperties.InformationFieldProperties_jRadioButtonYes_actionAdapter** Helper inner class created automatically by development tool.

Nested classes inherited from class javax.swing.JDialog
javax.swing.JDialog.AccessibleJDialog

Nested classes inherited from class java.awt.Dialog
java.awt.Dialog.AccessibleAWTDialog

Nested classes inherited from class java.awt.Window
java.awt.Window.AccessibleAWTWindow

Nested classes inherited from class java.awt.Container

java.awt.Container.AccessibleAWTContainer

Nested classes inherited from class java.awt.Component
java.awt.Component.AccessibleAWTComponent, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Field Summary

(package private) javax.swing.ImageIcon	**applyIcon** Icons used.
(package private) javax.swing.ImageIcon	**associatedIcon** Icons used.
private java.lang.String	**associationPath** Association Path of the Information Field , which is displayed in the the JTree.
(package private) javax.swing.ImageIcon	**associationPathIcon** Icons used.
(package private) javax.swing.ButtonGroup	**buttonGroup1**
(package private) javax.swing.ImageIcon	**cancelIcon** Icons used.
(package private) javax.swing.ImageIcon	**CategoryIcon** Icons used.
private InformationFieldProperties.ChangeListenerImpl	**changeListener**
private org.jdom.Element	**el** The Information Field.
(package private) javax.swing.ImageIcon	**FieldIcon** Icons used.
private boolean	**hasvaluerange** Has value range property of the Field
(package private) java.awt.Color	**informationFieldAssociatedColor** Color used.
(package private) java.awt.Color	**informationFieldAssociationIncompleteColor** Color used.
(package private) java.awt.Color	**informationFieldNotAssociatedColor** Color used.

private boolean	**inputNo** The new value that the user entered in radio button No.
private boolean	**inputYes** The new value that the user entered in radio button Yes.
(package private) javax.swing.JButton	**jButtonApply** Apply button.
(package private) javax.swing.JButton	**jButtonCancel** Cancel button.
(package private) javax.swing.JButton	**jButtonOK** OK button.
(package private) javax.swing.JCheckBox	**jCheckBoxPredefined** Predefine values property of the Information Field.
(package private) javax.swing.JCheckBox	**jCheckBoxValueRange** Value range property of the Information Field.
(package private) javax.swing.JLabel	**jLabelAssociated**
(package private) javax.swing.JLabel	**jLabelAssocPath**
(package private) javax.swing.JLabel	**jLabelField**
(package private) javax.swing.JLabel	**jLabelInfoCat**
(package private) javax.swing.JLabel	**jLabelSource**
(package private) javax.swing.JLabel	**jLabelType**
(package private) javax.swing.JLabel	**jLabelXPath**
(package private) javax.swing.JPanel	**jPanel1**
(package private) javax.swing.JRadioButton	**jRadioButtonNo** Associated property of the Information Field.
(package private) javax.swing.JRadioButton	**jRadioButtonYes** Associated property of the Information Field.
(package private) javax.swing.JTextField	**jTextFieldAssocPath** Association path of the Association.
(package private) javax.swing.JTextField	**jTextFieldInfoCat** Information Category of the

	Information Field.
(package private) javax.swing.JTextField	**jTextFieldName** Name of the Information Field.
(package private) javax.swing.JTextField	**jTextFieldSource** Source where the XPath points.
(package private) javax.swing.JTextField	**jTextFieldType** Type of the Information Field.
(package private) javax.swing.JTextField	**jTextFieldXPath** XPath of the Association.
private java.lang.String	**name** Name of the Field.
private boolean	**No** If it is not associatied, then it is true.
(package private) javax.swing.ImageIcon	**okIcon** Icons used.
(package private) javax.swing.JPanel	**panel1**
private java.lang.String	**parentName** The information Category of the Field.
private javax.swing.tree.TreePath	**path** TreePath to the informationField element
private boolean	**predefined** Predefined values property of the Field
(package private) javax.swing.ImageIcon	**predefinedValuesIcon** Icons used.
private boolean	**propertiesChanged** Status variable that indicates whether or not the field is modified by user and thus needs to be saved.
private java.lang.String	**source** The file or the Data Source where the XPath point to.
(package private) org.jdom.filter.Filter	**textFilter** In JDOM both Text and CDATA are considered to be Text content.
private JDOMTreeModel	**treeModel** JDOMTreeModel of the JTree
private java.lang.String	**type** Type of the Field
(package private) javax.swing.ImageIcon	**typeIcon**

	Icons used.
(package private) javax.swing.ImageIcon	**valueRangeIcon** Icons used.
private java.lang.String	**xPath** XPath of the Information Field.
(package private) javax.swing.ImageIcon	**xPathIcon** Icons used.
(package private) com.borland.jbcl.layout.XYLayout	**xYLayout1**
(package private) com.borland.jbcl.layout.XYLayout	**xYLayout2**
private boolean	**Yes** If it is associated, then true.

Fields inherited from class javax.swing.JDialog
accessibleContext, rootPane, rootPaneCheckingEnabled

Fields inherited from class java.awt.Dialog

Fields inherited from class java.awt.Window

Fields inherited from class java.awt.Container

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface javax.swing.WindowConstants

DISPOSE_ON_CLOSE, DO_NOTHING_ON_CLOSE, EXIT_ON_CLOSE, HIDE_ON_CLOSE

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

InformationFieldProperties(java.awt.Frame frame, java.lang.String title, boolean modal, org.jdom.Element element, JDOMTreeModel tm, javax.swing.tree.TreePath p)
Constructor.

Method Summary

private void	**eraseParameters**() Erases the XPath text node and the attributes that are related to it.
private void	**jbInit**() Sets up the GUI.
(package private) void	**jButtonApply_actionPerformed**(java.awt.event.ActionEvent e) Apply button action performed.
(package private) void	**jButtonCancel_actionPerformed**(java.awt.event.ActionEvent e) Cancel button action performed.
(package private) void	**jButtonOK_actionPerformed**(java.awt.event.ActionEvent e) OK button action performed.
(package private) void	**jRadioButtonNo_actionPerformed**(java.awt.event.ActionEvent e) No Radio button action performed.
(package private) void	**jRadioButtonYes_actionPerformed**(java.awt.event.ActionEvent e) Yes Radio button action performed.
private void	**modifyInformationField**() Modifies the properties' values of the Information Field.
private void	**readInformationFieldValues**() Reads Information Field Properties' values and initializes the Dialog's variables.
private void	**readInputValues**() Reads the values entered by user and updates the class' data variables.
private void	**writeParameters**() Writes the XPath text node and the attributes that are related to it.

Methods inherited from class javax.swing.JDialog
addImpl, createRootPane, dialogInit, getAccessibleContext, getContentPane, getDefaultCloseOperation, getGlassPane, getJMenuBar, getLayeredPane, getRootPane, isDefaultLookAndFeelDecorated, isRootPaneCheckingEnabled, paramString, processWindowEvent, remove, setContentPane, setDefaultCloseOperation, setDefaultLookAndFeelDecorated, setGlassPane, setJMenuBar, setLayeredPane, setLayout, setRootPane, setRootPaneCheckingEnabled, update

Methods inherited from class java.awt.Dialog
addNotify, dispose, getTitle, hide, isModal, isResizable, isUndecorated, setModal, setResizable, setTitle, setUndecorated, show

Methods inherited from class java.awt.Window
addPropertyChangeListener, addPropertyChangeListener, addWindowFocusListener, addWindowListener, addWindowStateListener, applyResourceBundle, applyResourceBundle, createBufferStrategy, createBufferStrategy, finalize, getBufferStrategy, getFocusableWindowState, getFocusCycleRootAncestor, getFocusOwner, getFocusTraversalKeys, getGraphicsConfiguration, getInputContext, getListeners, getLocale, getMostRecentFocusOwner, getOwnedWindows, getOwner, getToolkit, getWarningString, getWindowFocusListeners, getWindowListeners, getWindowStateListeners, isActive, isFocusableWindow, isFocusCycleRoot, isFocused, isShowing, pack, postEvent, processEvent, processWindowFocusEvent, processWindowStateEvent, removeWindowFocusListener, removeWindowListener, removeWindowStateListener, setCursor, setFocusableWindowState, setFocusCycleRoot, setLocationRelativeTo, toBack, toFront

Methods inherited from class java.awt.Container
add, add, add, add, add, addContainerListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getContainerListeners, getFocusTraversalPolicy, getInsets, getLayout, getMaximumSize, getMinimumSize, getPreferredSize, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paint, paintComponents, preferredSize, print, printComponents, processContainerEvent, remove, removeAll, removeContainerListener, removeNotify, setFocusTraversalKeys, setFocusTraversalPolicy, setFont, transferFocusBackward, transferFocusDownCycle, validate, validateTree

Methods inherited from class java.awt.Component
action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, coalesceEvents, contains, contains, createImage, createImage, createVolatileImage, createVolatileImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, getBackground, getBounds, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphics, getHeight, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocation, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isOpaque, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, prepareImage, prepareImage, printAll, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processKeyEvent, processMouseEvent, processMouseMotionEvent, processMouseWheelEvent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, reshape, resize, resize, setBackground, setBounds, setBounds, setComponentOrientation, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeysEnabled, setForeground, setIgnoreRepaint, setLocale, setLocation, setLocation, setName, setSize, setSize, setVisible, show, size, toString, transferFocus, transferFocusUpCycle

Methods inherited from class java.lang.Object
clone, equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

applyIcon

javax.swing.ImageIcon **applyIcon**

Icons used.

---

associatedIcon

`javax.swing.ImageIcon` **`associatedIcon`**

Icons used.

---

associationPath

`private java.lang.String` **`associationPath`**

Association Path of the Information Field , which is displayed in the the JTree.

---

associationPathIcon

`javax.swing.ImageIcon` **`associationPathIcon`**

Icons used.

---

buttonGroup1

`javax.swing.ButtonGroup` **`buttonGroup1`**

---

cancelIcon

`javax.swing.ImageIcon` **`cancelIcon`**

Icons used.

CategoryIcon

`javax.swing.ImageIcon` **`CategoryIcon`**

Icons used.

---

changeListener

`private` `InformationFieldProperties.ChangeListenerImpl` **`changeListener`**

---

el

`private org.jdom.Element` **`el`**

The Information Field. The "data" of this class.

FieldIcon

`javax.swing.ImageIcon` **`FieldIcon`**

Icons used.

---

hasvaluerange

`private boolean` **`hasvaluerange`**

Has value range property of the Field

---

informationFieldAssociatedColor

`java.awt.Color` **`informationFieldAssociatedColor`**

Color used.

---

informationFieldAssociationIncompleteColor

`java.awt.Color` **`informationFieldAssociationIncompleteColor`**

Color used.

---

informationFieldNotAssociatedColor

`java.awt.Color` **`informationFieldNotAssociatedColor`**

Color used.

---

inputNo

`private boolean` **`inputNo`**

The new value that the user entered in radio button No. The user input here is kept in different variable in order to know if save is needed on exit.

See Also:

`modifyInformationField()`

---

inputYes

`private boolean` **`inputYes`**

The new value that the user entered in radio button Yes. The user input here is kept in different variable in order to know if save is needed on exit.

See Also:

`modifyInformationField()`

---

jButtonApply

`javax.swing.JButton` **`jButtonApply`**

Apply button.

---

jButtonCancel

`javax.swing.JButton` **`jButtonCancel`**

Cancel button.

---

jButtonOK

`javax.swing.JButton` **`jButtonOK`**

OK button.

---

jCheckBoxPredefined

`javax.swing.JCheckBox` **`jCheckBoxPredefined`**

Predefine values property of the Information Field.

---

jCheckBoxValueRange
`javax.swing.JCheckBox` **`jCheckBoxValueRange`**

Value range property of the Information Field.

---

jLabelAssociated
`javax.swing.JLabel` **`jLabelAssociated`**

---

jLabelAssocPath
`javax.swing.JLabel` **`jLabelAssocPath`**

---

jLabelField
`javax.swing.JLabel` **`jLabelField`**

---

jLabelInfoCat
`javax.swing.JLabel` **`jLabelInfoCat`**

---

jLabelSource
`javax.swing.JLabel` **`jLabelSource`**

---

jLabelType
`javax.swing.JLabel` **`jLabelType`**

---

jLabelXPath
`javax.swing.JLabel` **`jLabelXPath`**

---

jPanel1
`javax.swing.JPanel` **`jPanel1`**

---

jRadioButtonNo
`javax.swing.JRadioButton` **`jRadioButtonNo`**

Associated property of the Information Field.

---

jRadioButtonYes
`javax.swing.JRadioButton` **`jRadioButtonYes`**

Associated property of the Information Field.

---

jTextFieldAssocPath

`javax.swing.JTextField` **`jTextFieldAssocPath`**

Association path of the Association. This is the path displayed in the JTree. It also is refered as display path.

---

jTextFieldInfoCat

`javax.swing.JTextField` **`jTextFieldInfoCat`**

Information Category of the Information Field.

---

jTextFieldName

`javax.swing.JTextField` **`jTextFieldName`**

Name of the Information Field.

---

jTextFieldSource

`javax.swing.JTextField` **`jTextFieldSource`**

Source where the XPath points.

---

jTextFieldType

`javax.swing.JTextField` **`jTextFieldType`**

Type of the Information Field.

---

jTextFieldXPath

`javax.swing.JTextField` **`jTextFieldXPath`**

XPath of the Association.

---

name

`private java.lang.String` **`name`**

Name of the Field.

No

`private boolean` **`No`**

If it is not associatied, then it is true.

---

okIcon

`javax.swing.ImageIcon` **`okIcon`**

Icons used.

---

panel1

`javax.swing.JPanel` **`panel1`**

---

parentName
`private java.lang.String` **`parentName`**

The information Category of the Field.

---

path
`private javax.swing.tree.TreePath` **`path`**

TreePath to the informationField element

---

predefined
`private boolean` **`predefined`**

Predefined values property of the Field

---

predefinedValuesIcon
`javax.swing.ImageIcon` **`predefinedValuesIcon`**

Icons used.

---

propertiesChanged
`private boolean` **`propertiesChanged`**

Status variable that indicates whether or not the field is modified by user and thus needs to be saved.

---

source
`private java.lang.String` **`source`**

The file or the Data Source where the XPath point to.

---

textFilter
`org.jdom.filter.Filter` **`textFilter`**

In JDOM both Text and CDATA are considered to be Text content. So content is filtered to be only Text.

---

treeModel
`private` `JDOMTreeModel` **`treeModel`**

JDOMTreeModel of the JTree

---

type
`private java.lang.String` **`type`**

Type of the Field

---

typeIcon

`javax.swing.ImageIcon` **`typeIcon`**

Icons used.

---

valueRangeIcon

`javax.swing.ImageIcon` **`valueRangeIcon`**

Icons used.

---

xPath

`private java.lang.String` **`xPath`**

XPath of the Information Field.

---

xPathIcon

`javax.swing.ImageIcon` **`xPathIcon`**

Icons used.

---

xYLayout1

`com.borland.jbcl.layout.XYLayout` **`xYLayout1`**

---

xYLayout2

`com.borland.jbcl.layout.XYLayout` **`xYLayout2`**

Yes

`private boolean` **`Yes`**

If it is associated, then true.

Constructor Detail

InformationFieldProperties

```
public InformationFieldProperties(java.awt.Frame frame,
                                  java.lang.String title,
                                  boolean modal,
                                  org.jdom.Element element,
                                  JDOMTreeModel tm,
                                  javax.swing.tree.TreePath p)
```

Constructor.

Does an initialization and then calls the jbInit method to set up the GUI.

Parameters:

`frame` - the parent frame, that is the AdminToolFrame

`title` - the title

`modal` - modal option

`element` - the informationField element.

`tm` - the JDOMTreeModel of the JTree

`p` - the TreePath to the informationField element.

Method Detail

eraseParameters

```
private void eraseParameters()
```

Erases the XPath text node and the attributes that are related to it.
Helper method used by modifyInformationField method. Finds the Text node and passes it to the JDOMTreeModel.removeNodeFromParent method to do the removal.

See Also:
`JDOMTreeModel.removeNodeFromParent(Text, TreePath)`

---

jbInit

```
private void jbInit()
            throws java.lang.Exception
```

Sets up the GUI.
Sets the components' properties and lays them out. Change Listeners and Document Listeners are added to the GUI componets, so as to know when data are modified.
Most of this code is automatically generated by JBuilder. Especially the laying out of the components.
This is the JBuilders method for setting up the GUI. It is necessary to use this method it order to be able to modify the GUI from JBuilder's Design view.

Throws:
`java.lang.Exception`

---

jButtonApply_actionPerformed

```
void jButtonApply_actionPerformed(java.awt.event.ActionEvent e)
```

Apply button action performed.

- Asks whether to save modifications to Information field or not.
- If Yes:
  - Read values that user entered
  - Modify field
  - Change status variable which indicates that field is modified
- if Cancel, do nothing.

Parameters:
`e` - ActionEvent

See Also:
`readInputValues()`, `modifyInformationField()`

---

jButtonCancel_actionPerformed

```
void jButtonCancel_actionPerformed(java.awt.event.ActionEvent e)
```

Cancel button action performed.
Close the Dialog window discarding changes.

Parameters:
`e` - ActionEvent

---

jButtonOK_actionPerformed

`void jButtonOK_actionPerformed(java.awt.event.ActionEvent e)`

OK button action performed.

- Infrmation Field Properties are changed, ask if whether to save or not:
  - if yes:
    - read values entered by user
    - update field
    - exit dialog
  - if cancel: do nothing
  - if no: exit dialog without saving
- Infrmation Field Properties are not changed: just exit

Parameters:
`e` - ActionEvent

---

jRadioButtonNo_actionPerformed

`void jRadioButtonNo_actionPerformed(java.awt.event.ActionEvent e)`

No Radio button action performed.
Disables Association path, XPath and Source text field for user to manually add values.
Parameters:
`e` - ActionEvent

---

jRadioButtonYes_actionPerformed

`void jRadioButtonYes_actionPerformed(java.awt.event.ActionEvent e)`

Yes Radio button action performed.
Enables Association path and XPath text field for user to manually add values.
Parameters:
`e` - ActionEvent

---

modifyInformationField

`private void modifyInformationField()`

Modifies the properties' values of the Information Field.
There are four cases of modifing the Association of an Information Field.

- Field was not associated and user let it as it was (No -> No): Do nothing (case 0)
- Field was not associated and user associated it (No - Yes): Write new parameters (case 1)
- Field was associated and user removed the association (Yes -> No): Erase parameters (case 2)
- Field was associated and whether user let it as it was or modified the association (Yes -> Yes): Erase the old parameters, write new (case 3)

where parameters are: associated, associationPath, XPath

---

readInformationFieldValues

`private void readInformationFieldValues()`

Reads Information Field Properties' values and initializes the Dialog's variables. When reading XPath the following must be considered: In JDOM both Text and CDATA are considered to be Text content. So when filter content to be only Text.

---

readInputValues

`private void readInputValues()`

Reads the values entered by user and updates the class' data variables. The association status of the Information Field is kept in inputNo and inputYes variables in order to know if save is needed on exit.

See Also:

`modifyInformationField()`

---

writeParameters

`private void writeParameters()`

Writes the XPath text node and the attributes that are related to it.

Helper method used by modifyInformationField method. Creates a dummy Element in which it puts the Text node and the relative attributes. Then it calls the insertNodeInto to do the insertion. The dummy node is used because insertNodeInto was design to take an Element.

See Also:

`JDOMTreeModel.insertNodeInto(Element, Element, int, TreePath)`

Overview Package Class Tree Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES All Classes

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

---

admintool

Class ReportViewer

```
java.lang.Object
  └java.awt.Component
      └java.awt.Container
          └java.awt.Window
              └java.awt.Dialog
```

└ javax.swing.JDialog
└ **admintool.ReportViewer**

All Implemented Interfaces:

javax.accessibility.Accessible, java.awt.image.ImageObserver, java.awt.MenuContainer, javax.swing.RootPaneContainer, java.io.Serializable, javax.swing.WindowConstants

---

public class **ReportViewer**
extends javax.swing.JDialog

A Dialog for dispaying Report on Information Structure, XML of Information Structure and XML of a selection in Data Structure.

It is called on comands:

- View Report on Information Structure
- View XML of Information Structure
- View XML of selection in Data Structure

The term *Report* is refered to the html generated when the XSL file is applied to the XML of the Information Structure.

See Also:

Serialized Form

Nested Class Summary

(package private) class	**ReportViewer.ReportViewer_jButtonOK_actionAdapter** Helper inner class created automatically by development tool.

Nested classes inherited from class javax.swing.JDialog
javax.swing.JDialog.AccessibleJDialog

Nested classes inherited from class java.awt.Dialog
java.awt.Dialog.AccessibleAWTDialog

Nested classes inherited from class java.awt.Window
java.awt.Window.AccessibleAWTWindow

Nested classes inherited from class java.awt.Container
java.awt.Container.AccessibleAWTContainer

Nested classes inherited from class java.awt.Component
java.awt.Component.AccessibleAWTComponent, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Field Summary

(package private) java.awt.BorderLayout	**borderLayout1**
(package private) java.awt.BorderLayout	**borderLayout2**
(package private) java.awt.BorderLayout	**borderLayout3**
private org.jdom.Document	**doc** The displayed document.
(package private) java.awt.Font	**editorFont** Editor pane's font.
private java.io.File	**file** The temp file used for displaying in the JEditorPane.
(package private) javax.swing.JButton	**jButtonOK** OK button.
(package private) javax.swing.JEditorPane	**jEditorPane**
(package private) javax.swing.JLabel	**jLabel1**
(package private) javax.swing.JPanel	**jPanel1**
(package private) javax.swing.JPanel	**jPanel2**
(package private) javax.swing.JPanel	**jPanel3**
(package private) javax.swing.JTextField	**jTextFieldDisplaying** Displaying Bar.
(package private) javax.swing.ImageIcon	**okIcon** Icon used
(package private) javax.swing.JPanel	**panel1**
private AdminToolFrame	**parent**

	The parent frame.
private boolean	**report** Indicates whether or not it is a report.
(package private) javax.swing.border.TitledBorder	**titledBorder1**
private java.io.File	**xslFile** The XSL file used to generate the html report from the Information Structure's XML.

Fields inherited from class javax.swing.JDialog
accessibleContext, rootPane, rootPaneCheckingEnabled

Fields inherited from class java.awt.Dialog

Fields inherited from class java.awt.Window

Fields inherited from class java.awt.Container

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface javax.swing.WindowConstants
DISPOSE_ON_CLOSE, DO_NOTHING_ON_CLOSE, EXIT_ON_CLOSE, HIDE_ON_CLOSE

Fields inherited from interface java.awt.image.ImageObserver

ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

ReportViewer(java.awt.Frame frame, java.lang.String title, boolean modal, org.jdom.Document doc, boolean report) Constructor.

Method Summary

private void	**createReport**() Creates the *.html report document from the XML and the XSL files.
private void	**displayDocument**() Displays the document.
private void	**jbInit**() Set up GUI.
(package private) void	**jButtonOK_actionPerformed**(java.awt.event.ActionEvent e) OK button action performed.
private void	**setDisplayingBar**() Sets the text of the Displaying Bar.

Methods inherited from class javax.swing.JDialog

addImpl, createRootPane, dialogInit, getAccessibleContext, getContentPane, getDefaultCloseOperation, getGlassPane, getJMenuBar, getLayeredPane, getRootPane, isDefaultLookAndFeelDecorated, isRootPaneCheckingEnabled, paramString, processWindowEvent, remove, setContentPane, setDefaultCloseOperation, setDefaultLookAndFeelDecorated, setGlassPane, setJMenuBar, setLayeredPane, setLayout, setRootPane, setRootPaneCheckingEnabled, update

Methods inherited from class java.awt.Dialog

addNotify, dispose, getTitle, hide, isModal, isResizable, isUndecorated, setModal, setResizable, setTitle, setUndecorated, show

Methods inherited from class java.awt.Window

addPropertyChangeListener, addPropertyChangeListener, addWindowFocusListener, addWindowListener, addWindowStateListener,

applyResourceBundle, applyResourceBundle, createBufferStrategy, createBufferStrategy, finalize, getBufferStrategy, getFocusableWindowState, getFocusCycleRootAncestor, getFocusOwner, getFocusTraversalKeys, getGraphicsConfiguration, getInputContext, getListeners, getLocale, getMostRecentFocusOwner, getOwnedWindows, getOwner, getToolkit, getWarningString, getWindowFocusListeners, getWindowListeners, getWindowStateListeners, isActive, isFocusableWindow, isFocusCycleRoot, isFocused, isShowing, pack, postEvent, processEvent, processWindowFocusEvent, processWindowStateEvent, removeWindowFocusListener, removeWindowListener, removeWindowStateListener, setCursor, setFocusableWindowState, setFocusCycleRoot, setLocationRelativeTo, toBack, toFront

Methods inherited from class java.awt.Container

add, add, add, add, add, addContainerListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getContainerListeners, getFocusTraversalPolicy, getInsets, getLayout, getMaximumSize, getMinimumSize, getPreferredSize, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paint, paintComponents, preferredSize, print, printComponents, processContainerEvent, remove, removeAll, removeContainerListener, removeNotify, setFocusTraversalKeys, setFocusTraversalPolicy, setFont, transferFocusBackward, transferFocusDownCycle, validate, validateTree

Methods inherited from class java.awt.Component

action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, coalesceEvents, contains, contains, createImage, createImage, createVolatileImage, createVolatileImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, getBackground, getBounds, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphics, getHeight, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocation, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isOpaque, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, prepareImage, prepareImage, printAll, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent,

processInputMethodEvent, processKeyEvent, processMouseEvent, processMouseMotionEvent, processMouseWheelEvent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, reshape, resize, resize, setBackground, setBounds, setBounds, setComponentOrientation, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeysEnabled, setForeground, setIgnoreRepaint, setLocale, setLocation, setLocation, setName, setSize, setSize, setVisible, show, size, toString, transferFocus, transferFocusUpCycle

Methods inherited from class java.lang.Object
clone, equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

borderLayout1
java.awt.BorderLayout **borderLayout1**

---

borderLayout2
java.awt.BorderLayout **borderLayout2**

---

borderLayout3
java.awt.BorderLayout **borderLayout3**

---

doc
private org.jdom.Document **doc**

The displayed document.

---

editorFont
java.awt.Font **editorFont**

Editor pane's font.

---

file
private java.io.File **file**

The temp file used for displaying in the JEditorPane. JEditorPane displays a page by loading a file. So in order to display the documents, they are first outputted with XMLOutputter into a file and then they are loaded into the JEditorPane as files.

jButtonOK
`javax.swing.JButton` **`jButtonOK`**

OK button.

---

jEditorPane
`javax.swing.JEditorPane` **`jEditorPane`**

---

jLabel1
`javax.swing.JLabel` **`jLabel1`**

---

jPanel1
`javax.swing.JPanel` **`jPanel1`**

---

jPanel2
`javax.swing.JPanel` **`jPanel2`**

---

jPanel3
`javax.swing.JPanel` **`jPanel3`**

---

jTextFieldDisplaying
`javax.swing.JTextField` **`jTextFieldDisplaying`**

Displaying Bar. Shows what is cirrently been displayed.

---

okIcon
`javax.swing.ImageIcon` **`okIcon`**

Icon used

---

panel1
`javax.swing.JPanel` **`panel1`**

---

parent
`private` `AdminToolFrame` **`parent`**

The parent frame.

---

report
`private boolean` **`report`**

Indicates whether or not it is a report.

---

titledBorder1
```
javax.swing.border.TitledBorder titledBorder1
```

---

xslFile
```
private java.io.File xslFile
```
The XSL file used to generate the html report from the Information Structure's XML.

Constructor Detail

ReportViewer
```
public ReportViewer(java.awt.Frame frame,
                    java.lang.String title,
                    boolean modal,
                    org.jdom.Document doc,
                    boolean report)
```
Constructor.
Does an initialization, then calls the jbInit method to set up the GUI and finally calls the proper mrthod to do the displaying. Also, sets the XSL file used to display the report.

The constructor takes only a document. So, when the "View xml of selection in Data Structure" command is enetered, the element that was selected by the user is passed to the constructor through a dummy document.

Parameters:

`frame` - parent frame
`title` - title
`modal` - modal
`doc` - the document that is going to be displayed.
`report` - True if it is a report (html) and false if it is just XML.

Method Detail

createReport
```
private void createReport()
```
Creates the *.html report document from the XML and the XSL files. Loads the XSL file into a document and transforms the document using an org.jdom.transform.XSLTransformer.
See Also:
`XSLTransformer`

---

displayDocument
```
private void displayDocument()
```
Displays the document.
JEditorPane displays a page loading a file. So in order to display the documents, they are first outputted with XMLOutputter into a file and then they are loaded into the

JEditorPane as files. When a report is to be displayed, the document is first transformed using createReport method.

See Also:
`createReport()`

---

jbInit

```
private void jbInit()
             throws java.lang.Exception
```

Set up GUI.

Sets the components' properties and lays them out.

Most of this code is automatically generated by JBuilder. Especially the laying out of the components.

This is the JBuilders method for setting up the GUI. It is necessary to use this method it order to be able to modify the GUI from JBuilder's Design view.

Throws:
`java.lang.Exception`

---

jButtonOK_actionPerformed

```
void jButtonOK_actionPerformed(java.awt.event.ActionEvent e)
```

OK button action performed.

Deletes the temp file and exits.

Parameters:
`e` - ActionEvent

---

setDisplayingBar

```
private void setDisplayingBar()
```

Sets the text of the Displaying Bar.

Overview **Package** **Class** **Tree** **Index** **Help**

PREV CLASS NEXT CLASS FRAMES NO FRAMES All Classes

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

---

admintool

Class TreeTransferHandler

```
java.lang.Object
  └javax.swing.TransferHandler
      └admintool.TreeTransferHandler
```
All Implemented Interfaces:
java.io.Serializable

---

public class **TreeTransferHandler**
extends javax.swing.TransferHandler

The TransferHanlder that provides the support for associating nodes in the JTree through Drag and Drop (DnD).

This is the class that provides the DnD support of the application. After J2SE 1.4, many Swing components provide out-of-the-box support for transferring data. The only thing need to be done is to provide to the application a custom implementation of the TransferHandler class. Both JTree must have this class to have DnD support.

This class is a custom implementation of the TransferHandler for the AdminTool, and does the following: When dragging an node of the source JTree (Data Structure tree), its XPath and related properties are tranfered to the destination JTree (Informaion Structure tree). On drop, the XPath is appended to the target Element a Text node. The rest of the tranfered properties are also appended to the target.

Tip: For more see The Java Tutorial: Creating a GUI with JFC/Swing: How to Use Drag and Drop and Data Transfer
See Also:
Serialized Form

Nested Class Summary

Nested classes inherited from class javax.swing.TransferHandler

Field Summary

(package private) DisplayedPathEvaluator	**displayedPathEvaluator** Estimates the associaion path or displayed path of the dragged element.
(package private) java.awt.datatransfer.DataFlavor	**elementFlavor** Element Flavor, that is a DataFlavor for org.jdom.Element, used in transfering.
(package private) java.lang.String	**elementType** The string used to identify the MIME type for the Element flavor.

(package private) javax.swing.JTree	**sourceTree** Source JTree.
(package private) javax.swing.JTree	**targetTree** Target JTree.
(package private) XPathEvaluator	**xPathEvaluator** Estimates the XPath of the dragged element.

Fields inherited from class javax.swing.TransferHandler
COPY, COPY_OR_MOVE, MOVE, NONE

Constructor Summary

TreeTransferHandler()
Constructor.

Method Summary

boolean	**canImport**(javax.swing.JComponent c, java.awt.datatransfer.DataFlavor[] flavors) Indicates whether the JTree that had the drop, would accept an import of the given set of data flavors prior to actually attempting to import it.
protected java.awt.datatransfer.Transferable	**createTransferable**(javax.swing.JComponent c) Creates the transfered data when a Drag occurs.
void	**exportAsDrag**(javax.swing.JComponent comp, java.awt.event.InputEvent e, int action) Causes the Swing drag support to be initiated.
protected void	**exportDone**(javax.swing.JComponent source, java.awt.datatransfer.Transferable data, int action) Invoked after data has been exported.
int	**getSourceActions**(javax.swing.JComponent c) Returns the type of transfer action (COPY) supported by the source.
private java.lang.String	**getSourceName**(java.lang.Object object) Gets the data source or file loaded to data

	structure tree and returns its name.
boolean	**importData**(javax.swing.JComponent c, java.awt.datatransfer.Transferable t) Imports the transfered data to the drop target.

Methods inherited from class javax.swing.TransferHandler
exportToClipboard, getCopyAction, getCutAction, getPasteAction, getVisualRepresentation

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

displayedPathEvaluator
DisplayedPathEvaluator **displayedPathEvaluator**

Estimates the associaion path or displayed path of the dragged element.

---

elementFlavor
java.awt.datatransfer.DataFlavor **elementFlavor**

Element Flavor, that is a DataFlavor for org.jdom.Element, used in transfering.

---

elementType
java.lang.String **elementType**

The string used to identify the MIME type for the Element flavor.

---

sourceTree
javax.swing.JTree **sourceTree**

Source JTree. Where the Drag occured.

---

targetTree
javax.swing.JTree **targetTree**

Target JTree. Where the Drop occured.

---

xPathEvaluator
`XPathEvaluator` **`xPathEvaluator`**

Estimates the XPath of the dragged element.

Constructor Detail

TreeTransferHandler
`public` **`TreeTransferHandler`**`()`

Constructor. Does the initialization.

Method Detail

canImport
```
public boolean canImport(javax.swing.JComponent c,
                         java.awt.datatransfer.DataFlavor[] flavors)
```

Indicates whether the JTree that had the drop, would accept an import of the given set of data flavors prior to actually attempting to import it.
When a JTree has a drop on it, calls this method to find out that the data flavors of the Transferable object matches the data flavor that it supports.
This method answers the question: *Has the target JTree support for this kind of dropped data?*

Parameters:

`c` - the component to receive the transfer

`flavors` - the data formats available

Returns:

true if the data can be inserted into the component, false otherwise

---

createTransferable
```
protected java.awt.datatransfer.Transferable
createTransferable(javax.swing.JComponent c)
```

Creates the transfered data when a Drag occurs.

Creates a Transferable to use as the source for a data transfer. Returns the representation of the data to be transferred, or null if the component's property is null.
The created Transferable is an instance of XMLTransferable for transfering org.jdom.Element. To make the application expandible, it was best considered that the Transferable should transfer an Element. The reason is that although now only one Text node and two attributes need to be transfered, in future an entire subtree might needed. So, by transfering an Element you can transfer everything else this element is parent of.

- Ensures that only JTrees can initiate a drag
- Gets selected object. Cases of selected object:
  - case of: Element
    - Check if selected element is root of data structure. if yes then abort - user cannot DnD root
    - create an Element to transfer the data
    - Estimates displayedPath and XPath of selected object.
    - Adds XPath to selected object

        - Adds association path to selected object
            - if the selected object is a data source field, add association path
            - otherwise the source tree displays a file structure then association path is the same with the XPath
        - Adds source, that is the data source name or the file name that XPath and Assocaion Path refers to.
        - return an XMLTransferable containing the Element
    - case of: Attribute
        - same actions are performed

Parameters:
c - the component holding the data to be transferred; this argument is provided to enable sharing of TransferHandlers by multiple components
Returns:
the representation of the data to be transferred (XMLTransferable), or null if the property associated with c is null
See Also:
XMLTransferable

---

exportAsDrag

```
public void exportAsDrag(javax.swing.JComponent comp,
                         java.awt.event.InputEvent e,
                         int action)
```

Causes the Swing drag support to be initiated.
Just calls the superclass' method. See TransferHandler API for more.
Parameters:
comp - the component holding the data to be transferred; this argument is provided to enable sharing of TransferHandlers by multiple components
e - the event that triggered the transfer
action - the transfer action initially requested; this should be a value of either COPY or MOVE; the value may be changed during the course of the drag operation
See Also:
TransferHandler

---

exportDone

```
protected void exportDone(javax.swing.JComponent source,
                          java.awt.datatransfer.Transferable data,
                          int action)
```

Invoked after data has been exported. Dummy implementation. It is not needed for COPY. Only for MOVE to remove the dragged data from source.
Parameters:
source - the component that was the source of the data
data - The data that was transferred or possibly null if the action is NONE.
action - the actual action that was performed

---

getSourceActions

```
public int getSourceActions(javax.swing.JComponent c)
```

Returns the type of transfer action (COPY) supported by the source. The source JTree does not need to modified so a transfer operation of COPY only should be advertised in our case.

Parameters:
`c` - JComponent
Returns:
COPY action only

---

getSourceName

```
private java.lang.String getSourceName(java.lang.Object object)
```

Gets the data source or file loaded to data structure tree and returns its name.
The name of the source is stored in the attribute of the corresponding element.

Parameters:
`object` - the data source or file loaded to data structure tree
Returns:
name of the source

---

importData

```
public boolean importData(javax.swing.JComponent c,
                          java.awt.datatransfer.Transferable t)
```

Imports the transfered data to the drop target.

Causes a transfer to a component from a DND drop operation. The Transferable represents the data to be imported into the component.

- Checks if import can be performed, by calling canImport method.
- Gets the transfered element from the Transferable
- Gets selection path of target tree and then the target object of the target JTree
- If target object is a informationfield element, then delegate the insertion of the XPath Text node and the rest of the data to tree model

Parameters:
`c` - he component to receive the transfer; this argument is provided to enable sharing of TransferHandlers by multiple components
`t` - he data to import
Returns:
if the data was inserted into the component, false otherwise
See Also:
`XMLTransferable`

---

admintool

Class XMLTransferable

```
java.lang.Object
  └admintool.XMLTransferable
```

All Implemented Interfaces:

java.awt.datatransfer.Transferable

public class **XMLTransferable**
extends java.lang.Object
implements java.awt.datatransfer.Transferable

Transferable interface implementation that supports the transfer of org.jdom.Element objects. Every custom TransferHandler implementation needs an implementation of Transferable interface to wrap the transfered object. This class is an implementation that supports the transfer of org.jdom.Element objects.

Field Summary

(package private) org.jdom.Element	**dataElement** Element that is transfered.
(package private) java.awt.datatransfer.DataFlavor	**elementFlavor** Element Flavor.

Constructor Summary

XMLTransferable(java.lang.Object o, java.awt.datatransfer.DataFlavor elFl) Constructor.

Method Summary

java.lang.Object	**getTransferData**(java.awt.datatransfer.DataFlavor flavor)

	Returns the transfered Element.
java.awt.datatransfer.DataFlavor[]	**getTransferDataFlavors**() Returns an array with the Element Flavor.
boolean	**isDataFlavorSupported**(java.awt.datatransfer.DataFlavor flavor) Returns whether or not the Element flavor is supported for this object.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

dataElement
org.jdom.Element **dataElement**

Element that is transfered.

---

elementFlavor
java.awt.datatransfer.DataFlavor **elementFlavor**

Element Flavor. Flavor of the transfered data.

Constructor Detail

XMLTransferable
public **XMLTransferable**(java.lang.Object o,
java.awt.datatransfer.DataFlavor elFl)

Constructor. Does the initialization. DataFlavor is also passed to constructor in order to be easier to add support of more flavors in the future .

Parameters:

o - the transfered data.

elFl - the DataFlavor of the transfered data

Method Detail

getTransferData
public java.lang.Object
getTransferData(java.awt.datatransfer.DataFlavor flavor)
throws
java.awt.datatransfer.UnsupportedFlavorException

Returns the transfered Element. The class of the object returned is defined by the representation class of the flavor.

Specified by:
`getTransferData` in interface `java.awt.datatransfer.Transferable`
Parameters:
`flavor` - DataFlavor
Returns:
the transfered object
Throws:
`java.awt.datatransfer.UnsupportedFlavorException`

---

getTransferDataFlavors

`public java.awt.datatransfer.DataFlavor[] getTransferDataFlavors()`

Returns an array with the Element Flavor.
Specified by:
`getTransferDataFlavors` in interface `java.awt.datatransfer.Transferable`
Returns:
array with the Element Flavor.[]

---

isDataFlavorSupported

`public boolean isDataFlavorSupported(java.awt.datatransfer.DataFlavor flavor)`

Returns whether or not the Element flavor is supported for this object.
Specified by:
`isDataFlavorSupported` in interface `java.awt.datatransfer.Transferable`
Parameters:
`flavor` - the DataFlavor of the object.
Returns:
whether or not the Element flavor is supported for this object.

Overview Package Class Tree Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES All Classes
SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

---

admintool

Interface XMLTransformer

public interface **XMLTransformer**

This is the interface that XML_JDOMTransformLoader class needs a class to implement, in order for that class to be considered a Transformation class.
It is used becasue there is not a priory knowledge of what transformer classes might be available in the future.

Method Summary

`void`	**`setName`**`(java.lang.String displayName)` Sets the name of the class returned with toString method.
`java.lang.String`	**`toString`**`()` Returns the name of the class displayed the XML_JDOMTransformer's comboBox.
`org.jdom.Document`	**`transform`**`(org.jdom.Document inputDoc)` Transforms a document into another document.

Method Detail

setName

`public void` **`setName`**`(java.lang.String displayName)`

Sets the name of the class returned with toString method.

Parameters:

`displayName` - the name of the class

toString

`public java.lang.String` **`toString`**`()`

Returns the name of the class displayed the XML_JDOMTransformer's comboBox.

Returns:

The transformation class name

transform

`public org.jdom.Document` **`transform`**`(org.jdom.Document inputDoc)`

Transforms a document into another document. Call that method to do the transformation.

Parameters:

`inputDoc` - document that is going to be transformed

Returns:

the transformed document

Overview Package Class Tree Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES All Classes
SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

admintool

Class XML_JDOMTransformLoader

```
java.lang.Object
  └java.awt.Component
      └java.awt.Container
          └java.awt.Window
              └java.awt.Dialog
                  └javax.swing.JDialog
                      └admintool.XML_JDOMTransformLoader
```

All Implemented Interfaces:

javax.accessibility.Accessible, java.awt.image.ImageObserver, java.awt.MenuContainer, javax.swing.RootPaneContainer, java.io.Serializable, javax.swing.WindowConstants

---

public class **XML_JDOMTransformLoader**
extends javax.swing.JDialog

Load generic XML file through JDOM transformation Dialog.
Prompts the user to choose a a generic XML and a transformation class Xto transform the generic XML in the appropriate structure, and then does the transform. The transformed the document is returned to the AdminToolFrame with a getter method.

See Also:

Serialized Form

Nested Class Summary

(package private) class	**XML_JDOMTransformLoader.XML_JDOMTransformLoader_jButtonCancel_actionAdapter** Helper inner class created automatically by development tool.
(package	**XML_JDOMTransformLoader.XML_JDOMTransformLoader_jButtonOK_actionAdap**

private) class	**ter** Helper inner class created automatically by development tool.
(package private) class	**XML_JDOMTransformLoader.XML_JDOMTransformLoader_jButtonXML_actionAdapter** Helper inner class created automatically by development tool.

Nested classes inherited from class javax.swing.JDialog
javax.swing.JDialog.AccessibleJDialog

Nested classes inherited from class java.awt.Dialog
java.awt.Dialog.AccessibleAWTDialog

Nested classes inherited from class java.awt.Window
java.awt.Window.AccessibleAWTWindow

Nested classes inherited from class java.awt.Container
java.awt.Container.AccessibleAWTContainer

Nested classes inherited from class java.awt.Component
java.awt.Component.AccessibleAWTComponent, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Field Summary

(package private) java.awt.BorderLayout	**borderLayout1**
(package private) java.awt.BorderLayout	**borderLayout2**
(package private) javax.swing.ImageIcon	**cancelIcon** Icons used.
(package private) java.awt.FlowLayout	**flowLayout1**

(package private) java.awt.FlowLayout	**flowLayout2**
(package private) javax.swing.ImageIcon	**folderIcon** Icons used.
(package private) java.awt.GridLayout	**gridLayout1**
(package private) javax.swing.JButton	**jButtonCancel** Cancel button.
(package private) javax.swing.JButton	**jButtonOK** OK button.
(package private) javax.swing.JButton	**jButtonXML** Choose a generic XML file button.
(package private) javax.swing.JComboBox	**jComboBoxTransformers** Combo box with the transformation classes
(package private) javax.swing.ImageIcon	**jdomIcon** Icons used.
(package private) javax.swing.JFileChooser	**jFileChooserXMLXSLT** The file chooser used in this dialog.
(package private) javax.swing.JLabel	**jLabelJDOMTransformClass**
(package private) javax.swing.JLabel	**jLabelPath1**
(package private) javax.swing.JLabel	**jLabelTitle**
(package private) javax.swing.JLabel	**jLabelXML**
(package private) javax.swing.JPanel	**jPanelButtons**
(package private) javax.swing.JPanel	**jPanelGrid**
(package private) javax.swing.JPanel	**jPanelJDOM**
(package private) javax.swing.JPanel	**jPanelMain**
(package private) javax.swing.JPanel	**jPanelTitle**
(package private) javax.swing.JPanel	**jPanelXML**
(package private) javax.swing.JTextField	**jTextFieldPath1** File path of XML file.
(package private) javax.swing.ImageIcon	**okIcon** Icons used.
(package private) javax.swing.JPanel	**panel1**

private AdminToolFrame	**parent** Parent frame.
private java.io.File	**propertiesFile** The Properties file which indicates which classes must be loaded.
private org.jdom.Document	**transformedDoc** The transformed document.
private org.jdom.Document	**xmlDoc** Generic XML file.
(package private) FilesFilter	**xmlFileFilter** File filter for xml files *.xml used in JFileChooser
(package private) javax.swing.ImageIcon	**xmlIcon** Icons used.
private java.util.Vector	**xmlTransformersList** The transformation classes list that the combo box displays.
(package private) com.borland.jbcl.layout.XYLayout	**xYLayout2**
(package private) com.borland.jbcl.layout.XYLayout	**xYLayout3**

Fields inherited from class javax.swing.JDialog
accessibleContext, rootPane, rootPaneCheckingEnabled

Fields inherited from class java.awt.Dialog

Fields inherited from class java.awt.Window

Fields inherited from class java.awt.Container

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface javax.swing.WindowConstants
DISPOSE_ON_CLOSE, DO_NOTHING_ON_CLOSE, EXIT_ON_CLOSE, HIDE_ON_CLOSE

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

XML_JDOMTransformLoader(java.awt.Frame frame, java.lang.String title, boolean modal) Constructor.

Method Summary

org.jdom.Document	**getTransformedDoc**() Gets the transformed document.
private void	**initializeXmlTransformers**() Initializes the Transformation Classes List.
private void	**jbInit**() Set up GUI.
(package private) void	**jButtonCancel_actionPerformed**(java.awt.event.ActionEvent e) Cancel button action performed.
(package private) void	**jButtonOK_actionPerformed**(java.awt.event.ActionEvent e) OK button action performed.
(package private) void	**jButtonXML_actionPerformed**(java.awt.event.ActionEvent e) Choose a generic XML file action performed.

Methods inherited from class javax.swing.JDialog
addImpl, createRootPane, dialogInit, getAccessibleContext, getContentPane, getDefaultCloseOperation, getGlassPane, getJMenuBar, getLayeredPane, getRootPane, isDefaultLookAndFeelDecorated, isRootPaneCheckingEnabled, paramString, processWindowEvent, remove, setContentPane,

setDefaultCloseOperation, setDefaultLookAndFeelDecorated, setGlassPane, setJMenuBar, setLayeredPane, setLayout, setRootPane, setRootPaneCheckingEnabled, update

Methods inherited from class java.awt.Dialog
addNotify, dispose, getTitle, hide, isModal, isResizable, isUndecorated, setModal, setResizable, setTitle, setUndecorated, show

Methods inherited from class java.awt.Window
addPropertyChangeListener, addPropertyChangeListener, addWindowFocusListener, addWindowListener, addWindowStateListener, applyResourceBundle, applyResourceBundle, createBufferStrategy, createBufferStrategy, finalize, getBufferStrategy, getFocusableWindowState, getFocusCycleRootAncestor, getFocusOwner, getFocusTraversalKeys, getGraphicsConfiguration, getInputContext, getListeners, getLocale, getMostRecentFocusOwner, getOwnedWindows, getOwner, getToolkit, getWarningString, getWindowFocusListeners, getWindowListeners, getWindowStateListeners, isActive, isFocusableWindow, isFocusCycleRoot, isFocused, isShowing, pack, postEvent, processEvent, processWindowFocusEvent, processWindowStateEvent, removeWindowFocusListener, removeWindowListener, removeWindowStateListener, setCursor, setFocusableWindowState, setFocusCycleRoot, setLocationRelativeTo, toBack, toFront

Methods inherited from class java.awt.Container
add, add, add, add, add, addContainerListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getContainerListeners, getFocusTraversalPolicy, getInsets, getLayout, getMaximumSize, getMinimumSize, getPreferredSize, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paint, paintComponents, preferredSize, print, printComponents, processContainerEvent, remove, removeAll, removeContainerListener, removeNotify, setFocusTraversalKeys, setFocusTraversalPolicy, setFont, transferFocusBackward, transferFocusDownCycle, validate, validateTree

Methods inherited from class java.awt.Component
action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, coalesceEvents, contains, contains, createImage, createImage, createVolatileImage,

createVolatileImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, getBackground, getBounds, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphics, getHeight, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocation, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isOpaque, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, prepareImage, prepareImage, printAll, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processKeyEvent, processMouseEvent, processMouseMotionEvent, processMouseWheelEvent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, reshape, resize, resize, setBackground, setBounds, setBounds, setComponentOrientation, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeysEnabled, setForeground, setIgnoreRepaint, setLocale, setLocation, setLocation, setName, setSize, setSize, setVisible, show, size, toString, transferFocus, transferFocusUpCycle

Methods inherited from class java.lang.Object
clone, equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

borderLayout1
java.awt.BorderLayout **borderLayout1**

---

borderLayout2
java.awt.BorderLayout **borderLayout2**

---

cancelIcon
javax.swing.ImageIcon **cancelIcon**

Icons used.

---

flowLayout1
`java.awt.FlowLayout` **`flowLayout1`**

---

flowLayout2
`java.awt.FlowLayout` **`flowLayout2`**

---

folderIcon
`javax.swing.ImageIcon` **`folderIcon`**

Icons used.

---

gridLayout1
`java.awt.GridLayout` **`gridLayout1`**

---

jButtonCancel
`javax.swing.JButton` **`jButtonCancel`**

Cancel button.

---

jButtonOK
`javax.swing.JButton` **`jButtonOK`**

OK button.

---

jButtonXML
`javax.swing.JButton` **`jButtonXML`**

Choose a generic XML file button.

---

jComboBoxTransformers
`javax.swing.JComboBox` **`jComboBoxTransformers`**

Combo box with the transformation classes

---

jdomIcon
`javax.swing.ImageIcon` **`jdomIcon`**

Icons used.

---

jFileChooserXMLXSLT
`javax.swing.JFileChooser` **`jFileChooserXMLXSLT`**

The file chooser used in this dialog.

---

jLabelJDOMTransformClass

javax.swing.JLabel **jLabelJDOMTransformClass**

---

jLabelPath1
javax.swing.JLabel **jLabelPath1**

---

jLabelTitle
javax.swing.JLabel **jLabelTitle**

---

jLabelXML
javax.swing.JLabel **jLabelXML**

---

jPanelButtons
javax.swing.JPanel **jPanelButtons**

---

jPanelGrid
javax.swing.JPanel **jPanelGrid**

---

jPanelJDOM
javax.swing.JPanel **jPanelJDOM**

---

jPanelMain
javax.swing.JPanel **jPanelMain**

---

jPanelTitle
javax.swing.JPanel **jPanelTitle**

---

jPanelXML
javax.swing.JPanel **jPanelXML**

---

jTextFieldPath1
javax.swing.JTextField **jTextFieldPath1**

File path of XML file.

---

okIcon
javax.swing.ImageIcon **okIcon**

Icons used.

---

panel1
`javax.swing.JPanel` **`panel1`**

---

parent
`private` `AdminToolFrame` **`parent`**

Parent frame.

---

propertiesFile
`private java.io.File` **`propertiesFile`**

The Properties file which indicates which classes must be loaded.

---

transformedDoc
`private org.jdom.Document` **`transformedDoc`**

The transformed document.

---

xmlDoc
`private org.jdom.Document` **`xmlDoc`**

Generic XML file.

---

xmlFileFilter
`FilesFilter` **`xmlFileFilter`**

File filter for xml files *.xml used in JFileChooser

---

xmlIcon
`javax.swing.ImageIcon` **`xmlIcon`**

Icons used.

---

xmlTransformersList
`private java.util.Vector` **`xmlTransformersList`**

The transformation classes list that the combo box displays.

---

xYLayout2
`com.borland.jbcl.layout.XYLayout` **`xYLayout2`**

---

xYLayout3
`com.borland.jbcl.layout.XYLayout` **`xYLayout3`**

Constructor Detail

XML_JDOMTransformLoader

```
public XML_JDOMTransformLoader(java.awt.Frame frame,
                               java.lang.String title,
                               boolean modal)
```

Constructor.
Does a general initialization, initializes the Transformation Classes List and then calls the jbInit method to set up the GUI.
Here is set the ile path of Proprties file.

Parameters:

`frame` - parent frame
`title` - the title
`modal` - modal

Method Detail

getTransformedDoc

```
public org.jdom.Document getTransformedDoc()
```

Gets the transformed document.
Used by AdminToolFrame to get the document that was transformed.

---

initializeXmlTransformers

```
private void initializeXmlTransformers()
```

Initializes the Transformation Classes List. Loads the Properties file that contains the names of the classes that must be loaded. Loads each class at runtime, instanciates an object, cast it to be an XMLTransformer and adds it to the list.
The directory of the *.class files that are loaded must be in the CLASSPATH.

Future support for new transformers must be added here. Transformation class MUST implement interface XMLTransformer.

See Also:
XMLTransformer

---

jbInit

```
private void jbInit()
             throws java.lang.Exception
```

Set up GUI.
Sets the components' properties and lays them out.
Initializes the combo box with the Transformation Classes List.
Most of this code is automatically generated by JBuilder. Especially the laying out of the components.
This is the JBuilders method for setting up the GUI. It is necessary to use this method it order to be able to modify the GUI from JBuilder's Design view.

Throws:
`java.lang.Exception`

---

jButtonCancel_actionPerformed

`void jButtonCancel_actionPerformed(java.awt.event.ActionEvent e)`

Cancel button action performed.
Exits the Dialog.

Parameters:

`e` - ActionEvent

---

jButtonOK_actionPerformed

`void jButtonOK_actionPerformed(java.awt.event.ActionEvent e)`

OK button action performed.
Transformes the document using the selected Transformation class and exits.

Parameters:

`e` - ActionEvent

---

jButtonXML_actionPerformed

`void jButtonXML_actionPerformed(java.awt.event.ActionEvent e)`

Choose a generic XML file action performed.
Use a file chooser to chooser a file. Set up file chooser with filefilter and title, get file, load it by calling loadXML method, show file path in corresponding text field and reset filechooser.

Parameters:

`e` - ActionEvent

See Also:

`AdminToolFrame.loadXML(File)`

Overview Package Class Tree Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES All Classes

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

---

admintool

Class XML_XSLTLoader

```
java.lang.Object
  └java.awt.Component
```

```
└java.awt.Container
    └java.awt.Window
        └java.awt.Dialog
            └javax.swing.JDialog
                └admintool.XML_XSLTLoader
```

All Implemented Interfaces:

javax.accessibility.Accessible, java.awt.image.ImageObserver, java.awt.MenuContainer, javax.swing.RootPaneContainer, java.io.Serializable, javax.swing.WindowConstants

---

public class **XML_XSLTLoader**
extends javax.swing.JDialog

Load generic XML file through XSLT transformation Dialog.
Prompts the user to choose a a generic XML and a XSL file to transform the generic XML in the appropriate structure, and then does the transform. The transformed the document is returned to the AdminToolFrame with a getter method.

See Also:

Serialized Form

Nested Class Summary

(package private) class	**XML_XSLTLoader.XML_XSLTLoader_jButtonCancel_actionAdapter** Helper inner class created automatically by development tool.
(package private) class	**XML_XSLTLoader.XML_XSLTLoader_jButtonOK_actionAdapter** Helper inner class created automatically by development tool.
(package private) class	**XML_XSLTLoader.XML_XSLTLoader_jButtonXML_actionAdapter** Helper inner class created automatically by development tool.
(package private) class	**XML_XSLTLoader.XML_XSLTLoader_jButtonXSLT_actionAdapter** Helper inner class created automatically by development tool.

Nested classes inherited from class javax.swing.JDialog
javax.swing.JDialog.AccessibleJDialog

Nested classes inherited from class java.awt.Dialog
java.awt.Dialog.AccessibleAWTDialog

Nested classes inherited from class java.awt.Window
java.awt.Window.AccessibleAWTWindow

Nested classes inherited from class java.awt.Container
java.awt.Container.AccessibleAWTContainer

Nested classes inherited from class java.awt.Component
java.awt.Component.AccessibleAWTComponent, java.awt.Component.BltBufferStrategy, java.awt.Component.FlipBufferStrategy

Field Summary

(package private) java.awt.BorderLayout	**borderLayout1**
(package private) java.awt.BorderLayout	**borderLayout2**
(package private) javax.swing.ImageIcon	**cancelIcon** Icons used.
(package private) java.awt.FlowLayout	**flowLayout1**
(package private) java.awt.FlowLayout	**flowLayout2**
(package private) javax.swing.ImageIcon	**folderIcon** Icons used.
(package private) java.awt.GridLayout	**gridLayout1**
(package private) javax.swing.JButton	**jButtonCancel** Cancel button.
(package private) javax.swing.JButton	**jButtonOK** OK button.
(package private) javax.swing.JButton	**jButtonXML** Choose a generic XML file button.
(package private) javax.swing.JButton	**jButtonXSLT** Choose a XSL file button.
(package private) javax.swing.JFileChooser	**jFileChooserXMLXSLT** The file chooser used in this dialog.
(package private) javax.swing.JLabel	**jLabelPath1**

(package private) javax.swing.JLabel	**jLabelPath2**
(package private) javax.swing.JLabel	**jLabelTitle**
(package private) javax.swing.JLabel	**jLabelXML**
(package private) javax.swing.JLabel	**jLabelXSLT**
(package private) javax.swing.JPanel	**jPanelButtons**
(package private) javax.swing.JPanel	**jPanelGrid**
(package private) javax.swing.JPanel	**jPanelMain**
(package private) javax.swing.JPanel	**jPanelTitle**
(package private) javax.swing.JPanel	**jPanelXML**
(package private) javax.swing.JPanel	**jPanelXSLT**
(package private) javax.swing.JTextField	**jTextFieldPath1** File path of XML file.
(package private) javax.swing.JTextField	**jTextFieldPath2** File path of XSL file.
(package private) javax.swing.ImageIcon	**okIcon** Icons used.
(package private) javax.swing.JPanel	**panel1**
private AdminToolFrame	**parent** Parent frame.
private org.jdom.Document	**transformedDoc** The transformed document.
private org.jdom.Document	**xmlDoc** Generic XML file.
(package private) FilesFilter	**xmlFileFilter** File filter for xml files *.xml used in JFileChooser
(package private) javax.swing.ImageIcon	**xmlIcon** Icons used.
private org.jdom.Document	**xslDoc** XSL file.
(package private) FilesFilter	**xslFileFilter** File filter for xsl files *.xsl used in JFileChooser
(package private)	**xsltIcon**

javax.swing.ImageIcon	Icons used.
(package private) com.borland.jbcl.layout.XYLayout	**xYLayout2**
(package private) com.borland.jbcl.layout.XYLayout	**xYLayout3**

Fields inherited from class javax.swing.JDialog
accessibleContext, rootPane, rootPaneCheckingEnabled

Fields inherited from class java.awt.Dialog

Fields inherited from class java.awt.Window

Fields inherited from class java.awt.Container

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface javax.swing.WindowConstants
DISPOSE_ON_CLOSE, DO_NOTHING_ON_CLOSE, EXIT_ON_CLOSE, HIDE_ON_CLOSE

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

XML_XSLTLoader(java.awt.Frame frame, java.lang.String title, boolean modal) Constructor.	

Method Summary

org.jdom.Document	**getTransformedDoc**() Gets the transformed document.
private void	**jbInit**() Set up GUI.
(package private) void	**jButtonCancel_actionPerformed**(java.awt.event.ActionEvent e) Cancel button action performed.
(package private) void	**jButtonOK_actionPerformed**(java.awt.event.ActionEvent e) OK button action performed.
(package private) void	**jButtonXML_actionPerformed**(java.awt.event.ActionEvent e) Choose a generic XML file action performed.
(package private) void	**jButtonXSLT_actionPerformed**(java.awt.event.ActionEvent e) Choose a generic XSL file action performed.

Methods inherited from class javax.swing.JDialog

addImpl, createRootPane, dialogInit, getAccessibleContext, getContentPane, getDefaultCloseOperation, getGlassPane, getJMenuBar, getLayeredPane, getRootPane, isDefaultLookAndFeelDecorated, isRootPaneCheckingEnabled, paramString, processWindowEvent, remove, setContentPane, setDefaultCloseOperation, setDefaultLookAndFeelDecorated, setGlassPane, setJMenuBar, setLayeredPane, setLayout, setRootPane, setRootPaneCheckingEnabled, update

Methods inherited from class java.awt.Dialog

addNotify, dispose, getTitle, hide, isModal, isResizable, isUndecorated, setModal, setResizable, setTitle, setUndecorated, show

Methods inherited from class java.awt.Window

addPropertyChangeListener, addPropertyChangeListener, addWindowFocusListener, addWindowListener, addWindowStateListener, applyResourceBundle, applyResourceBundle, createBufferStrategy, createBufferStrategy, finalize, getBufferStrategy, getFocusableWindowState, getFocusCycleRootAncestor, getFocusOwner, getFocusTraversalKeys, getGraphicsConfiguration, getInputContext, getListeners, getLocale,

getMostRecentFocusOwner, getOwnedWindows, getOwner, getToolkit, getWarningString, getWindowFocusListeners, getWindowListeners, getWindowStateListeners, isActive, isFocusableWindow, isFocusCycleRoot, isFocused, isShowing, pack, postEvent, processEvent, processWindowFocusEvent, processWindowStateEvent, removeWindowFocusListener, removeWindowListener, removeWindowStateListener, setCursor, setFocusableWindowState, setFocusCycleRoot, setLocationRelativeTo, toBack, toFront

Methods inherited from class java.awt.Container

add, add, add, add, add, addContainerListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getContainerListeners, getFocusTraversalPolicy, getInsets, getLayout, getMaximumSize, getMinimumSize, getPreferredSize, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paint, paintComponents, preferredSize, print, printComponents, processContainerEvent, remove, removeAll, removeContainerListener, removeNotify, setFocusTraversalKeys, setFocusTraversalPolicy, setFont, transferFocusBackward, transferFocusDownCycle, validate, validateTree

Methods inherited from class java.awt.Component

action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, coalesceEvents, contains, contains, createImage, createImage, createVolatileImage, createVolatileImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, getBackground, getBounds, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphics, getHeight, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocation, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, imageUpdate, inside, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isOpaque, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, prepareImage, prepareImage, printAll, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processKeyEvent, processMouseEvent, processMouseMotionEvent, processMouseWheelEvent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener,

removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, reshape, resize, resize, setBackground, setBounds, setBounds, setComponentOrientation, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeysEnabled, setForeground, setIgnoreRepaint, setLocale, setLocation, setLocation, setName, setSize, setSize, setVisible, show, size, toString, transferFocus, transferFocusUpCycle

Methods inherited from class java.lang.Object
clone, equals, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

borderLayout1
java.awt.BorderLayout **borderLayout1**

---

borderLayout2
java.awt.BorderLayout **borderLayout2**

---

cancelIcon
javax.swing.ImageIcon **cancelIcon**

Icons used.

---

flowLayout1
java.awt.FlowLayout **flowLayout1**

---

flowLayout2
java.awt.FlowLayout **flowLayout2**

---

folderIcon
javax.swing.ImageIcon **folderIcon**

Icons used.

---

gridLayout1
java.awt.GridLayout **gridLayout1**

---

jButtonCancel
`javax.swing.JButton` **`jButtonCancel`**

Cancel button.

---

jButtonOK
`javax.swing.JButton` **`jButtonOK`**

OK button.

---

jButtonXML
`javax.swing.JButton` **`jButtonXML`**

Choose a generic XML file button.

---

jButtonXSLT
`javax.swing.JButton` **`jButtonXSLT`**

Choose a XSL file button.

---

jFileChooserXMLXSLT
`javax.swing.JFileChooser` **`jFileChooserXMLXSLT`**

The file chooser used in this dialog.

---

jLabelPath1
`javax.swing.JLabel` **`jLabelPath1`**

---

jLabelPath2
`javax.swing.JLabel` **`jLabelPath2`**

---

jLabelTitle
`javax.swing.JLabel` **`jLabelTitle`**

---

jLabelXML
`javax.swing.JLabel` **`jLabelXML`**

---

jLabelXSLT
`javax.swing.JLabel` **`jLabelXSLT`**

---

jPanelButtons
`javax.swing.JPanel` **`jPanelButtons`**

---

jPanelGrid
`javax.swing.JPanel` **`jPanelGrid`**

---

jPanelMain
`javax.swing.JPanel` **`jPanelMain`**

---

jPanelTitle
`javax.swing.JPanel` **`jPanelTitle`**

---

jPanelXML
`javax.swing.JPanel` **`jPanelXML`**

---

jPanelXSLT
`javax.swing.JPanel` **`jPanelXSLT`**

---

jTextFieldPath1
`javax.swing.JTextField` **`jTextFieldPath1`**

File path of XML file.

---

jTextFieldPath2
`javax.swing.JTextField` **`jTextFieldPath2`**

File path of XSL file.

---

okIcon
`javax.swing.ImageIcon` **`okIcon`**

Icons used.

---

panel1
`javax.swing.JPanel` **`panel1`**

---

parent
`private` `AdminToolFrame` **`parent`**

Parent frame.

---

transformedDoc
`private org.jdom.Document` **`transformedDoc`**

The transformed document.

---

xmlDoc
`private org.jdom.Document` **`xmlDoc`**

Generic XML file.

---

xmlFileFilter
`FilesFilter` **`xmlFileFilter`**

File filter for xml files *.xml used in JFileChooser

---

xmlIcon
`javax.swing.ImageIcon` **`xmlIcon`**

Icons used.

---

xslDoc
`private org.jdom.Document` **`xslDoc`**

XSL file.

---

xslFileFilter
`FilesFilter` **`xslFileFilter`**

File filter for xsl files *.xsl used in JFileChooser

---

xsltIcon
`javax.swing.ImageIcon` **`xsltIcon`**

Icons used.

---

xYLayout2
`com.borland.jbcl.layout.XYLayout` **`xYLayout2`**

---

xYLayout3
`com.borland.jbcl.layout.XYLayout` **`xYLayout3`**

Constructor Detail

XML_XSLTLoader
```
public XML_XSLTLoader(java.awt.Frame frame,
                      java.lang.String title,
                      boolean modal)
```

Constructor. Does an initialization and then calls the jbInit method to set up the GUI.

Parameters:

`frame` - parent frame

`title` - the title

`modal` - modal

Method Detail

getTransformedDoc

```
public org.jdom.Document getTransformedDoc()
```

Gets the transformed document.
Used by AdminToolFrame to get the document that was transformed.

---

jbInit

```
private void jbInit()
            throws java.lang.Exception
```

Set up GUI.
Sets the components' properties and lays them out.
Most of this code is automatically generated by JBuilder. Especially the laying out of the components.
This is the JBuilders method for setting up the GUI. It is necessary to use this method it order to be able to modify the GUI from JBuilder's Design view.
Throws:
`java.lang.Exception`

---

jButtonCancel_actionPerformed

```
void jButtonCancel_actionPerformed(java.awt.event.ActionEvent e)
```

Cancel button action performed.
Exits the Dialog.
Parameters:
`e` - ActionEvent

---

jButtonOK_actionPerformed

```
void jButtonOK_actionPerformed(java.awt.event.ActionEvent e)
```

OK button action performed.
Transformes the document using JDOM's XSLTransformer and exits.
Parameters:
`e` - ActionEvent
See Also:
`XSLTransformer`

---

jButtonXML_actionPerformed

```
void jButtonXML_actionPerformed(java.awt.event.ActionEvent e)
```

Choose a generic XML file action performed.
Use a file chooser to chooser a file. Set up file chooser with filefilter and title, get file, load it by calling loadXML method, show file path in corresponding text field and reset filechooser.
Parameters:
`e` - ActionEvent
See Also:
`AdminToolFrame.loadXML(File)`

jButtonXSLT_actionPerformed

```
void jButtonXSLT_actionPerformed(java.awt.event.ActionEvent e)
```

Choose a generic XSL file action performed.
Use a file chooser to chooser a file. Set up file chooser with filefilter and title, get file, load it by calling loadXML method, show file path in corresponding text field and reset filechooser.

Parameters:

e - ActionEvent

See Also:

`AdminToolFrame.loadXML(File)`

Package admintool.jdomtreemodel

Provides the necessary classes for displaying a JDOM document into a JTree.

See:

Description

Class Summary	
DataSrcMapper	Retrives metadata from a datasources and maps them to a Data Source Document.
DataSrcTreeCellRenderer	Tree Cell Renderer of the Data Sstructure JTree.
DisplayedPathEvaluator	Utility class that encloses the mechanism of evaluting the assocoation path given data source field of a data source that is loaded in Data Structure tree.
InfoStructTreeCellRenderer	Tree Cell Renderer of the Information Structure JTree.
JDOMTreeModel	Tree Model implementation to display a JDOM Document in a JTree, with insert and remove support but without displaying

	attributes.
JDOMTreeModelAttr	Tree Model implementation to display a JDOM Document in a JTree, with insert and remove support and displaying attributes.
XMLWhitespaceFilter	XML filter used in order to avoid loading ignorable whitespace as Text nodes.
XPathEvaluator	Utility class that encloses the mechanism of evaluting the xpath of a given node in a JDOM document.

Package admintool.jdomtreemodel Description

Provides the necessary classes for displaying a JDOM document into a JTree.

Provides:

- TreeModel implementations for mapping directly a JDOM document into the JTree based on the JDOM document structure. One supporting attributes and not without.
- Tree Cell Renderers for displaying the document in the JTree in specified way.
- Class for mapping a Data Source into a Doument that can be displayed in a JTree
- Class that estimates the XPath od a given node of the tree
- XML filter for SAXBuilder that enables the loading of XML files with no corresponding validation file (DTD or XSD)ignoring the unwanted whitespace nodes.

See Also:
`admintool`

---

admintool.jdomtreemodel

Class DataSrcMapper

`java.lang.Object`

└ **admintool.jdomtreemodel.DataSrcMapper**

---

public class **DataSrcMapper**
extends java.lang.Object

Retrives metadata from a datasources and maps them to a Data Source Document.

Field Summary

(package private) org.jdom.Attribute	**attName** Name attribute variable.
(package private) org.jdom.Attribute	**attType** Type attribute variable.
private java.sql.Connection	**connection** Connection to the database.
private java.lang.String	**DATA_SRC_NAME** Data source name.
private java.lang.String	**DATA_SRC_TYPE** Data source type.
(package private) org.jdom.Document	**doc** Data Source Document where the retrieved data source is mapped.
private java.lang.String	**JDBC_DRIVER** JDBC driver.
private java.sql.ResultSetMetaData	**metaData** Meta data results retrieved from database.
private java.sql.Statement	**statement** Statement for accessing and querying database.

Constructor Summary

DataSrcMapper(java.lang.String driver, java.lang.String dataSrcName, java.lang.String dataSrcType) Creates the Data source document based on the meta data it retrieves from the corresponding data source.

Method Summary

org.jdom.Document	**getMappedDataSrc**() Gets the Data source document that the Data Source is retrieved and mapped on.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

attName
org.jdom.Attribute **attName**

Name attribute variable.

---

attType
org.jdom.Attribute **attType**

Type attribute variable.

---

connection
private java.sql.Connection **connection**

Connection to the database.

DATA_SRC_NAME
private java.lang.String **DATA_SRC_NAME**

Data source name.

DATA_SRC_TYPE
private java.lang.String **DATA_SRC_TYPE**

Data source type. Type can be: ODBC, Oracle, SQL server etc.

---

doc
org.jdom.Document **doc**

Data Source Document where the retrieved data source is mapped.

JDBC_DRIVER
private java.lang.String **JDBC_DRIVER**

JDBC driver.

---

metaData
private java.sql.ResultSetMetaData **metaData**

Meta data results retrieved from database.

---

statement

```
private java.sql.Statement statement
```

Statement for accessing and querying database.

Constructor Detail

DataSrcMapper

```
public DataSrcMapper(java.lang.String driver,
                     java.lang.String dataSrcName,
                     java.lang.String dataSrcType)
```

Creates the Data source document based on the meta data it retrieves from the corresponding data source.

Parameters:

`driver` - the JDBC driver

`dataSrcName` - data source name

`dataSrcType` - data source type.

Method Detail

getMappedDataSrc

```
public org.jdom.Document getMappedDataSrc()
```

Gets the Data source document that the Data Source is retrieved and mapped on.

Returns:

Document

Overview Package Class Tree Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES All Classes

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

---

admintool.jdomtreemodel

Class DataSrcTreeCellRenderer

```
java.lang.Object
  └java.awt.Component
```

```
└java.awt.Container
    └javax.swing.JComponent
        └javax.swing.JLabel
            └javax.swing.tree.DefaultTreeCellRenderer
                └admintool.jdomtreemodel.DataSrcTreeCellRenderer
```

All Implemented Interfaces:

javax.accessibility.Accessible, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, javax.swing.SwingConstants, javax.swing.tree.TreeCellRenderer

---

public class **DataSrcTreeCellRenderer**
extends javax.swing.tree.DefaultTreeCellRenderer

Tree Cell Renderer of the Data Sstructure JTree.

See Also:

`DefaultTreeCellRenderer`, Serialized Form

Nested Class Summary

Nested classes inherited from class javax.swing.JLabel
`javax.swing.JLabel.AccessibleJLabel`

Nested classes inherited from class javax.swing.JComponent
`javax.swing.JComponent.AccessibleJComponent`

Nested classes inherited from class java.awt.Container
`java.awt.Container.AccessibleAWTContainer`

Nested classes inherited from class java.awt.Component
`java.awt.Component.AccessibleAWTComponent`, `java.awt.Component.BltBufferStrategy`, `java.awt.Component.FlipBufferStrategy`

Field Summary

(package private)	**attIcon**

javax.swing.ImageIcon	Image used.
(package private) javax.swing.ImageIcon	**cdataIcon** Image used.
(package private) javax.swing.ImageIcon	**conditionedField** Image used.
(package private) java.awt.Color	**databaseFieldColor** Color used.
(package private) javax.swing.ImageIcon	**databaseFieldIcon** Image used.
(package private) javax.swing.ImageIcon	**databaseLoadedIcon** Image used.
(package private) javax.swing.ImageIcon	**databaseNotLoadedIcon** Image used.
(package private) javax.swing.ImageIcon	**databasesLoadedIcon** Image used.
(package private) javax.swing.ImageIcon	**databasesNotLoadedIcon** Image used.
(package private) java.awt.Color	**databaseTableColor** Color used.
(package private) javax.swing.ImageIcon	**databaseTableExpandedIcon** Image used.
(package private) javax.swing.ImageIcon	**databaseTableNotExpandedIcon** Image used.
(package private) javax.swing.ImageIcon	**dataFileIcon** Image used.
(package private) java.awt.Color	**dbColor** Color used.
(package private) java.awt.Font	**dbFieldFont** Font used.
(package private) java.awt.Font	**dbFont** Font used.
(package private) java.awt.Font	**dbTableFont** Font used.
(package private) javax.swing.ImageIcon	**elementIcon** Image used.
(package private) javax.swing.ImageIcon	**noIcon** Image used.
(package private) javax.swing.ImageIcon	**textIcon** Image used.
(package private) javax.swing.ImageIcon	**xmlIcon** Image used.

Fields inherited from class javax.swing.tree.DefaultTreeCellRenderer
backgroundNonSelectionColor, backgroundSelectionColor, borderSelectionColor, closedIcon, hasFocus, leafIcon, openIcon, selected, textNonSelectionColor, textSelectionColor

Fields inherited from class javax.swing.JLabel
labelFor

Fields inherited from class javax.swing.JComponent
accessibleContext, listenerList, TOOL_TIP_TEXT_KEY, ui, UNDEFINED_CONDITION, WHEN_ANCESTOR_OF_FOCUSED_COMPONENT, WHEN_FOCUSED, WHEN_IN_FOCUSED_WINDOW

Fields inherited from class java.awt.Container

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface javax.swing.SwingConstants
BOTTOM, CENTER, EAST, HORIZONTAL, LEADING, LEFT, NEXT, NORTH, NORTH_EAST, NORTH_WEST, PREVIOUS, RIGHT, SOUTH, SOUTH_EAST, SOUTH_WEST, TOP, TRAILING, VERTICAL, WEST

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

DataSrcTreeCellRenderer()
Constructor.

Method Summary

java.awt.Component	**getTreeCellRendererComponent**(javax.swing.JTree tree, java.lang.Object value, boolean sel, boolean expanded, boolean leaf, int row, boolean hasFocus) The method that does the rendering.

Methods inherited from class javax.swing.tree.DefaultTreeCellRenderer

firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, getBackgroundNonSelectionColor, getBackgroundSelectionColor, getBorderSelectionColor, getClosedIcon, getDefaultClosedIcon, getDefaultLeafIcon, getDefaultOpenIcon, getFont, getLeafIcon, getOpenIcon, getPreferredSize, getTextNonSelectionColor, getTextSelectionColor, paint, repaint, repaint, revalidate, setBackground, setBackgroundNonSelectionColor, setBackgroundSelectionColor, setBorderSelectionColor, setClosedIcon, setFont, setLeafIcon, setOpenIcon, setTextNonSelectionColor, setTextSelectionColor, validate

Methods inherited from class javax.swing.JLabel

checkHorizontalKey, checkVerticalKey, getAccessibleContext, getDisabledIcon, getDisplayedMnemonic, getDisplayedMnemonicIndex, getHorizontalAlignment, getHorizontalTextPosition, getIcon, getIconTextGap, getLabelFor, getText, getUI, getUIClassID, getVerticalAlignment, getVerticalTextPosition, imageUpdate, paramString, setDisabledIcon, setDisplayedMnemonic, setDisplayedMnemonic, setDisplayedMnemonicIndex, setHorizontalAlignment, setHorizontalTextPosition, setIcon, setIconTextGap, setLabelFor, setText, setUI, setVerticalAlignment, setVerticalTextPosition, updateUI

Methods inherited from class javax.swing.JComponent

addAncestorListener, addNotify, addPropertyChangeListener, addPropertyChangeListener, addVetoableChangeListener, computeVisibleRect, contains, createToolTip, disable, enable, fireVetoableChange, getActionForKeyStroke, getActionMap, getAlignmentX, getAlignmentY, getAncestorListeners, getAutoscrolls, getBorder, getBounds, getClientProperty, getComponentGraphics, getConditionForKeyStroke, getDebugGraphicsOptions, getDefaultLocale, getGraphics, getHeight, getInputMap, getInputMap, getInputVerifier, getInsets, getInsets, getListeners, getLocation, getMaximumSize, getMinimumSize, getNextFocusableComponent, getPropertyChangeListeners, getPropertyChangeListeners, getRegisteredKeyStrokes, getRootPane, getSize, getToolTipLocation, getToolTipText, getToolTipText, getTopLevelAncestor, getTransferHandler, getVerifyInputWhenFocusTarget, getVetoableChangeListeners, getVisibleRect, getWidth, getX, getY, grabFocus,

isDoubleBuffered, isLightweightComponent, isManagingFocus, isMaximumSizeSet, isMinimumSizeSet, isOpaque, isOptimizedDrawingEnabled, isPaintingTile, isPreferredSizeSet, isRequestFocusEnabled, isValidateRoot, paintBorder, paintChildren, paintComponent, paintImmediately, paintImmediately, print, printAll, printBorder, printChildren, printComponent, processComponentKeyEvent, processKeyBinding, processKeyEvent, processMouseMotionEvent, putClientProperty, registerKeyboardAction, registerKeyboardAction, removeAncestorListener, removeNotify, removePropertyChangeListener, removePropertyChangeListener, removeVetoableChangeListener, requestDefaultFocus, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, resetKeyboardActions, reshape, scrollRectToVisible, setActionMap, setAlignmentX, setAlignmentY, setAutoscrolls, setBorder, setDebugGraphicsOptions, setDefaultLocale, setDoubleBuffered, setEnabled, setForeground, setInputMap, setInputVerifier, setMaximumSize, setMinimumSize, setNextFocusableComponent, setOpaque, setPreferredSize, setRequestFocusEnabled, setToolTipText, setTransferHandler, setUI, setVerifyInputWhenFocusTarget, setVisible, unregisterKeyboardAction, update

Methods inherited from class java.awt.Container

add, add, add, add, add, addContainerListener, addImpl, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getContainerListeners, getFocusTraversalKeys, getFocusTraversalPolicy, getLayout, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusCycleRoot, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paintComponents, preferredSize, printComponents, processContainerEvent, processEvent, remove, remove, removeAll, removeContainerListener, setFocusCycleRoot, setFocusTraversalKeys, setFocusTraversalPolicy, setLayout, transferFocusBackward, transferFocusDownCycle, validateTree

Methods inherited from class java.awt.Component

action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, coalesceEvents, contains, createImage, createImage, createVolatileImage, createVolatileImage, disableEvents, dispatchEvent, enable, enableEvents, enableInputMethods, getBackground, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusCycleRootAncestor, getFocusListeners, getFocusTraversalKeysEnabled, getFontMetrics, getForeground, getGraphicsConfiguration, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputContext, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocale, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMouseWheelListeners, getName, getParent, getPeer, getSize, getToolkit, getTreeLock, gotFocus, handleEvent, hasFocus, hide, inside, isBackgroundSet, isCursorSet, isDisplayable, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isShowing, isValid, isVisible, keyDown,

keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, postEvent, prepareImage, prepareImage, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processMouseEvent, processMouseWheelEvent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, repaint, repaint, repaint, resize, resize, setBounds, setBounds, setComponentOrientation, setCursor, setDropTarget, setFocusable, setFocusTraversalKeysEnabled, setIgnoreRepaint, setLocale, setLocation, setLocation, setName, setSize, setSize, show, show, size, toString, transferFocus, transferFocusUpCycle

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

attIcon
`javax.swing.ImageIcon` **`attIcon`**

Image used.

---

cdataIcon
`javax.swing.ImageIcon` **`cdataIcon`**

Image used.

---

conditionedField
`javax.swing.ImageIcon` **`conditionedField`**

Image used.

---

databaseFieldColor
`java.awt.Color` **`databaseFieldColor`**

Color used.

---

databaseFieldIcon
`javax.swing.ImageIcon` **`databaseFieldIcon`**

Image used.

---

databaseLoadedIcon
`javax.swing.ImageIcon` **`databaseLoadedIcon`**

Image used.

---

databaseNotLoadedIcon
`javax.swing.ImageIcon` **`databaseNotLoadedIcon`**

Image used.

---

databasesLoadedIcon
`javax.swing.ImageIcon` **`databasesLoadedIcon`**

Image used.

---

databasesNotLoadedIcon
`javax.swing.ImageIcon` **`databasesNotLoadedIcon`**

Image used.

---

databaseTableColor
`java.awt.Color` **`databaseTableColor`**

Color used.

---

databaseTableExpandedIcon
`javax.swing.ImageIcon` **`databaseTableExpandedIcon`**

Image used.

---

databaseTableNotExpandedIcon
`javax.swing.ImageIcon` **`databaseTableNotExpandedIcon`**

Image used.

---

dataFileIcon
`javax.swing.ImageIcon` **`dataFileIcon`**

Image used.

---

dbColor
`java.awt.Color` **`dbColor`**

Color used.

---

dbFieldFont
`java.awt.Font` **`dbFieldFont`**

Font used.

dbFont
`java.awt.Font` **`dbFont`**

Font used.

---

dbTableFont
`java.awt.Font` **`dbTableFont`**

Font used.

---

elementIcon
`javax.swing.ImageIcon` **`elementIcon`**

Image used.

---

noIcon
`javax.swing.ImageIcon` **`noIcon`**

Image used.

---

textIcon
`javax.swing.ImageIcon` **`textIcon`**

Image used.

---

xmlIcon
`javax.swing.ImageIcon` **`xmlIcon`**

Image used.

Constructor Detail

DataSrcTreeCellRenderer
`public` **`DataSrcTreeCellRenderer`**`()`

Constructor. Initializes fonts, colors and loads images. When loading an image, if image is not found then the application won't start. The way that the images are loaded ensures that are all loaded if the appplication starts.

Method Detail

getTreeCellRendererComponent
```
public java.awt.Component
getTreeCellRendererComponent(javax.swing.JTree tree,

java.lang.Object value,
                                                    boolean sel,
                                                    boolean expanded,
                                                    boolean leaf,
                                                    int row,
```

```
                                                    boolean hasFocus)
```

The method that does the rendering.

Cases of rendered object:

- Case of: Element.
  Element can be:
  - case of: Data structure
    - is empty
    - or not empty
  - case of: loaded data structures. Load structures can be:
    - case of: data source structure, that is a data source which is retrived and loaded in the tree
    - case of: data structure file, that is a data structure loaded from file
  - case of: Data structure category
  - case of: Data structure field
  - case of: any other element, that is the case when a generic XML is loaded. It has the following subcases:
    - case of: the root of a generic xml
    - case of: the rest elements of a generic xml
- case of: Attribute
- case of: CDATA
- Case of: Text.
- Case of: any other node

In each case the method sets the appropriate color, font and text that the rendered objecy would be displayed. CDATA test must preceed Text test! Otherwise it mistakes CDATA nodes to be Text nodes

Parameters:

`tree` - JTree
`value` - the Object that is going to be renderered
`sel` - boolean
`expanded` - boolean
`leaf` - boolean
`row` - int
`hasFocus` - boolean

Returns:

Component

---

admintool.jdomtreemodel

Class DisplayedPathEvaluator

```
java.lang.Object
  └admintool.jdomtreemodel.DisplayedPathEvaluator
```

public class **DisplayedPathEvaluator**
extends java.lang.Object

Utility class that encloses the mechanism of evaluting the assocoation path given data source field of a data source that is loaded in Data Structure tree. In fact, that is the path to the data source field.

Tip: The `displayedPath` is also refered and as `associationPath` in this application code.
See Also:
`TreeTransferHandler`

Constructor Summary

`DisplayedPathEvaluator``()` Default constructor.

Method Summary

`java.lang.String`	**`getDisplayedPath`**`(java.lang.Object[] objects)` Returns the displayed path (or association path).

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

DisplayedPathEvaluator

`public` **`DisplayedPathEvaluator`**`()`

Default constructor.

Method Detail

getDisplayedPath

```
public java.lang.String getDisplayedPath(java.lang.Object[] objects)
```

Returns the displayed path (or association path).
It parses the objects that represent a path from selection to root, and for each objects test the following cases:

- Case of: Element:
  - if it is not a field append only its name
  - if it is a field and this is the last object in the path, append type too.
- Case of: Attribute
  - append attribute name

Parameters:
`objects` - the objects that represent a path from selection to root.
Returns:
the displayed path.

Overview **Package** **Class** **Tree** **Index** **Help**

PREV CLASS NEXT CLASS FRAMES NO FRAMES All Classes
SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

---

admintool.jdomtreemodel

Class InfoStructTreeCellRenderer

```
java.lang.Object
  └java.awt.Component
      └java.awt.Container
          └javax.swing.JComponent
              └javax.swing.JLabel
                  └javax.swing.tree.DefaultTreeCellRenderer
                      └admintool.jdomtreemodel.InfoStructTreeCellRenderer
```

All Implemented Interfaces:

javax.accessibility.Accessible, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable, javax.swing.SwingConstants, javax.swing.tree.TreeCellRenderer

---

public class **InfoStructTreeCellRenderer**
extends javax.swing.tree.DefaultTreeCellRenderer

Tree Cell Renderer of the Information Structure JTree.

See Also:

`DefaultTreeCellRenderer`, Serialized Form

Nested Class Summary

Nested classes inherited from class javax.swing.JLabel
`javax.swing.JLabel.AccessibleJLabel`

Nested classes inherited from class javax.swing.JComponent
`javax.swing.JComponent.AccessibleJComponent`

Nested classes inherited from class java.awt.Container
`java.awt.Container.AccessibleAWTContainer`

Nested classes inherited from class java.awt.Component
`java.awt.Component.AccessibleAWTComponent`, `java.awt.Component.BltBufferStrategy`, `java.awt.Component.FlipBufferStrategy`

Field Summary

(package private) java.awt.Font	**associationPathFont** Font used.
(package private) javax.swing.ImageIcon	**associationPathIcon** Image used.
(package private) java.awt.Color	**informationCategoryColor** Color used.

(package private) javax.swing.ImageIcon	**informationCategoryExpandedIcon** Image used.
(package private) java.awt.Font	**informationCategoryFont** Font used.
(package private) javax.swing.ImageIcon	**informationCategoryNotExpandedIcon** Image used.
(package private) java.awt.Color	**informationFieldAssociatedColor** Color used.
(package private) javax.swing.ImageIcon	**informationFieldAssociatedIcon** Image used.
(package private) java.awt.Color	**informationFieldAssociationIncompleteColor** Color used.
(package private) javax.swing.ImageIcon	**informationFieldAssociationIncompleteIcon** Image used.
(package private) java.awt.Font	**informationFieldFont** Font used.
(package private) java.awt.Color	**informationFieldNotAssociatedColor** Color used.
(package private) javax.swing.ImageIcon	**informationFieldNotAssociatedIcon** Image used.
(package private) java.awt.Color	**informationStructureColor** Color used.
(package private) java.awt.Font	**informationStructureFont** Font used.
(package private) javax.swing.ImageIcon	**informationStructureLoadedIcon** Image used.
(package private) javax.swing.ImageIcon	**informationStructureNotLoadedIcon** Image used.

Fields inherited from class javax.swing.tree.DefaultTreeCellRenderer
backgroundNonSelectionColor, backgroundSelectionColor, borderSelectionColor, closedIcon, hasFocus, leafIcon, openIcon, selected, textNonSelectionColor, textSelectionColor

Fields inherited from class javax.swing.JLabel
labelFor

Fields inherited from class javax.swing.JComponent

accessibleContext, listenerList, TOOL_TIP_TEXT_KEY, ui, UNDEFINED_CONDITION, WHEN_ANCESTOR_OF_FOCUSED_COMPONENT, WHEN_FOCUSED, WHEN_IN_FOCUSED_WINDOW

Fields inherited from class java.awt.Container

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface javax.swing.SwingConstants
BOTTOM, CENTER, EAST, HORIZONTAL, LEADING, LEFT, NEXT, NORTH, NORTH_EAST, NORTH_WEST, PREVIOUS, RIGHT, SOUTH, SOUTH_EAST, SOUTH_WEST, TOP, TRAILING, VERTICAL, WEST

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

InfoStructTreeCellRenderer() Constructor.

Method Summary

java.awt.Component	**getTreeCellRendererComponent**(javax.swing.JTree tree, java.lang.Object value, boolean sel, boolean expanded, boolean leaf, int row, boolean hasFocus) The method that does the rendering.

Methods inherited from class javax.swing.tree.DefaultTreeCellRenderer
firePropertyChange, firePropertyChange, firePropertyChange,

firePropertyChange, firePropertyChange, firePropertyChange,
firePropertyChange, firePropertyChange, firePropertyChange,
getBackgroundNonSelectionColor, getBackgroundSelectionColor,
getBorderSelectionColor, getClosedIcon, getDefaultClosedIcon,
getDefaultLeafIcon, getDefaultOpenIcon, getFont, getLeafIcon, getOpenIcon,
getPreferredSize, getTextNonSelectionColor, getTextSelectionColor, paint,
repaint, repaint, revalidate, setBackground, setBackgroundNonSelectionColor,
setBackgroundSelectionColor, setBorderSelectionColor, setClosedIcon,
setFont, setLeafIcon, setOpenIcon, setTextNonSelectionColor,
setTextSelectionColor, validate

Methods inherited from class javax.swing.JLabel

checkHorizontalKey, checkVerticalKey, getAccessibleContext, getDisabledIcon,
getDisplayedMnemonic, getDisplayedMnemonicIndex, getHorizontalAlignment,
getHorizontalTextPosition, getIcon, getIconTextGap, getLabelFor, getText,
getUI, getUIClassID, getVerticalAlignment, getVerticalTextPosition,
imageUpdate, paramString, setDisabledIcon, setDisplayedMnemonic,
setDisplayedMnemonic, setDisplayedMnemonicIndex, setHorizontalAlignment,
setHorizontalTextPosition, setIcon, setIconTextGap, setLabelFor, setText,
setUI, setVerticalAlignment, setVerticalTextPosition, updateUI

Methods inherited from class javax.swing.JComponent

addAncestorListener, addNotify, addPropertyChangeListener,
addPropertyChangeListener, addVetoableChangeListener, computeVisibleRect,
contains, createToolTip, disable, enable, fireVetoableChange,
getActionForKeyStroke, getActionMap, getAlignmentX, getAlignmentY,
getAncestorListeners, getAutoscrolls, getBorder, getBounds,
getClientProperty, getComponentGraphics, getConditionForKeyStroke,
getDebugGraphicsOptions, getDefaultLocale, getGraphics, getHeight,
getInputMap, getInputMap, getInputVerifier, getInsets, getInsets,
getListeners, getLocation, getMaximumSize, getMinimumSize,
getNextFocusableComponent, getPropertyChangeListeners,
getPropertyChangeListeners, getRegisteredKeyStrokes, getRootPane, getSize,
getToolTipLocation, getToolTipText, getToolTipText, getTopLevelAncestor,
getTransferHandler, getVerifyInputWhenFocusTarget,
getVetoableChangeListeners, getVisibleRect, getWidth, getX, getY, grabFocus,
isDoubleBuffered, isLightweightComponent, isManagingFocus, isMaximumSizeSet,
isMinimumSizeSet, isOpaque, isOptimizedDrawingEnabled, isPaintingTile,
isPreferredSizeSet, isRequestFocusEnabled, isValidateRoot, paintBorder,
paintChildren, paintComponent, paintImmediately, paintImmediately, print,
printAll, printBorder, printChildren, printComponent,
processComponentKeyEvent, processKeyBinding, processKeyEvent,
processMouseMotionEvent, putClientProperty, registerKeyboardAction,
registerKeyboardAction, removeAncestorListener, removeNotify,
removePropertyChangeListener, removePropertyChangeListener,
removeVetoableChangeListener, requestDefaultFocus, requestFocus,
requestFocus, requestFocusInWindow, requestFocusInWindow,
resetKeyboardActions, reshape, scrollRectToVisible, setActionMap,
setAlignmentX, setAlignmentY, setAutoscrolls, setBorder,
setDebugGraphicsOptions, setDefaultLocale, setDoubleBuffered, setEnabled,
setForeground, setInputMap, setInputVerifier, setMaximumSize,
setMinimumSize, setNextFocusableComponent, setOpaque, setPreferredSize,

setRequestFocusEnabled, setToolTipText, setTransferHandler, setUI, setVerifyInputWhenFocusTarget, setVisible, unregisterKeyboardAction, update

Methods inherited from class java.awt.Container
add, add, add, add, add, addContainerListener, addImpl, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getContainerListeners, getFocusTraversalKeys, getFocusTraversalPolicy, getLayout, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusCycleRoot, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paintComponents, preferredSize, printComponents, processContainerEvent, processEvent, remove, remove, removeAll, removeContainerListener, setFocusCycleRoot, setFocusTraversalKeys, setFocusTraversalPolicy, setLayout, transferFocusBackward, transferFocusDownCycle, validateTree

Methods inherited from class java.awt.Component
action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, coalesceEvents, contains, createImage, createImage, createVolatileImage, createVolatileImage, disableEvents, dispatchEvent, enable, enableEvents, enableInputMethods, getBackground, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusCycleRootAncestor, getFocusListeners, getFocusTraversalKeysEnabled, getFontMetrics, getForeground, getGraphicsConfiguration, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputContext, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocale, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMouseWheelListeners, getName, getParent, getPeer, getSize, getToolkit, getTreeLock, gotFocus, handleEvent, hasFocus, hide, inside, isBackgroundSet, isCursorSet, isDisplayable, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isShowing, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, postEvent, prepareImage, prepareImage, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processMouseEvent, processMouseWheelEvent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, repaint, repaint, repaint, resize, resize, setBounds, setBounds, setComponentOrientation, setCursor, setDropTarget, setFocusable, setFocusTraversalKeysEnabled, setIgnoreRepaint, setLocale, setLocation, setLocation, setName, setSize, setSize, show, show, size, toString, transferFocus, transferFocusUpCycle

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

associationPathFont
java.awt.Font **associationPathFont**

Font used.

---

associationPathIcon
javax.swing.ImageIcon **associationPathIcon**

Image used.

---

informationCategoryColor
java.awt.Color **informationCategoryColor**

Color used.

---

informationCategoryExpandedIcon
javax.swing.ImageIcon **informationCategoryExpandedIcon**

Image used.

---

informationCategoryFont
java.awt.Font **informationCategoryFont**

Font used.

---

informationCategoryNotExpandedIcon
javax.swing.ImageIcon **informationCategoryNotExpandedIcon**

Image used.

---

informationFieldAssociatedColor
java.awt.Color **informationFieldAssociatedColor**

Color used.

---

informationFieldAssociatedIcon
javax.swing.ImageIcon **informationFieldAssociatedIcon**

Image used.

---

informationFieldAssociationIncompleteColor
`java.awt.Color` **`informationFieldAssociationIncompleteColor`**

Color used.

---

informationFieldAssociationIncompleteIcon
`javax.swing.ImageIcon` **`informationFieldAssociationIncompleteIcon`**

Image used.

---

informationFieldFont
`java.awt.Font` **`informationFieldFont`**

Font used.

---

informationFieldNotAssociatedColor
`java.awt.Color` **`informationFieldNotAssociatedColor`**

Color used.

---

informationFieldNotAssociatedIcon
`javax.swing.ImageIcon` **`informationFieldNotAssociatedIcon`**

Image used.

---

informationStructureColor
`java.awt.Color` **`informationStructureColor`**

Color used.

---

informationStructureFont
`java.awt.Font` **`informationStructureFont`**

Font used.

---

informationStructureLoadedIcon
`javax.swing.ImageIcon` **`informationStructureLoadedIcon`**

Image used.

---

informationStructureNotLoadedIcon
`javax.swing.ImageIcon` **`informationStructureNotLoadedIcon`**

Image used.

Constructor Detail

InfoStructTreeCellRenderer
`public` **`InfoStructTreeCellRenderer`**`()`

Constructor. Initializes fonts, colors and loads images. When loading an image, if image is not found then the application won't start. The way that the images are loaded ensures that are all loaded if the appplication starts.

Method Detail

getTreeCellRendererComponent

```
public java.awt.Component
getTreeCellRendererComponent(javax.swing.JTree tree,

java.lang.Object value,
                                                    boolean sel,
                                                    boolean expanded,
                                                    boolean leaf,
                                                    int row,
                                                    boolean hasFocus)
```

The method that does the rendering.

Cases of rendered object:

- Case of: Element.
  Element can be:
  - case of: Information structure
    - is empty
    - or not empty
  - case of: Information category
  - case of: information field
    - Not associated
    - Associated
- Case of: Text. The text node carries the XPath string. But, the text node would be dislpayed with the association path attribute of its parent element.
- Case of: any other node

In each case the method sets the appropriate color, font and text that the rendered objecy would be displayed.

Parameters:

`tree` - JTree

`value` - the Object that is going to be renderered

`sel` - boolean

`expanded` - boolean

`leaf` - boolean

`row` - int

`hasFocus` - boolean

Returns:

Component

---

admintool.jdomtreemodel

Class JDOMTreeModel

```
java.lang.Object
  └admintool.jdomtreemodel.JDOMTreeModel
```

All Implemented Interfaces:

javax.swing.tree.TreeModel

Direct Known Subclasses:

JDOMTreeModelAttr

public class **JDOMTreeModel**
extends java.lang.Object
implements javax.swing.tree.TreeModel

Tree Model implementation to display a JDOM Document in a JTree, with insert and remove support but without displaying attributes.

The structure shown by the JTrees using this Tree Model is the structure of the XML document of the Tree Model. The JTree is displayed by calling the methods of the Interface TreeModel that this class implements. And these methods are implemented in order to outline the structure of the XML document of the this class.

This class implements the TreeModel Interface, instead of extending the DefaultTreeModel class. As a result the data structure of the XML document is directly mapped to the structure shown by JTree. If a DefaultTreeModel class was used, the current structure shown by JTree would be the Structure of the DefaultTreeModel and not of the XML document.

This class does not provide support for displaying Attributes as leaves of the tree. Attributes are completely ignored. The subclass JDOMTrreModelAttr extends this functionality.

Used by Information Structure JTree.

Note: `Content` is refered to org.jdom.Content.

Field Summary

(package private) DataModifier	**dataModifier** Through this reference the GUIUpdater of AdminToolFrame is notified, when a modification on TreeModel is made.
(package private) org.jdom.Document	**doc** The JDOM Document that this Tree Model is based.
(package private) java.util.Vector	**listeners** Event Listeners List.
(package private) org.jdom.Element	**root** The root of the Tree Model, which is the root of the document too.

Constructor Summary

JDOMTreeModel(org.jdom.Document document, DataModifier dataModifier) Constructor.

Method Summary

void	**addTreeModelListener**(javax.swing.event.TreeModelListener l) Adds a listener for the TreeModelEvent posted after the tree changes.
private void	**fireTreeNodesInserted**(javax.swing.tree.TreePath path, int[] indices, java.lang.Object[] children) Notifies all listeners that have registered interest for notification on this event type (Tree Model Event).
private void	**fireTreeNodesRemoved**(java.lang.Object source, javax.swing.tree.TreePath path, int[] childIndices, java.lang.Object[] children) Notifies all listeners that have registered interest for notification on this event type (Tree Model Event).
java.lang.Object	**getChild**(java.lang.Object parent, int index) Returns the child of parent at "index" index in the parent's child array.
int	**getChildCount**(java.lang.Object parent) Returns the number of children of parent.
int	**getIndexOfChild**(java.lang.Object parent, java.lang.Object child) Returns the index of child in parent.
java.lang.Object	**getRoot**() Returns the root of the tree.
void	**insertNodeInto**(org.jdom.Element dataElement, org.jdom.Element parent, int index, javax.swing.tree.TreePath path) Invoked this to insert newChild at location index in parents

	children.
boolean	**isLeaf**(java.lang.Object node) Returns true if node is a leaf.
void	**nodesWereInserted**(org.jdom.Element newChild, org.jdom.Element parent, int index, javax.swing.tree.TreePath path) Invoke this method after you've inserted some Elements into Element.
void	**nodesWereInserted**(org.jdom.Text newChild, org.jdom.Element parent, int index, javax.swing.tree.TreePath path) Invoke this method after you've inserted some Text into Element.
void	**nodesWereRemoved**(org.jdom.Element node, int[] childIndices, java.lang.Object[] removedChildren, javax.swing.tree.TreePath path) Invoke this method after you've removed some nodes from an Element nodes.
private void	**notifyDataModification**() Notifies AdminToolFrame that its data (that is the Information Structure) is modified (and thus needs to be saved).
void	**removeAll**(java.lang.String childrenName) Removes all children of name childrenName or all the children with any name, if childrenName=="all" is entered, from the root of the Tree Model.
void	**removeNodeFromParent**(org.jdom.Element removedEl, javax.swing.tree.TreePath parentPath) Invoke this to remove an Element node from its parent Element.
void	**removeNodeFromParent**(org.jdom.Text text, javax.swing.tree.TreePath path) Invoke this to remove a Text node from its parent Element.
void	**removeTreeModelListener**(javax.swing.event.TreeModelListener l) Removes a listener previously added with addTreeModelListener.
void	**valueForPathChanged**(javax.swing.tree.TreePath path, java.lang.Object newValue) Not implemented.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

dataModifier
`DataModifier` **`dataModifier`**

Through this reference the GUIUpdater of AdminToolFrame is notified, when a modification on TreeModel is made. The information structure tree holds the *data* of the application. So, when a modification is done upon it, the GUIUpdater must be notified to change the GUI (for example to enable the Save Button).

---

doc
`org.jdom.Document` **`doc`**

The JDOM Document that this Tree Model is based.

---

listeners
`java.util.Vector` **`listeners`**

Event Listeners List. Listeners that listens to the Tree Model's changes.

---

root
`org.jdom.Element` **`root`**

The root of the Tree Model, which is the root of the document too.

Constructor Detail

JDOMTreeModel
```
public JDOMTreeModel(org.jdom.Document document,
                     DataModifier dataModifier)
```

Constructor.

The AdminToolFrame passed to the constructor is casted to a DataModifier reference. See Interface DataModifier for more.

Parameters:

`document` - - the document of the TreeModel.

`dataModifier` - - the reference through which GUIUpdater will be notified.

See Also:

`DataModifier`

Method Detail

addTreeModelListener
```
public void addTreeModelListener(javax.swing.event.TreeModelListener l)
```

Adds a listener for the TreeModelEvent posted after the tree changes.

Specified by:

`addTreeModelListener` in interface `javax.swing.tree.TreeModel`

Parameters:

`l` - - the listener to add

See Also:

`removeTreeModelListener(TreeModelListener)`

---

fireTreeNodesInserted

```
private void fireTreeNodesInserted(javax.swing.tree.TreePath path,
                                   int[] indices,
                                   java.lang.Object[] children)
```

Notifies all listeners that have registered interest for notification on this event type (Tree Model Event).
The event instance is lazily created using the parameters passed into the fire method. The TreePath parameter which is initially passed to insertNodeInto method is used to create the event.
It is necessary to fire events (TreeModelEvents) in order for the JTree to be notified and update its apperance. That is upon initialization JTree adds listeners to its Tree Model, to listen for events that change the structure of the tree (Tree Model Events).

Parameters:

`path` - the path in the tree where the new node was inserted
`indices` - indices of new nodes
`children` - new nodes

See Also:

`nodesWereInserted(Text , Element, int , TreePath)`, `nodesWereInserted(Element , Element , int , TreePath )`

---

fireTreeNodesRemoved

```
private void fireTreeNodesRemoved(java.lang.Object source,
                                  javax.swing.tree.TreePath path,
                                  int[] childIndices,
                                  java.lang.Object[] children)
```

Notifies all listeners that have registered interest for notification on this event type (Tree Model Event).
The event instance is lazily created using the parameters passed into the fire method. The TreePath parameter which is initially passed to removeNodeFromParent method is used to create the event.
It is necessary to fire events (TreeModelEvents) in order for the JTree to be notified and update its apperance. That is upon initialization JTree adds listeners to its Tree Model, to listen for events that change the structure of the tree (Tree Model Events).

Parameters:

`source` - the node where elements are being removed
`path` - the path to the parent of the node which was removed
`childIndices` - the indices of the removed elements
`children` - the removed elements

---

getChild

```
public java.lang.Object getChild(java.lang.Object parent,
                                 int index)
```

Returns the child of parent at "index" index in the parent's child array. parent must be a node previously obtained from this data source. This should not return null if index is a valid index for parent (that is index >= 0 && index < getChildCount(parent)).

Specified by:

`getChild` in interface `javax.swing.tree.TreeModel`

Parameters:

`parent` - a node in the tree, obtained from this data source

`index` - the index

Returns:

the child of parent at index index

---

getChildCount

```
public int getChildCount(java.lang.Object parent)
```

Returns the number of children of parent. Returns 0 if the node is a leaf or if it has no children. Parent must be a node previously obtained from this data source.

Specified by:

`getChildCount` in interface `javax.swing.tree.TreeModel`

Parameters:

`parent` - - a node in the tree, obtained from this data source

Returns:

the number of children of the node parent

See Also:

`org.jdom.Element.getContentSize()`, `org.jdom.Document.getContentSize()`

---

getIndexOfChild

```
public int getIndexOfChild(java.lang.Object parent,
                           java.lang.Object child)
```

Returns the index of child in parent. If parent is null or child is null, returns -1. Parent can be: a Document or an Element Child can be: an Element or Content.

Specified by:

`getIndexOfChild` in interface `javax.swing.tree.TreeModel`

Parameters:

`parent` - a node in the tree, obtained from this data source

`child` - the node we are interested in

Returns:

the index of the child in the parent, or -1 if either child or parent are null

---

getRoot

```
public java.lang.Object getRoot()
```

Returns the root of the tree.

Specified by:

`getRoot` in interface `javax.swing.tree.TreeModel`

Returns:

Object - the root of the tree

---

insertNodeInto

```
public void insertNodeInto(org.jdom.Element dataElement,
                           org.jdom.Element parent,
                           int index,
                           javax.swing.tree.TreePath path)
```

Invoked this to insert newChild at location index in parents children.

This will then message nodesWereInserted to create the appropriate event. This is the preferred way to add children as it will create the appropriate TreeModelEvent.

It supports three cases of insertion. Case where insertion occurs:

- on an Information Field, when a Associtions was created by a DnD. Cases of Information Filed:
  - not associated yet, so it can be associated
  - already associaited, show error message
- on the root of Data Structrue tree, when a data structure is loaded
- on the root of Inforation Structrue tree, when a information structure is loaded

It was written based on DefaultTreeModel's insertNodeInto method. So, after an insertion occurs, nodesWereInserted method is called to fire the TreeModelEvents in order for JTree to be notified that its Tree Model is changed, and update its appearance.

After insertion, it notifies that data is modified in all cases (also when an insertion occures in the Data Structure tree, which is not needed). Nevertheless, this way was prefered because it would be easier to extend the program (in future release when data structure might be considered to be data).

Parameters:
`dataElement` - the new child
`parent` - the parent
`index` - the index
`path` - the path in the tree where the new node will be inserted, used for firrong the TreeModelEvent
See Also:
`nodesWereInserted(Text , Element, int , TreePath)`,
`nodesWereInserted(Element , Element , int , TreePath )`

---

isLeaf

```
public boolean isLeaf(java.lang.Object node)
```

Returns true if node is a leaf.
Only Element and Document have Children. If a Document or an Element has no Content, then it is considered to be a leaf and thus the method returns true.
Specified by:
`isLeaf` in interface `javax.swing.tree.TreeModel`
Parameters:
`node` - - a node in the tree, obtained from this data source
Returns:
true if node is a leaf

---

nodesWereInserted

```
public void nodesWereInserted(org.jdom.Element newChild,
                              org.jdom.Element parent,
                              int index,
                              javax.swing.tree.TreePath path)
```

Invoke this method after you've inserted some Elements into Element. Gathers parameters for fireTreeNodesInserted method and makes the call to it. The

fireTreeNodesInserted method is the one that actually fires the Tree Model Events.
It is called by insertNodeInto.

Parameters:

`newChild` - Element child

`parent` - parent

`index` - the position in parent where the new node was inserted

`path` - the path in the tree where the new node was inserted

See Also:

`insertNodeInto(Element , Element , int , TreePath )`,
`fireTreeNodesInserted(TreePath , int[] , Object[] )`

---

nodesWereInserted

```
public void nodesWereInserted(org.jdom.Text newChild,
                              org.jdom.Element parent,
                              int index,
                              javax.swing.tree.TreePath path)
```

Invoke this method after you've inserted some Text into Element. Gathers parameters for fireTreeNodesInserted method and makes the call to it. The fireTreeNodesInserted method is the one that actually fires the Tree Model Events.
It is called by insertNodeInto.

Parameters:

`newChild` - Text child

`parent` - parent

`index` - the position in parent where the new node was inserted

`path` - the path in the tree where the new node was inserted

See Also:

`insertNodeInto(Element , Element , int , TreePath )`,
`fireTreeNodesInserted(TreePath , int[] , Object[] )`

---

nodesWereRemoved

```
public void nodesWereRemoved(org.jdom.Element node,
                             int[] childIndices,
                             java.lang.Object[] removedChildren,
                             javax.swing.tree.TreePath path)
```

Invoke this method after you've removed some nodes from an Element nodes. Gathers parameters for fireTreeNodesRemoved method and makes the call to it. The fireTreeNodesRemoved method is the one that actually fires the Tree Model Events.
It is called by reomoveNodeFromParent.

Parameters:

`node` - the parent

`childIndices` - array with the indices of these nodes in ascending order

`removedChildren` - array with the removed nodes

`path` - the path to the parent of the node which was removed

---

notifyDataModification

```
private void notifyDataModification()
```

Notifies AdminToolFrame that its data (that is the Information Structure) is modified (and thus needs to be saved).

It checks to see which tree it is and only calls dataModified method in AdminToolFrame when it founds that it is a information structure tree.
Data Structure is not considered to be data, so we do not need to do any notificationwhen an isertion or deletion is made.
The test, whether or not the notification should occured, was prefered to be done here for making the application easier to be extend in the future.

See Also:
`DataModifier.dataModified()`

---

removeAll

```
public void removeAll(java.lang.String childrenName)
```

Removes all children of name `childrenName` or all the children with any name, if childrenName=="all" is entered, from the root of the Tree Model.
The remove action is delegate to the removeNodeFromParent method. Used in AdminToolFrame to clear the trees.

Parameters:
`childrenName` - the name of the children of hte root that must be renoved. If "all" the everything is removed

See Also:
`removeNodeFromParent(Element , TreePath )`

---

removeNodeFromParent

```
public void removeNodeFromParent(org.jdom.Element removedEl,
                                 javax.swing.tree.TreePath parentPath)
```

Invoke this to remove an Element node from its parent Element.
This will then message nodesWereRemoved to create the appropriate event. This is the preferred way to remove children as it will create the appropriate TreeModelEvent.

It was written based on DefaultTreeModel's removeNodeFromPArent method. So, after an removal occurs, nodesWereRemoved method is called to fire the TreeModelEvents in order for JTree to be notified that its Tree Model is changed, and update its appearance.

After removal, it notifies that data is modified in all cases (also when a removal occures in the Data Structure tree, which is not needed). Nevertheless, this way was prefered because it would be easier to extend the program (in future release when data structure might be considered to be data).

Parameters:
`removedEl` - the Text node to be removed
`parentPath` - the path to the parent of the node which is going to be removed

See Also:
`nodesWereRemoved(Element , int[] , Object[] , TreePath )`

---

removeNodeFromParent

```
public void removeNodeFromParent(org.jdom.Text text,
```

javax.swing.tree.TreePath path)

Invoke this to remove a Text node from its parent Element.
This will then message nodesWereRemoved to create the appropriate event. This is the preferred way to remove children as it will create the appropriate TreeModelEvent.

It was written based on DefaultTreeModel's removeNodeFromPArent method. So, after an removal occurs, nodesWereRemoved method is called to fire the TreeModelEvents in order for JTree to be notified that its Tree Model is changed, and update its appearance.

After removal, it notifies that data is modified in all cases (also when a removal occures in the Data Structure tree, which is not needed). Nevertheless, this way was prefered because it would be easier to extend the program (in future release when data structure might be considered to be data).

See Also:
`nodesWereRemoved(Element , int[] , Object[] , TreePath )`

---

removeTreeModelListener

```
public void removeTreeModelListener(javax.swing.event.TreeModelListener l)
```

Removes a listener previously added with addTreeModelListener.

Specified by:
`removeTreeModelListener` in interface `javax.swing.tree.TreeModel`

Parameters:
`l` - - the listener to remove

See Also:
`addTreeModelListener(TreeModelListener)`

---

valueForPathChanged

```
public void valueForPathChanged(javax.swing.tree.TreePath path,
                                java.lang.Object newValue)
```

Not implemented. Because it is not been used.

Specified by:
`valueForPathChanged` in interface `javax.swing.tree.TreeModel`

Parameters:
`path` - TreePath
`newValue` - Object

---

admintool.jdomtreemodel

Class JDOMTreeModelAttr

```
java.lang.Object
  └admintool.jdomtreemodel.JDOMTreeModel
      └admintool.jdomtreemodel.JDOMTreeModelAttr
```

All Implemented Interfaces:

javax.swing.tree.TreeModel

public class **JDOMTreeModelAttr**
extends JDOMTreeModel

Tree Model implementation to display a JDOM Document in a JTree, with insert and remove support and displaying attributes.

Extends the functionality of its superclass by overiding the display tree structure methods in order to put Atribute display support.

Must be used only in a NON mutable JTree. No mutation support is added for attributes yet.

Used by Data Structure tree were Attributes need to be displayed.

Before reading this class' API, first read its superclass' API.

See Also:

JDOMTreeModel

Field Summary

Fields inherited from class admintool.jdomtreemodel.JDOMTreeModel
dataModifier, doc, listeners, root

Constructor Summary

JDOMTreeModelAttr(org.jdom.Document document, DataModifier admintoolFrameGUIUpdater) Constructor.

Method Summary

java.lang.Object	**getChild**(java.lang.Object parent, int index) Returns the child of parent at "index" index in the parent's child array.
int	**getChildCount**(java.lang.Object parent) Returns the number of children of parent.
int	**getIndexOfChild**(java.lang.Object parent, java.lang.Object child) Returns the index of child in parent.
boolean	**isLeaf**(java.lang.Object node) Returns true if node is a leaf.

Methods inherited from class admintool.jdomtreemodel.JDOMTreeModel

addTreeModelListener, getRoot, insertNodeInto, nodesWereInserted, nodesWereInserted, nodesWereRemoved, removeAll, removeNodeFromParent, removeNodeFromParent, removeTreeModelListener, valueForPathChanged

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

JDOMTreeModelAttr

```
public JDOMTreeModelAttr(org.jdom.Document document,
                         DataModifier admintoolFrameGUIUpdater)
```

Constructor. Just calls the superclass constructor.

Parameters:

`document` - the document of the TreeModel.

Method Detail

getChild

```
public java.lang.Object getChild(java.lang.Object parent,
                                 int index)
```

Returns the child of parent at "index" index in the parent's child array. See superclass' method description for more.

It manipulates the two list (the attribute list and the Content list) as one list. This list begins with the attribute list and continues with the content list. So, there are the following index cases when parent is an Element:

- index < number of attributes, so index indicates an attribute. The index stays the same.
- number of attributes < index < number of attributes and Content, so index indicates a Content. Thus Index = index - number of attributes, and use new index to get the content.
- index > number of attributes and Content OR index < 0, out of bounds

Specified by:
`getChild` in interface `javax.swing.tree.TreeModel`
Overrides:
`getChild` in class `JDOMTreeModel`
Parameters:
`parent` - a node in the tree, obtained from this data source
`index` - the index
Returns:
the child of parent at index index

---

getChildCount

```
public int getChildCount(java.lang.Object parent)
```

Returns the number of children of parent. Returns 0 if the node is a leaf or if it has no children. Parent must be a node previously obtained from this data source. See superclass' method description for more.
When the parent is an Element, it returns the sum of the number of elements and the number of attributes
Specified by:
`getChildCount` in interface `javax.swing.tree.TreeModel`
Overrides:
`getChildCount` in class `JDOMTreeModel`
Parameters:
`parent` - - a node in the tree, obtained from this data source
Returns:
the number of children of the node parent
See Also:
`org.jdom.Element.getContentSize()`, `org.jdom.Document.getContentSize()`

---

getIndexOfChild

```
public int getIndexOfChild(java.lang.Object parent,
                           java.lang.Object child)
```

Returns the index of child in parent. If parent is null or child is null, returns -1. See superclass' method description for more.
Parent can be: a Document or an Element Child can be: an Element, a Content or an Attribute.
In the case of an Element or a Content child, it returns the sum of the number of attributes that the parent Element has, plus the index of the Element or Content child.

Specified by:
`getIndexOfChild` in interface `javax.swing.tree.TreeModel`
Overrides:
`getIndexOfChild` in class `JDOMTreeModel`
Parameters:
`parent` - a node in the tree, obtained from this data source
`child` - the node we are interested in
Returns:
the index of the child in the parent, or -1 if either child or parent are null

---

isLeaf

`public boolean isLeaf(java.lang.Object node)`

Returns true if node is a leaf. See superclass' method description for more.
Specified by:
`isLeaf` in interface `javax.swing.tree.TreeModel`
Overrides:
`isLeaf` in class `JDOMTreeModel`
Parameters:
`node` - - a node in the tree, obtained from this data source
Returns:
true if node is a leaf

Overview Package Class Tree Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES All Classes
SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

---

admintool.jdomtreemodel

Class XMLWhitespaceFilter

```
java.lang.Object
  └org.xml.sax.helpers.XMLFilterImpl
      └admintool.jdomtreemodel.XMLWhitespaceFilter
```

All Implemented Interfaces:
org.xml.sax.ContentHandler, org.xml.sax.DTDHandler, org.xml.sax.EntityResolver, org.xml.sax.ErrorHandler, org.xml.sax.XMLFilter, org.xml.sax.XMLReader

public class **XMLWhitespaceFilter**
extends org.xml.sax.helpers.XMLFilterImpl

XML filter used in order to avoid loading ignorable whitespace as Text nodes.

The application should be able to load XML files without their corresponinding DTD ot XSD for validation. So, if an XML filter is not used while loading the XML file with the SAXBuilder, and SAXBuilder would mistakely consider the whitespace Text nodes, as Text nodes with chatacter data. That is the SAXBuilder cannot distinguish which Text node is useful and which has unwanted whitespace text.
The only way to specify to SAXBuilder which are the useful Text nodes is to use a XML filter.

Field Summary

Fields inherited from class org.xml.sax.helpers.XMLFilterImpl

Constructor Summary

`XMLWhitespaceFilter()` Default constructor.
`XMLWhitespaceFilter(org.xml.sax.XMLReader parent)` Constructor passing a XMLReader in superclass constructor - not used in this version of the application.

Method Summary

`void`	`characters(char[] ch, int start, int length)` Overriding characters method of of SAXBuilder to filter character data the create the Text nodes.

Methods inherited from class org.xml.sax.helpers.XMLFilterImpl
`endDocument, endElement, endPrefixMapping, error, fatalError, getContentHandler, getDTDHandler, getEntityResolver, getErrorHandler, getFeature, getParent, getProperty, ignorableWhitespace, notationDecl, parse, parse, processingInstruction, resolveEntity, setContentHandler, setDocumentLocator, setDTDHandler, setEntityResolver, setErrorHandler,`

setFeature, setParent, setProperty, skippedEntity, startDocument, startElement, startPrefixMapping, unparsedEntityDecl, warning

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

XMLWhitespaceFilter

```
public XMLWhitespaceFilter()
```

Default constructor.

XMLWhitespaceFilter

```
public XMLWhitespaceFilter(org.xml.sax.XMLReader parent)
```

Constructor passing a XMLReader in superclass constructor - not used in this version of the application.

Parameters:

`parent` - XMLReader

Method Detail

characters

```
public void characters(char[] ch,
                       int start,
                       int length)
                throws org.xml.sax.SAXException
```

Overriding characters method of of SAXBuilder to filter character data the create the Text nodes.

Creates a string from character data and trims its whitespace with String.trim method. If trimmed string is not an empty string, then the character data are not whitspace data, thus pass them to the superclass' method to create the Text node. Else, do nothing.

Parameters:

`ch` - character data

`start` - start index

`length` - length

Throws:

`org.xml.sax.SAXException`

---

admintool.jdomtreemodel

Class XPathEvaluator

```
java.lang.Object
  └ admintool.jdomtreemodel.XPathEvaluator
```

public class **XPathEvaluator**
extends java.lang.Object

Utility class that encloses the mechanism of evaluting the xpath of a given node in a JDOM document. A *node* can be an Element, an Attribute, CDATA or Text.

The XPath is return by the getXPath method. For example to estimate the following XPath:
`/dataStructureSource/dataStructureTable[3]/dataStructureField[5]`
the XPath is divided into 3 parts. One part for each element. These parts are estimated by the getElementXPath method.

See Also:
`TreeTransferHandler`

Field Summary

private java.lang.Object	**object** XPath of this node is evaluted.

Constructor Summary

XPathEvaluator() Default constructor.

Method Summary

private	**getElementXPath**(java.lang.Object currentNode)

`java.lang.String`	Estimates the part of XPath of the object passed.
`java.lang.String`	**`getXPath`**`(java.lang.Object o)` Returns the complete XPath expression for this node.

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Field Detail

object
`private java.lang.Object` **`object`**

XPath of this node is evaluted.

Constructor Detail

XPathEvaluator
`public` **`XPathEvaluator`**`()`

Default constructor.

Method Detail

getElementXPath
`private java.lang.String` **`getElementXPath`**`(java.lang.Object currentNode)`

Estimates the part of XPath of the object passed.

- Creates a string buffer to store the string:
- adds the element's name
- adds the selector ([x]) that this element has in its parent child list. if it's the only child, no selector is added

Parameters:
`currentNode` - the element that its part of the XPath expression is estimated
Returns:
String the part of XPath expression of this element

---

getXPath
`public java.lang.String` **`getXPath`**`(java.lang.Object o)`

Returns the complete XPath expression for this node. A *node* can be an Element or an Attribute.

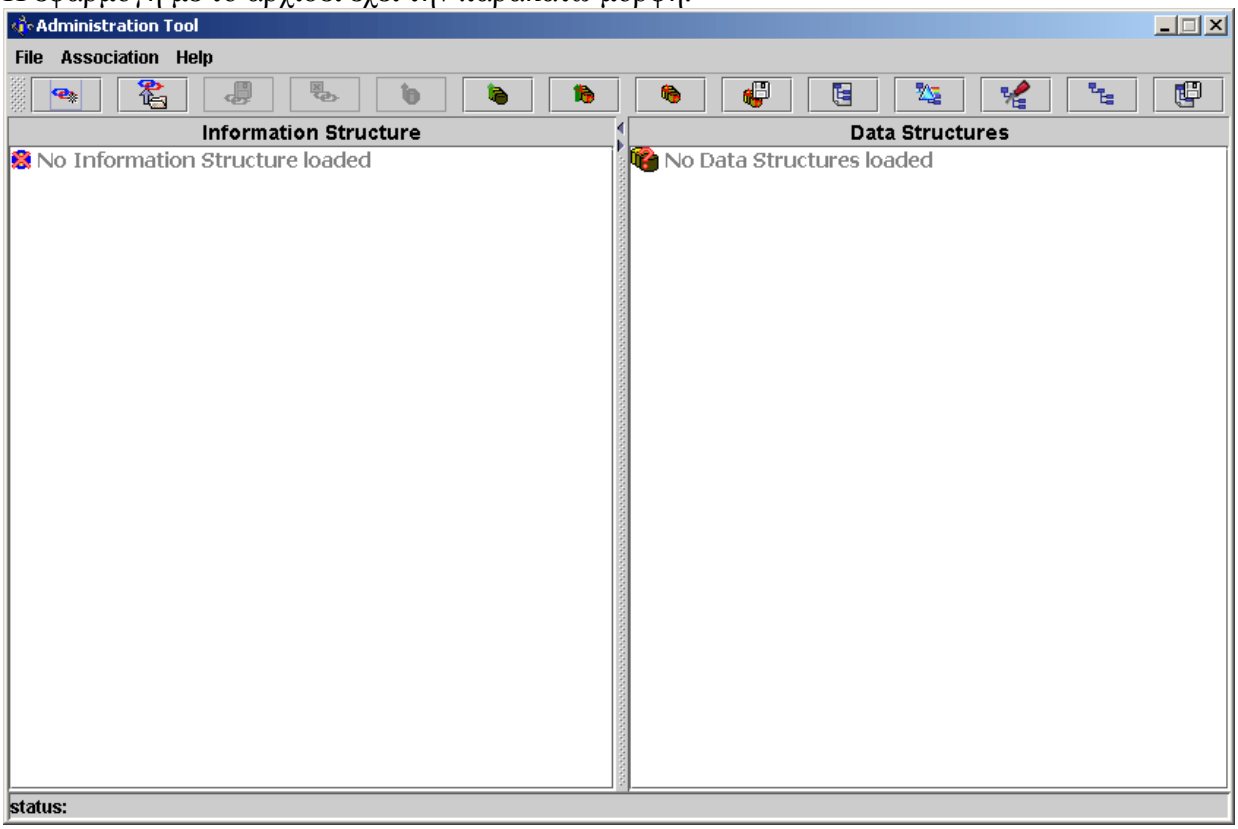
Firstly, it initializes the object variable. There are several cases of the object:

- Case of: Element
  - if element is root element return error message
  - store in a filo queue the Elements from selection to root
  - create a string buffer to store the XPath string and for every element from root to selection, append to buffer the XPath part of each element. The XPath part is estimated from getElementXPath method.
- Case of: Attribute
  - Creates a new instance of XPathEvaluator to estimate the XPath of its parent element.
  - Appends the XPath part of the Attribute
- Case of: CDATA. XPath does not support CDATA the way it supports Text, so only the parent element XPath is returned.
  - Creates a new instance of XPathEvaluator to estimate the XPath of its parent element.
- Case of: Text
  - Creates a new instance of XPathEvaluator to estimate the XPath of its parent element.
  - Appends the XPath part of the Text

Parameters:
o - the object that we want to estimate its XPath
Returns:
complete XPath.
See Also:
`getElementXPath(Object )`

Οδηγίες χρήσεως

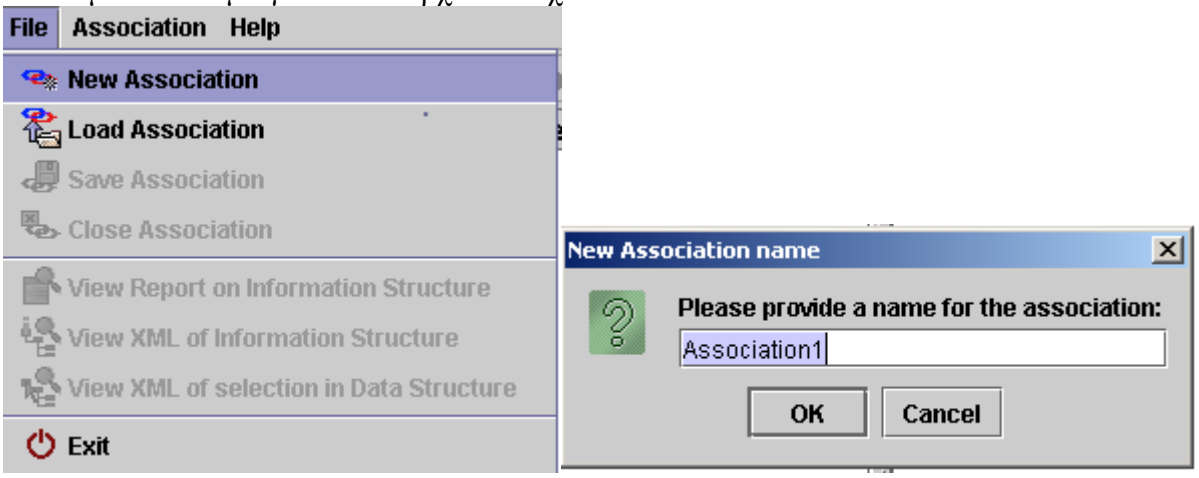
Ακολουθεί ένα σενάριο χρήσης της εφαρμογής.
Η εφαρμογή με το αρχίσει έχει την παρακάτω μορφή.



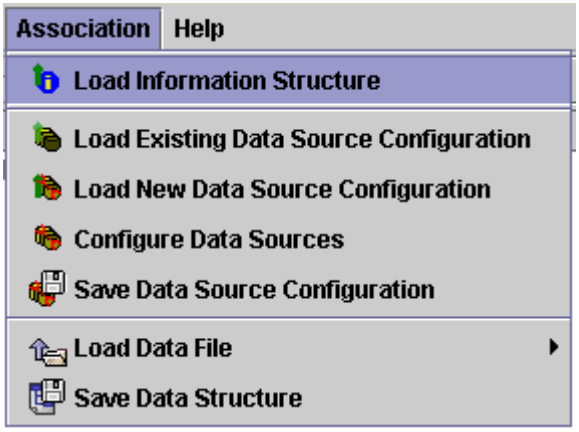
Κανένα αρχειο δεν είναι φορτωμένο, ενώ αξίζει να παρατηρήσουμε οτι μερικές επιλογές (buttons) είναι απενεργοποιημένα.
Στην συνέχεια υπάρχουν δυο πιθανα σενάρια:

- Δημιουργία νέο αρχειου συσχετισμών (New Association), και να φορτώσουμε σε αυτό μια Information Structure
- Φόρτωση υπάρχοντος αρχείου (Load Association)

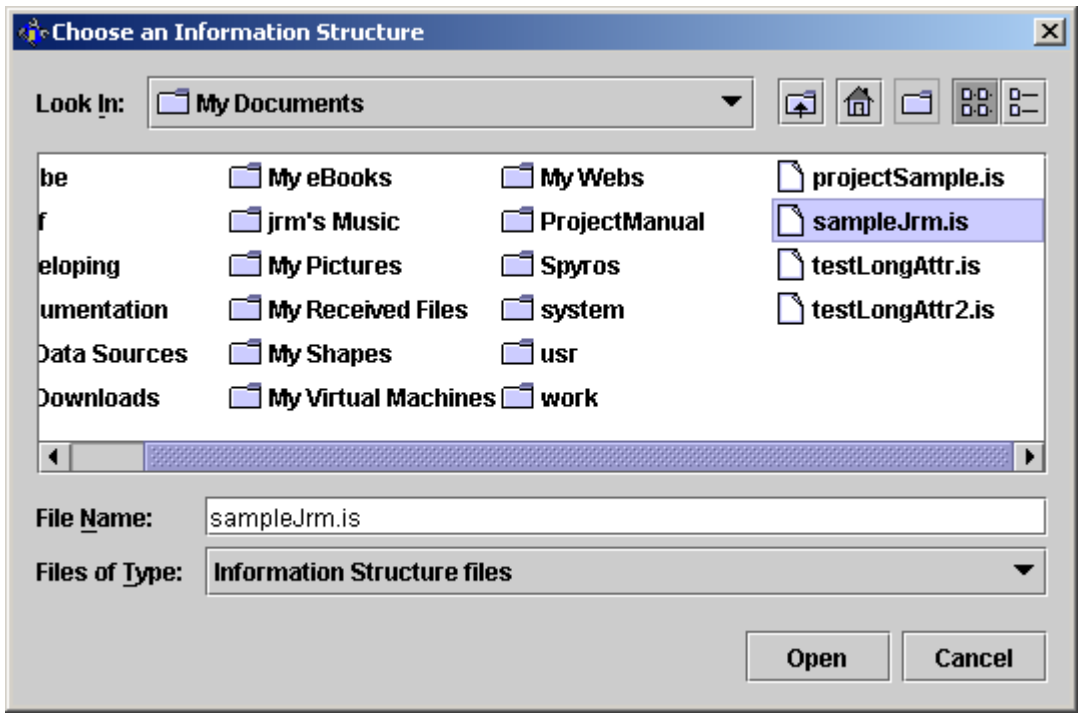
Δημιουργία νέο αρχειου συσχετισμών (New Association)
Επιλεγοντας New Association, εμαφανίζεται το διπλανο παράθυρο που μας προτρέπει να δώσουμε ένα όνομα για το νέο αρχείο συσχετίσεων



Στην συνέχεια επιλέγουμε να φορτώσουμε μια Information Structure

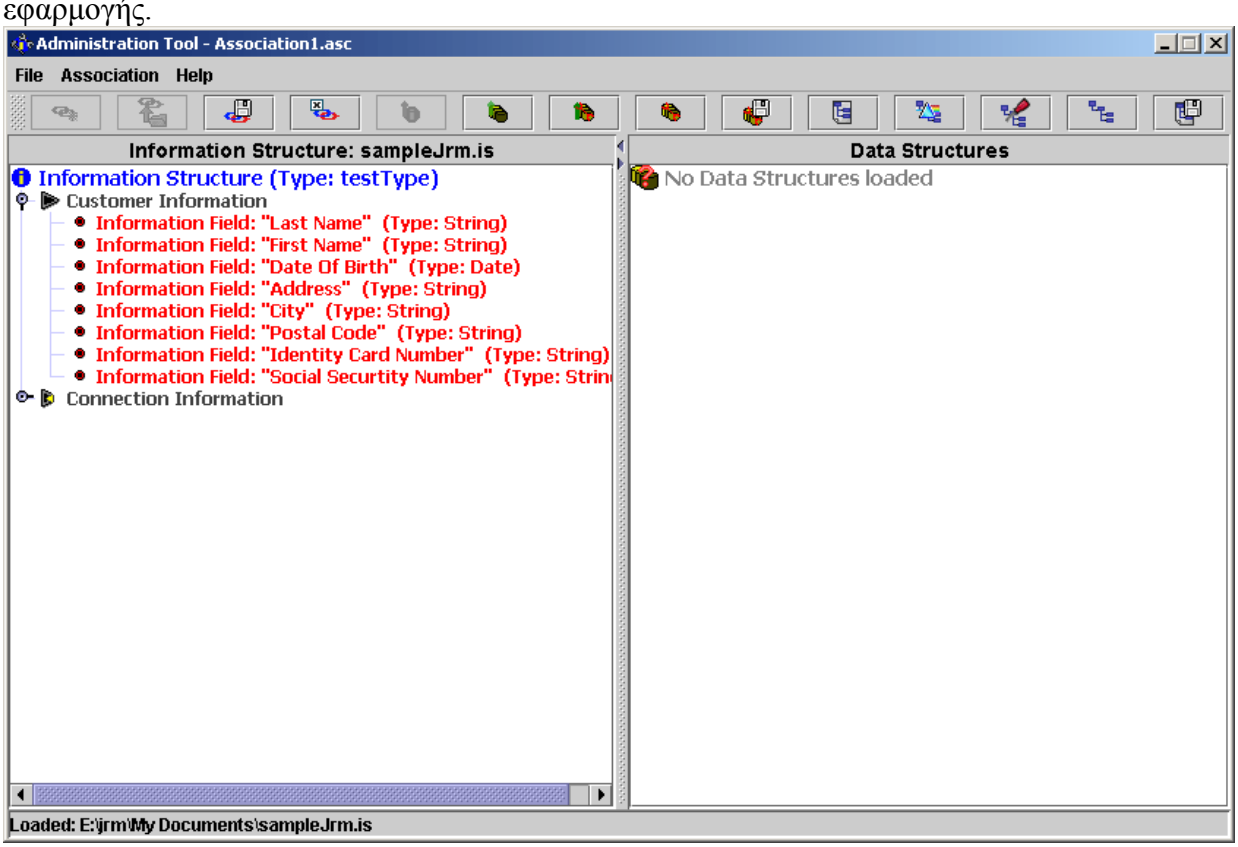


Εμφανίζεται το παράθυρο για την επιλογή αρχείου.



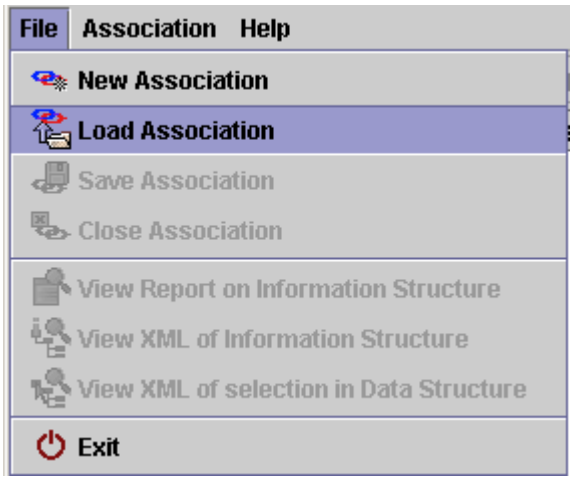
Παρατηρούμε ότι φαίνονται μόνο κατάλογοι και αρχεία τύπου Information Structure (*.is)

Αφού φορτωθεί το Information Structure αρχείο, απεικονίζεται στο αριστερό με΄ρος της εφαρμογής.

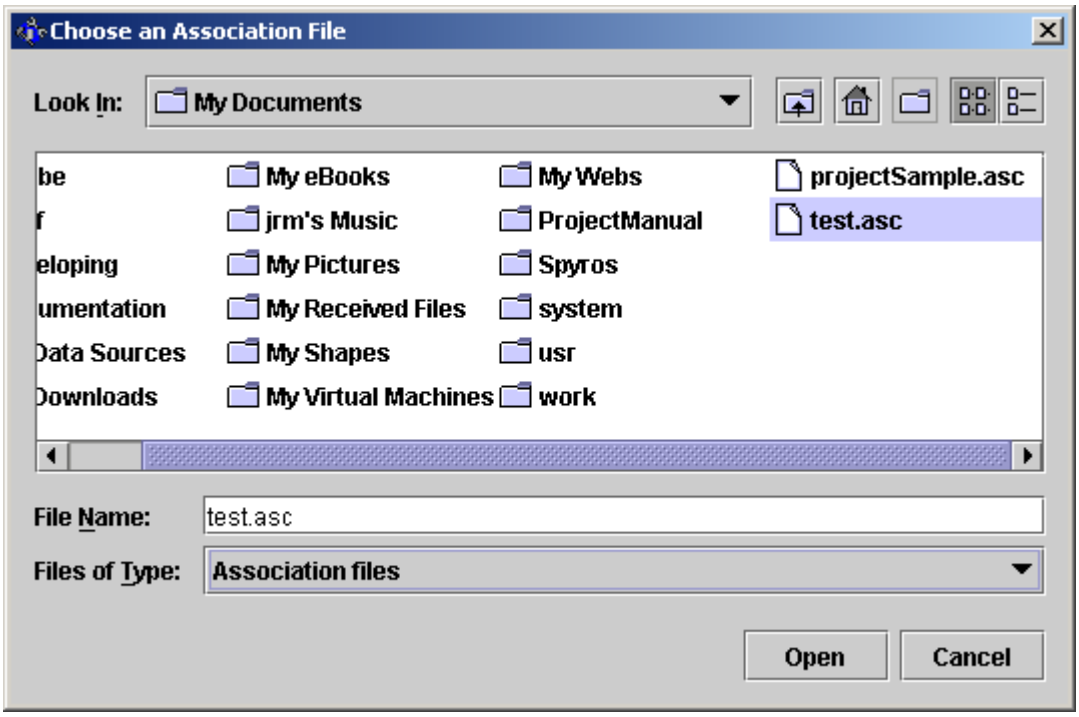


Παρατηρούμε ότι απο την στιγμή που δημιουργήθηκε ένα αρχειο συσχετισμού έχουν ενεργοποιηθεί και οι υπόλοιπες επιλογές. Παρόλα αυτά στην παραπάνω εικόνα η επιλογή “Load Information Structure” ειναι απενεργοποιημένη διότι εχουμε ήδη φορτώσει αρχειο τέτοιου είδους.

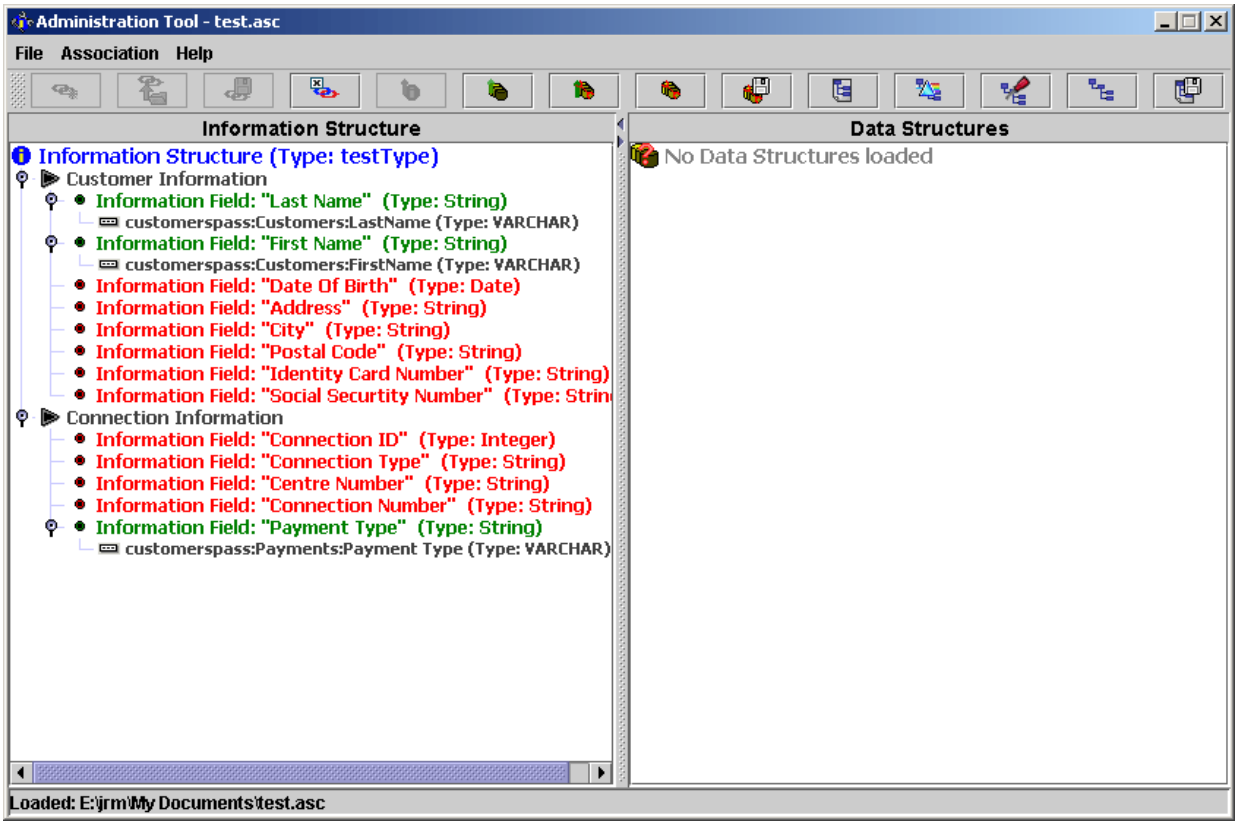
Φόρτωση υπάρχοντος αρχείου (Load Association)
Επιλέγουμε Load Association,



και η εφαρμογή μας προτρέπει να επιλέξουμε ένα αρχειο.



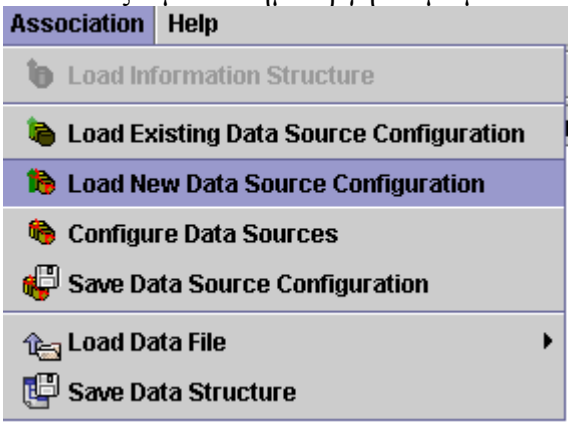
Αφού φορτώσουμε το αρχείο, η εφαρμογή το απεικονίζει.



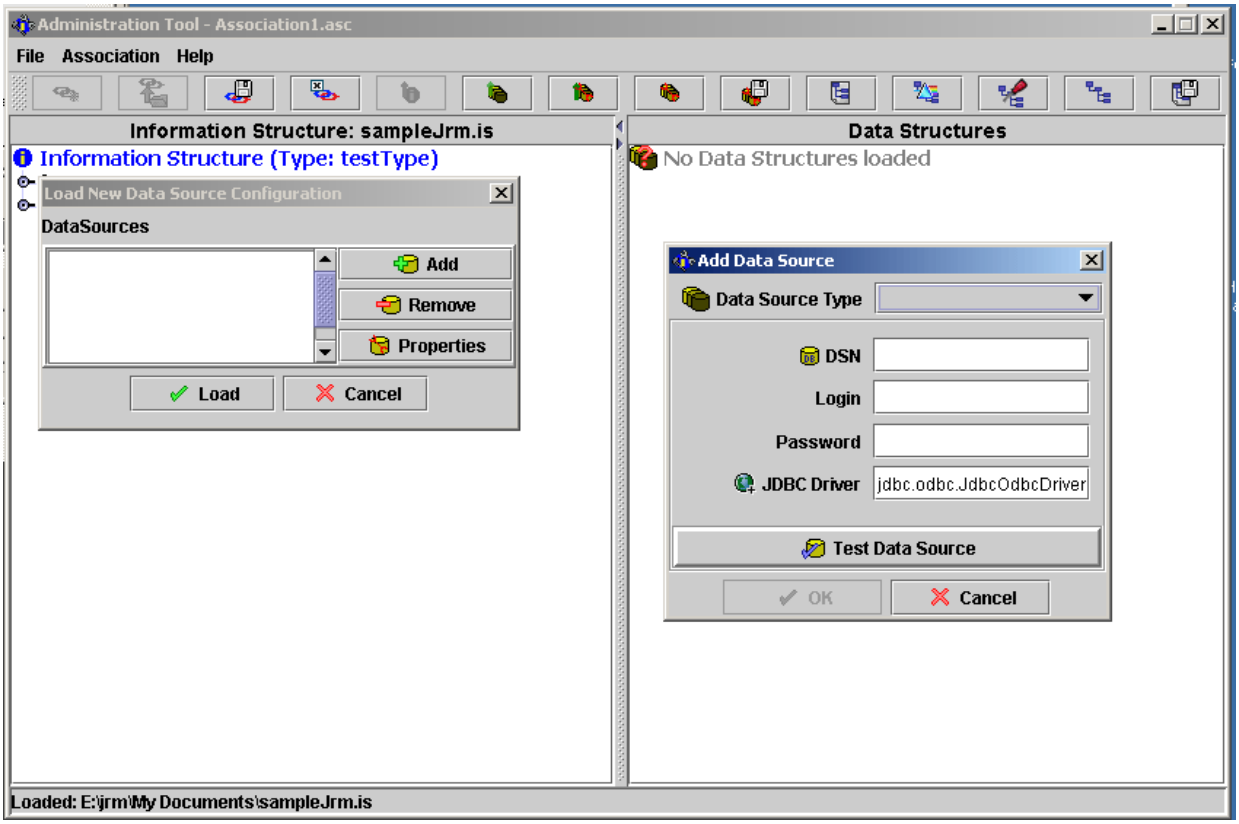
Παρατηρούμε ότι φαίνονται και οι συσχετισμοί που έχουν δημιουργηθεί στο αρχείο απο την προηγούμενη φορά.

Συνεχίζοντας στο πρώτο σενάριο (Δημιουργία νέο αρχειου συσχετισμών (New Association), και να φορτώσουμε σε αυτό μια Information Structure), έστω οτι θέλουμε να φορτώσουμε πηγές δεδομένων για να τις συσχετίσουμε.

Αν επιλέξουμε να δημιουργήσουμε μια νέα διαμόρφωση πηγών δεδομένων,

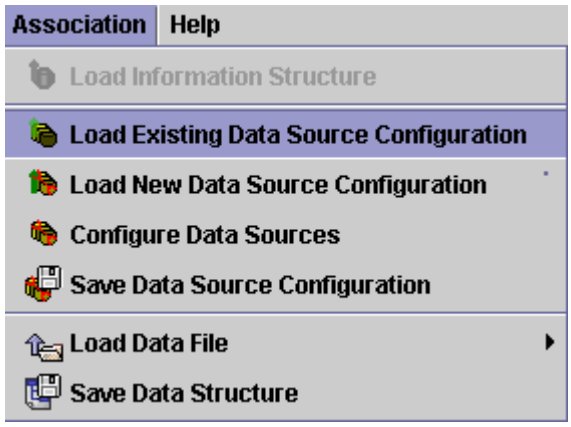


έχουμε:

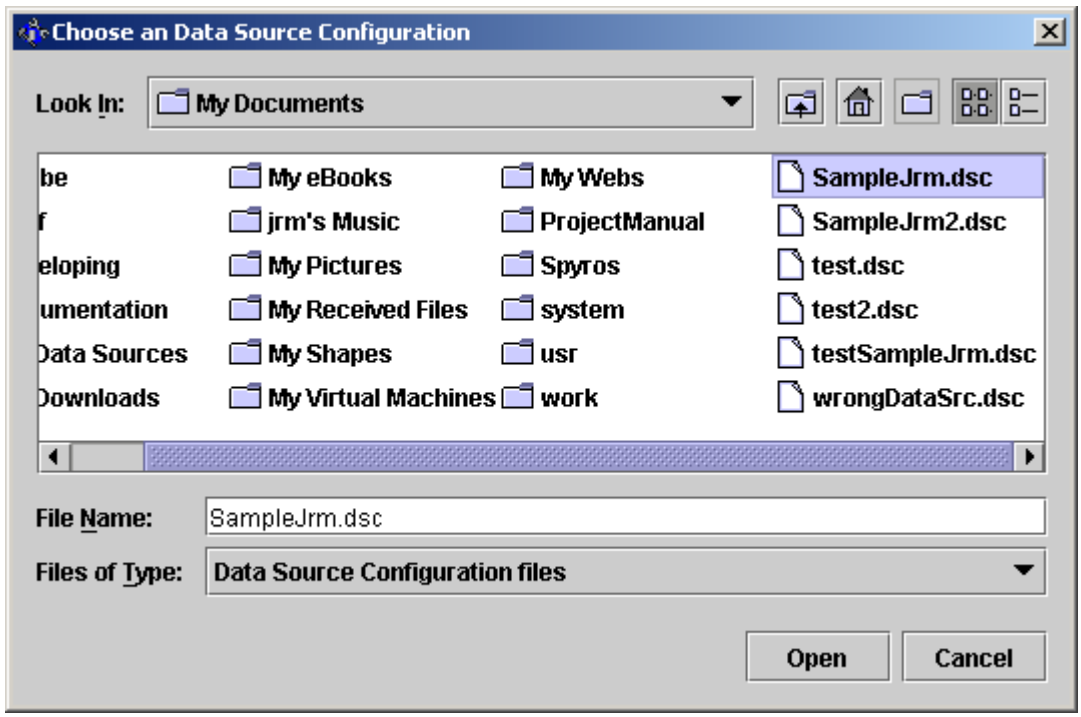


Στην παραπάνω εικόνα, έχουμε δυο παράθυρα πάνω απο το κυρίως παράθυρο της εφαρμογής. Στο αριστερό παράθυρο επιλέγουμε «Add» και εμφανίζεται το δεξί παράθυρο όπου μπορούμε να εισάγουμε τις παραμέτρους της νέας πηγής δεδομένων. Παρατηρούμε ότι πρέπει πρώτα να τεστάρουμε ότι οι παράμετροι τις πηγής είναι σωστές, για να μπορέσουμε να την δημιουργήσουμε (να ενεργοποιηθεί το OK κουμπί). Πατώντας OK, η πηγή προστίθεται στη λίστα του αριστερού παραθύρου. Στην συνέχεια μπορούμε να δημιουργήσουμε και άλλη πηγή, να διαγράψουμε καποια απο τις οποίες δημιουργήσαμε ή να τροποποιήσουμε καποια απο αυτές.

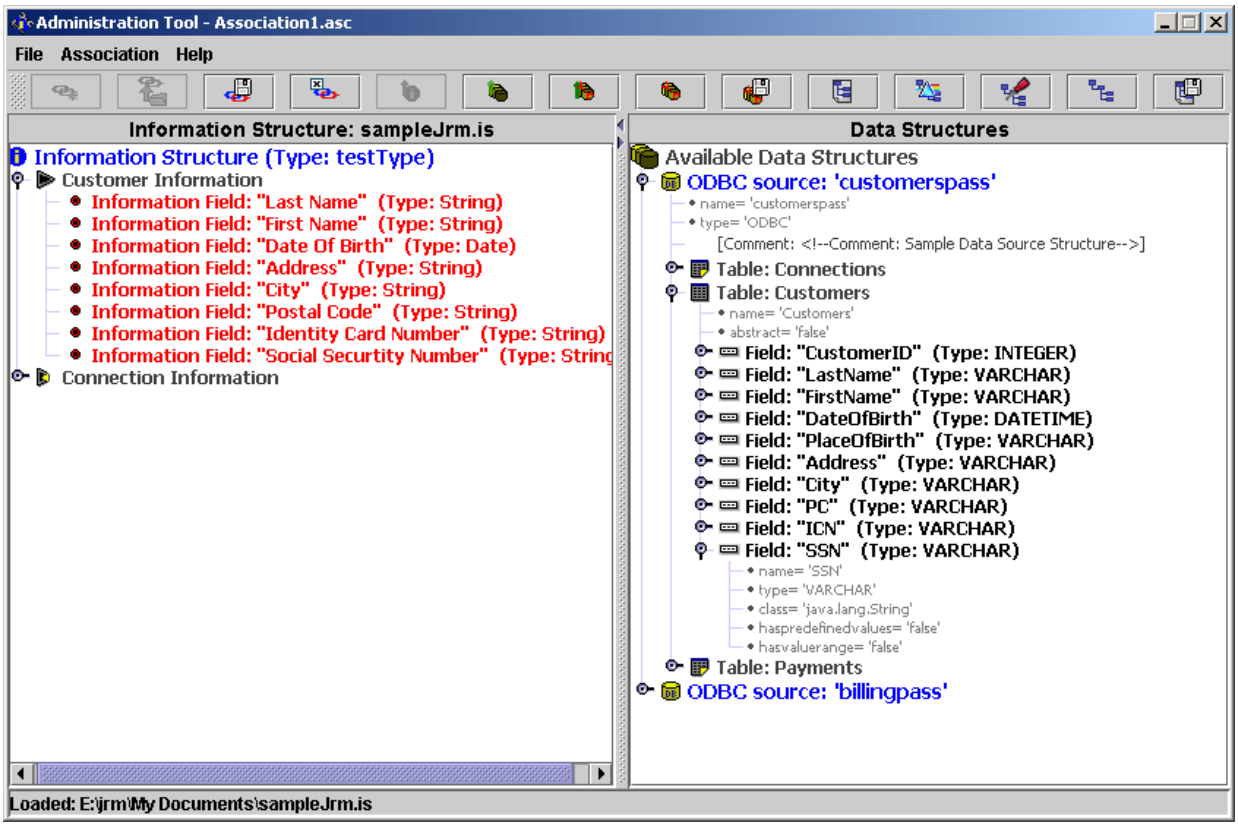
Αν επιλέξουμε να φορτώσουμε μια ήδη υπάρχουσα διαμορφωση, επιλέγουμε



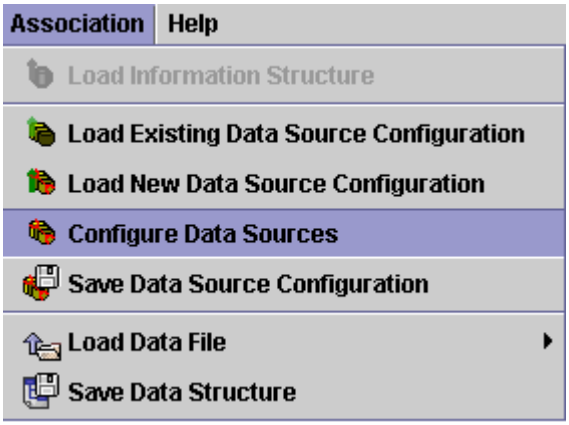
Όπου εμφανίζεται το παράθυρο επιλογής



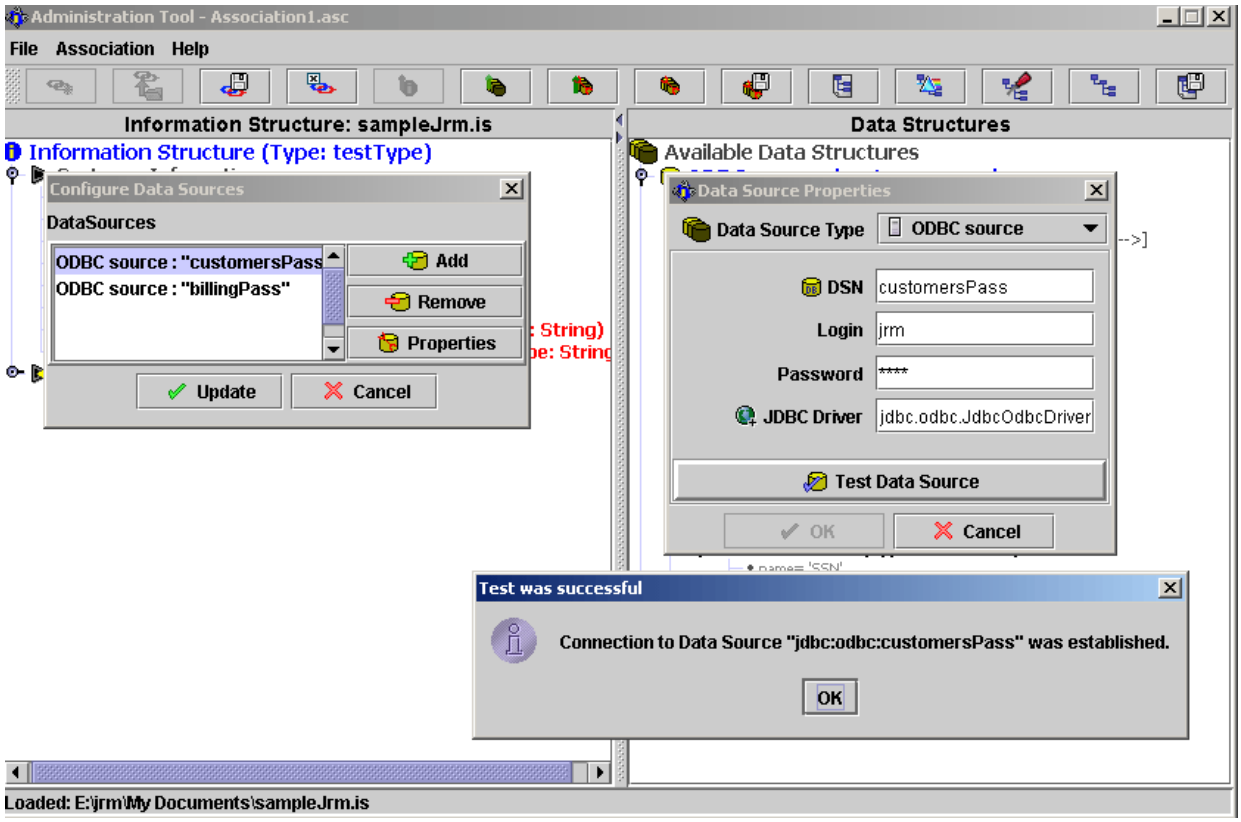
Έστω ότι έχουμε φορτώσει με οποιοδήποτε απο τους δυο παραπάνω τρόπους δυο πηγές δεδομένων, τότε η εφαρμογή τις απεικονίζει στο δεξί της μέρος



Μπορούμε να τροποποιήσουμε την πηγές που έχουμε φορτώσει, επιλέγοντας

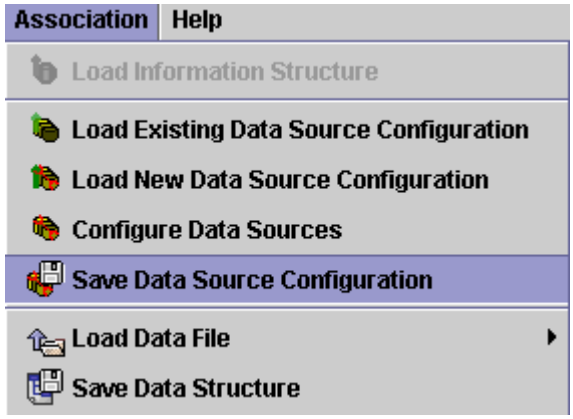


Εμφανίζονται τα πατακάτω παράθυρα

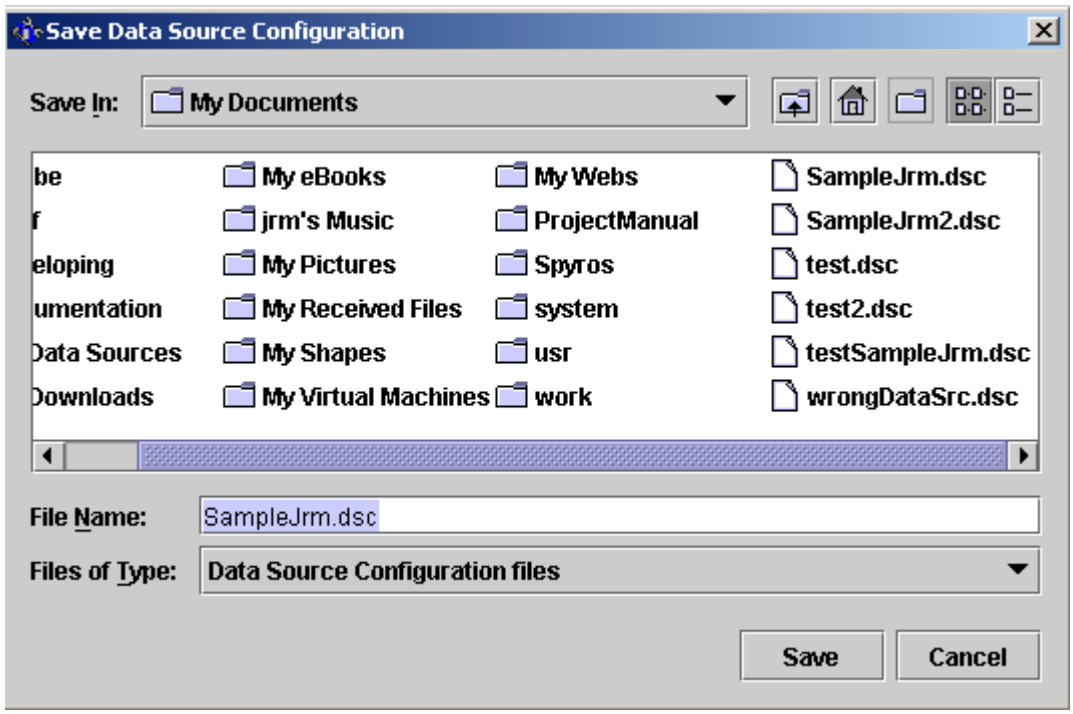


Στα οποία μπορούμε να φορτώσουμε μια νέα πηγή, να κλείσουμε ήδη φορτωμένη ή να την τροποποιήσουμε. Κάτω δεξιά φαίνεται το μήνυμα ότι η πηγή περασε επιτυχως το τεστ.

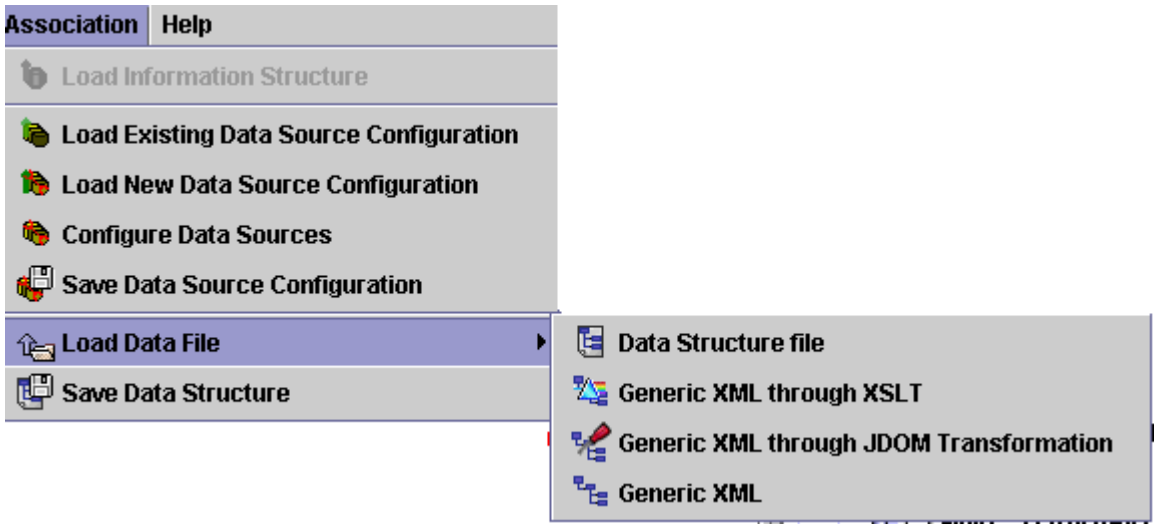
Η διαμόρφωση που δημιουργήσαμε μπορούμε να την αποθηκεύσουμε, επιλέγοντας



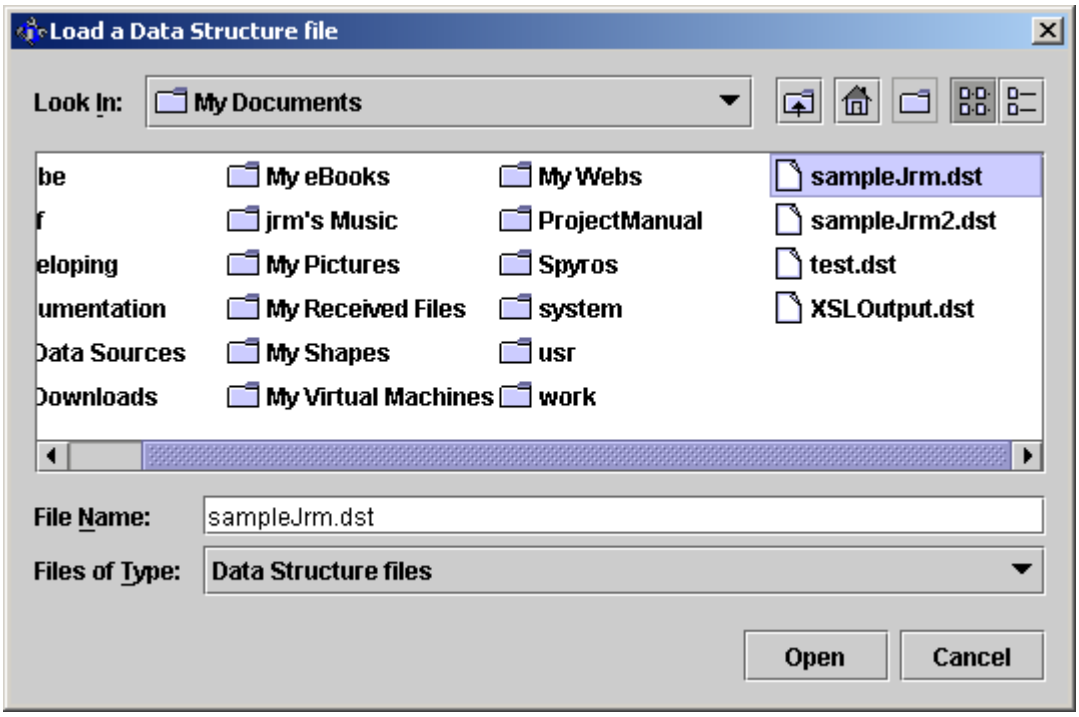
εμφανίζεται το παράθυρο για να επιλέξουμε που θα την αποθηκεύσουμε και το όνομα του αρχείου της.



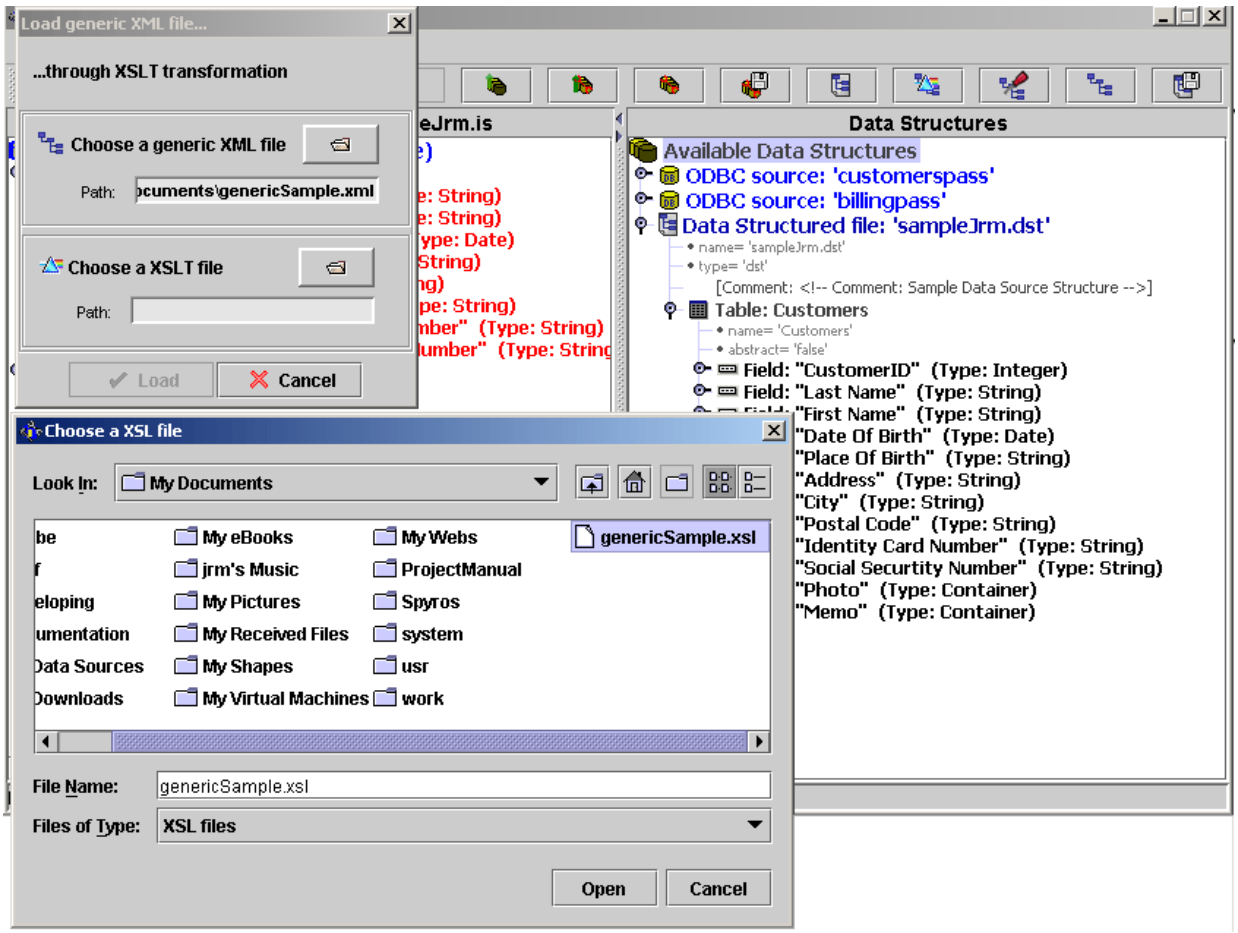
Έκτός απο πηγές δεδομένων μπορούμε να φορτώσουμε και αρχεια στην εφαρμογή. Ακολουθούν οι επιλογές που έχουμε



Επιλέγοντας Data Structure file εμφανίζεται το παρακάτω παράθυρο για να το επιλέξουμε



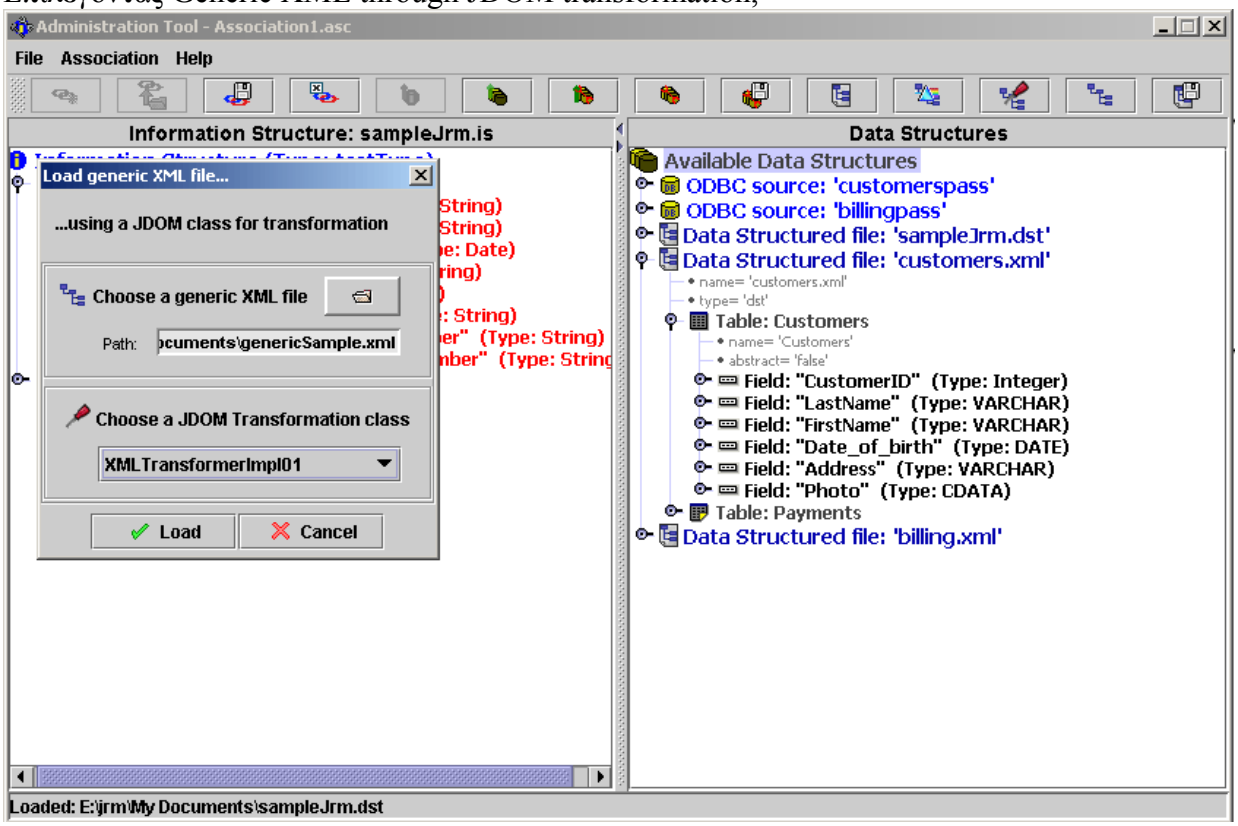
Επιλέγοντας Generic XML through XSLT,



εμφανίζεται το παράθυρο που βρίσκεται πάνω αριστερα στην εικόνα, όπου μας προτρέπει αν επιλέξουμε ένα XML και το αντιστοιχο XSL που θέλουμε να χρησιμοποιήσουμε για το μετασχηματισμό. Έστω ότι έχει ήδη επιλεγεί το XML. Επιλέγοντας το δεύτερο κουμπί 

εμφανίζεται το κάτω αριστερο παραθυρο για να επιλέξουμε και το αντίστοιχο XSL. Το path του επιλεγμένου αρχειου εμφανίζεται στο αντίστοιχο πεδίο.
Τέλος, στο κυρίως παράθυρο της εφαρμογής φαίνεται και το Data Structure file που φορτώσαμε προηγουμένως (το 3^{ο} παιδί της ρίζας Available Data Sources). Το αρχείο αυτό όπως προδίδει το ονομά του έχει την προκαθορισμένη μορφή που φαίνεται.

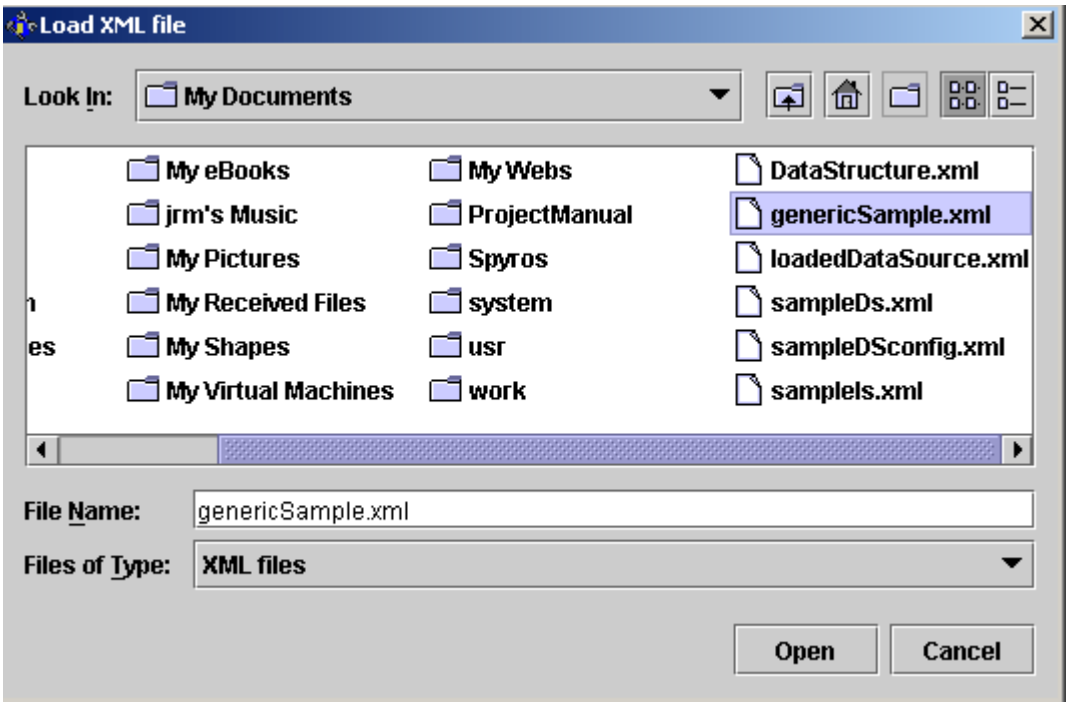
Επιλέγοντας Generic XML through JDOM transformation,



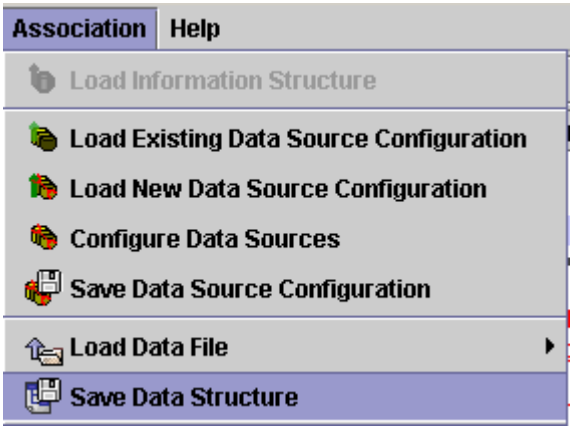
εμφανίζεται το αριστερό παράθυρο όπου με ίδιο τρόπο επιλέγουμε ένα XML αρχείο. Στην συνέχεια επιλέγουμε μια απο τις κλάσεις μετασχηματισμού που φορτώνονται με το γίνεται η «Generic XML through JDOM transformation» επιλογή απο το χρήστη. Πρέπει η κλάση που θα επιλέξουμε να μπορεί να μετασχηματίσει το αρχείο.

Παρατηρούμε πάλι του δυο τελευταίους κόμβους που προστέθηκα απο την προηγούμενη μας ενέργεια (Επιλέγοντας Generic XML through XSLT). Εδώ πρέπει να επισημάνουμε ότι όταν φορτώνουμε αρχείο μέσω μετασχηματισμού αυτό, μετασχηματίζεται στην Data Structure δομή.

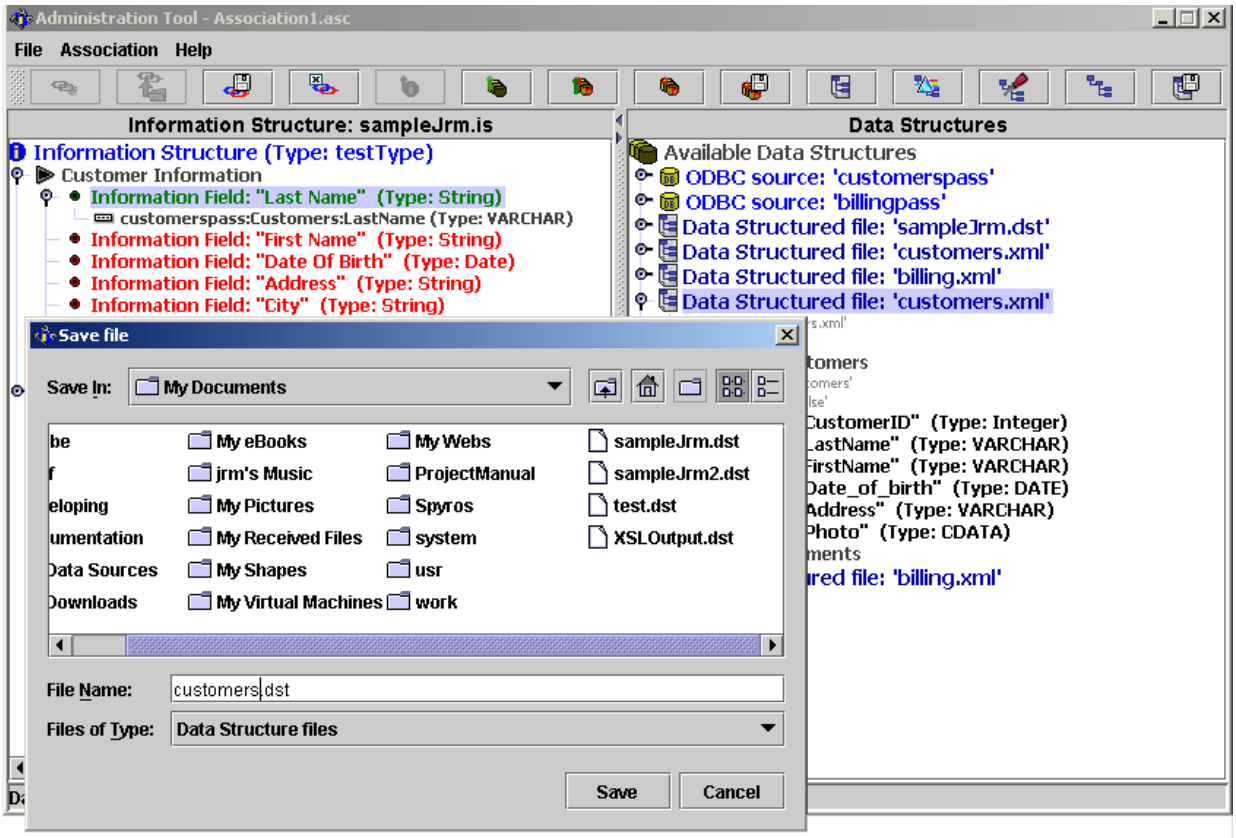
Τέλος, επιλέγοντας Generic XML, εμφανίζεται το παρακάτω παράθυρο επιλογής αρχείου,



Μπορούμε να αποθηκεύσουμε το μετασχηματισμένο αρχείο, επιλέγοντας ενα αρχείο και μετα

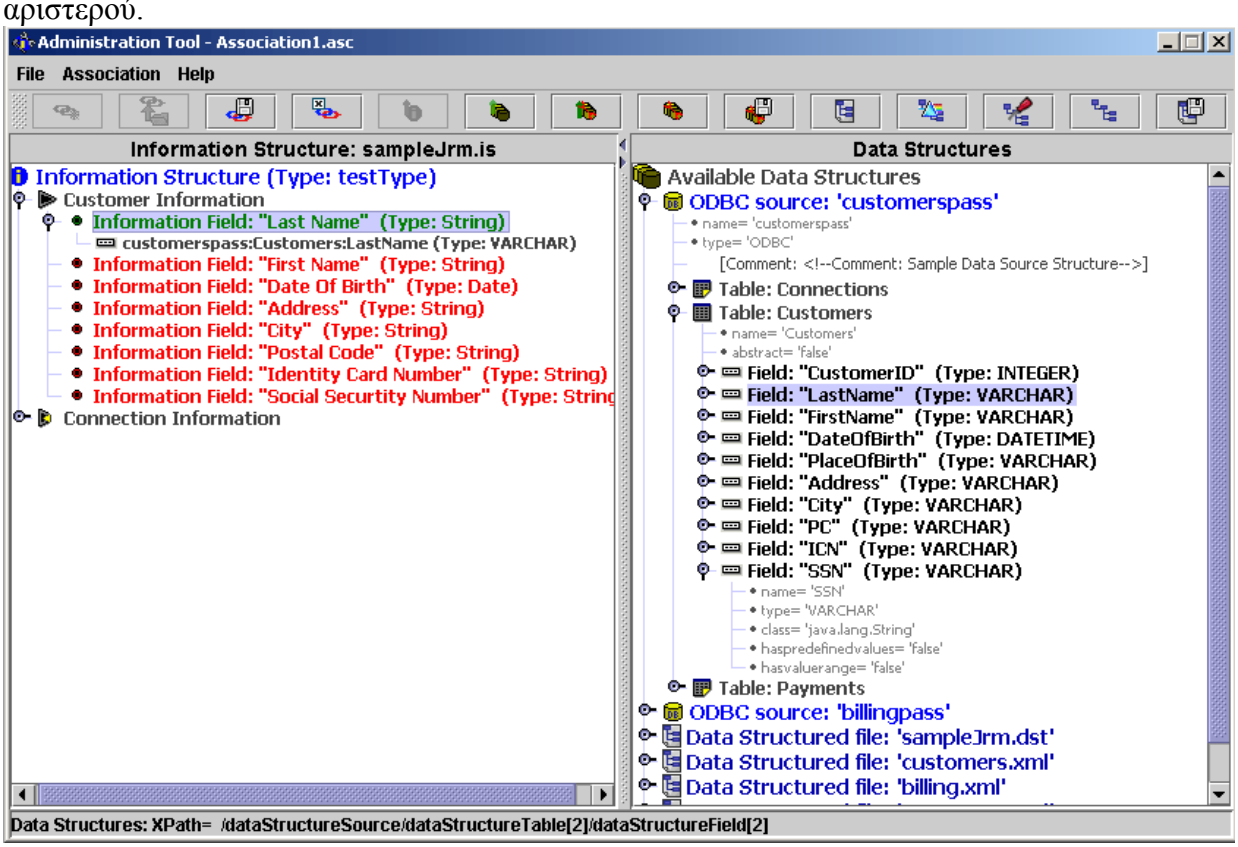


όπου εμφανίζεται το παραθυρο παράθυρο για να επιλέξουμε που θα την αποθηκεύσουμε και το όνομα του αρχείου του.

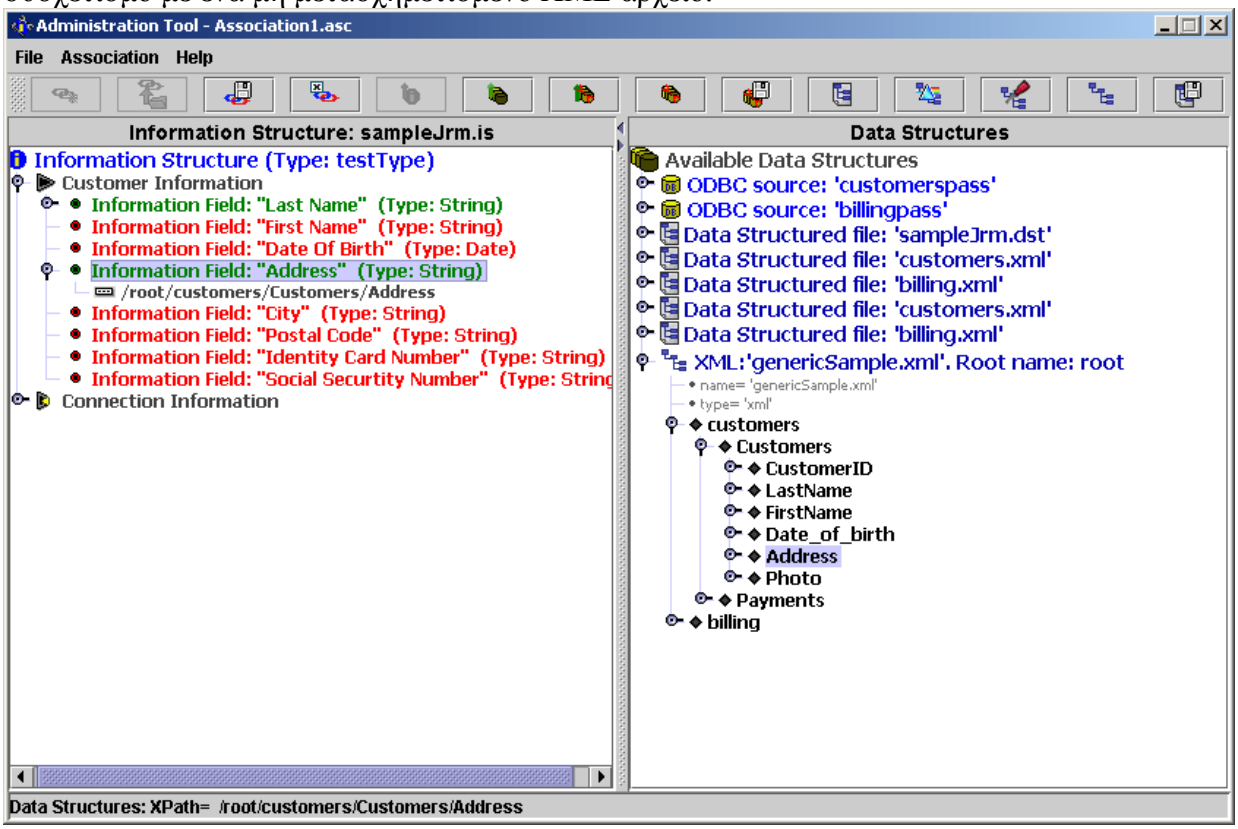


Στο κυρίως παράθυρο της εφαρμογής φαίνεται η δομή που επιλέξαμε για να σώσουμε.

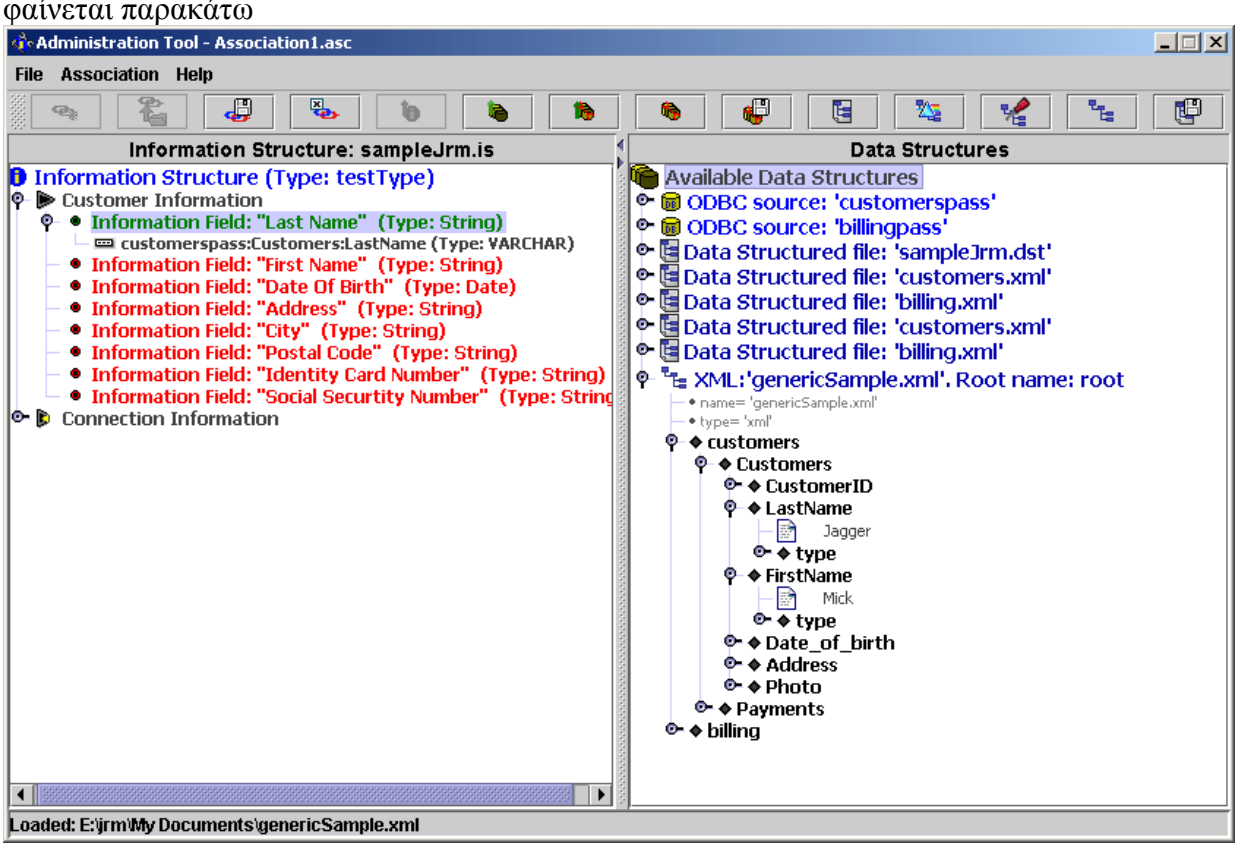
Αφου φορτώσαμε τα αρχεία που θέλουμε μπορούμε με ένα απλό Drag and Drop να συσχετίσουμε τις πηγές και τα αρχεία του δεξιού μέρους της εφαρμογής με το αρχείο του αριστερού.



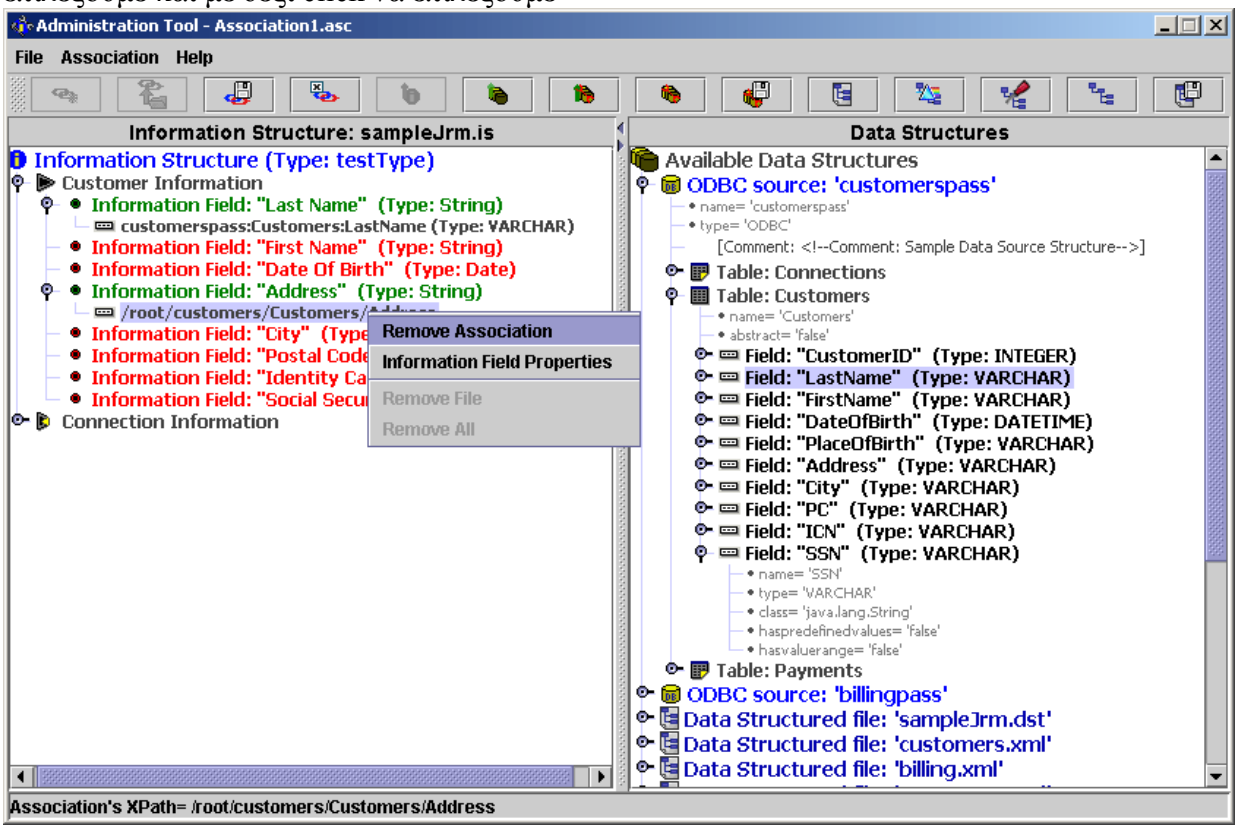
Στο προηγούμενο παράθυρο φαίνεται ο συσχετισμός που δημιουργήθηκε απο το DnD του επιλεγμένου δεξιου κόμβου και του κόμβου που δημιουργήθηκε ο συσχετισμός. Ο παραπάνω συσχετιμός ήταν συσχετισμός με μία πηγή δεδομένων. Το ακόλουθο παράδειγμα δείχνει ένα συσχετισμό με ένα μη μετασχηματισμένο XML αρχείο.



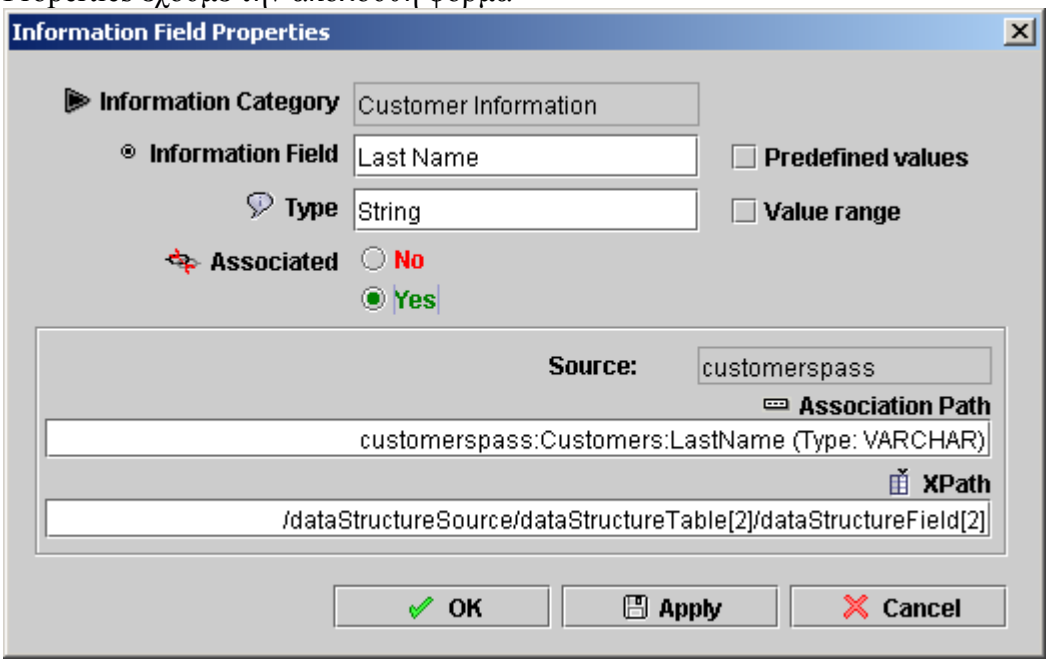
Εδώ να επισημάνουμε ότι το μη μετασχηματισμένο αρχειο διατηρεί την δική του δομή όπως φαίνεται παρακάτω



Αν θελουμε να σβήσουμε ένα συσχετισμό που δημιουργήσαμε, δεν έχουμε απο το να τον επιλέξουμε και με δεξί click να επιλέξουμε

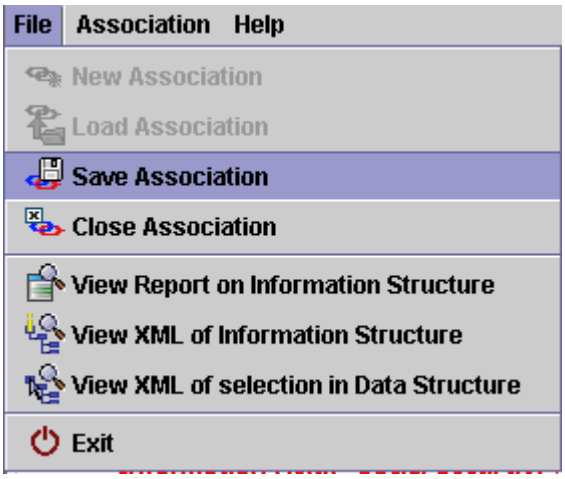


Επίσης αν επιλέξουμε ένα συσχετισμένο Information field, με δεξί click και Information Field Properties έχουμε την ακόλουθη φόρμα

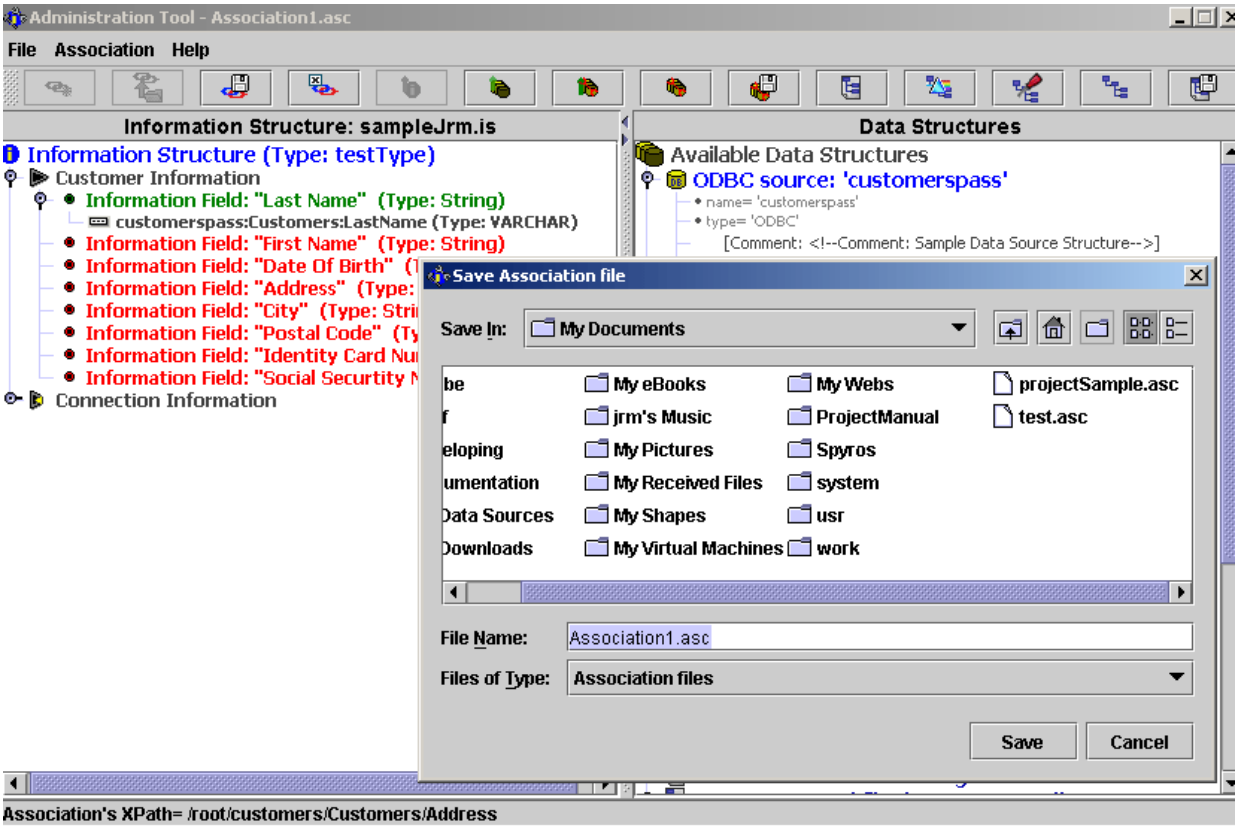


όπου εμφανίζονται όλες οι ιδιότητες του πεδίου. Μπορούμε να κάνουμε οποιεσδήποτε αλλαγές σε αυτές τις ιδιότητες.

Τέλος, για να αποθηκεύσουμε το αρχείο συσχετισμού επιλέγουμε,

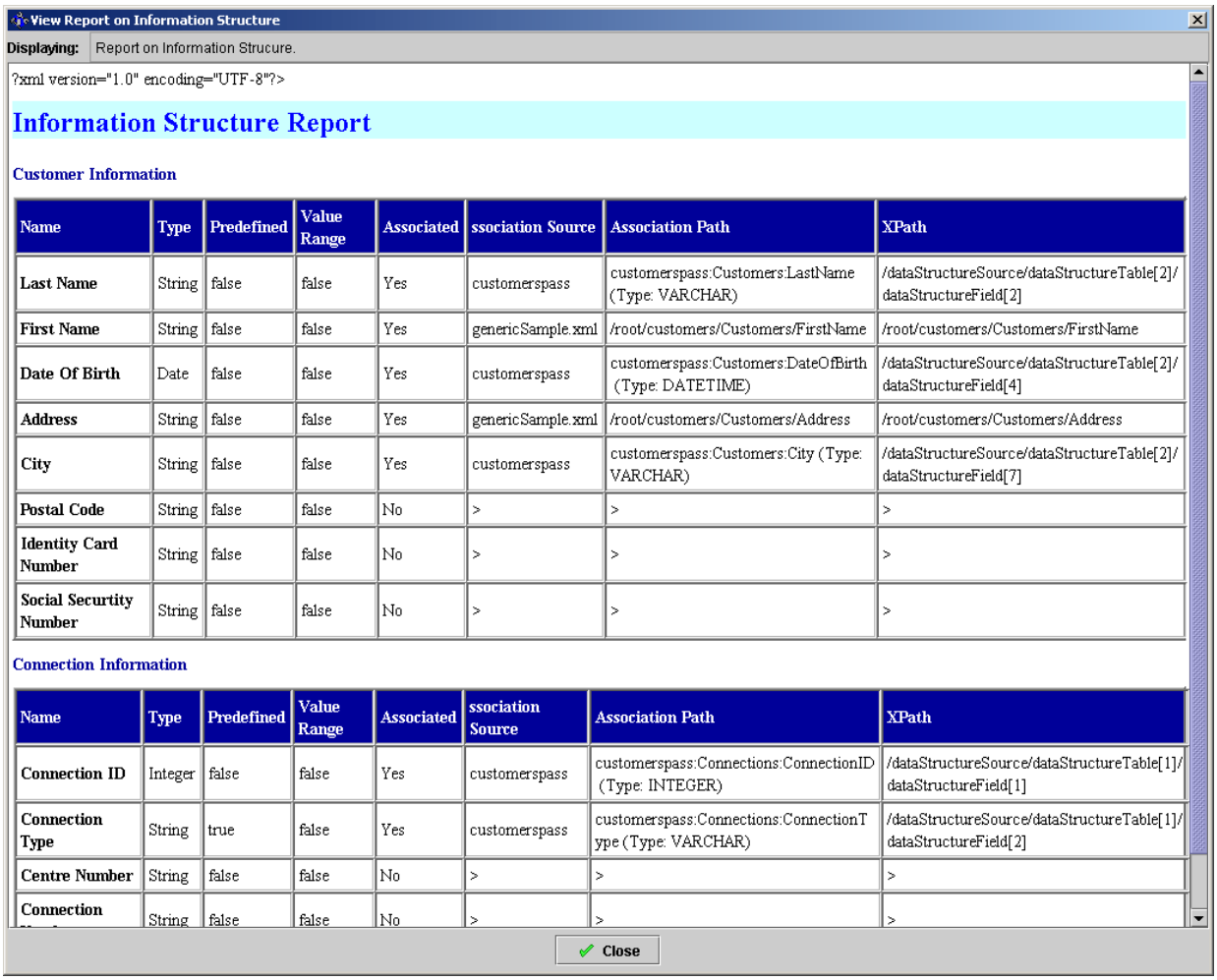


και εμφανίζεται το παρακατω παράθυρο

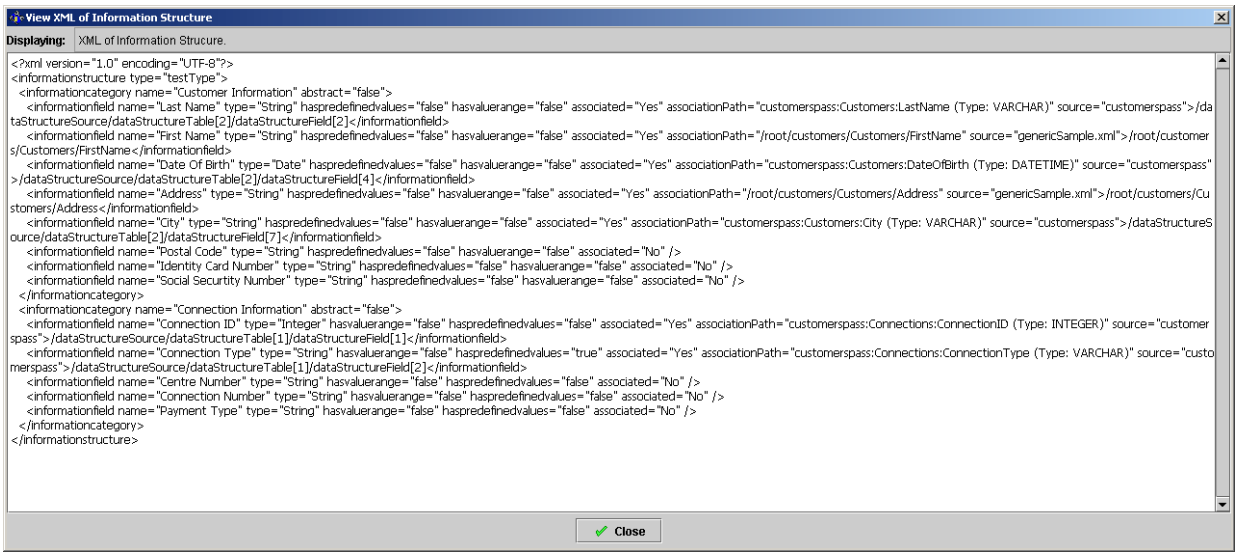


Παρατηρούμε ότι το όνομα που μας προτρέπει η εφαρμογή είναι αυτό που είχαμε δώσει όταν δημιουργήσαμε τον αρχείο συσχετιμού (New Association).

Τέλος, επιλέγοντας View Report of Information Structure, εμφανίζεται μια αναφορα του αρχείο συσχετισμού



Αν επιλέξουμε View XML of Information Structure, μπορούμε να δόμη την δομή του XML που αντιστοιχεί στην Information Structure και που διαχειρίζεται εσωτερικα η εφαρμογή.



Επίσης μπορούμε να επιλέξουμε ένα κόμβο της Data Structure και επιλέγοντας View XML of selection in Data Structure να δουμε τη XML δομή το κόμβου αυτο και όλου του υποδέντρου του.

Τέλος, επιλέγοντας About εχουμε καποιες πληροφορίες για την εφαρμογή.

Όπου φαίνεται και το email του προγραμματιστή το οποίο θα κυνηγήσουμε για τυχόν ατέλειες που σίγουρα θα υπαρχουν σε μια τόσο μεγάλη εφαρμογή.

Τέλος να επισημανουμε ότι η εφαρμογής εμφανίζει πολλά παράθυρα μυνημάτων όταν ο χρήστης κάνει λανθασμένη επιλογή, τα οποία είναι δύσκολο να παρουσιαστούν εδώ.

Αναφορές

Βιβλιογραφικές

- Java™ How to Program Fifth Edition, DEITEL
- Advanced Java™ 2 Platform How to Program. DEITEL
- Java & XML, 2nd Edition, By Brett McLaughlin, O'Reilly

Διαδικτιακές

JAVA:

- API -- Java 2 Platform SE v1.4.2 - [http://java.sun.com/j2se/1.4.2/docs/api/index.html]
- Applets - [http://java.sun.com/applets/]
- Deitel & Associates, Inc. - [http://www.deitel.com/books/downloads.html]
- Developer Resources - Code Samples - [http://developers.sun.com/resources/codesamples.html]
- Developer Resources - Downloads - [http://developers.sun.com/resources/downloads.html] > Early Access - [http://java.sun.com/downloads/ea/index.html]
- Focus on Java - [http://java.about.com/]
- Getting Started - [http://java.sun.com/docs/books/tutorial/getStarted/index.html]
- JARS.COM The #1 Java Review Service - [http://www.jars.com/]
- Java 2D API Home Page - [http://java.sun.com/products/java-media/2D/index.html]
- Java Forums - Developer Forums - [http://forum.java.sun.com/index.jspa]
- Java Foundation Classes (JFC-Swing) - [http://java.sun.com/products/jfc/]
- Java HotSpot Technology - [http://java.sun.com/products/hotspot/]
- Java Reference Guide - [http://www.informit.com/guides/guide.asp?g=java]
- Java SE Platform Documentation - [http://java.sun.com/docs/]
- java.com The marketplace for Java technology - [http://www.java.com/en/index.jsp]
- java.net - The Source for Java Technology Collaboration - [http://www.java.net/]
- JavaWorld's Toolbox - [http://www.javaworld.com/columns/jw-toolbox-index.shtml]
- JavaWorld.com - [http://www.javaworld.com/]
- New to Java Programming Center - [http://developer.java.sun.com/developer/onlineTraining/new2java/]
- ObjectWeb - Home Page - [http://www.objectweb.org/]
- Overview (Java 2 Platform SE v1.4.1) - API Specification - [http://java.sun.com/j2se/1.4.1/docs/api/index.html]

- Patterns for Java and Distributed Computing - [http://www.cmcrossroads.com/bradapp/javapats.html]
- Sun Developer Network Help - [http://servlet.java.sun.com/help]
- Sun Microsystems - Developer Home - [http://developers.sun.com/]
- The Java Boutique The Ultimate Java Applet Resource Tutorials - [http://javaboutique.internet.com/tutorials/]
- The Java Community Process(SM) Program - [http://www.jcp.org/en/home/index]
- The Java Language Specification - [http://java.sun.com/docs/books/jls/second_edition/html/j.title.doc.html]
- The Java Tutorial - [http://java.sun.com/docs/books/tutorial/index.html]
- Threads and Multithreading - [http://java.sun.com/developer/technicalArticles/Threads/#top]
- UML - [http://www.uml.org/]
- Unicode Home Page - [http://www.unicode.org/]
- W3Schools Online Web Tutorials - [http://www.w3schools.com/]
- Welcome to Java How to Program 5-e - [file:///G:/My%20Documents/JAVA/Deitel%26Deitel%20examples/welcome.htm]

CLASSPATH:

- Classpath Tutorial - [http://www.dynamic-apps.com/tutorials/classpath.jsp]
- CS 235 Classpath Tutorial - Fall 2003 - [http://students.cs.byu.edu/~cs235ta/summer2004/help/classpath.html]
- Learn Java Learn how to download and decompile Java classes, part 2 - [http://www.devdaily.com/java/edu/pj/pj010021/]
- Using the SAX Parser - [http://www.javacommerce.com/tutorial/xmldev/SaxParser1.htm]

JAVArious:

Design Patterns:

- Design Patterns - [http://umbc7.umbc.edu/~tarr/dp/dp.html]
- Hillside.net - Your Patterns Library - [http://hillside.net/patterns/]
- Net Objectives Design Patterns - [http://www.netobjectives.com/design.htm]
- Patterns for Java and Distributed Computing - [http://www.cmcrossroads.com/bradapp/javapats.html]
- Portland Pattern Repository - [http://c2.com/ppr/]

Architectural Patterns:

- (ootips) Model-View-Controller - [http://www.ootips.org/mvc-pattern.html]
- Architectural Patterns, Design Patterns, - [http://www.tml.hut.fi/Opinnot/Tik-109.450/1998/niska/sld001.htm]
- MVC meets Swing - [http://www.javaworld.com/javaworld/jw-04-1998/jw-04-howto.html]

DnD:

- Drag and Drop with Swing - [http://java.sun.com/products/jfc/tsc/articles/dragndrop/index.html]
- How to drag and drop with Java 2, Part 1 - [http://www.javaworld.com/javaworld/jw-03-1999/jw-03-dragndrop.html]
- How to drag and drop with Java 2, Part 2 - [http://www.javaworld.com/jw-08-1999/jw-08-draganddrop-p4.html]
- J2EE Training Course - [http://www.rockhoppertech.com/java-drag-and-drop-faq.html]
- Java - JFC - Drag and Drop - [http://www.objectsdevelopment.com/portal/modules/freecontent/content/javadragdrop.html]
- Java 1.2 Unleashed -- Ch 16 -- Working with Drag and Drop - [http://utenti.lycos.it/yanorel6/2/ch16.htm]
- Java Forums !!! - An example of Drag and Drop in JTree - [http://forum.java.sun.com/thread.jspa?forumID=57&threadID=296255]
- Java Forums - Drag & Drop multiple node in JTree - [http://forum.java.sun.com/thread.jspa?forumID=57&threadID=546005]
- Java Forums - Drag & Drop- Multiple Selection - [http://forum.java.sun.com/thread.jspa?forumID=57&threadID=122261]
- Java Forums - JTree Drag N Drop - [http://forum.java.sun.com/thread.jspa?forumID=57&threadID=414578]
- Java Reference Guide - [http://www.informit.com/guides/content.asp?g=java&seqNum=54]
- Java Tip 97 Add drag and drop to your JTrees - [http://www.javaworld.com/javaworld/javatips/jw-javatip97.html]
- New Data Transfer Capabilities - Nov 2001 - [http://java.sun.com/developer/technicalArticles/releases/data/]
- Swing Data Transfer - Sun Microsystems - [http://access1.sun.com/tutorials/Swing_Tutorial/Dnd-Merlin-Tutorial/index.html]
- Unfurling Java's data transfer API - [http://www.javaworld.com/javaworld/jw-05-1998/jw-05-howto.html]

GUI:

- community.java.net - JavaDesktop - [http://community.java.net/javadesktop/]
- Creating a GUI with JFC-Swing ---Tutorial - [http://java.sun.com/docs/books/tutorial/uiswing/index.html]
- Drag and Drop - [file:///E:/jrm/JAVA/Downloaded/J2SE_SDK-1_4_2-Docs/docs/guide/dragndrop/index.html]
- Java Foundation Classes (JFC-Swing) - [http://java.sun.com/products/jfc/]
- Learning Swing by Example - [http://java.sun.com/docs/books/tutorial/uiswing/learn/index.html]
- The Swing Connection - [http://java.sun.com/products/jfc/tsc/index.html]

- Java Programming, Main Index - [http://math.hws.edu/javanotes/]
- java.net - The Source for Java Technology Collaboration - [http://www.java.net/]

javadoc:

- How to Write Doc Comments for the Javadoc Tool - [http://java.sun.com/j2se/javadoc/writingdoccomments/index.html]
- Javadoc Tool Home Page - [http://java.sun.com/j2se/javadoc/]

JDBC:

- Cloudscape and ODBC - [http://www-106.ibm.com/developerworks/db2/library/techarticle/dm-0409cline2/index.html]
- Frequently Asked Questions about JDBC - [http://java.sun.com/products/jdbc/faq.html]
- JDBC Drivers - [http://servlet.java.sun.com/products/jdbc/drivers]
- JDBC Technology - [http://java.sun.com/products/jdbc/index.jsp]
- jGuru JDBC 2.0 Fundamentals - [http://java.sun.com/developer/onlineTraining/Database/JDBC20Intro/JDBC20.html]
- jGuru JDBC FAQ Home Page - [http://www.jguru.com/faq/JDBC]
- The JDBC Tutorial Chapter 3 - Advanced Tutorial - [http://java.sun.com/developer/Books/JDBCTutorial/]

JDBC Drivers:

- JDBC Drivers - [http://servlet.java.sun.com/products/jdbc/drivers]
- Types of JDBC technology drivers - [http://java.sun.com/products/jdbc/driverdesc.html]

JDBC-ODBC:

- Download details MDAC 2.8 - [http://www.microsoft.com/downloads/details.aspx?FamilyID=6c050fe3-c795-4b7d-b037-185d0506396c&DisplayLang=en]
- Duke's Bakery - A JDBC Order Entry Prototype - Part I - [http://java.sun.com/developer/technicalArticles/Database/dukesbakery/]
- Easysoft JDBC-ODBC Bridge - [http://www.easysoft.com/products/job/]
- http--java.sun.com-j2se-1.3-docs-guide-jdbc-getstart-bridge.doc.html - [http://java.sun.com/j2se/1.3/docs/guide/jdbc/getstart/bridge.doc.html#996747]
- JDBC Overview - JDBC-ODBC Bridge - [http://java.sun.com/products/jdbc/overview.html]
- ODBC Start Page - [http://msdn.microsoft.com/library/default.asp?url=/library/en-us/odbc/htm/dasdkodbcoverview.asp]
- ODBC Tutorial - [http://frankscaslpage.home.att.net/tips_pages/odbc_tutorial.htm]
- ODBC--Open Database Connectivity Overview - [http://support.microsoft.com/default.aspx?scid=kb;en-us;110093]
- Register the New Database with ODBC - [http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vctutor98/html/_gs_register_the_new_database_with_odbc.asp]
- Search Microsoft.com - [http://search.microsoft.com/search/search.aspx?st=b&View=en-us]
- Welcome to the MSDN Library - [ ]

Sun Developer Network:

- SDN Newsletter Archive - [http://developers.sun.com/inside/sdnnewsarchive.html]
- Sun Microsystems - Developer Home - [http://developers.sun.com/]

- Sun Microsystems - [http://access1.sun.com/]

JBuilder X:

- Borland JBuilder The leading development solution for Java - [http://www.borland.com/jbuilder/]
- Borland JBuilder X Product Documentation - [http://info.borland.com/techpubs/jbuilder/indexx.html]
- JBuilder additional downloads - [http://info.borland.com/downloads/jbuilder/addons.html]

XML:

- W3 Schools [www.w3schools.com]
- Cooktop The free XML editor for Windows - [http://www.xmlcooktop.com/]
- xml examples - [http://www.ibiblio.org/xml/examples/]
- XML java.sun.com-xml - [http://java.sun.com/xml/]
- XML Pitstop Largest Source of XML Examples on the Web - [http://www.xmlpitstop.com/]
- XML Pull Parsing - [http://www.xmlpull.org/]
- XML.com XML From the Inside Out -- XML development, XML resources, XML specifications - [http://www.xml.com/]

Data Binding:

- Data binding from XML to Java Apps - Part 1 - [http://www-106.ibm.com/developerworks/xml/library/x-data-binding1/]
- Enhydra Zeus - Home Page - [http://zeus.objectweb.org/index.html]
- From bits to brackets - [http://www-106.ibm.com/developerworks/xml/library/x-databind4/]
- From text to byte code - [http://www-106.ibm.com/developerworks/xml/library/x-data-binding3/]
- Make classes from XML data - [http://www-106.ibm.com/developerworks/xml/library/x-data-binding2/]

DOM:

- Document Object Model (DOM) Level 1 Specification - [http://www.w3.org/TR/REC-DOM-Level-1/]
- Document Object Model (DOM) Level 2 Core Specification - [http://www.w3.org/TR/DOM-Level-2-Core/]
- DOM Tutorial - [http://www.w3schools.com/dom/default.asp]
- W3C Document Object Model - [http://www.w3.org/DOM/]

dom4j:

- dom4j - dom4j the flexible XML framework for Java - [http://dom4j.org/]

- ebXML - Enabling A Global Electronic Market - [http://www.ebxml.org/]
- Extensible Markup Language (XML) 1.0 - [http://www.w3.org/TR/1998/REC-xml-19980210]

JAXP:

- Java API for XML Processing (JAXP) - [http://java.sun.com/xml/jaxp/index.jsp]

JDOM:

- Java(TM) Boutique - Working with JDOM, XPath and XSLT - [http://javaboutique.internet.com/tutorials/jdom&/]
- jaxen universal java xpath engine - jaxen - [http://jaxen.org/]
- JDOM Documentation - [http://www.jdom.org/downloads/docs.html]
- JDOM Mailing Lists - [http://www.jdom.org/involved/lists.html]
- JDOM - [http://www.jdom.org/]
- Servlets.com Archives jdom-interest - [http://www.servlets.com/archive/servlet/SearchList?listName=jdom-interest&by=thread]

- newInstance.com resources - [http://www.newinstance.com/resources.jsp]
- oreilly.com -- Online Catalog Java & XML, 2nd Edition - [http://www.oreilly.com/catalog/javaxml2/index.html]
- Processing XML with Java - [http://www.cafeconleche.org/books/xmljava/]

Documentation on JDOM:

- JDOM and XML Parsing, Part 1 - [http://www.oracle.com/technology/oramag/oracle/02-sep/o52jdom.html]
- JDOM and XML Parsing, Part 2 - [http://www.oracle.com/technology/oramag/oracle/02-nov/o62odev_jdom.html]
- JDOM in the Real World, Part 3 - [http://www.oracle.com/technology/oramag/oracle/03-mar/o23javaxml.html]
- Processing XML with Java - [http://www.cafeconleche.org/books/xmljava/]
- Simplify XML programming with JDOM - [http://www-106.ibm.com/developerworks/java/library/j-jdom/]
- Tip Converting from DOM - [http://www-106.ibm.com/developerworks/java/library/x-tipcdm.html]
- Tip Converting from SAX - [http://www-106.ibm.com/developerworks/java/library/x-tipcsx.html]
- Tip Using JDOM and XSLT - [http://www-106.ibm.com/developerworks/java/library/x-tipjdom.html]

SAX:

- SAX Filters - [http://www.saxproject.org/?selected=filters]
- SAX official website - [http://www.saxproject.org/]

XML Parsers:

- Xerces2 Java Parser Readme - [http://xml.apache.org/xerces2-j/]

XPath:

- Chapter 16. XPath - [http://www.cafeconleche.org/books/xmljava/chapters/ch16.html]
- Java(TM) Boutique - Working with JDOM, XPath and XSLT - Page 3= XPath - [http://javaboutique.internet.com/tutorials/jdom&/index-3.html]
- SourceForge.net Project Info - XPath Explorer - [http://sourceforge.net/projects/xpe/]
- Xalan-Java version 2.6.0 - [http://xml.apache.org/xalan-j/]
- XML Path Language (XPath) - [http://www.w3.org/TR/xpath]
- XPath Tutorial - [http://www.w3schools.com/xpath/default.asp]

XSL:

- Creating CDATA sections with XSLT - [http://www.azureus.com/xml/art_creating_cdata_sections_in_xslt.htm]
- Extensible Stylesheet Language (XSL) - [http://www.w3.org/TR/xsl/]
- XSLT Tutorial - [http://www.w3schools.com/xsl/default.asp]

Παράρτημα 1: Κώδικας

Package: admintool:

admintool.AbsractChangeListenerImpl.java

```
package admintool;

import javax.swing.event.*;
/**
* <p>
 * Class that provides a dummy implementation of the DocumentListener and ChangeListener
Interfaces.
 * Document Listener is for listening changes on JTestFields and Change Listener is for
listening
 * changes on JCheckBoxex and JRadioButtons.
* </p>
 *<p>
 * This class is only used in InformationFieldProperties class for listening changes of its
 * GUI components, which is also the reason that this class was written.
InformationFieldProperties
*  had GUI conmponents that needed a ChangeListener or a DocumentListener. So, in order to avoid
to
* put so much code in a private helper class of InformationFieldProperties, the interfaces
* are dummy implemented here, resulting in a abstract class which needs only one method to
* be implement in order to work.
 * </p>
 * @see InformationFieldProperties
 */
public abstract class AbsractChangeListenerImpl implements DocumentListener, ChangeListener  {
  /**Default Constructor.*/
  public AbsractChangeListenerImpl() {
  }

  /**
   * Calls <code>changesOccured</code> helper method.
   * @see changesOccured()
   *
   * @param e ChangeEvent
   */
  public void stateChanged(ChangeEvent e) {
    changesOccured();
  }

  /**
   * Calls <code>changesOccured</code> helper method.
   * @see changesOccured()
   *
   * @param e DocumentEvent
   */
  public void changedUpdate(DocumentEvent e) {
        changesOccured();
  }

  /**
   * Calls <code>changesOccured</code> helper method.
  * @see changesOccured()
   *
   * @param e DocumentEvent
   */
  public void insertUpdate(DocumentEvent e) {
        changesOccured();
  }

  /**
   * Calls <code>changesOccured</code> helper method.
```

```
  * @see changesOccured()
   *
   * @param e DocumentEvent
   */
  public void removeUpdate(DocumentEvent e) {
        changesOccured();
  }

  /**
   * The actions that need to be made when a change is made.
   * Implement this method according to the actions need to be made when the data
   * kept on the GUI components that have this class as a Change/Document listener
   * are modified.
   *
   *
   */
  abstract void changesOccured() ;

}
```

admintool.AdminTool.java

```
package admintool;

import javax.swing.UIManager;
import java.awt.*;

/**
 * <p>Administration Tool</p>
 * <p>Description: Main class. It only creates an AdminToolFrame by calling its
*  constructor and displays it. The all of the application's functionallity is
*  implemented in AdminToolFrame </p>
*
* @see AdminToolFrame
 */

public class AdminTool {
   /** Set it true to pack the frame or false to validate it.   */
   boolean packFrame = false;


  /**
   * Construct the application. Generates the application frame and displayes it.
   * The frame is an AdminToolFrame. It Validate frames that have preset sizes,
   * Pack frames that have useful preferred size info, e.g. from their layout
   * and centers the window
   */
  public AdminTool() {
    AdminToolFrame frame = new AdminToolFrame();
    //Validate frames that have preset sizes
    //Pack frames that have useful preferred size info, e.g. from their layout
    if (packFrame) {
      frame.pack();
    }
    else {
      frame.validate();
    }
    //Center the window
    Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
    Dimension frameSize = frame.getSize();
    if (frameSize.height > screenSize.height) {
      frameSize.height = screenSize.height;
    }
    if (frameSize.width > screenSize.width) {
      frameSize.width = screenSize.width;
    }
    frame.setLocation((screenSize.width - frameSize.width) / 2, (screenSize.height -
frameSize.height) / 2);
    frame.setVisible(true);
  }

  //
```

```
  /**
   * Main method. Sets the LookandFeel and calls the constructor.s
   *
   * @param args String[] not used
   */
  public static void main(String[] args) {
    try {
      UIManager.setLookAndFeel("javax.swing.plaf.metal.MetalLookAndFeel");
      //UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());


    }
    catch(Exception e) {
      e.printStackTrace();
    }
    new AdminTool();
  }
}
```

admintool.AdminToolFrame.java

```
package admintool;

import java.io.*;
import java.util.*;
import java.util.List;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
import javax.swing.tree.*;

import org.jdom.*;
import org.jdom.input.*;
import org.jdom.output.*;
import admintool.jdomtreemodel.*;

/**
 * <p>Title: AdministrationFrame</p>
 * <p>Description: Class the everything else is link into. This is actually
 *    the main application class, because all the functionality is implemented
 *     on this class or in other classes connected to it. AdminTool class is only
 *     a caller of this class.
 * </p>
 * <p>
 * The application has three Documents:
 * <ul>
 * <li>the document that holds the Information Structure that is loaded.</li>
 * <li>the document that holds the Data Structures which are loaded from Data Sources
 * or from files. </li>
 * <li>the document that holds which data source configurations that are loaded.</li>
 * </ul>
 * Each Data structure loaded from a Data Source has a corresponding Data Source configuration.
 * </p>
 * <p>
 * <i>
 * Quick Reference:
 * The word <b>document</b> is refered to a JDOM Document.
 *  <br>An XML which is loaded with
 * JDOM is also refered as a "document". XML is a file in the disk not the data
 * structure created by JDOM when a XML file is loaded.
 * </i>
 * </p>
 *
 * @see AdminTool
 */
public class AdminToolFrame
    extends JFrame
    implements DataModifier {
  /** It holds the passwords*/
  private Map passwordMap;
```

```
  /**The GUIUpdater.   */
  GUIUpdater guiUpdater = null;
  /**The file that the association is been saved. <br>
   * It is also been used as a flag. When it is null, then the association is not yet
   * yet been saved. The next time time that is going to be saved, it will be the first time.
   * @see jMenuItemFileSaveAssoc_actionPerformed(ActionEvent)
   * */
  File savefile = null;
  /**The file for loading an informatin Structure*/
  File infoStructFile = null;
  /**The name of the association. <br>
   * It holds the name of the association provided
   * when New Association is choosen, so as to use the given name on save.<br>
   * It is also been used as a flag. When it is null, then a New Association is not yet created
or loaded.
   * */
  String associationName = null;

  //JDOM
  /**Document that holds the information structure loaded*/
  Document infoStructDoc = null;
  /**Document that holds the data structures loaded*/
  Document dataStructDoc = null;
  /**Document that holds the data sources configurations (*.dsc) that are loaded.<br>
   * Data source configuration are loaded as data structures in the data
   * structure tree*/
  Document dataSrcConfig = null;
  /**The root of the dataStructDoc*/
  Element dataStructRoot = null;
  /** The root of the infoStructDoc*/
  Element infoStructRoot = null;
  //FileFilters
  /**File filter for information structures files *.is used in JFileChooser*/
  FilesFilter isFileFilter = new FilesFilter("is");
  /**File filter for data source configurations files *.dsc used in JFileChooser */
  FilesFilter dscFileFilter = new FilesFilter("dsc");
  /**File filter for data structures files *.dst used in JFileChooser*/
  FilesFilter dstFileFilter = new FilesFilter("dst");
  /**File filter for xml files *.xml used in JFileChooser*/
  FilesFilter xmlFileFilter = new FilesFilter("xml"); //data struct
  /**File filter for associations *.asc used in JFileChooser*/
  FilesFilter ascFileFilter = new FilesFilter("asc"); //data struct
  /**For the evaluation of the XPaths*/
  XPathEvaluator xPathEvaluator = new XPathEvaluator();
  /**The file chooser used. <br>It was consider better to use one filechooser, which
   * is set, used and reset, that to user more than one. <i>Setting</i> a
   * filechooser refers to setting title, file filter, selected file etc. */
  JFileChooser jFileChooser1 = new JFileChooser();

  /**Main menu bar*/
  JMenuBar jMenuBar1 = new JMenuBar();
  //menus
  /**File menu*/
  JMenu jMenuFile = new JMenu();
  /**Association menu*/
  JMenu jMenuAssoc = new JMenu();
  /**Help menu.*/
  JMenu jMenuHelp = new JMenu();
  //menu items
  /**File | New Association menu item.*/
  JMenuItem jMenuItemFileNewAssoc = new JMenuItem();
  /**File | Load Association menu item.*/
  JMenuItem jMenuItemFileLoadAssoc = new JMenuItem();
  /**File | Save Association menu item.*/
  JMenuItem jMenuItemFileSaveAssoc = new JMenuItem();
  /**File | Close Association menu item.*/
  JMenuItem jMenuItemFileCloseAssoc = new JMenuItem();
  /** File | View Report on Information Structure menu item.*/
  JMenuItem jMenuItemReport = new JMenuItem();
  /** File | View XML of Information Structure menu item.*/
  JMenuItem jMenuItemInfoStructXML = new JMenuItem();
  /** File | View XML of selection of Structure menu item.*/
  JMenuItem jMenuItemSelectedXML = new JMenuItem();
  /**File | Exit menu item.*/
  JMenuItem jMenuFileExit = new JMenuItem();
  /**Association | Load Information Structure menu item.*/
```

```
JMenuItem jMenuItemLoadInfoStruct = new JMenuItem();
/**Association | Load Existing Data Source Configuration menu item.*/
JMenuItem jMenuItemLoadExistDataSrcConfig = new JMenuItem();
/**Association | Configure Data Sources menu item.*/
JMenuItem jMenuItemConfigDataSrc = new JMenuItem();
/**Association | Save Data Source Configuration menu item.*/
JMenuItem jMenuItemSaveDataSrcConfig = new JMenuItem();
/**Association |  LOad new Data Source Configuration menu item.*/
JMenuItem jMenuItemLoadNewDataSrcConfig = new JMenuItem();
/**Association | Save Data Structure menu item.*/
JMenuItem jMenuItemSaveDataStruct = new JMenuItem();
/**Association | Load Data File menu.*/
JMenu jMenuLoadDataFile = new JMenu();
/**Association | Load Data File | Generic XML through XSLT menu item.*/
JMenuItem jMenuItemLoadXML_XSLT = new JMenuItem();
/**Association | Load Data File | Generic XML through JDOM Transformation menu item.*/
JMenuItem jMenuItemLoadXML_jdomTransform = new JMenuItem();
/**Association | Load Data File | Data Structure File menu item.*/
JMenuItem jMenuItemLoadDataStruct = new JMenuItem();
/**Association | Load Data File | Generic XML menu item.*/
JMenuItem jMenuItemLoadXml = new JMenuItem();
/**Help | About menu item.*/
JMenuItem jMenuHelpAbout = new JMenuItem();

/**Pop up menu.*/
JPopupMenu jPopupMenu1 = new JPopupMenu();
/**Pop up | Information Field Prperties menu item*/
JMenuItem jMenuItemInfoFieldProps = new JMenuItem();
/**Pop up | Remove Association menu item*/
JMenuItem jMenuItemRemoveAssoc = new JMenuItem();
/**Pop up | Remove All menu item*/
JMenuItem jMenuItemRemoveAll = new JMenuItem();
/**Pop up | Remove File menu item*/
JMenuItem jMenuItemRemoveFile = new JMenuItem();

/**The Toolbar of the frame.*/
JToolBar jToolBar = new JToolBar();
/**Toolbar | New Association button*/
JButton jButtonNewAssoc = new JButton();
/**Toolbar | Load Association button*/
JButton jButtonLoadAssoc = new JButton();
/**Toolbar | Save Association button*/
JButton jButtonSaveAssoc = new JButton();
/**Toolbar | Close Association button*/
JButton jButtonCloseAssoc = new JButton();
/**Toolbar | Load Informaiton Structure button*/
JButton jButtonLoadIS = new JButton();
/**Toolbar | Save Data Source Configuration button*/
JButton jButtonSaveDSC = new JButton();
/**Toolbar | Load Data Structured File button*/
JButton jButtonLoadDST = new JButton();
/**Toolbar | Generic XML button*/
JButton jButtonLoadXML = new JButton();
/**Toolbar | Generic XML throough XSLT button*/
JButton jButtonLoadXMLXSLT = new JButton();
/**Toolbar | Generic XML through JDOM Transformation button*/
JButton jButtonLoadXMLJDOM = new JButton();
/**Toolbar | Save Data Structured File button*/
JButton jButtonSaveDST = new JButton();
/**Toolbar | Load Existing Data Source Configuration button*/
JButton jButtonLoadExistDSC = new JButton();
/**Toolbar | Load New Data Source Configuration button*/
JButton jButtonLoadNewDSC = new JButton();
/**Toolbar | Configure Data Sources button*/
JButton jButtonConfigDSC = new JButton();

//layouts and the rest of the components
JPanel contentPane;
BorderLayout borderLayout1 = new BorderLayout();
JPanel mainPanel = new JPanel();
BorderLayout borderLayout2 = new BorderLayout();
JSplitPane jSplitPane = new JSplitPane();
JScrollPane jRightScrollPane = new JScrollPane();
JScrollPane jLefttScrollPane = new JScrollPane();
/**Label of Data Structure tree*/
JLabel jLabelDataSrc = new JLabel();
/**Label of Information Structure tree*/
```

```
JLabel jLabelInfoStruct = new JLabel();
JPanel ToolBarPanel = new JPanel();
GridLayout gridLayout2 = new GridLayout();
JPanel LeftPanel = new JPanel();
JPanel RightPanel = new JPanel();
BorderLayout borderLayout4 = new BorderLayout();
BorderLayout borderLayout5 = new BorderLayout();
JTextField statusBar = new JTextField();

//JTrees and relative stuff
/**Data Structure tree */
JTree jTree1 = new JTree();
/**Information Structure tree*/
JTree jTree2 = new JTree();
// TreeModel jrm
/**Tree model of the Data Structure JTree.*/
JDOMTreeModel treeModel1 = null;
/**Selection model of the Data Sstructure JTree.*/
DefaultTreeSelectionModel selectionModel1;
/**Tree Selection Handler of the Data Sstructure JTree.*/
TreeSelectionHandler treeSelectionHandler = new TreeSelectionHandler();
/**Tree Cell Renderer of the Data Sstructure JTree.*/
DataSrcTreeCellRenderer dataSrcTreeCellRenderer = new DataSrcTreeCellRenderer();
// Jtree 2 = the left tree
/**Tree model of the Information Structure tree.*/
JDOMTreeModel treeModel2 = null;
/**Selection model of the Information Structure JTree.*/
DefaultTreeSelectionModel selectionModel2;
/**Tree Cell Renderer  of the Information Structure JTree.*/
InfoStructTreeCellRenderer infoStructTreeCellRenderer = new
    InfoStructTreeCellRenderer();
/**TransferHandler that provides support for Drag and Drop functionality between the Jtrees*/
TreeTransferHandler treeTransferHandler = new TreeTransferHandler();
/**Tree model listener implementation for listening for tree model changes*/
TreeModelAdapter treeModelAdapter = new TreeModelAdapter();

//Images and Image Icons
/**Frame icon.*/
private Image frameImage;

/**Icon used in corresponding button or menu item.*/
private ImageIcon newAssocIcon, loadAssocIcon, saveAssocIcon, closeAssocIcon,
    viewReportIcon, viewXmlInfoStruct, viewXmlSelection,
    exitIcon,
    loadInfoStructIcon, loadExistDSCIcon, loadNewDSCIcon, configDSCIcon,
    saveDSCIcon,
    loadDataFileIcon, jdomXmlTransformIcon, xsltXmlIcon, dstIcon, xmlIcon,
    saveDSTIcon,
    aboutIcon;

/**
 * Construct the frame.
 * It initialize variables, loads  the images for the gui, calls jbInit to set up the gui and
 * initializes the guiupdater.
 * If the image files are not found an exception will be thrown and the
 * application will not start. The way images are loaded
 * (e.g.    * new ImageIcon(AdminToolFrame.class.getResource("xmlJdom.gif"));)
 * guarantees that if the application starts all the images are found and
 * loaded.
 *
 */
public AdminToolFrame() {
  passwordMap = new HashMap();
  infoStructRoot = new Element("informationstructure");
  dataStructRoot = new Element("dataStructure");
  infoStructDoc = new Document(infoStructRoot);
  dataStructDoc = new Document(dataStructRoot);
  treeModel1 = new JDOMTreeModelAttr(dataStructDoc, (DataModifier) this);
  treeModel2 = new JDOMTreeModel(infoStructDoc, (DataModifier) this);
  Element root = new Element("datasources");
  dataSrcConfig = new Document(root);
  //initialize some passwords
  initializePasswordMap();

  //load the images
  jdomXmlTransformIcon = new ImageIcon(AdminToolFrame.class.getResource(
      "xmlJdom.gif"));
```

```
  xsltXmlIcon = new ImageIcon(AdminToolFrame.class.getResource(
      "xmlXslt.gif"));
  dstIcon = new ImageIcon(AdminToolFrame.class.getResource(
      "dst.gif"));
  xmlIcon = new ImageIcon(AdminToolFrame.class.getResource(
      "xml.gif"));
  frameImage = Toolkit.getDefaultToolkit().getImage(AdminToolFrame.class.
      getResource("frameIcon.gif"));
  newAssocIcon = new ImageIcon(AdminToolFrame.class.getResource(
      "newAssociation.gif"));
  loadAssocIcon = new ImageIcon(AdminToolFrame.class.getResource(
      "loadAssociation.gif"));
  saveAssocIcon = new ImageIcon(AdminToolFrame.class.getResource(
      "saveAssociation.gif"));
  closeAssocIcon = new ImageIcon(AdminToolFrame.class.getResource(
      "closeAssociation.gif"));
  viewReportIcon = new ImageIcon(AdminToolFrame.class.getResource(
      "ViewReport.gif"));
  viewXmlInfoStruct = new ImageIcon(AdminToolFrame.class.getResource(
      "ViewXMLInfoStruct.gif"));
  viewXmlSelection = new ImageIcon(AdminToolFrame.class.getResource(
      "ViewXMLSelection.gif"));
  exitIcon = new ImageIcon(AdminToolFrame.class.getResource(
      "exitApp.gif"));
  loadInfoStructIcon = new ImageIcon(AdminToolFrame.class.getResource(
      "loadInfoStruct.gif"));
  loadExistDSCIcon = new ImageIcon(AdminToolFrame.class.getResource(
      "loadExistDSC.gif"));
  loadNewDSCIcon = new ImageIcon(AdminToolFrame.class.getResource(
      "loadNewDSC.gif"));
  configDSCIcon = new ImageIcon(AdminToolFrame.class.getResource(
      "configDSC.gif"));
  saveDSCIcon = new ImageIcon(AdminToolFrame.class.getResource(
      "saveDSC.gif"));
  loadDataFileIcon = new ImageIcon(AdminToolFrame.class.getResource(
      "loadDataFile.gif"));
  saveDSTIcon = new ImageIcon(AdminToolFrame.class.getResource(
      "saveDST.gif"));
  aboutIcon = new ImageIcon(AdminToolFrame.class.getResource(
      "About16.gif"));

  enableEvents(AWTEvent.WINDOW_EVENT_MASK);
  //set up gui
  try {
    jbInit();
  }
  catch (Exception e) {
    e.printStackTrace();
  }

  //start guiUpdater
  guiUpdater = new GUIUpdater();

}

/**
 * Set up GUI. <br>
 * Sets the components' properties and lays them out. <br>
 * Most of this code is automatically generated by JBuilder. Especially the
 * laying out of the components.<br>
 * This is the JBuilders method for setting up the GUI. It is necessary to
 * use this method it order to be able to modify the  GUI from JBuilder's
 * Design view.
 *
 *
 * @throws Exception
 */
private void jbInit() throws Exception {
  //set frame image
  super.setIconImage(frameImage);

  contentPane = (JPanel)this.getContentPane();
  contentPane.setLayout(borderLayout1);
  this.setResizable(false);
  this.setSize(new Dimension(842, 558));
  this.setTitle("Administration Tool");
```

```
    jMenuFile.setText("File");
    jMenuFileExit.setText("Exit");
    jMenuFileExit.addActionListener(new
                                    AdminToolFrame_jMenuFileExit_ActionAdapter(this));
    jMenuHelp.setText("Help");
    jMenuHelpAbout.setText("About");
    jMenuHelpAbout.addActionListener(new
                                     AdminToolFrame_jMenuHelpAbout_ActionAdapter(this));

    jButtonNewAssoc.setMaximumSize(new Dimension(25, 25));
    jButtonNewAssoc.setMinimumSize(new Dimension(15, 25));
    jButtonNewAssoc.setPreferredSize(new Dimension(25, 25));
    jButtonNewAssoc.setToolTipText("New Association");
    jButtonNewAssoc.addActionListener(new
                                      AdminToolFrame_jButtonNewAssoc_actionAdapter(this));
    jButtonLoadAssoc.setMaximumSize(new Dimension(25, 25));
    jButtonLoadAssoc.setMinimumSize(new Dimension(15, 25));
    jButtonLoadAssoc.setPreferredSize(new Dimension(25, 25));
    jButtonLoadAssoc.setToolTipText("Load Association");
    jButtonLoadAssoc.setIcon(null);
    jButtonSaveAssoc.setMaximumSize(new Dimension(25, 25));
    jButtonSaveAssoc.setMinimumSize(new Dimension(15, 25));
    jButtonSaveAssoc.setPreferredSize(new Dimension(25, 25));
    jButtonSaveAssoc.setToolTipText("Save Association");
    jButtonSaveAssoc.setIcon(null);

    mainPanel.setLayout(borderLayout2);
    jSplitPane.setDoubleBuffered(false);
    jSplitPane.setToolTipText("");
    jSplitPane.setOneTouchExpandable(true);
    jMenuItemFileNewAssoc.setToolTipText("");
    jMenuItemFileNewAssoc.setText("New Association");
    jMenuItemFileNewAssoc.addActionListener(new

AdminToolFrame_jMenuItemFileNewAssoc_actionAdapter(this));
    jMenuItemFileLoadAssoc.setText("Load Association");
    jMenuItemFileLoadAssoc.addActionListener(new

AdminToolFrame_jMenuItemFileLoadAssoc_actionAdapter(this));
    jMenuItemFileSaveAssoc.setText("Save Association");
    jMenuItemFileSaveAssoc.addActionListener(new

AdminToolFrame_jMenuItemFileSaveAssoc_actionAdapter(this));
    jMenuAssoc.setText("Association");
    jMenuItemLoadInfoStruct.setText("Load Information Structure");
    jMenuItemLoadInfoStruct.addActionListener(new

AdminToolFrame_jMenuItemLoadInfoStruct_actionAdapter(this));
    jMenuItemReport.addActionListener(new
                                      AdminToolFrame_jMenuItemReport_actionAdapter(this));
    jMenuItemLoadExistDataSrcConfig.setToolTipText("");
    jMenuItemLoadExistDataSrcConfig.setText(
        "Load Existing Data Source Configuration");
    jMenuItemLoadExistDataSrcConfig.addActionListener(new
        AdminToolFrame_jMenuItemLoadExistDataSrcConfig_actionAdapter(this));
    jMenuItemConfigDataSrc.setText("Configure Data Sources");
    jMenuItemConfigDataSrc.addActionListener(new

AdminToolFrame_jMenuItemConfigDataSrc_actionAdapter(this));
    jMenuItemSaveDataSrcConfig.setText("Save Data Source Configuration");
    jMenuItemSaveDataSrcConfig.addActionListener(new

AdminToolFrame_jMenuItemSaveDataSrcConfig_actionAdapter(this));
    jLabelDataSrc.setFont(new java.awt.Font("Dialog", 1, 14));
    jLabelDataSrc.setText("Data Structures");
    jLabelInfoStruct.setFont(new java.awt.Font("Dialog", 1, 14));
    jLabelInfoStruct.setText("Information Structure");
    jButtonLoadExistDSC.setMaximumSize(new Dimension(25, 25));
    jButtonLoadExistDSC.setMinimumSize(new Dimension(15, 25));
    jButtonLoadExistDSC.setPreferredSize(new Dimension(25, 25));
    jButtonLoadExistDSC.setToolTipText(
        "Load Existing Data Sourse Configuration");
    jButtonLoadExistDSC.addActionListener(new

AdminToolFrame_jButtonLoadExistDSC_actionAdapter(this));
    jButtonLoadNewDSC.setMaximumSize(new Dimension(25, 25));
    jButtonLoadNewDSC.setMinimumSize(new Dimension(15, 25));
```

```
    jButtonLoadNewDSC.setPreferredSize(new Dimension(25, 25));
    jButtonLoadNewDSC.setToolTipText("Load New Data Sourse Configuration");
    jButtonLoadNewDSC.setText("");
    jButtonLoadNewDSC.addActionListener(new
                                        AdminToolFrame_jButtonLoadNewDSC_actionAdapter(this));
    jButtonConfigDSC.setMaximumSize(new Dimension(25, 25));
    jButtonConfigDSC.setMinimumSize(new Dimension(15, 25));
    jButtonConfigDSC.setPreferredSize(new Dimension(25, 25));
    jButtonConfigDSC.setToolTipText("Configure Data Sourses");
    jButtonConfigDSC.addActionListener(new
                                       AdminToolFrame_jButtonConfigDSC_actionAdapter(this));
    jToolBar.setFloatable(true);
    ToolBarPanel.setLayout(gridLayout2);
    LeftPanel.setLayout(borderLayout4);
    RightPanel.setLayout(borderLayout5);
    gridLayout2.setHgap(10);
    jTree2.setCellRenderer(infoStructTreeCellRenderer);
    jTree1.setMinimumSize(new Dimension(0, 0));
    jTree1.setCellRenderer(dataSrcTreeCellRenderer);
    jTree1.setModel(null);

    jMenuItemInfoFieldProps.setToolTipText("");
    jMenuItemInfoFieldProps.setText("Information Field Properties");
    jMenuItemInfoFieldProps.addActionListener(new

AdminToolFrame_jMenuItemInfoFieldProps_actionAdapter(this));
    statusBar.setFont(new java.awt.Font("Dialog", 1, 12));
    statusBar.setAlignmentX( (float) 0.0);
    statusBar.setBorder(BorderFactory.createLoweredBevelBorder());
    statusBar.setDisabledTextColor(Color.gray);
    statusBar.setEditable(false);
    statusBar.setText("status:");
    jMenuItemLoadNewDataSrcConfig.setText("Load New Data Source Configuration");
    jMenuItemLoadNewDataSrcConfig.addActionListener(new
        AdminToolFrame_jMenuItemLoadNewDataSrcConfig_actionAdapter(this));
    jMenuItemSaveDataStruct.setText("Save Data Structure");
    jMenuItemSaveDataStruct.addActionListener(new

AdminToolFrame_jMenuItemSaveDataStruct_actionAdapter(this));
    jMenuItemRemoveFile.setText("Remove File");
    jMenuItemRemoveFile.addActionListener(new

AdminToolFrame_jMenuItemRemoveFile_actionAdapter(this));

    jMenuLoadDataFile.setText("Load Data File");
    jMenuItemLoadDataStruct.addActionListener(new

AdminToolFrame_jMenuItemLoadDataStruct_actionAdapter(this));
    jMenuItemLoadXML_XSLT.addActionListener(new

AdminToolFrame_jMenuItemLoadXML_XSLT_actionAdapter(this));
    jMenuItemLoadXML_jdomTransform.addActionListener(new
        AdminToolFrame_jMenuItemLoadXML_jdomTransform_actionAdapter(this));
    jMenuItemLoadXml.addActionListener(new
                                       AdminToolFrame_jMenuItemLoadXml_actionAdapter(this));
    jMenuItemReport.setText("View Report on Information Structure");
    jMenuItemInfoStructXML.setText("View XML of Information Structure");
    jMenuItemInfoStructXML.addActionListener(new

AdminToolFrame_jMenuItemInfoStructXML_actionAdapter(this));
    jMenuItemSelectedXML.setText("View XML of selection in Data Structure");
    jMenuItemSelectedXML.addActionListener(new

AdminToolFrame_jMenuItemSelectedXML_actionAdapter(this));
    jMenuItemFileCloseAssoc.setText("Close Association");
    jMenuItemFileCloseAssoc.addActionListener(new

AdminToolFrame_jMenuItemFileCloseAssoc_actionAdapter(this));
    jMenuItemRemoveAssoc.setText("Remove Association");
    jMenuItemRemoveAssoc.addActionListener(new

AdminToolFrame_jMenuItemRemoveAssoc_actionAdapter(this));
    jButtonLoadXMLXSLT.setMaximumSize(new Dimension(25, 25));
    jButtonLoadXMLXSLT.setMinimumSize(new Dimension(15, 25));
    jButtonLoadXMLXSLT.setPreferredSize(new Dimension(25, 25));
    jButtonLoadXMLXSLT.setToolTipText(
        "Load XML file through XSLT transformation");
```

```
jButtonLoadXMLXSLT.setText("");
jButtonLoadXMLXSLT.addActionListener(new
                                     AdminToolFrame_jButtonLoadXMLXSLT_actionAdapter(this));
jButtonLoadXMLJDOM.setMaximumSize(new Dimension(25, 25));
jButtonLoadXMLJDOM.setMinimumSize(new Dimension(15, 25));
jButtonLoadXMLJDOM.setPreferredSize(new Dimension(25, 25));
jButtonLoadXMLJDOM.setToolTipText(
    "Load an XML file through JDOM transformation");
jButtonLoadXMLJDOM.addActionListener(new
                                     AdminToolFrame_jButtonLoadXMLJDOM_actionAdapter(this));
jButtonSaveDST.setMaximumSize(new Dimension(25, 25));
jButtonSaveDST.setMinimumSize(new Dimension(15, 25));
jButtonSaveDST.setPreferredSize(new Dimension(25, 25));
jButtonSaveDST.setToolTipText("Save Data Structured file");
jButtonSaveDST.addActionListener(new
                                 AdminToolFrame_jButtonSaveDST_actionAdapter(this));
jButtonSaveDSC.setMaximumSize(new Dimension(25, 25));
jButtonSaveDSC.setMinimumSize(new Dimension(15, 25));
jButtonSaveDSC.setPreferredSize(new Dimension(25, 25));
jButtonSaveDSC.setToolTipText("Save Data Sourse Configuration");
jButtonSaveDSC.setText("");
jButtonSaveDSC.addActionListener(new
                                 AdminToolFrame_jButtonSaveDSC_actionAdapter(this));
jButtonLoadXML.setMaximumSize(new Dimension(25, 25));
jButtonLoadXML.setMinimumSize(new Dimension(15, 25));
jButtonLoadXML.setPreferredSize(new Dimension(25, 25));
jButtonLoadXML.setToolTipText("Load a generic XML file");
jButtonLoadXML.setText("");
jButtonLoadXML.addActionListener(new
                                 AdminToolFrame_jButtonLoadXML_actionAdapter(this));
jButtonLoadDST.setMaximumSize(new Dimension(25, 25));
jButtonLoadDST.setMinimumSize(new Dimension(15, 25));
jButtonLoadDST.setPreferredSize(new Dimension(25, 25));
jButtonLoadDST.setToolTipText("Load Data Structured file");
jButtonLoadDST.setText("");
jButtonLoadDST.addActionListener(new
                                 AdminToolFrame_jButtonLoadDST_actionAdapter(this));
jButtonCloseAssoc.setMaximumSize(new Dimension(25, 25));
jButtonCloseAssoc.setMinimumSize(new Dimension(15, 25));
jButtonCloseAssoc.setPreferredSize(new Dimension(25, 25));
jButtonCloseAssoc.setToolTipText("Close Association");
jButtonLoadIS.setMaximumSize(new Dimension(25, 25));
jButtonLoadIS.setMinimumSize(new Dimension(15, 25));
jButtonLoadIS.setPreferredSize(new Dimension(25, 25));
jButtonLoadIS.setToolTipText("Load Information Structure");
jButtonLoadIS.addActionListener(new
                                AdminToolFrame_jButtonLoadIS_actionAdapter(this));

jButtonNewAssoc.setActionCommand(GUIUpdater.NEW);
jButtonLoadAssoc.setActionCommand(GUIUpdater.LOAD);
jButtonLoadAssoc.addActionListener(new
                                   AdminToolFrame_jButtonLoadAssoc_actionAdapter(this));
jButtonSaveAssoc.setActionCommand(GUIUpdater.SAVE);
jButtonSaveAssoc.addActionListener(new
                                   AdminToolFrame_jButtonSaveAssoc_actionAdapter(this));
jButtonCloseAssoc.setActionCommand(GUIUpdater.CLOSE);
jButtonCloseAssoc.addActionListener(new
                                    AdminToolFrame_jButtonCloseAssoc_actionAdapter(this));

jMenuItemRemoveAll.setToolTipText("It clears the Data Structure");
jMenuItemRemoveAll.setText("Remove All");
jMenuItemRemoveAll.addActionListener(new
                                     AdminToolFrame_jMenuItemRemoveAll_actionAdapter(this));
jToolBar.add(ToolBarPanel, null);
ToolBarPanel.add(jButtonNewAssoc, null);
ToolBarPanel.add(jButtonLoadAssoc, null);
ToolBarPanel.add(jButtonSaveAssoc, null);
ToolBarPanel.add(jButtonCloseAssoc, null);
ToolBarPanel.add(jButtonLoadIS, null);
ToolBarPanel.add(jButtonLoadExistDSC, null);
ToolBarPanel.add(jButtonLoadNewDSC, null);
ToolBarPanel.add(jButtonConfigDSC, null);
ToolBarPanel.add(jButtonSaveDSC, null);
ToolBarPanel.add(jButtonLoadDST, null);
ToolBarPanel.add(jButtonLoadXMLXSLT, null);
ToolBarPanel.add(jButtonLoadXMLJDOM, null);
ToolBarPanel.add(jButtonLoadXML, null);
```

```
ToolBarPanel.add(jButtonSaveDST, null);
jButtonNewAssoc.setIcon(newAssocIcon);
jButtonLoadAssoc.setIcon(loadAssocIcon);
jButtonSaveAssoc.setIcon(saveAssocIcon);
jButtonCloseAssoc.setIcon(closeAssocIcon);
jButtonLoadIS.setIcon(loadInfoStructIcon);
jButtonLoadExistDSC.setIcon(loadExistDSCIcon);
jButtonLoadNewDSC.setIcon(loadNewDSCIcon);
jButtonConfigDSC.setIcon(configDSCIcon);
jButtonSaveDSC.setIcon(saveDSCIcon);
jButtonLoadDST.setIcon(dstIcon);
jButtonLoadXMLXSLT.setIcon(xsltXmlIcon);
jButtonLoadXMLJDOM.setIcon(jdomXmlTransformIcon);
jButtonLoadXML.setIcon(xmlIcon);
jButtonSaveDST.setIcon(saveDSTIcon);
jMenuFile.add(jMenuItemFileNewAssoc);
jMenuFile.add(jMenuItemFileLoadAssoc);
jMenuFile.add(jMenuItemFileSaveAssoc);
jMenuFile.add(jMenuItemFileCloseAssoc);
jMenuFile.addSeparator();
jMenuFile.add(jMenuItemReport);
jMenuFile.add(jMenuItemInfoStructXML);
jMenuFile.add(jMenuItemSelectedXML);
jMenuFile.addSeparator();
jMenuFile.add(jMenuFileExit);
jMenuHelp.add(jMenuHelpAbout);
jMenuBar1.add(jMenuFile);
jMenuBar1.add(jMenuAssoc);
jMenuBar1.add(jMenuHelp);
this.setJMenuBar(jMenuBar1);
contentPane.add(jToolBar, BorderLayout.NORTH);
contentPane.add(mainPanel, BorderLayout.CENTER);
LeftPanel.add(jLefttScrollPane, BorderLayout.CENTER);
jLefttScrollPane.getViewport().add(jTree2, null);
LeftPanel.add(jLabelInfoStruct, BorderLayout.NORTH);
contentPane.add(statusBar, BorderLayout.SOUTH);
jSplitPane.add(RightPanel, JSplitPane.RIGHT);
jSplitPane.add(LeftPanel, JSplitPane.LEFT);
RightPanel.add(jRightScrollPane, BorderLayout.CENTER);
jRightScrollPane.getViewport().add(jTree1, null);
//jTree1 jrm
jTree1.setName("Data Structures");
jTree1.setModel(treeModel1); //map TreeModel toy Jtree
jTree1.addTreeSelectionListener(treeSelectionHandler);
selectionModel1 = (DefaultTreeSelectionModel) jTree1.getSelectionModel();
selectionModel1.setSelectionMode(TreeSelectionModel.SINGLE_TREE_SELECTION);
jTree1.setTransferHandler(treeTransferHandler); //dnd
jTree1.setDragEnabled(true); //dnd
jTree1.setEditable(false);
treeModel1.addTreeModelListener(treeModelAdapter);
//jTree2
jTree2.setName("Information Structure");
jTree2.setModel(treeModel2); //map TreeMode2 toy Jtree
jTree2.addTreeSelectionListener(treeSelectionHandler);
selectionModel2 = (DefaultTreeSelectionModel) jTree2.getSelectionModel();
selectionModel2.setSelectionMode(TreeSelectionModel.SINGLE_TREE_SELECTION);
jTree2.setTransferHandler(treeTransferHandler); //dnd
jTree2.setDragEnabled(false); //dnd
jTree2.setEditable(true);
treeModel2.addTreeModelListener(treeModelAdapter);
jTree1.addMouseListener(new PopupListener());
jTree2.addMouseListener(new PopupListener());

jMenuItemLoadXML_XSLT.setText("Generic XML through XSLT");
jMenuItemLoadXML_XSLT.setIcon(xsltXmlIcon);
jMenuItemLoadXML_jdomTransform.setText(
    "Generic XML through JDOM Transformation");
jMenuItemLoadXML_jdomTransform.setIcon(jdomXmlTransformIcon);
jMenuItemLoadDataStruct.setText("Data Structure file");
jMenuItemLoadDataStruct.setIcon(dstIcon);
jMenuItemLoadXml.setText("Generic XML");
jMenuItemLoadXml.setIcon(xmlIcon);
//--
RightPanel.add(jLabelDataSrc, BorderLayout.NORTH);
mainPanel.add(jSplitPane, BorderLayout.CENTER);
jMenuAssoc.add(jMenuItemLoadInfoStruct);
jMenuAssoc.addSeparator();
```

```
  jMenuAssoc.add(jMenuItemLoadExistDataSrcConfig);
  jMenuAssoc.add(jMenuItemLoadNewDataSrcConfig);
  jMenuAssoc.add(jMenuItemConfigDataSrc);
  jMenuAssoc.add(jMenuItemSaveDataSrcConfig);
  jMenuAssoc.addSeparator();
  jMenuLoadDataFile.add(jMenuItemLoadDataStruct);
  jMenuLoadDataFile.add(jMenuItemLoadXML_XSLT);
  jMenuLoadDataFile.add(jMenuItemLoadXML_jdomTransform);
  jMenuLoadDataFile.add(jMenuItemLoadXml);
  jMenuAssoc.add(jMenuLoadDataFile);
  jMenuAssoc.add(jMenuItemSaveDataStruct);
  jPopupMenu1.add(jMenuItemRemoveAssoc);
  jPopupMenu1.add(jMenuItemInfoFieldProps);
  jPopupMenu1.addSeparator();
  jPopupMenu1.add(jMenuItemRemoveFile);
  jPopupMenu1.add(jMenuItemRemoveAll);
  jSplitPane.setDividerLocation(415);
  jLabelInfoStruct.setHorizontalTextPosition(SwingConstants.CENTER);
  jLabelDataSrc.setHorizontalAlignment(SwingConstants.CENTER);
  jLabelInfoStruct.setHorizontalAlignment(SwingConstants.CENTER);

  jMenuItemFileNewAssoc.setIcon(newAssocIcon);
  jMenuItemFileLoadAssoc.setIcon(loadAssocIcon);
  jMenuItemFileSaveAssoc.setIcon(saveAssocIcon);
  jMenuItemFileCloseAssoc.setIcon(closeAssocIcon);
  jMenuItemReport.setIcon(viewReportIcon);
  jMenuItemInfoStructXML.setIcon(viewXmlInfoStruct);
  jMenuItemSelectedXML.setIcon(viewXmlSelection);
  jMenuFileExit.setIcon(exitIcon);
  jMenuItemLoadInfoStruct.setIcon(loadInfoStructIcon);
  jMenuItemLoadExistDataSrcConfig.setIcon(loadExistDSCIcon);
  jMenuItemLoadNewDataSrcConfig.setIcon(loadNewDSCIcon);
  jMenuItemConfigDataSrc.setIcon(configDSCIcon);
  jMenuItemSaveDataSrcConfig.setIcon(saveDSCIcon);
  jMenuItemSaveDataStruct.setIcon(saveDSTIcon);
  jMenuLoadDataFile.setIcon(loadDataFileIcon);
  jMenuHelpAbout.setIcon(aboutIcon);

} //end jbInit

/**
 *  Initializes the application with some passwords.
 *  key=login, value=password
 *
 *
 */
private void initializePasswordMap() {
  try {
    passwordMap.put("jrm", "pass");
  }
  catch (Exception ex) {
    ex.printStackTrace();
  }
  //
}

/**
 * Listens for mouse events and when right click occurs,
 * it shows the popup menu.
 * <p>
 * Only maybeShowPopup method is used.
 * </p>
 *
 */
private class PopupListener
    extends MouseAdapter {
  /**
   * Calls maybeShowPopup
   * @param e MouseEvent
   */
  public void mousePressed(MouseEvent e) {
    maybeShowPopup(e);
  }

  /**
   * Calls maybeShowPopup
   * @param e MouseEvent
```

```
   */
  public void mouseReleased(MouseEvent e) {
    maybeShowPopup(e);
  }

  /**
   * Check if the event is a right click and shows the popup in the clicked
   * position. <br>
   * Also disables and enables menu items of the popup menu
   * according to the component (Jtree) the click occured.
   * @param e MouseEvent
   */
  private void maybeShowPopup(MouseEvent e) {
    //if proper event occured show popup
    if (e.isPopupTrigger()) {
      jPopupMenu1.show(e.getComponent(),
                       e.getX(), e.getY());
      //eneble-disable menuitems
      String componentName = e.getComponent().getName();
      if (componentName.equals(jTree1.getName())) { //Data Structures
        jMenuItemInfoFieldProps.setEnabled(false);
        jMenuItemRemoveAssoc.setEnabled(false);
        jMenuItemRemoveFile.setEnabled(true);
        jMenuItemRemoveAll.setEnabled(true);
      }
      else if (componentName.equals(jTree2.getName())) { //Info Structure
        jMenuItemInfoFieldProps.setEnabled(true);
        jMenuItemRemoveAssoc.setEnabled(true);
        jMenuItemRemoveFile.setEnabled(false);
        jMenuItemRemoveAll.setEnabled(false);
      }
    }
  }

}

/**
 * File | Exit action performed.
 * Exits the application. <br>
 * First checks to see if an assocoation exists (To protect user from losing data). If
 * it exists shows warning message and cancels exiting. If no assocoation exists, then exits.
 *
 */
public void jMenuFileExit_actionPerformed(ActionEvent e) {
  //if there is an Associsation which is not closed, then associationName != null
  if (associationName != null) {
    JOptionPane.showMessageDialog(null,
                                  "Please close the Association before attempting to exit.",
                                  "Application cannot exit!",
                                  JOptionPane.WARNING_MESSAGE);
  }
  else  if (associationName == null) {
    System.exit(0);
  }
  else {
    System.err.println("ERRRO in jMenuFileExit_actionPerformed - unexpected program flow!");
  }
}

/**
 * Help | About action performed.
 * DIsplays the About Box.
 */

public void jMenuHelpAbout_actionPerformed(ActionEvent e) {
  AdminToolFrame_AboutBox dlg = new AdminToolFrame_AboutBox(this);
  Dimension dlgSize = dlg.getPreferredSize();
  Dimension frmSize = getSize();
  Point loc = getLocation();
  dlg.setLocation( (frmSize.width - dlgSize.width) / 2 + loc.x,
                  (frmSize.height - dlgSize.height) / 2 + loc.y);
  dlg.setModal(true);
  dlg.pack();
  dlg.show();
}

/**
```

```
   * Overridden so we can exit when window is closed and when no Association exists. <br>
   * If window (that is the AdminToolFrame) is closed, then it delegates the closing action
   * to File | Exit action performed mnethod.
   *
   * @see jMenuFileExit_actionPerformed(ActionEvent)
   */
  protected void processWindowEvent(WindowEvent e) {
   if (e.getID() == WindowEvent.WINDOW_CLOSING) {
      jMenuFileExit_actionPerformed(null);
    }
  }

  /**
   * It calculates the XPath expression of the selected node of the tree.
   * <p>
   * It takes a tree selection event , it figures out in which JTree node it
   * took place, retrieves the JDOM Content object of that node and calls
   * buildXPath with that JDOM node.
   * </p>
   * To detect when the user selects a node in a tree , you need to register
   * a tree selection listener. That is why the TreeSelectionListener is
   * implemented.
   *
   */
  class TreeSelectionHandler
      implements TreeSelectionListener {
    /**
     * Interface TreeSelectionListener implemented method. <p>
     * The method that performs the actions of the TreeSelectionHandler. <br>
     * The following steps are performed:
     * <ul>
     * <li>Define which tree the selection occured, because
     *     it servers events for both jTree1 and jTree2</li>
     * <li>Acquire Treepath from selection model, and then from the treepath
     *     the JDOM object</li>
     * <li>Call XPathEvaluator.getXPath method to find XPath</li>
     * </ul>
     * </p>
     * @param e TreeSelectionEvent
     */
    public void valueChanged(TreeSelectionEvent e) {
      //vars
      TreePath path = null;

      //Define the tree the selection occured
      Object source = e.getSource();
      if (source.equals(jTree1)) { //tree1 --> dataSource
        path = selectionModel1.getSelectionPath(); //Returns the first path in the selection.
        if (path != null) { // If you are just collapsing the tree, you may not have a new path
          Object selectedObject = (Object) path.getLastPathComponent(); //Returns the last
component of this path.
          //call XPathEvalutor to calculate the XPAth
          statusBar.setText(jTree1.getName() + ": XPath=  " +
                            xPathEvaluator.getXPath(selectedObject));
        }
        //else {
        //  System.out.println(
        //      "*** ERROR - TreeSelectionHandler1: Treepath is null");
        //}
      }
      else if (source.equals(jTree2)) { //tree2 --> infoStruct
        path = selectionModel2.getSelectionPath(); // Returns the first path in the selection.-
        if (path != null) { // If you are just collapsing the tree, you may not have a new path
          Object selectedObject = (Object) path.getLastPathComponent(); //Returns the last
component of this path.
          if (selectedObject instanceof Text) { //call XPathEvalutor to calculate the XPAth
            Text text = (Text) selectedObject;
            statusBar.setText("Association's XPath= " + text.getText().trim());
          }
          //else {
          //DO NOTHING
          //Because you may want to see DataSource XPath while displaying Information field's
properties
          //Uncomment to enable displaing the XPath of Info Struct too.
          //statusBar.setText(jTree2.getName() + ": " +xPathEvaluator.getXPath(selectedObject));
          //statusBar.setText(jTree2.getName());
          // }
```

```
      }

    }
    else { //It should never be executed
      System.err.println(
          "ERROR - TreeSelectionHandler: No such event source defined");
      // If you are just collapsing the tree, you may not have a new path
    }

  } //valueChanged
}

/**
 * For expanding automatically the inserted nodes after Drag and Drop.
 * <p>
 * So only treeNodesInserted is implemented, because only when nodes are
 * inserted it is needed to perform an action (expand the nodes).
 * It servers events for both jTree1 and jTree2.
 * </p>
 * To be able to listen to TreeModelEvents the class implements the
 * TreeModelListener interface. TreeModelEvent is generated when the tree
 * model of the jtree is modified that is a node is inserted.
 */
class TreeModelAdapter
    implements TreeModelListener {
  /**
   * Not implemented. Because no action need to be performed when a node is
   * changed.
   *
   * @param e TreeModelEvent
   */
  public void treeNodesChanged(TreeModelEvent e) {
  }

  /**
   * Expands the inserted nodes.
   * <p>
   * <ul>
   * <li>Defines which from which tree model generated the event</li>
   * <li>Expands the path of the relevant jtree</li>
   * </ul>
   * </p>
   * <p>
   * If deeper expantion is needed then
   * replace path with childPath
   * "TreePath childPath = e.getTreePath().pathByAddingChild(e.getChildren()[0]);"
   * that is "jTree12.expandPath(childPath);"
   * </p>
   * @param e TreeModelEvent
   */
  public void treeNodesInserted(TreeModelEvent e) {
    //Get path to be expanded
    TreePath path = e.getTreePath();

    //define tree model that generated the event and expand relavant jtree
    Object source = e.getSource();
    if (source.equals(treeModel1)) {
      jTree1.expandPath(path);
    }
    else if (source.equals(treeModel2)) {
      jTree2.expandPath(path);
    }
    else { //It should never be executed
      System.err.println(
          "ERROR - TreeModelAdapter: No such event source defined");

    }
  }

  /**
   * Not implemented. Because no action need to be performed when a node is
   * changed.
   *
   * @param e TreeModelEvent
   */
  public void treeNodesRemoved(TreeModelEvent e) {
  }
```

```
    /**
     * Not implemented. Because no action need to be performed when a node is
     * changed.
     *
     * @param e TreeModelEvent
     */
    public void treeStructureChanged(TreeModelEvent e) {
    }

  }

  /**
   * Association | Load Information Structure action performed.
   * <p>
   * Calls the JFileChooser for user to pick a file, and then calls the necessary
   * methods to load the file as the information structure. <br>
   * The steps taken are:
   * <ul>
   * <li>Sets up the jFileChooser (set title anf file filter) and calls it for the user to
choose a file. </li>
   * <li>Loads the file into a JDOM Document with loadXML method.</li>
   * <li>Loads the loaded Document to the information structure by calling
   *      method loadInfoStruct. </li>
   * <li>updates the gui: updates related Jtree, statusbar, notifies GUIUpdater of the
   *     action that was performed </li>
   * <li>removes the file filter used from jfilechooser</li>
   * </ul>
   * </p>
   *
   * @param e ActionEvent
   */
  void jMenuItemLoadInfoStruct_actionPerformed(ActionEvent e) {
    //Sets up the jFileChooser (set title anf file filter) and calls it for the user to choose a
file.
    jFileChooser1.setDialogTitle("Choose an Information Structure");
    jFileChooser1.addChoosableFileFilter(isFileFilter);
    if (JFileChooser.APPROVE_OPTION == jFileChooser1.showOpenDialog(this)) {
      //get the file
      infoStructFile = jFileChooser1.getSelectedFile();
      //test if it is the proper file type
      if (jFileChooser1.accept(infoStructFile)) {
        //get the document for the file
        Document doc = loadXML(infoStructFile);
        //load the document to the information structure
        loadInfoStruct(doc);

        //update Jtree GUI
        updateJTreeGUI(jTree2);
        statusBar.setText("Loaded: " + jFileChooser1.getSelectedFile().getPath());
        //notify the guiupdater of the action
        guiUpdater.update(GUIUpdater.LOAD_INFO_STRUCT); //jLabelInfoStruct.setText("Information
Structure: " + infoStructFile.getName());
      }
      else {
        JOptionPane.showMessageDialog(null,
                                      "Please choose a Information Structure (*.is) file.",
                                      "Wrong file type",
                                      JOptionPane.ERROR_MESSAGE);
      }
    }
    //remove file filter form the file chooser
    jFileChooser1.removeChoosableFileFilter(isFileFilter);
  }

  /**
   * Refreshes the view of the JTree.
   * <p>
   * Refresh is needed because modifications in the tree model may not be shown
   * be the Jtree yet.
   * The update is done by collapsing and expanding the root of the JTree.
   * </p>
   *
   * @param jTree JTree - The tree to be updated
   */
  private void updateJTreeGUI(JTree jTree) {
    //first Build root path
```

```
  Element[] pathToRoot = new Element[1];
  if (jTree.equals(jTree2)) {
    pathToRoot[0] = infoStructRoot;
  }
  else if (jTree.equals(jTree1)) {
    pathToRoot[0] = dataStructRoot;
  }
  else {
    System.err.println("Error in updateJtreeGUI");
  }
  TreePath rootPath = new TreePath(pathToRoot);

  //then update
  jTree.treeDidChange();
  jTree.collapsePath(rootPath);
  jTree.expandPath(rootPath);

}

/**
 * Loads the document to the Information structure.
 *
 * <p>
 * <ul>
 * <li>Builds the root path necessary for calling the
 * JDOMTreeModel.insertNodeInto method</li>
 * <li>Clones the content of the document</li>
 * <li>Delegate insertion to JDOMTreeModel by calling insertNodeInto
 * method</li>
 * <li>Set Information Structure root's attribute </li>
 * </ul> </p>
 *
 * <p> Insert must be delegated to JDOMTreeModel in order to have coherence
 * between the Data hold in the tree model and what JTree shows. Otherwise, if
 * we add the Content by using a addContent method of JDOM Element, then the
 * content will be added but JTree will not know about the modification and so
 * it will not show it.
 * <br> But by delegating the insertion to JDOMTreeModel while looping the
 * document here results in a cocurrent modification exception thrown by JDOM.
 * So the content is cloned before insertion. </p>
 *
 * @see loadDataStruct(Document)
 * @param doc Document - loaded document
 */
private void loadInfoStruct(Document doc) {
  //Building path to root
  Element[] pathToRoot = new Element[1];
  pathToRoot[0] = infoStructRoot;
  TreePath rootPath = new TreePath(pathToRoot);

  //Get loaded content
  Element root = doc.getRootElement();
  List list = root.cloneContent();
  Iterator i = list.iterator();
  //insert it
  while (i.hasNext()) {
    Element el = (Element) i.next();
    //delegate insetr action to JDOMTreeModel
    treeModel2.insertNodeInto(el, infoStructRoot, 0, rootPath);
  }
  //set Attribute
  infoStructRoot.setAttribute("type", root.getAttribute("type").getValue());

}

/**
 * Loads an XML from a file to a JDOM Document. This is the only method
 * throughout the application that loads XML into a Document.<br>
 * The XML filter is used in order to avoid creating ignorable whitespace
 * <code>Text</code> nodes. See XMLWhitespaceFilter for more.
 *
 * <p>
 * <ul>
 * <li>Create builder</li>
 * <li>Set builder's XML filter.  </li>
 * <li>Load document catching Exceptions</li>
 * </ul> </p>
```

```
 * It is declared public because it is used be other classes too.
 * <p>
 *
 * @return Document - the loaded XML
 * @see XMLWhitespaceFilter
 * @param file - the XML file to be loaded
 */
public Document loadXML(File file) {
  Document doc = null;
  // --  Load the JDOM Document --
  //create SAX builder for loading
  SAXBuilder builder = new SAXBuilder();
  //set XMLFilter of builder
  builder.setXMLFilter(new XMLWhitespaceFilter());
  //load
  try {
    doc = builder.build(file);
  }
  catch (java.io.IOException ioe) {
    System.err.println(
        " IO Exception -- CANNOT OPEN FILE OR FILE NOT FOUND!");
  }
  catch (JDOMException jdome) {
    System.out.println(file.getName() + " is not well-formed.");
    System.out.println(jdome.getMessage());
  }
  return doc;
}

/**
 * Loads a Data Source Cofiguration.
 * <br> Takes a data source configuration Document, that is a *.dsc file that
 * it is already loaded as a JDOM Document and loads it to the Data Structure
 * tree.
 *
 * <p> More precisely the actions taken are:
 * <ul>
 * <li>Get content of cofiguration document and acquire Data Source parameters
 * from each configuraion node</li>
 * <li>Retrieve Data Sources with the use of <code>DataSrcMapper</code>.</li>
 * <li>Load data sources to data structure tree by calling
 * <code>loadDataStruct</code> method. </li>
 * </ul> </p>
 *
 * @see DataSrcMapper
 * @see loadDataStruct(Document)
 * @param dataSrcConfig the data source configuration document.
 */
private void loadDataSourceConfig(Document dataSrcConfig) {
  Element datasource = null;
  //data source type
  String DATA_SRC_TYPE = null;
  //JDBC driver
  String JDBC_DRIVER = null;
  //data source name
  String DATA_SRC_NAME = null;

  //Get content of cofiguration file
  Element datasources = dataSrcConfig.getRootElement();
  List datasourceList = datasources.getChildren();
  Iterator iterator = datasourceList.iterator();
  //for each node of cofiguration file contents
  while (iterator.hasNext()) {
    //Acquiring parameters from configuraion node
    datasource = (Element) iterator.next();
    DATA_SRC_TYPE = datasource.getAttributeValue("type");
    JDBC_DRIVER = datasource.getAttributeValue("driver");
    DATA_SRC_NAME = datasource.getChild("configuration").
        getAttributeValue("dsn");

    //load from data source
    DataSrcMapper dataSrcMapper = new DataSrcMapper(JDBC_DRIVER,
        DATA_SRC_NAME,
        DATA_SRC_TYPE);
    //Here we have Mapped a data Source of a database to a data struct
    //pass it to loadDataStruct in order to append it in the data Struct
    loadDataStruct(dataSrcMapper.getMappedDataSrc());
```

```
  } //end while
} //end loadDataSourceConfig

/**
 * Loads a document to the Data Structure tree.
 *  <p>
 * <ul>
 * <li>Builds the root path necessary for calling the
 *     JDOMTreeModel.insertNodeInto method</li>
 * <li>Clones the content of the document</li>
 * <li>Delegate insertion to JDOMTreeModel by calling insertNodeInto method</li>
 * </ul>
 * </p>
 * <p>
 * <p>Loading a document into the Data Source tree is not so
 * trivial. The loading must be performed by JDOMTreeModel in order to have
 * coherence between the tree model and the JTree. But by delegating the
 * insertion to JDOMTreeModel while looping the document here results in a
 * cocurrent modification exception thrown by JDOM.
 * So the content is cloned before insertion.
 * <p>
 *
 * @see loadInfoStruct(Document)
 */
private void loadDataStruct(Document loadedDoc) {
  //build root path used in insertNodeInto
  Element[] pathToRoot = new Element[1];
  pathToRoot[0] = dataStructDoc.getRootElement();
  TreePath dataStructRootPath = new TreePath(pathToRoot);

  //get data struct root
  Element dataStructRoot = dataStructDoc.getRootElement();
  Element loadedRoot = loadedDoc.getRootElement();
  //clone content to avoid Concurrent mdification thrown by JDOM
  List newList = loadedRoot.cloneContent();
  Iterator i = newList.iterator();
  //append its contents to data struct root one by one
  while (i.hasNext()) {
    Element loadedDataSrc = (Element) i.next();
    //delegate insertion to tree model
    treeModel1.insertNodeInto(loadedDataSrc, dataStructRoot, 0,
                              dataStructRootPath);
  } //end while
} //end loadDataStruct

/**
 * Loads a  generic XML document to the Data Structure.
 * <p> When a generic XML is loaded, some information (like the name of the \
 * file and its type) must be added to its root in order to be able to be
 * demostrated correctly from JTree. This is the use case of this method.
 * <ul>
 * <li>Get file name </li>
 * <li>Clone loaded document's root </li>
 * <li>Set Attributes </li>
 * <li>Call <code> loadDataStruct(Document)</code> to do the loading </li>
 * </ul>
 * </p>
 * @param loadedDoc - loaded document
 * @param file - the XML file
 */
void loadDataStruct(Document loadedDoc, File file) {
  //Get file name
  String filename = file.getName();
  //make a clone of loaded document root
  Element oldRoot = (Element) loadedDoc.getRootElement().clone();
  //put attributes required from demostration
  oldRoot.setAttribute("name", filename);
  oldRoot.setAttribute("type", "xml");
  //Call loadDataStruct(Document) by passing a new Document
  Element newRoot = new Element("root");
  newRoot.addContent(oldRoot);
  loadDataStruct(new Document(newRoot));
}

/**
 * Pop up | Information Field Properties action performed.
 * <p>
```

```
 * <ul>
 * <li>Gets the selected object.</li>
 * <li>Cases of object:
 *   <ul>
 *     <li>an information field element:
 *         <ul>
 *          <li> Get path of selection </li>
 *          <li>Calls the InformationFieldProperties
 *              Dialog</li>
 *         </ul>
 *     <li>not an information field: display relative message</li>
 *     <li>not an Element: display relative message</li>
 *    </ul>
 * </ul>
 * </p>
 *
 * @param e ActionEvent
 */
void jMenuItemInfoFieldProps_actionPerformed(ActionEvent e) {
  TreePath path = null;
  //Get selected object
  Object o = jTree2.getLastSelectedPathComponent();
  Element el = null;
  String elName = null;
  //cases of the object
  if (o instanceof Element) {
    el = (Element) o;
    elName = el.getName();
    if (elName.equals("informationfield")) {
      //Get path of selection used in INformationFieldProperties Dialog
      path = selectionModel2.getSelectionPath();
      //call InformationFieldProperties Dialog
      InformationFieldProperties ifp = new InformationFieldProperties(this,
          "Information Field Properties", true, el, treeModel2, path);
    }
    else { //not an information field
      JOptionPane.showMessageDialog(null,
                                    "Please choose an Information Field!",
                                    "No Information field",
                                    JOptionPane.WARNING_MESSAGE);
    }
  }
  else { // not element
    JOptionPane.showMessageDialog(null,
                                  "Please choose an Information Field!",
                                  "No Information field",
                                  JOptionPane.WARNING_MESSAGE);
  }
} //end jMenuItemInfoFieldProps_actionPerformed

/**
 * Association | Load Existing Data Source Configuration action performed. <br>
 * <p>
 * <ul>
 * <li>Sets up the jFileChooser (set title anf file filter) and calls it for
 *     the user to choose a *.dsc file. </li>
 * <li>Loads the file into a JDOM Document with loadXML method.</li>
 * <li>Loads the data sources defined in the loaded Document to the data
 *     structure by calling   method loadDataSourceConfig. </li>
 * <li>Updates the data source configuration document <code>dataSrcConfig</code>
 *     of the data sources loaded
 * <li>Updates related Jtree</li>
 * <li>Removes the file filter used from jfilechooser</li>
 * </ul>
 * </p>
 *
 * @param e ActionEvent
 */
void jMenuItemLoadExistDataSrcConfig_actionPerformed(ActionEvent e) {
  Document loadeDataSrcConfig = null;
  //Use a fileChooser to choose the file
  File file = null;
  //Use the OPEN version of the dialog, test return for Approve/Cancel
  jFileChooser1.setDialogTitle("Choose an Data Source Configuration");
  jFileChooser1.addChoosableFileFilter(dscFileFilter);
  if (JFileChooser.APPROVE_OPTION == jFileChooser1.showOpenDialog(this)) {
    file = jFileChooser1.getSelectedFile();
```

```
    //test if it is the proper file type
    if (jFileChooser1.accept(file)) {
      //load file to document
      loadeDataSrcConfig = loadXML(file);
      //load data sources defined in document
      loadDataSourceConfig(loadeDataSrcConfig);
      //update dataSrcConfig Doc
      dataSrcConfig.getRootElement().addContent(loadeDataSrcConfig.
                                                getRootElement().cloneContent());
    }
    else {
      JOptionPane.showMessageDialog(null,
                                    "Please choose a Data Source Configuration (*.dsc) file.",
                                    "Wrong file type",
                                    JOptionPane.ERROR_MESSAGE);
    }
  }
  //remove file filter from jfilechooser
  jFileChooser1.removeChoosableFileFilter(dscFileFilter);
  //update Jtree GUI
  updateJTreeGUI(jTree1);
}

/**
 * Association | Configure Data Sources action performed. <br>
 * Calls the  DataSourcesConfiguration Dialog.
 *
 * @param e ActionEvent
 * @see DataSourcesConfiguration
 */
void jMenuItemConfigDataSrc_actionPerformed(ActionEvent e) {
  DataSourcesConfiguration dsc = new DataSourcesConfiguration(this, true,
      dataSrcConfig, passwordMap);

}

/**
 * Association | New Data Source Configuration action performed. <br>
 * Calls the  DataSourcesConfiguration Dialog.
 *
 * @param e ActionEvent
 * @see DataSourcesConfiguration
 */
void jMenuItemLoadNewDataSrcConfig_actionPerformed(ActionEvent e) {
  DataSourcesConfiguration dsc = new DataSourcesConfiguration(this, true,
      dataSrcConfig, passwordMap);
}

/**
 * Sets the Data source configuration document <code>dataSrcConfig</code>.
 * Used by DataSourceConfiguration Diaog to restore the document to its inital
 * state and so discard any modifications.
 *
 *@see DataSourcesConfiguration
 */
void setDataSrcConfigDocument(Document doc) {
  dataSrcConfig = doc;
}

/**
 * Updates the data sources loaded on the Data Structure tree.
 * <p> Called from DataSourceConfiguration Dialog. Update is performed by
 * erasing all the datasources and write the data sources that the document
 * passed by DataSourcesConfiguration has. This might mean to rewrite the old
 * ones. That is:
 * <ul>
 * <li>Remove all data sources from data Structure tree</li>
 * <li>Load new the Data Src configuration</li>
 * <li>Update Jtree GUI</li>
 *</p>
 * @see DataSourcesConfiguration
 * @param doc - the Data source configuration document passed by calling
 *   object.
 */
public void updateDataStructure(Document doc) {
  //Remove all data sources from data Structure tree
  treeModel1.removeAll("dataStructureSource");
```

```
    //Load new the Data Src configuration
    loadDataSourceConfig(doc);
    //update Jtree GUI
    updateJTreeGUI(jTree1);
  }

  /**
   * Pop up | Remove File action performed.
   * <p>
   * <ul>
   * <li>Gets the path of the selection used in <code>removeNodeFromParent</code>
   * method.</li>
   * <li>Gets selected object</li>
   * <li>Cases of selected object:</li>
   *   <ul>
   *      <li>is a dataStructureFile: Delegate removal to
   *          <code>JDOMTreeModel.removeNodeFromParent</code>.</li>
   *      <li>Generic XML that means taht the selected Item has dataStructRoot
   *          as its Parent AND it is not a data source: Delegate removal to
   *          <code>JDOMTreeModel.removeNodeFromParent</code>.</li>
   *      <li>Any other case, which at this version is a data source:
   *          show an error message.</li>
   *   </ul>
   * </ul>
   * </p>
   * <p>
   * The reason for delegating the removal of a node to the tree model is the same
   * as in in loadInfoStruct and loadDataStruct methods.
   * </p>
   *
   * @param e ActionEvent
   * @see JDOMTreeModel
   * @see loadInfoStruct(Document)
   * @see loadDataStruct(Document)
   */
  void jMenuItemRemoveFile_actionPerformed(ActionEvent e) {

    TreePath path = null;

    //Get the path of the selection.
    path = selectionModel1.getSelectionPath();

    // If you are just collapsing the tree, you may not have a new path
    if (path != null) {
      //Gets the selected object.
      Object selectedObject = (Object) path.getLastPathComponent();
      //cases of selected object
      if (selectedObject instanceof Element) {
        Element removedEl = (Element) selectedObject;
        String elName = removedEl.getName();
        if (elName.equals("dataStructureFile")) { //is a dataStructureFile
          treeModel1.removeNodeFromParent(removedEl, path.getParentPath());
        }
        //the selected Item has dataStructRoot as its Parent AND it is not a data source
        else if ( (dataStructRoot.getName().equals(removedEl.getParentElement().
            getName())) && (!elName.equals("dataStructureSource"))) {
          treeModel1.removeNodeFromParent(removedEl, path.getParentPath());
        }
        else { //any other case, which at this version is a data source
          JOptionPane.showMessageDialog(this,
                                        "Please select a file.",
                                        "No file selected",
                                        JOptionPane.WARNING_MESSAGE);
        }
      }
    }
  } //end jMenuItemRemoveFile_actionPerformed

  /**
   * Association | Load Data File | Data Structure File action performed.
   * <p>
   * Calls the JFileChooser for user to pick a file, and then calls the necessary
   * methods to load the file as the information structure. <br>
   * The steps taken are:
   * <ul>
   * <li>Sets up the jFileChooser (set title anf file filter) and calls it for the user to
choose a file. </li>
```

```
   * <li>Loads the file into a JDOM Document with loadXML method.</li>
   * <li>Loads the loaded Document to the data structure by calling
   *      method loadDatasStruct. </li>
   * <li>updates the gui: updates related Jtree, statusbar, notifies GUIUpdater of the
   *     action that was performed </li>
   * <li>removes the file filter used from jfilechooser</li>
   * </ul>
   * </p>
   *
   * @param e ActionEvent
   * @see loadXML(File)
   * @see loadDataStruct(Document)
   */
  void jMenuItemLoadDataStruct_actionPerformed(ActionEvent e) {

    File file = null;
    //Use a fileChooser to choose the file. Use the OPEN version of the dialog, test return for
Approve/Cancel
    jFileChooser1.setDialogTitle("Load a Data Structure file");
    jFileChooser1.addChoosableFileFilter(dstFileFilter);
    if (JFileChooser.APPROVE_OPTION == jFileChooser1.showOpenDialog(this)) {
      //Get the file
      file = jFileChooser1.getSelectedFile();
      //test if it is the proper file type
      if (jFileChooser1.accept(file)) {
        //load document from xml
        Document loadedDoc = loadXML(file);
        //load document to data structure tree
        loadDataStruct(loadedDoc);
        //update Jtree GUI
        updateJTreeGUI(jTree1);
        statusBar.setText("Loaded: " + jFileChooser1.getSelectedFile().getPath());
      }
      else {
        JOptionPane.showMessageDialog(null,
                                      "Please choose a Data Structure (*.dst) file.",
                                      "Wrong file type",
                                      JOptionPane.ERROR_MESSAGE);
      }
    }
    //remove file filter from jfilechooser
    jFileChooser1.removeChoosableFileFilter(dstFileFilter);
  }

  /**
   * Association | Load Data File | Generic XML through XSLT action performed.<br>
   * Calls the XML_XSLTLoader Dialog, loads the transformed document and update Jtree GUI.
   *
   * @param e ActionEvent
   * @see XML_XSLTLoader
   */
  void jMenuItemLoadXML_XSLT_actionPerformed(ActionEvent e) {
    //Call XML_XSLTLoader Dialog
    XML_XSLTLoader xxl = new XML_XSLTLoader(this, "Load generic XML file...", true);
    //get the transformed document and
    Document doc = xxl.getTransformedDoc();
    if (doc != null) {
      //load it
      loadDataStruct(doc);
      //update Jtree GUI
      updateJTreeGUI(jTree1);
    }
  }

  /**
   * Association | Load Data File | Generic XML through JDOM Transformation
   * action performed.<br>
   * Calls the XML_JDOMTransformLoader Dialog, loads the transformed document and update Jtree
GUI.
   *
   * @param e ActionEvent
   * @see XML_JDOMTransformLoader
   */
  void jMenuItemLoadXML_jdomTransform_actionPerformed(ActionEvent e) {
    //Call the XML_JDOMTransformLoader
    XML_JDOMTransformLoader xjtl = new XML_JDOMTransformLoader(this,
        "Load generic XML file...", true);
```

```
    //get the transformed document and
    Document doc = xjtl.getTransformedDoc();
    if (doc != null) {
      //load it
      loadDataStruct(doc);
      //update Jtree GUI
      updateJTreeGUI(jTree1);
    }
  }

  /**
   * Association | Load Data File | Generic XML action performed.<br>
   * <p>
   * <ul>
   * <li>Sets up the jFileChooser (set title anf file filter) and calls it for
   * the user to choose a file. </li>
   * <li>Loads the file into a JDOM Document with loadXML method.</li>
   * <li>Loads the loaded Document to the data structure by calling
   *      method loadDatasStruct(Document, File), so as to put name and type to
   *      generic xml. </li>
   * <li>updates the gui: updates related Jtree, statusbar, notifies GUIUpdater of the
   *     action that was performed </li>
   * <li>removes the file filter used from jfilechooser</li>
   * </ul>
   * </p>
   *
   * @param e ActionEvent
   */
  void jMenuItemLoadXml_actionPerformed(ActionEvent e) {
    File file = null;
    //Use a fileChooser to choose the file.  Use the OPEN version of the dialog, test return for
Approve/Cancel
    jFileChooser1.setDialogTitle("Load XML file");
    jFileChooser1.addChoosableFileFilter(xmlFileFilter);
    if (JFileChooser.APPROVE_OPTION == jFileChooser1.showOpenDialog(this)) {
      // Get the file
      file = jFileChooser1.getSelectedFile();
      //test if it is the proper file type
      if (jFileChooser1.accept(file)) {
        //load the document from the file
        Document loadedDoc = loadXML(file);
        //call overloaded version in order to put name and type to generic xml
        loadDataStruct(loadedDoc, file);
        //update Jtree GUI
        updateJTreeGUI(jTree1);
        statusBar.setText("Loaded: " + jFileChooser1.getSelectedFile().getPath());
      }
      else {
        JOptionPane.showMessageDialog(null,
                                      "Please choose a XML (*.xml) file.",
                                      "Wrong file type",
                                      JOptionPane.ERROR_MESSAGE);
      }
    }
    //remove file filter
    jFileChooser1.removeChoosableFileFilter(xmlFileFilter);
  } //end jMenuItemLoadXml_actionPerformed

  /**
   * File | View Report on Information Structure action performed.<br>
   * Calls ReportViewer Dialog
   *
   * @param e ActionEvent
   * @see ReportViewer
   */
  void jMenuItemReport_actionPerformed(ActionEvent e) {
    //Call ReportViewer Dialog
    ReportViewer rv = new ReportViewer(this,
                                       "View Report on Information Structure", false,
                                       infoStructDoc, true);
  }

  /**
   * File | Save Association action performed.
   *
   * <p> Save cases:
   * <ul>
```

```
 * <li>It is the first time that it is saved:
 * <ul>
 * <li>create a file with name = associationName and set filechooser with it.
 * When user creates an new association, he/she is prompted to provide an
 * Association name. That name is stored so as to be used here. But in order
 * for the filechooser to prompt the user with the predifed association name,
 * a file with that name must be created to set as selected file in the
 * filechooser.</li>
 * <li>Use a fileChooser to choose the file name and location. Set up
 * filechooser</li>
 * <li><save it as file by calling fileOutputXML method</li>
 * <li>update the guiupdater</li>
 * <li>if cancel on filechooser then restore savefile initial status. Because
 * savefile == null indicates that the file is never saved, and thus the next
 * time will be the first time that it will be saved.</li>
 * <li><reset jFileChooser/li>
 * </ul>
 * <li>it is been saved before so the file is already created</li>
 * <ul>
 * <li><save it as file by calling fileOutputXML method</li>
 * <li>update the guiupdater</li>
 * </ul>
 * </ul>
 * </p>
 *
 * @param e ActionEvent
 * @see fileOutputXML(Document, File)
 */
void jMenuItemFileSaveAssoc_actionPerformed(ActionEvent e) {

  //it is the first time that it is saved, so choose a file
  if (savefile == null) {
    //create file with the finame= assotionName, and set filechooser with it
    File parent = null;
    savefile = new File(parent, associationName);
    jFileChooser1.setSelectedFile(savefile);
    //Use a fileChooser to choose the file name and location. Set up filechooser
    jFileChooser1.setDialogTitle("Save Association file");
    jFileChooser1.addChoosableFileFilter(ascFileFilter);
    int n = jFileChooser1.showSaveDialog(this);
    if (n == JFileChooser.APPROVE_OPTION) {
      //Get file
      savefile = jFileChooser1.getSelectedFile();
      //file name
      associationName = savefile.getName();
      //save it
      fileOutputXML(infoStructDoc, savefile);
      //update the guiupdater
      guiUpdater.update(e.getActionCommand());
    }
    //if cancel then restore savefile initial status
    else if (n == JFileChooser.CANCEL_OPTION) {
      savefile = null;
    }
    //reset jFileChooser
    jFileChooser1.removeChoosableFileFilter(ascFileFilter);
    jFileChooser1.setSelectedFile(null);

  }
  //it is been saved before so I have the file
  else {
    //save it
    fileOutputXML(infoStructDoc, savefile);
    //update the guiupdater
    guiUpdater.update(e.getActionCommand());
  }
}

/**
 * Writes the JDOM document to a XML file.
 * <p>
 * <ul>
 * <li>Creatse a XMLOutputter which uses a PrettyFormat. </li>
 * <li>Outputs document to the file</li>
 * </p>
 * <p>
 * This is the only method of the application that outputs XML documents to file.
```

```
 * It is declared public because it is also used by ReportViewer class.
 * </p>
 * @param doc - the document to be written to file.
 * @param file - the file in which the document will be written.
 * @see org.jdom.output.Format.getPrettyFormat()
 */
public void fileOutputXML(Document doc, File file) {
  //create a XMLOutputter which uses a PrettyFormat.
  XMLOutputter serializer = new XMLOutputter(Format.getPrettyFormat());
  //output document to the file
  try {
    FileOutputStream output = new FileOutputStream(file);
    serializer.output(doc, output);
    output.close();
  }
  catch (IOException ioex) {
    ioex.printStackTrace();
  }
}

/**
 * File | View XML of Information Structure action performed. <br>
 * Calls the ReportViewer Dialog.
 *
 * @param e ActionEvent
 * @see ReportViewer
 */
void jMenuItemInfoStructXML_actionPerformed(ActionEvent e) {
  //calls the ReportViewer Dialog
  ReportViewer rvx = new ReportViewer(this,
                                      "View XML of Information Structure", false,
                                      infoStructDoc, false);
}

/**
 * File | View XML of selection in Data Structure action performed.
 * <p>
 * <ul>
 * <li>Gets selection path.</li>
 * <li>Gets selected object.</li>
 * <li>If selection object is an Element. </li>
 * <ul>
 * <li>Clone the selected element </li>
 * <li>Creates a dummy document with clone as its root. </li>
 * <li>Calls ReportViewer passing to it the dummy document. The <i>dummy document</i>
 *     is created because ReportViewer manipulates only documents and not
 *     elements. </li>
 * </ul>
 * <li>Error cases: when selected object is not an element or user has not
 * selected anything.</li>
 * </ul>
 * </p>
 * @param e ActionEvent
 * @see ReportViewer
 */
void jMenuItemSelectedXML_actionPerformed(ActionEvent e) {
  TreePath path = null;

  //Get seletion path.
  path = jTree1.getSelectionModel().getSelectionPath();
  if (path != null) {
    //Get selected object
    Object o = path.getLastPathComponent();
    if (o instanceof Element) { //if selection is an Element
      Element el = (Element) o;
      //clone selected element
      Element clone = (Element) el.clone();
      //create dummy doc to pass it to the ReportVIewer
      Document dummyDoc = new Document(clone);
      //Call ReportVIewer with dummyDoc
      ReportViewer rvs = new ReportViewer(this,
                                          "View XML of selected subtree", false,
                                          dummyDoc, false);
    }
    //error cases
    else { //no element selected
      JOptionPane.showMessageDialog(this,
```

```
                                    "Please select an Element node in Data Structure.",
                                    "No Element is selected",
                                    JOptionPane.WARNING_MESSAGE);
      }
    }
    else { //nothing is selected
      JOptionPane.showMessageDialog(this,
                                    "Please select an Element node in Data Structure.",
                                    "No Selection",
                                    JOptionPane.WARNING_MESSAGE);
    }
  }

  /**
   * Association | Save Data Source Configuration action performed.
   * <ul>
   * <li>Set up file chooser. </li>
   * <li>Get file from file chooser. </li>
   * <li>Save data source configuration document to file by calling fileOutoutXML
   *     method. </li>
   * <li>Reset file chooser. </li>
   * </ul>
   * @param e ActionEvent
   * @see fileOutputXML(Document, File)
   */
  void jMenuItemSaveDataSrcConfig_actionPerformed(ActionEvent e) {
    //Use a fileChooser to choose the file
    File file = null;
    // Set up file chooser. Use the OPEN version of the dialog, test return for Approve/Cancel
    jFileChooser1.setDialogTitle("Save Data Source Configuration");
    jFileChooser1.addChoosableFileFilter(dscFileFilter);
    if (JFileChooser.APPROVE_OPTION == jFileChooser1.showSaveDialog(this)) {
      //get the file
      file = jFileChooser1.getSelectedFile();
      //save data source configuration document to file
      fileOutputXML(dataSrcConfig, file);
    }
    //reset file chooser
    jFileChooser1.removeChoosableFileFilter(dscFileFilter);

  }

  /**
   * Association | Save Data Structure action performed.
   *
   * <p> Saves only data structured files loaded. Data sources displayed in
   * the tree can not be saved as files.
   * <ul>
   * <li>Gets the selection path and the selected object.</li>
   * <li>Check the object to be an element. If it is:</li>
   * <ul>
   * <li>Checks to see if parent of the selection has a name. If not user has
   * probably selected the node "No Data Structures Loaded", so display
   * message that no data structures are loaded and exit method. </li>
   * <li>Checks if it is a data source. If yes, show relative error message
   * and exit method. Because data sources cannot be saved.</li>
   * <li>Check if the element is a direct child of the root of the data
   * structure tree. If it is not, this element and its content does not
   * represent a data structure but a subtree of the data structure. That is
   * the element is somewhere inside the content of the data structure and
   * does not represent the root od the data structure. A subtree of the data
   * structure cannot be saved. So, show error message and exit method. </li>
   * <li>If nothing of the above cases occured then it is a data structure:
   * </li>
   * <ul>
   * <li>Clone the content because the element cannot have two parents. </li>
   * <li>Put it in a dummy doc because fileOutputXML is called with a file and
   * a document. </li>
   * <li>Set up file chooser to find the file that the document will be
   * saved. </li>
   * <li>Get file from file chooser </li>
   * <li>Save document to file by calling fileOutputXML method </li>
   * <li>Reset file chooser </li>
   * </ul>
   * </ul>
   * <li>Error cases: No Element is selected or No Selection </li>
   * </ul> </p>
```

```
   *
   * @see fileOutputXML(Document, File)
   * @param e ActionEvent
   */
  void jMenuItemSaveDataStruct_actionPerformed(ActionEvent e) {
    File file = null;
    TreePath path = null;
    //Get selection path.
    path = jTree1.getSelectionModel().getSelectionPath();
    if (path != null) {
      //get selected object
      Object o = path.getLastPathComponent();
      //must be an element
      if (o instanceof Element) {
        Element el = (Element) o;
        //check if a datastucture is loaded. If not exit method.
        Element parent = el.getParentElement();
        String parentName = null;
        try {
          parentName = parent.getName();
        }
        catch (Exception ex) {
          JOptionPane.showMessageDialog(this,
                                        "You must first load a data structure...",
                                        "No Data Structure Loaded",
                                        JOptionPane.WARNING_MESSAGE);

          return;
        }
        //if it is a data source, do not save it and show error message and exit method.
        String elName = el.getName();
        if (elName.equals("dataStructureSource")) {
          JOptionPane.showMessageDialog(this,
                                        "Data Source structure cannot be saved as xml...",
                                        "Unsupported action",
                                        JOptionPane.WARNING_MESSAGE);
          return;
        }
        //if it selected element has not root of the document as its parent is
        //not a datastructure file but a node in the tree. Show error message and exit method.
        else if (!parentName.equals("dataStructure")) {
          JOptionPane.showMessageDialog(this,
                                        "Only file structures can be saved.\nNot their
subtrees...",
                                        "Unsupported action",
                                        JOptionPane.WARNING_MESSAGE);
          return;

        }
        //if not of the above, then it is a datastructure file, so save it
        //clone it, because it cannot have 2 parents
        Element clone = (Element) el.clone();
        //put it in a dummy doc
        Element root = new Element("dataStructure");
        root.setAttribute("type", "None");
        root.addContent(clone);
        Document dummyDoc = new Document(root);
        //Set up file chooser to get find the file that the document will be saved.
        jFileChooser1.setDialogTitle("Save file");
        jFileChooser1.addChoosableFileFilter(dstFileFilter);
        if (JFileChooser.APPROVE_OPTION == jFileChooser1.showSaveDialog(this)) {
          // get the file
          file = jFileChooser1.getSelectedFile();
          //save it
          fileOutputXML(dummyDoc, file);
        }
        //reset file chooser
        jFileChooser1.removeChoosableFileFilter(dstFileFilter);
      }
      //error cases
      else {
        JOptionPane.showMessageDialog(this,
                                      "Please select an Element node in Data Structure.",
                                      "No Element is selected",
                                      JOptionPane.WARNING_MESSAGE);
      }
    }
```

```
    else {
      JOptionPane.showMessageDialog(this,
                                    "Please select an Element node in Data Structure.",
                                    "No Selection",
                                    JOptionPane.WARNING_MESSAGE);
    }
  }

  /**
   * File | Close Association action performed.
   * <p>
   * Removes all content from information structure tree. But before removing, first
   * checks to see if data is modified, if save is needed etc.<br>
   * <i>Removing all content</i> means:  clear information structure, update GUIUpdater
   * and update JTree GUI. That means clear information structure from data (the actual
   * removing of data), notify GUIUpdater taht data is cleared and refresh JTree
   * appearance.<br>
   * Here the term <i>data</i> refers to the XML document heldd in Information
   * structure tree.
   * <ul>
   * <li>Data is modified, so prompt user to save data:</li>
   *  <ul>
   *   <li>if cancel: Do nothing  and exit method.</li>
   *   <li>if no: Remove all content, re-intialize file variables</li>
   *   <li>if yes: save, by delegating save action to File | Save Association
   *       action performed. Check again because user might hit yes or cancel on
   *       filechooser. </li>
   *       <ul>
   *         <li>if user hit yes in file chooser: Remove all content, re-intialize file
variables</li>
   *         <li>if user hit no in file chooser: do nothing and exit</li>
   *       </ul>
   *  </ul>
   * <li>Dara is not modified:</li>
   *  <ul>
   *   <li>Remove all content. re-intialize file variables</li>
   *  </ul>
   * </ul>
   * File variables represented the state of the data.
   *<p>
   *
   * @param e ActionEvent
   * @see jMenuItemFileSaveAssoc_actionPerformed(ActionEvent )
   * @see clearInfoStruct()
   * @see GUIUpdater
   * @see updateJTreeGUI(JTree)
   */
  void jMenuItemFileCloseAssoc_actionPerformed(ActionEvent e) {
    //if data is modified.
    if (guiUpdater.getModified()) {
      //prompt user to save data
      int n;
      Object[] options = {
          "Yes, save",
          "No, discard",
          "Cancel"};
      n = JOptionPane.showOptionDialog(null, "Save changes?", "Save?",
                                       JOptionPane.YES_NO_CANCEL_OPTION,
                                       JOptionPane.QUESTION_MESSAGE,
                                       null,
                                       options,
                                       options[2]);
      //if cancel
      if (n == JOptionPane.CANCEL_OPTION) {
        //if called from file | exit action performed, then do not allow app to exit
        //do nothing and exit
        return;
      }
      //if no
      else if (n == JOptionPane.NO_OPTION) {
        //clear information structure
        clearInfoStruct();
        //update guiupdater
        guiUpdater.update(e.getActionCommand());
        //update Jtree GUI
        updateJTreeGUI(jTree2);
```

```
      //re intialize file vars becasue the represented the state of the data
      savefile = null;
      associationName = null;
      infoStructFile = null;
    }
    //if yes
    else if (n == JOptionPane.YES_OPTION) {
      //save, by delegating save action to File | Save Association action performed.
      jMenuItemFileSaveAssoc_actionPerformed(e);
      //check again because user might hit cancel on filechooser, so
      //if not null, user hit yes on filechooser
      if (savefile != null) {
        //clear inforation structure
        clearInfoStruct();
        //update guiupdater for close action
        guiUpdater.update(e.getActionCommand());
        //update Jtree GUI
        updateJTreeGUI(jTree2);
        //re intialize file vars becasue the represented the state of the data
        savefile = null;
        associationName = null;
        infoStructFile = null;
      }
      //else user hit cancel on file chooser
      else {
        //do nothing and exit
        return;
      }
    }
    else { //should never be executed
      System.err.println(
          "ERROR in jMenuItemFileCloseAssoc_actionPerformed - Unsupported case... ");
    }
  }
  //data is not moified
  else {
    //clear information structure
    clearInfoStruct();
    //update guiupdater
    guiUpdater.update(e.getActionCommand());
    //update Jtree GUI
    updateJTreeGUI(jTree2);
    //re intialize file vars becasue the represented the state of the data
    savefile = null;
    associationName = null;
    infoStructFile = null;
 }
}

/**
 * Removes all content of Information Structure tree.
 * <ul>
 * <li>Remove everithing from Information structure tree, by delegating the
 *     the remove action to its tree model. That means calling JDOMTreeModel's
 *     removeAll method.</li>
 * <li>Remove type attribute </li>
 * </ul>
 *
 * @see JDOMTreeModel.removeAll(String)
 *
 */
private void clearInfoStruct() {
  //Remove everything from infoStruct Structure --
  treeModel2.removeAll("all");
  //removing attribute
  infoStructRoot.removeAttribute("type");
}

/**
 * File | New Association action performed.
 * <p>
 * Creates a new associaion. Asks user for the name of the new associon and
 * with the input initializes the necessary variable.
 * <ul>
 * <li>Ask user for association name.</li>
 * <li>if user enter a appropriate name: initialize association name variable
 *     and update guiupdater</li>
```

```
 * <li>if user enters an inappropriate name, that is an empty string, show error message.</li>
 * <li>if user hits escape key, show error message.</li>
 * </ul>
 * An <i>appropriate </i> name is any string except "".
 * </p>
 * @param e ActionEvent
 */
void jMenuItemFileNewAssoc_actionPerformed(ActionEvent e) {
  associationName = (String) JOptionPane.showInputDialog(
      this,
      "Please provide a name for the association: ",
      "New Association name",
      JOptionPane.QUESTION_MESSAGE,
      null,
      null,
      "Association1");

  //If a string was returned, say so.
  if (associationName != null) {
    if (associationName.length() > 0) {
      associationName = associationName + ".asc";
      guiUpdater.update(e.getActionCommand());
      return;
    }
    //empty string return by user
    else if (associationName.length() == 0) {
      //custom title, error icon
      JOptionPane.showMessageDialog(this,
                                    "Please provide a proper associaiton name.",
                                    "Invalid name",
                                    JOptionPane.ERROR_MESSAGE);
    }
  }
  //null -> esc pressed by user
  else {
    return;
  }
}

/**
 * File | Load Association action performed.<br>
 * <p>
 * Loads a association *.asc file.
 * <ul>
 * <li>Use the OPEN version of the file chooser, to get file. Set up file chooser. </li>
 * <li>Get file </li>
 * <li>Set association name from file name </li>
 * <li>Load XML file into a JDOM document, by calling loadXML method. </li>
 * <li>Load document in Information structure, bya calling loadInfoStruct method. </li>
 * <li>Update JTree, guiupdater and status bar</li>
 * <li>Reset file chooser. </li>
 * </ul>
 * </p>
 *
 * @param e ActionEvent
 * @see loadXML(File)
 * @see loadInfoStruct(Document)
 */
void jMenuItemFileLoadAssoc_actionPerformed(ActionEvent e) {
  // Use the Open version of the file chooser, to get file. Set up file chooser.
  jFileChooser1.setDialogTitle("Choose an Association File");
  jFileChooser1.addChoosableFileFilter(ascFileFilter);
  if (JFileChooser.APPROVE_OPTION == jFileChooser1.showOpenDialog(this)) {
    // get file
    savefile = jFileChooser1.getSelectedFile();
    //test if it is the proper file type
    if (jFileChooser1.accept(savefile)) {
      //set association name from file name
      associationName = savefile.getName();
      //load XML file into a JDOM document
      Document doc = loadXML(savefile);
      //load document in Information structure
      loadInfoStruct(doc);

      //update Jtree GUI
      updateJTreeGUI(jTree2);
      //update GUIUpdater
```

```
        guiUpdater.update(e.getActionCommand());

        statusBar.setText("Loaded: " + jFileChooser1.getSelectedFile().getPath());
      }
      else {
        JOptionPane.showMessageDialog(null,
                                      "Please choose a Association (*.asc) file.",
                                      "Wrong file type",
                                      JOptionPane.ERROR_MESSAGE);
      }
    }
    //reset file chooser
    jFileChooser1.removeChoosableFileFilter(ascFileFilter);
  }

  /**
   * When a class (in our case JDOMTreeModel) modifies the data (in our case the
   * Information structure tree), calls this method to inform that data are modified
   * (and need to be saved before exiting).<br>
   * The method just delegates the update of the status of data to the GUIUpdater.
   * Through the implementation of this method the DataModifier interface is implemented.
   *
   * @see DataModifier
   * @see GUIUpdater
   */
  public final void dataModified() {
    //upadate guiupadter
    guiUpdater.update(GUIUpdater.DATA_MODIFIER);
  }

  /**
   *
   * GUIUpdater is s class responsible for udating the menus items and buttons
   * of AdminToolFrame.
   * <p>
   * In order to enable or disable the AdminToolFrame's GUI components, it must always
   * know the status of the application according to whether or not an associaiton
   * is created, a  data is saved after modification etc.
   * To do that it accepts specific commands from the AdminToolFrame, which are predefined
   * as static public variables. The status is held in boolean variables. <br>
   * </p>
   * The String of static variables coresponds to the menu items' command text,
   * except of course from DATA_MODIFIER. So if it is changed in future releases, please update
the static variables.
   * <p>
   * The term <i>GUI</i> here refers to menus items and buttons
   * of AdminToolFrame only.
   * </p>
   *
   */
  private class GUIUpdater {
    //status variables
    /**Status variable indicating whether or not an association is created.*/
    private boolean associationCreated = false;

    /**Status variable indicating whether or not data (XML document hold in information
     * structure tree) is modified (and needs to be saved). */
    private boolean modified = false;

    //supported commands
    /**Command: New Association.*/
    public static final String NEW = "New Association";

    /**Command: Load Association.*/
    public static final String LOAD = "Load Association";

    /**Command: Save Association.*/
    public static final String SAVE = "Save Association";

    /**Command: Close Association.*/
    public static final String CLOSE = "Close Association";

    /**
     * Command: DATA_MODIFIER.
     * It is not a menu item command text.
     * DATA_MODIFIER corresponds to every action (Drag and Drop, Information Field Propertes,
Remove a assocition)
```

```
 * that can modify the Information Structure TreeModel. It is used basically from methods
 * methods of JDOMTreeModel, because all modifications to the information
 * structure tree is done through them.
 */
public static final String DATA_MODIFIER = "dataModifier";

/**Command: Load Information Structure. After loading an information structure
 * data are considered modified.*/
public static final String LOAD_INFO_STRUCT = "Load Information Structure";

/**
 * Constructor.
 * Sets the initial status of the GUI, which is the same as when
 * a ssociation is closed.
 *
 * @see processClose()
 */
GUIUpdater() {
  //set the initial status of GUI
  processClose();
}

/**
 * Updates the status of GUIUpdater.<br>
 * By calling this method with the proper command string (predefined in static variables),
 * a class can modify the status of GUIUpdater.
 * <p>
 * Decides on action that must be taken and calls appropriate
 * helper private method (processXXX() method) to do the updates.<br>
 * Cases of actions:
 * <ul>
 * <li>Initial status or when association is closed, that is no association is
 * created and of thus not modified. </li>
 * <ul>
 * <li>User enters command: New Associaion</li>
 * <li>User enters command: Load Association</li>
 * </ul>
 * <li>Associaion is created but not modified yet. </li>
 * <ul>
 * <li>User enters command: Close Association</li>
 * <li>Data is modified.</li>
 * </ul>
 * <li>Association is created and it is modified </li>
 * <ul>
 * <li>User enters command: Save Association</li>
 * <li>User enters command: Close Association</li>
 * <li>User enters command: Load an Information Structure</li>
 * </ul>
 * </ul>
 * </p>
 * <p>
 * <i>Note: it must "return;" in every case because status variables are modified. </i>
 * </p>
 * @param command String passed by caller
 *
 * @see processNew()
 * @see processLoad()
 * @see processClose()
 * @see processModifier()
 * @see processSave()
 * @see processLoadInfoStruct()
 */
void update(String command) {
  //Initial status or when association is closed, that is no association is
  //created and of thus not modified
  if ( (!associationCreated) && (!modified)) {
    if (command.equals(NEW)) {
      processNew();
    }
    else if (command.equals(LOAD)) {
      processLoad();
    }
    return; //on purpose
  }
  //Associaion is created but not modified yet
  else if ( (associationCreated) && (!modified)) {
    if (command.equals(CLOSE)) {
```

```
      processClose();
    }
    else if (command.equals(DATA_MODIFIER)) {
      processModifier();
    }
    return; //on purpose
  }
  //Association is created and it is modified
  else if ( (associationCreated) && (modified)) {
    if (command.equals(SAVE)) {
      processSave();
    }
    else if (command.equals(CLOSE)) {
      processClose();
    }
    else if (command.equals(LOAD_INFO_STRUCT)) {
      processLoadInfoStruct();
    }
    return; //on purpose
  }
  //Should never be executed
  else {
    System.err.println("ERROR in GUIUpdater - Unexpected state!");
    return; //on purpose
  }
} //end method update

/**
 * Processes the necessary modifications on GUI when there is a Load
 * Information Structure command.<br>
 * It disables the menu item and button for loading an Information Structure
 * in order to prevent user of loading a second information structure.
 *
 */
private void processLoadInfoStruct() {
  jLabelInfoStruct.setText("Information Structure: " +
                           infoStructFile.getName());
  //do not load a second info struct
  jMenuItemLoadInfoStruct.setEnabled(false);
  jButtonLoadIS.setEnabled(false);
}

/**
 * Processes the necessary modifications on GUI when there is a Save Association
 * command.<br>
 * Also, it change the status of GUIUpdater into "Data not modified."
 */
private void processSave() {
  //change status vars
  modified = false;

  jMenuItemFileSaveAssoc.setEnabled(false);
  jButtonSaveAssoc.setEnabled(false);
  //set frame's title
  setTitle("Administration Tool" + " - " + savefile.getName());

  jLabelInfoStruct.setText("Information Structure: " + savefile.getName());
}

/**
 * Processes the necessary modifications on GUI when there is a modification occured on the
 * the data (Command: DATA_MODIFIER).<br>
 * Also, it change the status of GUIUpdater into "Data modified."
 *
 */
private void processModifier() {
  //change status vars
  modified = true;

  jMenuItemFileSaveAssoc.setEnabled(true);
  jButtonSaveAssoc.setEnabled(true);
}

/**
 * Processes the necessary modifications on GUI when there is a Close Association
 * command.<br>
 * Also, it change the status of GUIUpdater into "Data not modified" and
```

```
 * "Association not created".
 *
 */
private void processClose() {

  //change status vars
  associationCreated = false;
  modified = false;

  //File menu
  jMenuItemFileNewAssoc.setEnabled(true);
  jMenuItemFileLoadAssoc.setEnabled(true);
  jMenuItemFileSaveAssoc.setEnabled(false);
  jMenuItemFileCloseAssoc.setEnabled(false);

  jMenuItemReport.setEnabled(false);
  jMenuItemInfoStructXML.setEnabled(false);
  jMenuItemSelectedXML.setEnabled(false);

  //Association Menu
  jMenuItemLoadInfoStruct.setEnabled(false);

  //Pop menu
  jMenuItemInfoFieldProps.setEnabled(false);
  jMenuItemRemoveFile.setEnabled(false);
  jMenuItemRemoveAll.setEnabled(false);

  setTitle("Administration Tool");

  jButtonNewAssoc.setEnabled(true);
  jButtonLoadAssoc.setEnabled(true);
  jButtonSaveAssoc.setEnabled(false);
  jButtonCloseAssoc.setEnabled(false);
  jButtonLoadIS.setEnabled(false);

  jLabelInfoStruct.setText("Information Structure");
}

/**
 * Processes the necessary modifications on GUI when there is a Load Association
 * command.<br>
 * Delegates the action to processNew() and then makes the  modifications
 * that are different between New and Load.
 *
 * @see processNew()
 */
private void processLoad() {
  //deledate action to processNew
  processNew();
  //make the  modifications that are different between New and Load
  jMenuItemLoadInfoStruct.setEnabled(false);
  jButtonLoadIS.setEnabled(false);
}

/**
 * Processes the necessary modifications on GUI when there is a New Association
 * command.<br>
 * Also, it change the status of GUIUpdater into "Association created".
 *
 */
private void processNew() {

  //change status vars
  associationCreated = true;

  //File menu
  jMenuItemFileNewAssoc.setEnabled(false);
  jMenuItemFileLoadAssoc.setEnabled(false);
  jMenuItemFileSaveAssoc.setEnabled(false);
  jMenuItemFileCloseAssoc.setEnabled(true);

  jMenuItemReport.setEnabled(true);
  jMenuItemInfoStructXML.setEnabled(true);
  jMenuItemSelectedXML.setEnabled(true);

  //Association Menu
  jMenuItemLoadInfoStruct.setEnabled(true);
```

```
      //Pop Menu
      jMenuItemInfoFieldProps.setEnabled(true);
      jMenuItemRemoveFile.setEnabled(true);
      jMenuItemRemoveAll.setEnabled(true);

      setTitle("Administration Tool" + " - " + associationName.toString());

      jButtonNewAssoc.setEnabled(false);
      jButtonLoadAssoc.setEnabled(false);
      jButtonSaveAssoc.setEnabled(false);
      jButtonCloseAssoc.setEnabled(true);
      jButtonLoadIS.setEnabled(true);
    }

    /**
     * Tells whether or not data is modified, according to GUIUpdater.<br>
     * Used in AdminToolFrame class when a Association is closed.
     * @return boolean - whether or not data is modified.
     *
     * @see AdminToolFrame.jMenuItemFileCloseAssoc_actionPerformed(ActionEvent)
     */
    public final boolean getModified() {
      return modified;
    }

  } //end GUIUpdater

  /**
   * Pop up | RemoveAssociation action performed.
   * <p>
   * <ul>
   * <li>Gets selection path required for callinf removeNodeFromParent method.</li>
   * <li>Gets selected object. </li>
   * <li>Checks to see if it is an association. Association are Text nodes in the JDOM document
   * of Information structure. </li>
   * <li>Delegate remove action to tree model removeFromParent method. </li>
   * <li>Error cases: Not an association </li>
   * </ul>
   *  The reason for delegating the removal of a node to the tree model is the
   * same as in in loadInfoStruct and loadDataStruct methods.
   * </p>
   * @param e ActionEvent
   * @see JDOMTreeModel
   * @see loadInfoStruct(Document)
   * @see loadDataStruct(Document)
   */
  void jMenuItemRemoveAssoc_actionPerformed(ActionEvent e) {
    TreePath path = null;

    //Get selection path
    path = selectionModel2.getSelectionPath();
    // If you are just collapsing the tree, you may not have a new path
    if (path != null) {
      //Get selected object
      Object selectedObject = (Object) path.getLastPathComponent(); //Returns the last component
of this path.
      //Check to see if it is an association. Association are Text nodes in the JDOM document.
      if (selectedObject instanceof Text) {
        Text text = (Text) selectedObject;
        //remove it by delegating action to tree model.
        treeModel2.removeNodeFromParent(text, path.getParentPath());
      }
      //not a Association
      else {
        JOptionPane.showMessageDialog(null,
                                      "Only association fields can be removed!",
                                      "Error in removing",
                                      JOptionPane.ERROR_MESSAGE);
      }
    }
  }

  /**
   * Pop up | Remove All action performed.
   * Removes all content from Data Structure tree.
   * <p>
```

```
 * <ul>
 * <li>Clears data structure tree, by delegating the calls to tree model's
 * removeAll method. </li>
 * <li>Clears data source configuration document. It must be cleared to
 * show that no data sources are loaded, because data structure tree is cleared
 * from its loaded data sources too. </li>
 * </ul>
 * The reason for delegating the removal of a node to the tree model is the
 * same as in in loadInfoStruct and loadDataStruct methods.
 * </p>
 * @param e ActionEvent
 * @see JDOMTreeModel
 * @see loadInfoStruct(Document)
 * @see loadDataStruct(Document)
 */
void jMenuItemRemoveAll_actionPerformed(ActionEvent e) {
  //clear data structure tree. String "all" means ALL nodes.
  treeModel1.removeAll("all");
  //clear data source config
  dataSrcConfig.getRootElement().removeContent();
}

/**
 * Toolbar | New Associaion button action performed. <br>
 * Delegates the call to the corresponding menu item's action performed method.
 * @param e ActionEvent
 * @see jMenuItemFileNewAssoc_actionPerformed(ActionEvent)
 */
void jButtonNewAssoc_actionPerformed(ActionEvent e) {
  jMenuItemFileNewAssoc_actionPerformed(e);
}

/**
 * Toolbar | Load Associaion button action performed. <br>
 * Delegates the call to the corresponding menu item's action performed method.
 * @param e ActionEvent
 * @see jMenuItemFileLoadAssoc_actionPerformed(ActionEvent)
 */
void jButtonLoadAssoc_actionPerformed(ActionEvent e) {
  jMenuItemFileLoadAssoc_actionPerformed(e);
}

/**
 * Toolbar | Save Associaion button action performed. <br>
 * Delegates the call to the corresponding menu item's action performed method.
 * @param e ActionEvent
 * @see jMenuItemFileSaveAssoc_actionPerformed(ActionEvent)
 */
void jButtonSaveAssoc_actionPerformed(ActionEvent e) {
  jMenuItemFileSaveAssoc_actionPerformed(e);
}

/**
 * Toolbar | Close Associaion button action performed. <br>
 * Delegates the call to the corresponding menu item's action performed method.
 * @param e ActionEvent
 * @see jMenuItemFileCloseAssoc_actionPerformed(ActionEvent)
 */
void jButtonCloseAssoc_actionPerformed(ActionEvent e) {
  jMenuItemFileCloseAssoc_actionPerformed(e);
}

/**
 * Toolbar | Load Information Structure button action performed. <br>
 * Delegates the call to the corresponding menu item's action performed method.
 * @param e ActionEvent
 * @see jMenuItemLoadInfoStruct_actionPerformed(ActionEvent)
 */
void jButtonLoadIS_actionPerformed(ActionEvent e) {
  jMenuItemLoadInfoStruct_actionPerformed(e);
}

/**
 * Toolbar | Save Data Source Configuration button action performed. <br>
 * Delegates the call to the corresponding menu item's action performed method.
 * @param e ActionEvent
 * @see jMenuItemSaveDataSrcConfig_actionPerformed(ActionEvent)
```

```
 */
void jButtonSaveDSC_actionPerformed(ActionEvent e) {
  jMenuItemSaveDataSrcConfig_actionPerformed(e);
}

/**
 * Toolbar | Load Data Structure button action performed. <br>
 * Delegates the call to the corresponding menu item's action performed method.
 * @param e ActionEvent
 * @see jMenuItemLoadDataStruct_actionPerformed(ActionEvent)
 */
void jButtonLoadDST_actionPerformed(ActionEvent e) {
  jMenuItemLoadDataStruct_actionPerformed(e);
}

/**
 * Toolbar | Load XML file through XSLT transformation button action performed. <br>
 * Delegates the call to the corresponding menu item's action performed method.
 * @param e ActionEvent
 * @see jMenuItemLoadXML_XSLT_actionPerformed(ActionEvent)
 */
void jButtonLoadXMLXSLT_actionPerformed(ActionEvent e) {
  jMenuItemLoadXML_XSLT_actionPerformed(e);
}

/**
 * Toolbar | Load XML file through JDOM transformation button action performed. <br>
 * Delegates the call to the corresponding menu item's action performed method.
 * @param e ActionEvent
 * @see jMenuItemLoadXML_jdomTransform_actionPerformed(ActionEvent)
 */
void jButtonLoadXMLJDOM_actionPerformed(ActionEvent e) {
  jMenuItemLoadXML_jdomTransform_actionPerformed(e);
}

/**
 * Toolbar | Load a generic XML file button action performed. <br>
 * Delegates the call to the corresponding menu item's action performed method.
 * @param e ActionEvent
 * @see jMenuItemLoadXml_actionPerformed(ActionEvent)
 */
void jButtonLoadXML_actionPerformed(ActionEvent e) {
  jMenuItemLoadXml_actionPerformed(e);
}

/**
 * Toolbar | Save Data Structure file button action performed. <br>
 * Delegates the call to the corresponding menu item's action performed method.
 * @param e ActionEvent
 * @see jMenuItemSaveDataStruct_actionPerformed(ActionEvent)
 */
void jButtonSaveDST_actionPerformed(ActionEvent e) {
  jMenuItemSaveDataStruct_actionPerformed(e);
}

/**
 * Toolbar | Load New Data Source Configuration button action performed. <br>
 * Delegates the call to the corresponding menu item's action performed method.
 * @param e ActionEvent
 * @see jMenuItemLoadNewDataSrcConfig_actionPerformed(ActionEvent)
 */
void jButtonLoadNewDSC_actionPerformed(ActionEvent e) {
  jMenuItemLoadNewDataSrcConfig_actionPerformed(e);
}

/**
 * Toolbar | Configure Data Sources button action performed. <br>
 * Delegates the call to the corresponding menu item's action performed method.
 *@param e ActionEvent
 * @see jMenuItemConfigDataSrc_actionPerformed(ActionEvent)
 */
void jButtonConfigDSC_actionPerformed(ActionEvent e) {
  jMenuItemConfigDataSrc_actionPerformed(e);

}

/**
```

```
   * Toolbar | Load Existing Data Source  COnfiguration button action performed. <br>
   * Delegates the call to the corresponding menu item's action performed method.
   * @param e ActionEvent
   * @see jMenuItemLoadExistDataSrcConfig_actionPerformed(ActionEvent e)
   */
  void jButtonLoadExistDSC_actionPerformed(ActionEvent e) {
    jMenuItemLoadExistDataSrcConfig_actionPerformed(e);
  }

//------------  Action Adapter classes ----------------------------------------
  /**
   * Helper inner class created automatically by development tool. <br>
   * I just calls the corresponding action performed method of AdminToolFrame.
   */
  class AdminToolFrame_jMenuFileExit_ActionAdapter
      implements ActionListener {
    AdminToolFrame adaptee;

    AdminToolFrame_jMenuFileExit_ActionAdapter(AdminToolFrame adaptee) {
      this.adaptee = adaptee;
    }

    public void actionPerformed(ActionEvent e) {
      adaptee.jMenuFileExit_actionPerformed(e);
    }
  }

  /**
   * Helper inner class created automatically by development tool. <br>
   * I just calls the corresponding action performed method of AdminToolFrame.
   */
  class AdminToolFrame_jMenuHelpAbout_ActionAdapter
      implements ActionListener {
    AdminToolFrame adaptee;

    AdminToolFrame_jMenuHelpAbout_ActionAdapter(AdminToolFrame adaptee) {
      this.adaptee = adaptee;
    }

    public void actionPerformed(ActionEvent e) {
      adaptee.jMenuHelpAbout_actionPerformed(e);
    }
  }

  /**
   * Helper inner class created automatically by development tool. <br>
   * I just calls the corresponding action performed method of AdminToolFrame.
   */
  class AdminToolFrame_jButtonLoadNewDSC_actionAdapter
      implements java.awt.event.ActionListener {
    AdminToolFrame adaptee;

    AdminToolFrame_jButtonLoadNewDSC_actionAdapter(AdminToolFrame adaptee) {
      this.adaptee = adaptee;
    }

    public void actionPerformed(ActionEvent e) {
      adaptee.jButtonLoadNewDSC_actionPerformed(e);
    }
  }

  /**
   * Helper inner class created automatically by development tool. <br>
   * I just calls the corresponding action performed method of AdminToolFrame.
   */
  class AdminToolFrame_jMenuItemInfoFieldProps_actionAdapter
      implements java.awt.event.ActionListener {
    AdminToolFrame adaptee;

    AdminToolFrame_jMenuItemInfoFieldProps_actionAdapter(AdminToolFrame adaptee) {
      this.adaptee = adaptee;
    }

    public void actionPerformed(ActionEvent e) {
      adaptee.jMenuItemInfoFieldProps_actionPerformed(e);
    }
  }
```

```
/**
 * Helper inner class created automatically by development tool. <br>
 * I just calls the corresponding action performed method of AdminToolFrame.
 */
class AdminToolFrame_jButtonConfigDSC_actionAdapter
    implements java.awt.event.ActionListener {
  AdminToolFrame adaptee;

  AdminToolFrame_jButtonConfigDSC_actionAdapter(AdminToolFrame adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jButtonConfigDSC_actionPerformed(e);
  }
}

/**
 * Helper inner class created automatically by development tool. <br>
 * I just calls the corresponding action performed method of AdminToolFrame.
 */
class AdminToolFrame_jMenuItemLoadInfoStruct_actionAdapter
    implements java.awt.event.ActionListener {
  AdminToolFrame adaptee;

  AdminToolFrame_jMenuItemLoadInfoStruct_actionAdapter(AdminToolFrame
      adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jMenuItemLoadInfoStruct_actionPerformed(e);
  }
}

/**
 * Helper inner class created automatically by development tool. <br>
 * I just calls the corresponding action performed method of AdminToolFrame.
 */
class AdminToolFrame_jMenuItemLoadExistDataSrcConfig_actionAdapter
    implements java.awt.event.ActionListener {
  AdminToolFrame adaptee;

  AdminToolFrame_jMenuItemLoadExistDataSrcConfig_actionAdapter(AdminToolFrame
      adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jMenuItemLoadExistDataSrcConfig_actionPerformed(e);
  }
}

/**
 * Helper inner class created automatically by development tool. <br>
 * I just calls the corresponding action performed method of AdminToolFrame.
 */
class AdminToolFrame_jMenuItemConfigDataSrc_actionAdapter
    implements java.awt.event.ActionListener {
  AdminToolFrame adaptee;

  AdminToolFrame_jMenuItemConfigDataSrc_actionAdapter(AdminToolFrame adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jMenuItemConfigDataSrc_actionPerformed(e);
  }
}

/**
 * Helper inner class created automatically by development tool. <br>
 * I just calls the corresponding action performed method of AdminToolFrame.
 */

class AdminToolFrame_jMenuItemLoadNewDataSrcConfig_actionAdapter
```

```
    implements java.awt.event.ActionListener {
  AdminToolFrame adaptee;

  AdminToolFrame_jMenuItemLoadNewDataSrcConfig_actionAdapter(AdminToolFrame
      adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jMenuItemLoadNewDataSrcConfig_actionPerformed(e);
  }
}

/**
 * Helper inner class created automatically by development tool. <br>
 * I just calls the corresponding action performed method of AdminToolFrame.
 */

class AdminToolFrame_jButtonLoadExistDSC_actionAdapter
    implements java.awt.event.ActionListener {
  AdminToolFrame adaptee;

  AdminToolFrame_jButtonLoadExistDSC_actionAdapter(AdminToolFrame adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jButtonLoadExistDSC_actionPerformed(e);
  }
}

/**
 * Helper inner class created automatically by development tool. <br>
 * I just calls the corresponding action performed method of AdminToolFrame.
 */

class AdminToolFrame_jMenuItemRemoveFile_actionAdapter
    implements java.awt.event.ActionListener {
  AdminToolFrame adaptee;

  AdminToolFrame_jMenuItemRemoveFile_actionAdapter(AdminToolFrame adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jMenuItemRemoveFile_actionPerformed(e);
  }
}

/**
 * Helper inner class created automatically by development tool. <br>
 * I just calls the corresponding action performed method of AdminToolFrame.
 */
class AdminToolFrame_jMenuItemLoadDataStruct_actionAdapter
    implements java.awt.event.ActionListener {
  AdminToolFrame adaptee;

  AdminToolFrame_jMenuItemLoadDataStruct_actionAdapter(AdminToolFrame adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jMenuItemLoadDataStruct_actionPerformed(e);
  }
}

/**
 * Helper inner class created automatically by development tool. <br>
 * I just calls the corresponding action performed method of AdminToolFrame.
 */

class AdminToolFrame_jMenuItemLoadXML_XSLT_actionAdapter
    implements java.awt.event.ActionListener {
  AdminToolFrame adaptee;

  AdminToolFrame_jMenuItemLoadXML_XSLT_actionAdapter(AdminToolFrame adaptee) {
    this.adaptee = adaptee;
```

```
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jMenuItemLoadXML_XSLT_actionPerformed(e);
  }
}

/**
 * Helper inner class created automatically by development tool. <br>
 * I just calls the corresponding action performed method of AdminToolFrame.
 */

class AdminToolFrame_jMenuItemLoadXML_jdomTransform_actionAdapter
    implements java.awt.event.ActionListener {
  AdminToolFrame adaptee;

  AdminToolFrame_jMenuItemLoadXML_jdomTransform_actionAdapter(AdminToolFrame
      adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jMenuItemLoadXML_jdomTransform_actionPerformed(e);
  }
}

/**
 * Helper inner class created automatically by development tool. <br>
 * I just calls the corresponding action performed method of AdminToolFrame.
 */

class AdminToolFrame_jMenuItemLoadXml_actionAdapter
    implements java.awt.event.ActionListener {
  AdminToolFrame adaptee;

  AdminToolFrame_jMenuItemLoadXml_actionAdapter(AdminToolFrame adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jMenuItemLoadXml_actionPerformed(e);
  }
}

/**
 * Helper inner class created automatically by development tool. <br>
 * I just calls the corresponding action performed method of AdminToolFrame.
 */

class AdminToolFrame_jMenuItemReport_actionAdapter
    implements java.awt.event.ActionListener {
  AdminToolFrame adaptee;

  AdminToolFrame_jMenuItemReport_actionAdapter(AdminToolFrame adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jMenuItemReport_actionPerformed(e);
  }
}

/**
 * Helper inner class created automatically by development tool. <br>
 * I just calls the corresponding action performed method of AdminToolFrame.
 */

class AdminToolFrame_jMenuItemFileSaveAssoc_actionAdapter
    implements java.awt.event.ActionListener {
  AdminToolFrame adaptee;

  AdminToolFrame_jMenuItemFileSaveAssoc_actionAdapter(AdminToolFrame adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jMenuItemFileSaveAssoc_actionPerformed(e);
```

```
  }
}

/**
 * Helper inner class created automatically by development tool. <br>
 * I just calls the corresponding action performed method of AdminToolFrame.
 */

class AdminToolFrame_jMenuItemInfoStructXML_actionAdapter
    implements java.awt.event.ActionListener {
  AdminToolFrame adaptee;

  AdminToolFrame_jMenuItemInfoStructXML_actionAdapter(AdminToolFrame adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jMenuItemInfoStructXML_actionPerformed(e);
  }
}

/**
 * Helper inner class created automatically by development tool. <br>
 * I just calls the corresponding action performed method of AdminToolFrame.
 */

class AdminToolFrame_jMenuItemSelectedXML_actionAdapter
    implements java.awt.event.ActionListener {
  AdminToolFrame adaptee;

  AdminToolFrame_jMenuItemSelectedXML_actionAdapter(AdminToolFrame adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jMenuItemSelectedXML_actionPerformed(e);
  }
}

/**
 * Helper inner class created automatically by development tool. <br>
 * I just calls the corresponding action performed method of AdminToolFrame.
 */

class AdminToolFrame_jMenuItemSaveDataSrcConfig_actionAdapter
    implements java.awt.event.ActionListener {
  AdminToolFrame adaptee;

  AdminToolFrame_jMenuItemSaveDataSrcConfig_actionAdapter(AdminToolFrame
      adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jMenuItemSaveDataSrcConfig_actionPerformed(e);
  }
}

/**
 * Helper inner class created automatically by development tool. <br>
 * I just calls the corresponding action performed method of AdminToolFrame.
 */

class AdminToolFrame_jMenuItemSaveDataStruct_actionAdapter
    implements java.awt.event.ActionListener {
  AdminToolFrame adaptee;

  AdminToolFrame_jMenuItemSaveDataStruct_actionAdapter(AdminToolFrame adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jMenuItemSaveDataStruct_actionPerformed(e);
  }
}

/**
```

```
 * Helper inner class created automatically by development tool. <br>
 * I just calls the corresponding action performed method of AdminToolFrame.
 */

class AdminToolFrame_jMenuItemFileCloseAssoc_actionAdapter
    implements java.awt.event.ActionListener {
  AdminToolFrame adaptee;

  AdminToolFrame_jMenuItemFileCloseAssoc_actionAdapter(AdminToolFrame adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jMenuItemFileCloseAssoc_actionPerformed(e);
  }
}

/**
 * Helper inner class created automatically by development tool. <br>
 * I just calls the corresponding action performed method of AdminToolFrame.
 */

class AdminToolFrame_jMenuItemFileNewAssoc_actionAdapter
    implements java.awt.event.ActionListener {
  AdminToolFrame adaptee;

  AdminToolFrame_jMenuItemFileNewAssoc_actionAdapter(AdminToolFrame adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jMenuItemFileNewAssoc_actionPerformed(e);
  }
}

/**
 * Helper inner class created automatically by development tool. <br>
 * I just calls the corresponding action performed method of AdminToolFrame.
 */

class AdminToolFrame_jMenuItemFileLoadAssoc_actionAdapter
    implements java.awt.event.ActionListener {
  AdminToolFrame adaptee;

  AdminToolFrame_jMenuItemFileLoadAssoc_actionAdapter(AdminToolFrame adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jMenuItemFileLoadAssoc_actionPerformed(e);
  }
}

/**
 * Helper inner class created automatically by development tool. <br>
 * I just calls the corresponding action performed method of AdminToolFrame.
 */

class AdminToolFrame_jMenuItemRemoveAssoc_actionAdapter
    implements java.awt.event.ActionListener {
  AdminToolFrame adaptee;

  AdminToolFrame_jMenuItemRemoveAssoc_actionAdapter(AdminToolFrame adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jMenuItemRemoveAssoc_actionPerformed(e);
  }
}

/**
 * Helper inner class created automatically by development tool. <br>
 * I just calls the corresponding action performed method of AdminToolFrame.
 */

class AdminToolFrame_jButtonNewAssoc_actionAdapter
```

```
    implements java.awt.event.ActionListener {
  AdminToolFrame adaptee;

  AdminToolFrame_jButtonNewAssoc_actionAdapter(AdminToolFrame adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jButtonNewAssoc_actionPerformed(e);
  }
}

/**
 * Helper inner class created automatically by development tool. <br>
 * I just calls the corresponding action performed method of AdminToolFrame.
 */

class AdminToolFrame_jButtonLoadAssoc_actionAdapter
    implements java.awt.event.ActionListener {
  AdminToolFrame adaptee;

  AdminToolFrame_jButtonLoadAssoc_actionAdapter(AdminToolFrame adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jButtonLoadAssoc_actionPerformed(e);
  }
}

/**
 * Helper inner class created automatically by development tool. <br>
 * I just calls the corresponding action performed method of AdminToolFrame.
 */

class AdminToolFrame_jButtonSaveAssoc_actionAdapter
    implements java.awt.event.ActionListener {
  AdminToolFrame adaptee;

  AdminToolFrame_jButtonSaveAssoc_actionAdapter(AdminToolFrame adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jButtonSaveAssoc_actionPerformed(e);
  }
}

/**
 * Helper inner class created automatically by development tool. <br>
 * I just calls the corresponding action performed method of AdminToolFrame.
 */

class AdminToolFrame_jButtonCloseAssoc_actionAdapter
    implements java.awt.event.ActionListener {
  AdminToolFrame adaptee;

  AdminToolFrame_jButtonCloseAssoc_actionAdapter(AdminToolFrame adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jButtonCloseAssoc_actionPerformed(e);
  }
}

/**
 * Helper inner class created automatically by development tool. <br>
 * I just calls the corresponding action performed method of AdminToolFrame.
 */

class AdminToolFrame_jButtonLoadIS_actionAdapter
    implements java.awt.event.ActionListener {
  AdminToolFrame adaptee;

  AdminToolFrame_jButtonLoadIS_actionAdapter(AdminToolFrame adaptee) {
    this.adaptee = adaptee;
```

```
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jButtonLoadIS_actionPerformed(e);
  }
}

/**
 * Helper inner class created automatically by development tool. <br>
 * I just calls the corresponding action performed method of AdminToolFrame.
 */

class AdminToolFrame_jButtonSaveDSC_actionAdapter
    implements java.awt.event.ActionListener {
  AdminToolFrame adaptee;

  AdminToolFrame_jButtonSaveDSC_actionAdapter(AdminToolFrame adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jButtonSaveDSC_actionPerformed(e);
  }
}

/**
 * Helper inner class created automatically by development tool. <br>
 * I just calls the corresponding action performed method of AdminToolFrame.
 */

class AdminToolFrame_jButtonLoadDST_actionAdapter
    implements java.awt.event.ActionListener {
  AdminToolFrame adaptee;

  AdminToolFrame_jButtonLoadDST_actionAdapter(AdminToolFrame adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jButtonLoadDST_actionPerformed(e);
  }
}

/**
 * Helper inner class created automatically by development tool. <br>
 * I just calls the corresponding action performed method of AdminToolFrame.
 */

class AdminToolFrame_jButtonLoadXMLXSLT_actionAdapter
    implements java.awt.event.ActionListener {
  AdminToolFrame adaptee;

  AdminToolFrame_jButtonLoadXMLXSLT_actionAdapter(AdminToolFrame adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jButtonLoadXMLXSLT_actionPerformed(e);
  }
}

/**
 * Helper inner class created automatically by development tool. <br>
 * I just calls the corresponding action performed method of AdminToolFrame.
 */

class AdminToolFrame_jButtonLoadXMLJDOM_actionAdapter
    implements java.awt.event.ActionListener {
  AdminToolFrame adaptee;

  AdminToolFrame_jButtonLoadXMLJDOM_actionAdapter(AdminToolFrame adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jButtonLoadXMLJDOM_actionPerformed(e);
  }
```

```
  }

  /**
   * Helper inner class created automatically by development tool. <br>
   * I just calls the corresponding action performed method of AdminToolFrame.
   */

  class AdminToolFrame_jButtonLoadXML_actionAdapter
      implements java.awt.event.ActionListener {
    AdminToolFrame adaptee;

    AdminToolFrame_jButtonLoadXML_actionAdapter(AdminToolFrame adaptee) {
      this.adaptee = adaptee;
    }

    public void actionPerformed(ActionEvent e) {
      adaptee.jButtonLoadXML_actionPerformed(e);
    }
  }

  /**
   * Helper inner class created automatically by development tool. <br>
   * I just calls the corresponding action performed method of AdminToolFrame.
   */

  class AdminToolFrame_jButtonSaveDST_actionAdapter
      implements java.awt.event.ActionListener {
    AdminToolFrame adaptee;

    AdminToolFrame_jButtonSaveDST_actionAdapter(AdminToolFrame adaptee) {
      this.adaptee = adaptee;
    }

    public void actionPerformed(ActionEvent e) {
      adaptee.jButtonSaveDST_actionPerformed(e);
    }
  }

  /**
   * Helper inner class created automatically by development tool. <br>
   * I just calls the corresponding action performed method of AdminToolFrame.
   */

  class AdminToolFrame_jMenuItemRemoveAll_actionAdapter
      implements java.awt.event.ActionListener {
    AdminToolFrame adaptee;

    AdminToolFrame_jMenuItemRemoveAll_actionAdapter(AdminToolFrame adaptee) {
      this.adaptee = adaptee;
    }

    public void actionPerformed(ActionEvent e) {
      adaptee.jMenuItemRemoveAll_actionPerformed(e);
    }
  }

//-----------------------------------------------------------------------------

} //end AdminToolFrame
```

admintool.AdminToolFrame_AboutBox.java

```
package admintool;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;
import com.borland.jbcl.layout.*;

/**
* <p>
```

```
* The About Dialog.
* </p>
*
*
 */

public class AdminToolFrame_AboutBox extends JDialog implements ActionListener {
  //GUI componets
  JPanel panel1 = new JPanel();
  JPanel insetsPanel1 = new JPanel();
  JButton button1 = new JButton();
  ImageIcon image1 = new ImageIcon();
  BorderLayout borderLayout1 = new BorderLayout();
  String product = "Administration Tool 0.1";
  String version = "Ver. 0.1";
  String copyright = "Copyright (c) 2004";
  String comments = "GUI Developing. Without event-handling.";
  JPanel jPanel1 = new JPanel();
  XYLayout xYLayout1 = new XYLayout();
  JTabbedPane jTabbedPane1 = new JTabbedPane();
  JPanel jPanelSupervisors = new JPanel();
  JPanel jPanelDeveloper = new JPanel();
  BorderLayout borderLayout2 = new BorderLayout();
  JLabel jLabelAboutDev = new JLabel();
  JLabel jLabeliImage = new JLabel();
  ImageIcon devIcon, empIcon;
 JLayeredPane layeredPane;
  BorderLayout borderLayout3 = new BorderLayout();
  XYLayout xYLayout3 = new XYLayout();
  JLabel jLabel1 = new JLabel();
  JLabel jLabel2 = new JLabel();
  JLabel jLabelEmp = new JLabel();
  JPanel jPanel2 = new JPanel();
  JLabel jLabel3 = new JLabel();
  JLabel jLabel4 = new JLabel();
  JLabel jLabel5 = new JLabel();
  JLabel jLabel6 = new JLabel();
  XYLayout xYLayout2 = new XYLayout();
  JPanel jPanelProgram = new JPanel();
  JLabel label2 = new JLabel();
  JPanel insetsPanel3 = new JPanel();
  JLabel label4 = new JLabel();
  JLabel label1 = new JLabel();
  BorderLayout borderLayout4 = new BorderLayout();
  XYLayout xYLayout5 = new XYLayout();
  JLabel jLabel7 = new JLabel();
  JLabel jLabel8 = new JLabel();
  JLabel jLabel9 = new JLabel();
  JPanel jPanel3 = new JPanel();
  BorderLayout borderLayout5 = new BorderLayout();
  JLabel jLabel10 = new JLabel();
  XYLayout xYLayout4 = new XYLayout();
  /**
   * Constructor.
   * Does an initialization and then calls the jbInit method to set up the GUI.
   * @param parent parent frame
   */
  public AdminToolFrame_AboutBox(Frame parent) {
    super(parent);

    //icon initialization
    devIcon = new ImageIcon(AdminToolFrame_AboutBox.class.getResource(
        "dev.gif"));
    empIcon = new ImageIcon(AdminToolFrame_AboutBox.class.getResource(
        "emp.gif"));

    // exit when window is closed.
    enableEvents(AWTEvent.WINDOW_EVENT_MASK);

  //set up gui.
    try {
      jbInit();
    }
    catch(Exception e) {
      e.printStackTrace();
    }
  }
```

```
  /**
   * Default Constructor.
   * Overriden to call the other constructor.
   */
  AdminToolFrame_AboutBox() {
    this(null);
  }

  /**
  * Set up GUI. <br>
  * Sets the components' properties and lays them out. <br>
  * Most of this code is automatically generated by JBuilder. Especially the
  * laying out of the components.<br>
  * This is the JBuilders method for setting up the GUI. It is necessary to
  * use this method it order to be able to modify the  GUI from JBuilder's
  * Design view.
  */
  private void jbInit() throws Exception  {
    layeredPane = new JLayeredPane();
    jLabeliImage.setFont(new java.awt.Font("Dialog", 0, 12));
    jLabeliImage.setMaximumSize(new Dimension(500, 650));
    jLabeliImage.setMinimumSize(new Dimension(500, 650));
    layeredPane.setLayout(xYLayout3);
    jLabel1.setFont(new java.awt.Font("Dialog", 1, 12));
    jLabel1.setForeground(Color.white);
    jLabel1.setText("Spyros Doulgeridis");
    jLabel2.setText("A.M.: 03099716, email: jrmmorrow@yahoo.gr");
    jLabel2.setForeground(Color.white);
    jLabel2.setFont(new java.awt.Font("Dialog", 1, 12));
    jLabelEmp.setText("");
    jLabelEmp.setPreferredSize(new Dimension(100, 97));
    jLabelEmp.setIcon(empIcon);
    jPanel2.setLayout(xYLayout2);
    jLabel3.setText("Professor:");
    jLabel4.setForeground(Color.blue);
    jLabel4.setText("G. I. Stassinopoulos");
    jLabel5.setText("Supervisor:");
    jLabel6.setForeground(Color.blue);
    jLabel6.setText("Dimitris M. Kyriazanos");
    label2.setText("Description: Tool for the association of XML data");
    insetsPanel3.setLayout(xYLayout5);
    insetsPanel3.setBorder(BorderFactory.createEmptyBorder(10, 60, 10, 10));
    label4.setText(" with Data Sources and other XML data.");
    label1.setText("Administration Tool");
    jPanelProgram.setLayout(borderLayout4);
    jLabel7.setFont(new java.awt.Font("Microsoft Sans Serif", 2, 11));
    jLabel7.setForeground(Color.white);
    jLabel7.setText("Special Thanks:");
    jLabel8.setFont(new java.awt.Font("Microsoft Sans Serif", 2, 11));
    jLabel8.setForeground(Color.white);
    jLabel8.setText("I would like to thank my family and everyone that was close to me " +
    "during the development");
    jLabel9.setFont(new java.awt.Font("Microsoft Sans Serif", 2, 11));
    jLabel9.setForeground(Color.white);
    jLabel9.setText(" of this Project");
    jPanelSupervisors.setMaximumSize(new Dimension(2147483647, 2147483647));
    jPanel3.setPreferredSize(new Dimension(10, 100));
    jPanel3.setLayout(xYLayout4);
    jLabel10.setText("I would like to thank the following persons for their wonderful
collaboration:");
    layeredPane.add(jLabel1,   new XYConstraints(61, 99, 443, 31), 0);
    layeredPane.add(jLabelAboutDev,  new XYConstraints(6, 82, -1, -1), 0);
    layeredPane.add(jLabel2,   new XYConstraints(61, 129, 443, 31), 0);
    layeredPane.add(jLabel8,  new XYConstraints(20, 197, 480, 23),0);
    layeredPane.add(jLabeliImage, new XYConstraints(1, 1, -1, -1), -1);
    layeredPane.add(jLabel7,  new XYConstraints(6, 177, 130, 20),0);
    layeredPane.add(jLabel9, new XYConstraints(20, 220, 475, 21),0);
    jTabbedPane1.add(jPanelProgram,  "Administration Tool");
    jPanelProgram.add(insetsPanel3,  BorderLayout.CENTER);
    insetsPanel3.add(label1, new XYConstraints(-18, 62, 406, 23));
    insetsPanel3.add(label2, new XYConstraints(-18, 93, 406, 23));
    insetsPanel3.add(label4,      new XYConstraints(62, 116, 257, 23));
    layeredPane.setPreferredSize(new Dimension(500, 650));
    image1 = new ImageIcon(admintool.AdminToolFrame.class.getResource("about.png"));
    this.setTitle("About");
    panel1.setLayout(borderLayout1);
    button1.setText("Ok");
```

```
    button1.addActionListener(this);
    jPanel1.setLayout(borderLayout2);
    jPanelDeveloper.setLayout(borderLayout3);
    jPanelDeveloper.add(layeredPane,  BorderLayout.CENTER);
    jLabelAboutDev.setFont(new java.awt.Font("Dialog", 1, 11));
    jLabelAboutDev.setText("Developed by");
    jLabelAboutDev.setForeground(Color.WHITE);
    jLabeliImage.setText("");
    jLabeliImage.setPreferredSize(new Dimension(500, 650));
    jLabeliImage.setIcon(devIcon);
    jPanelSupervisors.setLayout(borderLayout5);
    jPanelSupervisors.setPreferredSize(new Dimension(500, 650));
    panel1.add(insetsPanel1, BorderLayout.SOUTH);
    insetsPanel1.add(button1,   new XYConstraints(5, 5, 389, 22));
    panel1.add(jPanel1,  BorderLayout.CENTER);
    jPanel1.add(jTabbedPane1,  BorderLayout.CENTER);
    jTabbedPane1.add(jPanelSupervisors, "Supervisors");
    jPanel2.add(jLabelEmp, new XYConstraints(42, 1, 194, 151));
    jPanel2.add(jLabel5, new XYConstraints(256, 68, 185, 30));
    jPanel2.add(jLabel3, new XYConstraints(254, 25, 185, 30));
    jPanel2.add(jLabel4, new XYConstraints(302, 47, 185, 30));
    jPanel2.add(jLabel6, new XYConstraints(296, 90, 185, 30));
    jPanelSupervisors.add(jPanel3, BorderLayout.NORTH);
    jPanel3.add(jLabel10,      new XYConstraints(8, 13, 482, 25));
    jPanelSupervisors.add(jPanel2, BorderLayout.CENTER);
    jTabbedPane1.add(jPanelDeveloper, "Developer");
    this.getContentPane().add(panel1, BorderLayout.WEST);
    setResizable(true);
  }

  /**
   * Overridden so we can exit when window is closed.
   */
    protected void processWindowEvent(WindowEvent e) {
    if (e.getID() == WindowEvent.WINDOW_CLOSING) {
      cancel();
    }
    super.processWindowEvent(e);
  }

  /**
   * Close the dialog
   */

  void cancel() {
    dispose();
  }
  /**
   * OK button action performed.
   * Close the dialog on a button event.
   * @param e ActionEvent
   */
  public void actionPerformed(ActionEvent e) {
    if (e.getSource() == button1) {
      cancel();
    }
  }
}
```

admintool.DataModifier.java

```
package admintool;

/**
 * Objects can update the status of GUIUpdater that ADminToolFrame class has,
 * through this interface. Used by JDOMTreeModel objects to notify GUIUpdater that
 * tree structure is modified.<br>
 * AdminToolFrame implements this interface, and so a reference to AdminToolFrame is passed to
* JDOMTreeModel constructor. But JDOMTreeModel constructor casts the reference it gets to
* be DataModifier. So as to hide the rest of AdminToolFrame's methods.
 */
public interface DataModifier {
```

```
  /**
   * Notifies that data is modified. This is the only way to inform AdminToolFrame
   * that its data (Information Structure) is modified. Only AdminToolFrame implements this
method.
   * See the implementation of the method in AdminToolFrame for more.
   *
   * @see AdminToolFrame.dataModified()
   */
  public void dataModified();

}
```

admintool.DataSourceProperties.java

```
package admintool;

import java.sql.*;
import java.util.*;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

import org.jdom.*;
import com.borland.jbcl.layout.*;

/**
 * <p>
 *  A Dialog for manipulating Data Source Configuration Properties. Is called from
 * DataSourceConfiguration Dialog.
 * </p>
 * <p>
 * It is called both on "Add" and on "Properties" commands. When it is called,
 * it defines which of the two cases is called for, from the element
 * that is passed to the constructor and modifies some of its GUI components.
 * </p>
 * <p>
 * In the case of "Properties" Dialog, it puts on the text fields the values of the
 * data source configuration element passed to the constructor.
 * </p>

 */
public class DataSourceProperties
    extends JDialog {
  //data variables
  /**ODBC data source used in Combo box.*/
  private final String ODBC = "ODBC";

  /**Oracle data source used in Combo box.*/
  private final String ORACLE = "Oracle";

  /**Postgresql data source used in Combo box.*/
  private final String POSTGRESQL = "Postgresql";

  /**SQL server data source used in Combo box.*/
  private final String SQL_SERVER = "SQLServer";

  /**MySQL data source used in Combo box.*/
  private final String MY_SQL = "MySQL";

  /**Array of the data source used in combo box.*/
  private String sources[] = {
      ODBC, ORACLE, POSTGRESQL, SQL_SERVER, MY_SQL};

  /**Data Source which is being manipulated or created.*/
  private Element dataSrc;

  //values that was pass to the constructor.
  /**Type of the Data Source.*/
  private String type = null;

  /**DSN of the Data Source.*/
```

```
private String dsn = null;

/**Login required to access the Data Source.*/
private String login = null;

/**Password required to access the Data Source.*/
private String password = null;

/**Driver used to access te Data Source.*/
private String driver = null;

//values entered from user
/**Type of the Data Source entered by user.*/
private String inputType = null;

/**DSN of the Data Source entered by user.*/
private String inputDsn = null;

/**Login required to access the Data Source entered by user.*/
private String inputLogin = null;

/**Password required to access the Data Source entered by user.*/
private String inputPassword = null;

/**Driver used to access te Data Source entered by user.*/
private String inputDriver = null;

/**Passwords.*/
private Map passwordMap = null;

/**Indicates whether the data source properties are modified or not.*/
private boolean modified = false;

/**Renderer used in combo box that displays available data source types.*/
ComboBoxRenderer comboBoxRenderer = null;
/**To isten for changes.*/
private ChangeListenerImpl changeListener = null;

/**?*/
private ItemListenerImpl itemListener = null;

/**Icon used.*/
ImageIcon dataSrcTypeIcon, dsnIcon, jdbcDriverIcon, testIcon, okIcon,
    cancelIon, serverIcon;
//GUI components
JPanel panel1 = new JPanel();
JPanel jPanel1 = new JPanel();
JPanel jPanelButtons = new JPanel();
JPanel jPanelODBC = new JPanel();
JPanel jPanelChooseSource = new JPanel();
JPanel jPanelTestButton = new JPanel();
JPanel jPanelFields = new JPanel();
XYLayout xYLayout1 = new XYLayout();
JLabel jLabelType = new JLabel();
/**Available data source types.*/
JComboBox jComboBox1 = new JComboBox(sources);
/**OK button.*/
JButton jButtonOK = new JButton();
/**Cancel button.*/
JButton jButtonCancel = new JButton();
/**Test button.*/
JButton jButtonTest = new JButton();
/**JDBC Driver text field.*/
JTextField jTextFieldJDBC = new JTextField();
/**Login text field.*/
JTextField jTextFieldLogin = new JTextField();
/**DSN text field.*/
JTextField jTextFieldDSN = new JTextField();
/** Password. */
JPasswordField jPasswordField = new JPasswordField();
XYLayout xYLayout3 = new XYLayout();
BorderLayout borderLayout3 = new BorderLayout();
FlowLayout flowLayout1 = new FlowLayout();
BorderLayout borderLayout2 = new BorderLayout();
JLabel jLabel1 = new JLabel();
JLabel jLabel2 = new JLabel();
JLabel jLabel3 = new JLabel();
```

```
  JLabel jLabel4 = new JLabel();
  BorderLayout borderLayout1 = new BorderLayout();
  BorderLayout borderLayout4 = new BorderLayout();
  /**
   * Constructor.
   * Does an initialization and then calls the jbInit method to set up the GUI.
   * @param dialog the parent dialog from which this dialog was called.
   * @param modal modal
   * @param ds data source configuration which is going to be  manipulated.
   * @param passwordMap Passwords
   */
  public DataSourceProperties(Dialog dialog, boolean modal, Element ds,
                              Map passwordMap) {
    //call super class (JDialog) constructor to create the Dialog
    super(dialog, "", modal);
    //intialization
    dataSrc = ds;
    this.passwordMap = passwordMap;
    changeListener = new ChangeListenerImpl();
    itemListener = new ItemListenerImpl();
    comboBoxRenderer = new ComboBoxRenderer();

    //read the values of the data source configuration element.
    readDataSource();

    //initialize icons
    dataSrcTypeIcon = new ImageIcon(DataSourceProperties.class.getResource(
        "databasesLoaded.gif"));
    dsnIcon = new ImageIcon(DataSourceProperties.class.getResource(
        "databaseLoaded.gif"));
    jdbcDriverIcon = new ImageIcon(DataSourceProperties.class.getResource(
        "WebComponentAdd16.gif"));
    testIcon = new ImageIcon(DataSourceProperties.class.getResource(
        "testDataSource.gif"));
    okIcon = new ImageIcon(DataSourceProperties.class.getResource("done.gif"));
    cancelIon = new ImageIcon(DataSourceProperties.class.getResource(
        "cancel.gif"));
    serverIcon = new ImageIcon(DataSourceProperties.class.getResource(
        "Server16.gif"));

    //set up  GUI
    try {
      jbInit();
      pack();
      setVisible(true);
    }
    catch (Exception ex) {
      ex.printStackTrace();
    }
  }

  /**
   * Reads Data Source Configuration Properties. <br>
   * It decides whether it is an Add Data Source Dialog or a
   * Data Source Properties Dialog. In each case it set proper title. In the cases of
   * Properties Dialog it reads the valeus from the element which was passed to the
   * Constructor. The values are stored in its data variables.
   *
   */
  private void readDataSource() {
    //if null, then it is an Add Dialog.
    if (dataSrc == null) { //Add
      super.setTitle("Add Data Source");
      //set default value for driver
      driver = "sun.jdbc.odbc.JdbcOdbcDriver";
    }
    //if not null, the it is a Properties Dialog.
    else {
      super.setTitle("Data Source Properties");
      //read values from passed element.
      type = dataSrc.getAttribute("type").getValue();
      driver = dataSrc.getAttribute("driver").getValue();
      Element configuration = dataSrc.getChild("configuration");
      dsn = configuration.getAttribute("dsn").getValue();
      login = configuration.getAttribute("login").getValue();
      password = (String) passwordMap.get(login); //null if the map contains no mapping for this
key.
```

```
    }
  }

  /**
   * Set up GUI. <br>
   * Sets the components' properties and lays them out. <br>
   * In the case of the Properties Dialog, the text fields are initialized with the data
variables. Also OK button's
   * initial ststus is disable, so user must first test the data source before exiting.<br>
   * Most of this code is automatically generated by JBuilder. Especially the
   * laying out of the components.<br>
   * This is the JBuilders method for setting up the GUI. It is necessary to
   * use this method it order to be able to modify the  GUI from JBuilder's
   * Design view.
   */
  private void jbInit() throws Exception {
    panel1.setLayout(borderLayout1);
    jPanel1.setLayout(borderLayout4);
    jPanelODBC.setLayout(borderLayout2);
    jPanelFields.setLayout(xYLayout3);
    jPanelChooseSource.setLayout(xYLayout1);
    jPanelChooseSource.setPreferredSize(new Dimension(10, 30));
    jLabelType.setIcon(dataSrcTypeIcon);
    jLabelType.setText("Data Source Type");
    jPanelButtons.setLayout(flowLayout1);
    jPanelButtons.setPreferredSize(new Dimension(10, 30));
    jButtonOK.setPreferredSize(new Dimension(110, 25));
    jButtonOK.setIcon(okIcon);
    jButtonOK.setText("OK");
    jButtonOK.addActionListener(new
                                DataSourceProperties_jButtonOK_actionAdapter(this));
    jButtonOK.setEnabled(false);
    jButtonCancel.setAlignmentY( (float) 0.5);
    jButtonCancel.setPreferredSize(new Dimension(110, 25));
    jButtonCancel.setIcon(cancelIon);
    jButtonCancel.setText("Cancel");
    jButtonCancel.addActionListener(new
                                  DataSourceProperties_jButtonCancel_actionAdapter(this));
    jPanelTestButton.setLayout(borderLayout3);
    jPanelTestButton.setBorder(BorderFactory.createEtchedBorder());
    jPanelTestButton.setPreferredSize(new Dimension(10, 30));
    jButtonTest.setBorder(BorderFactory.createRaisedBevelBorder());
    jButtonTest.setIcon(testIcon);
    jButtonTest.setText("Test Data Source");
    jButtonTest.addActionListener(new
                                 DataSourceProperties_jButtonTest_actionAdapter(this));
    jPanelODBC.setBorder(BorderFactory.createEtchedBorder());
    jPanelFields.setPreferredSize(new Dimension(227, 90));
    borderLayout3.setHgap(10);
    borderLayout3.setVgap(10);
    panel1.setPreferredSize(new Dimension(300, 234));
    jLabel1.setHorizontalAlignment(SwingConstants.RIGHT);
    jLabel1.setHorizontalTextPosition(SwingConstants.RIGHT);
    jLabel1.setIcon(dsnIcon);
    jLabel1.setText("DSN");
    jLabel2.setHorizontalAlignment(SwingConstants.RIGHT);
    jLabel2.setHorizontalTextPosition(SwingConstants.RIGHT);
    jLabel2.setText("Login");
    jLabel3.setHorizontalAlignment(SwingConstants.RIGHT);
    jLabel3.setHorizontalTextPosition(SwingConstants.RIGHT);
    jLabel3.setText("Password");
    jLabel4.setHorizontalAlignment(SwingConstants.RIGHT);
    jLabel4.setHorizontalTextPosition(SwingConstants.RIGHT);
    jLabel4.setIcon(jdbcDriverIcon);
    jLabel4.setText("JDBC Driver");
    jPanel1.setPreferredSize(new Dimension(300, 234));
    jTextFieldDSN.setText("");
    jTextFieldLogin.setText("");
    jPasswordField.setText("");
    getContentPane().add(panel1);
    panel1.add(jPanel1, BorderLayout.CENTER);
    jPanel1.add(jPanelChooseSource, BorderLayout.NORTH);
    jPanelChooseSource.add(jLabelType, new XYConstraints(9, 5, 127, 22));
    jPanelChooseSource.add(jComboBox1, new XYConstraints(141, 5, 157, 22));
    jPanelButtons.add(jButtonOK, null);
    jPanelButtons.add(jButtonCancel, null);
    jPanel1.add(jPanelODBC, BorderLayout.CENTER);
```

```
  jPanelFields.add(jLabel4, new XYConstraints(14, 99, 116, 24));
  jPanelFields.add(jLabel2, new XYConstraints(14, 40, 116, 24));
  jPanelFields.add(jLabel1, new XYConstraints(14, 11, 116, 24));
  jPanelFields.add(jLabel3, new XYConstraints(14, 70, 116, 24));
  jPanelFields.add(jTextFieldDSN, new XYConstraints(138, 11, 149, 24));
  jPanelFields.add(jTextFieldJDBC, new XYConstraints(138, 99, 149, 24));
  jPanelFields.add(jPasswordField, new XYConstraints(138, 70, 149, 24));
  jPanelFields.add(jTextFieldLogin, new XYConstraints(138, 40, 149, 24));
  jPanel1.add(jPanelButtons, BorderLayout.SOUTH);
  jPanelODBC.add(jPanelTestButton, BorderLayout.SOUTH);
  jPanelTestButton.add(jButtonTest, BorderLayout.CENTER);
  jPanelODBC.add(jPanelFields, BorderLayout.CENTER);

  jComboBox1.addItemListener(itemListener);
  jComboBox1.setRenderer(comboBoxRenderer);

  jTextFieldDSN.setText(dsn);
  jTextFieldLogin.setText(login);
  jPasswordField.setText(password);
  jTextFieldJDBC.setText(driver);
  jComboBox1.setSelectedItem(type);

  jTextFieldDSN.getDocument().addDocumentListener(changeListener);
  jTextFieldLogin.getDocument().addDocumentListener(changeListener);
  jPasswordField.getDocument().addDocumentListener(changeListener);
  jTextFieldJDBC.getDocument().addDocumentListener(changeListener);

}

/**
 * Cancel button action performed. <br>
 * Exits the Dialog.
 * @param e ActionEvent
 */
void jButtonCancel_actionPerformed(ActionEvent e) {
  setVisible(false);
  dispose();
}

/**
 * Test button action performed.<br>
 * Reads values entered by user for data source configuration and test them.
 * Only ODBC sources are support so far. The test is delegated to testODBCSource method.
 * If test is passed the OK button is enabled.
 *
 * @param e ActionEvent
 * @see testODBCSource()
 */
void jButtonTest_actionPerformed(ActionEvent e) {
  //Read values entered by user for data source configuration
  readInputValues();
  //create JDBC string
  //check if a type is selected
  if (inputType == null) {
    JOptionPane.showMessageDialog(null,
                                  "Please select a Data Source type.",
                                  "No Data Source type is selected",
                                  JOptionPane.WARNING_MESSAGE);
    return;
  }
  //start testing
  String jdbcString = null;
  try {
    //if it is a ODBC source
    if (inputType.equals(ODBC)) {
      //create jdbc string and test it
      jdbcString = testODBCSource();
    }
    //other sources testing code goes here
    //else not data source not supported yet, show error message.
    else {
      JOptionPane.showMessageDialog(null,
                                    "Data Source type not supported yet!",
                                    "Unsupported Data Source type",
                                    JOptionPane.ERROR_MESSAGE);
    }
```

```
  }
  //catch errors
  catch (ClassNotFoundException ex) {
    //custom title, error icon
    JOptionPane.showMessageDialog(null,
                                  "Could not load DriverManager for " +
                                  jdbcString +
                                  "! \nClassNotFoundException thrown...",
                                  "Failure in connecting to Data Source",
                                  JOptionPane.ERROR_MESSAGE);
    return;
  }
  catch (SQLException sqlex) {
    JOptionPane.showMessageDialog(null,
                                  "Could not establish connection for \"" +
                                  jdbcString + "\"! \nSQLException thrown...",
                                  "Failure in connecting to Data Source",
                                  JOptionPane.ERROR_MESSAGE);
    return;
  }

  //if everything went OK
  JOptionPane.showMessageDialog(null,
                                "Connection to Data Source \"" + jdbcString +
                                "\" was established.",
                                "Test was successful",
                                JOptionPane.INFORMATION_MESSAGE);
  //enable OK button
  jButtonOK.setEnabled(true);

}

/**
 * Creats an JDBC connection string  and tests it. <br>
 * Takes the type and the name of the ODBC source creats the connection string
 * and tests if connects using the Default ODBC driver provided by Sun.
 * Other drivers are not supported yet.
 * @throws ClassNotFoundException
 * @throws SQLException
 * @return jdbc connection string, eg. jdbc:odbc:customers
 */
private String testODBCSource() throws ClassNotFoundException, SQLException {
  //create jdbc driver string
  String jdbcString = new String("jdbc:" +
                                 inputType.toLowerCase() + ":" +
                                 inputDsn);
  //load driver
  Class.forName(inputDriver);
  // establish connection to database
  Connection connection = DriverManager.getConnection(jdbcString, inputLogin,
      inputPassword); //eg. jdbc:odbc:customers
  //read meta data to ensure that connection is successful
  DatabaseMetaData databaseMetaData = connection.getMetaData();
  //now we are sure that the connection is successful. Job done.

  //disconnect
  if (connection != null) {
    connection.close();
  }
  //return the jdbc string that passed the test
  return jdbcString;
}

/**
 * Reads values that the user entered forthe data source configuration.
 */
private void readInputValues() {
  //red values from GUI components
  inputType = (String) jComboBox1.getSelectedItem();
  inputDsn = jTextFieldDSN.getText();
  inputLogin = jTextFieldLogin.getText();
  char[] charArray = jPasswordField.getPassword();
  inputPassword = new String(charArray);
  inputDriver = jTextFieldJDBC.getText();
}

/**
```

```
 * <p>
 * Maps the selection of the ComboBox to the appropriate Data source type.
 * The String displayed in the Combo box is just a string that displays the data
 * source type. The type variable of this dialog contains the corresponding string
 * that is used in the jdbc connection string, e.g. for jdbc:odbc:nameofdatasource
 * the type is odbc, but in other types of data sources the displayed name in the
 * combo box might not be the same with the string type.
 * Supports only ODBC sources so far.
 * </p>
 * <p>
 * Item Listener inplementation used by ComboBox.
 * </p>
 */
class ItemListenerImpl
    implements ItemListener {
  /**
   * Default constructor. Does nothing.
   */
  ItemListenerImpl() {
  }

  /**
   *  Maps the selection of the JComboBox to the appropriate string type.
   * @param event ItemEvent
   */
  public void itemStateChanged(ItemEvent event) {
    // if a check box selected
    if (event.getStateChange() == ItemEvent.SELECTED) {
      //get selection
      String selection = (String) jComboBox1.getSelectedItem();
      //if it is an ODBC selection
      if (selection.equals(ODBC)) {
        //then type is ODBC
        inputType = ODBC;
      }
      //support code for other data sources goes here
      //error messages
      else {
        JOptionPane.showMessageDialog(null,
                                      "Data Source type not supported yet!",
                                      "Unsupported Data Source type",
                                      JOptionPane.ERROR_MESSAGE);
        //set as Default type the ODBC type
        jComboBox1.setSelectedItem(ODBC);
      }
    }
  } //end itemStateChanged
} //end ItemListenerImpl

/**
 * The renderer that modifies the appearance of the Conbo box. <br>
 * It sets the description of the data source and its icon.
 *
 */
class ComboBoxRenderer
    extends JLabel
    implements ListCellRenderer {
  /**
   * Array with the descriptions of the data sources used in combo box.
   */
  private String sourcesDisplay[] = {
      new String(ODBC + " source"),
      new String(ORACLE + " database"),
      new String(POSTGRESQL + " database"),
      new String(SQL_SERVER + " database"), new String(MY_SQL + " database")};

  /**
   * Default constructor. Does nothing.
   */
  public ComboBoxRenderer() {
  }

  /**
   * Does the rendering.
   *
   * @param list The JList we're painting.
```

```
     * @param value  The component that is going to be rendered. The value returned by
list.getModel().getElementAt(index).
     * @param index  The cells index.
     * @param isSelected True if the specified cell was selected.
     * @param cellHasFocus True if the specified cell has the focus.
     * @return A component whose paint() method will render the specified value.
     */
    public Component getListCellRendererComponent(
        JList list,
        Object value,
        int index,
        boolean isSelected,
        boolean cellHasFocus) {
      //if data source not defined
      String source = (String) value;
      if (source == null) {
        //exit
        return this;
      }
      //set icon of data sources
      setIcon(serverIcon);
      //find source and...
      for (int i = 0; i < source.length(); i++) {
        if (sources[i].equals(source)) {
          //set set its description
          setText(sourcesDisplay[i]);
          break;
        }
      }
      return this;
    } //end getListCellRendererComponent
  } //end ComboBoxRenderer

  /**
   * OK button action peformed. <br>
   * Saves the data source configuration element and exits dialog.
   * <ul>
   * <li>if it is an Add Dialog, create a new data source configuration element</li>
   * <li>if it is a Properties Dialog and data source configuration is modified, then update the
existing element</li>
   * <li>exit</li>
   * </ul>
   *
   * @param e ActionEvent
   */
  void jButtonOK_actionPerformed(ActionEvent e) {
    //if it is an Add Dialog
    if (dataSrc == null) {
      //create a new elememement
      dataSrc = new Element("datasource");
      dataSrc.setAttribute("type", inputType);
      dataSrc.setAttribute("driver", inputDriver);
      dataSrc.addContent(new Element("configuration"));
      dataSrc.getChild("configuration").setAttribute("dsn", inputDsn);
      dataSrc.getChild("configuration").setAttribute("login", inputLogin);
      passwordMap.put(inputLogin, inputPassword);
    }
    //if it is a Properties Dialog
    else {
      //and is modified
      if (modified) {
        //update it
        dataSrc.setAttribute("type", inputType);
        dataSrc.setAttribute("driver", inputDriver);
        dataSrc.getChild("configuration").setAttribute("dsn", inputDsn);
        dataSrc.getChild("configuration").setAttribute("login", inputLogin);
        passwordMap.put(inputLogin, inputPassword);
      }
    }
    //exit
    jButtonCancel_actionPerformed(e);
  }

  /**
   * Gets the data source configuration element.
   * Used by DataSourceConfiguration Dialog to access the data source configuration.
   * @return Element
```

```
   */
  public Element getDataSrc() {
    return dataSrc;
  }

  /**
   * An implementation of abstract class AbsractChangeListenerImpl.
   * See supperclass' API for more.
   */
  private class ChangeListenerImpl
      extends AbsractChangeListenerImpl {
    /**
     * Actions taken when data on GUI components are modified.<br>
     * Changes status of properties into modified.
     */
    void changesOccured() {
      modified = true;
    }
  } //ChangeListenerImpl

//----------------------Action Adapter classes --------------------------------
  /**
   * Helper inner class created automatically by development tool. <br>
   * I just calls the corresponding action performed method of AdminToolFrame.
   */

  class DataSourceProperties_jButtonCancel_actionAdapter
      implements java.awt.event.ActionListener {
    DataSourceProperties adaptee;

    DataSourceProperties_jButtonCancel_actionAdapter(DataSourceProperties
        adaptee) {
      this.adaptee = adaptee;
    }

    public void actionPerformed(ActionEvent e) {
      adaptee.jButtonCancel_actionPerformed(e);
    }
  }
  /**
     * Helper inner class created automatically by development tool. <br>
     * I just calls the corresponding action performed method of AdminToolFrame.
     */

  class DataSourceProperties_jButtonTest_actionAdapter
      implements java.awt.event.ActionListener {
    DataSourceProperties adaptee;

    DataSourceProperties_jButtonTest_actionAdapter(DataSourceProperties adaptee) {
      this.adaptee = adaptee;
    }

    public void actionPerformed(ActionEvent e) {
      adaptee.jButtonTest_actionPerformed(e);
    }
  }
  /**
     * Helper inner class created automatically by development tool. <br>
     * I just calls the corresponding action performed method of AdminToolFrame.
     */

  class DataSourceProperties_jButtonOK_actionAdapter
      implements java.awt.event.ActionListener {
    DataSourceProperties adaptee;

    DataSourceProperties_jButtonOK_actionAdapter(DataSourceProperties adaptee) {
      this.adaptee = adaptee;
    }

    public void actionPerformed(ActionEvent e) {
      adaptee.jButtonOK_actionPerformed(e);
    }
  }

//-----------------------------------------------------------------------------

} //end DataSourceProperties
```

admintool.DataSourcesConfiguration.java

```
package admintool;

import java.util.*;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;

import org.jdom.*;

/**
 * <p>
 * A Dialog for manipulating Data Source Configurations.
 * </p>
 * <p>
 * It is called both on "New Data Source Configuration" and on "Configure Data Sources"
 * commands. When it is called it defines which of the two cases is called for, from the
document
 * that is passed to the constructor and modifies some of its GUI components.
 * </p>
 * <p>
 * In this class, the term <i>document</i> is refered to the Document that holds the data
sources configurations (*.dsc)
 * that are loaded on the application, and not to the documents of the Information Structure
tree
 * and Data Structure tree.
 * </p>
 *
 */
public class DataSourcesConfiguration
    extends JDialog {
  //data variables
  /**The parent frame.*/
  private AdminToolFrame parent = null;

  /**The initial Data source configuration document that was passed to constructor. Used to
store the document
   * to the constructor as a back up. If user hits cancel on exit, it is used to restore the
initial state of the document.*/
  private Document initialDoc = null;

  /**Data source configuration document which contains the Data sources currently loaded. The
modification are applied directly
   * on it.*/
  private Document doc = null;

  /**The root of the doc document.*/
  private Element root = null;

  /**Passwords.*/
  private Map passwordMap = null;

  /**List model used in the list that displays the data sources.*/
  private DefaultListModel listModel = null;

  /**OK button text. OK button might say "Load" or "Update".*/
  private String okButtonName = null;

  /**Indicates whether the data source configurations are modified or not.*/
  private boolean modified = false;

  //GUI Components
  JPanel panel1 = new JPanel();
  JPanel jPanel1 = new JPanel();
  JPanel jPanel2 = new JPanel();
  JPanel jPanel3 = new JPanel();
  JLabel jLabelDataSrc = new JLabel();
  JPanel jPanel5 = new JPanel();
```

```
/**Add button.*/
JButton jButtonAdd = new JButton();
/**Remove button.*/
JButton jButtonRemove = new JButton();
/**Properties button.*/
JButton jButtonProps = new JButton();
JPanel jPanel4 = new JPanel();
/**Cancel button*/
JButton jButtonCancel = new JButton();
/**OK (displayed as Load or Update) button. When command "New Data Source Configuration"
 * is entered then the button text is "Load". When "Configure Data Sources"  is entered then
 * button text is "Update".
 * */
JButton jButtonOK = new JButton();
TitledBorder titledBorder1;
BorderLayout borderLayout1 = new BorderLayout();
BorderLayout borderLayout2 = new BorderLayout();
GridLayout gridLayout1 = new GridLayout();
BorderLayout borderLayout3 = new BorderLayout();
BorderLayout borderLayout4 = new BorderLayout();
FlowLayout flowLayout1 = new FlowLayout();
JScrollPane jScrollPane1 = new JScrollPane();
/**The list that the data sources are displayed.*/
JList jListDataSrc = new JList();

//Imaging
/**Icon used.*/
ImageIcon addDataSrcIcon, removeDataSrcIcon, dataSrcPropsIcon, doneIcon,
    cancelIcon;
/**
 * Constructor.
 * Does an initialization and then calls the jbInit method to set up the GUI.
 *
 * @param frame the parent frame
 * @param modal
 * @param dsc the data source configuration document.
 * @param passwordMap the passwords.
 */
public DataSourcesConfiguration(Frame frame, boolean modal,
                                Document dsc, Map passwordMap) {
  //call super class (JDialog) constructor to create the Dialog
  super(frame, "", true);
  //Initialize data variables
  doc = dsc;
  root = doc.getRootElement();
  parent = (AdminToolFrame) frame;
  this.passwordMap = passwordMap;
  listModel = new DefaultListModel();
  //read the data source configurations passed
  readDataSrcConfig();

  //initialize icons
  addDataSrcIcon = new ImageIcon(DataSourcesConfiguration.class.getResource(
      "addDataSource.gif"));
  removeDataSrcIcon = new ImageIcon(DataSourcesConfiguration.class.
                                    getResource("removeDataSource.gif"));
  dataSrcPropsIcon = new ImageIcon(DataSourcesConfiguration.class.getResource(
      "dataSourceProperties.gif"));
  doneIcon = new ImageIcon(DataSourcesConfiguration.class.getResource(
      "done.gif"));
  cancelIcon = new ImageIcon(DataSourcesConfiguration.class.getResource(
      "cancel.gif"));

  //set up GUI.
  try {
    jbInit();
    pack();
    setVisible(true);
  }
  catch (Exception ex) {
    ex.printStackTrace();
  }
} //end consstructor

/**
 * Reads data source configurations. <br>
 * It decides whether it is a New Data Source Configuration Dialog or a
```

```
   * Configure Data Sources Dialog. In each case it set proper title and OK button name.
   * Also in the case of Configure Data Sources Dialog it initilizes the list of data sources.
   * In both cases it backs up the document that passed to the constructor in order to be able
to
   * discard changes if user hits cancel on exit.
   * @see initializeList()
   */
  private void readDataSrcConfig() {
    //if the document has no content, then it is a New Data Source Configuration Dialog.
    if (doc.getRootElement().getContentSize() == 0) {
      super.setTitle("Load New Data Source Configuration");
      okButtonName = "Load";
      //.. do nothing
    }
    //if the document has content, then it is a Configure Data Sources Dialog.
    else { // has root --> Configure existing
      super.setTitle("Configure Data Sources");
      okButtonName = "Update";
      //Map data Sources in xml document TO List model
      initializeList();

    }
    //back up the document that passed to the constructor used in cancel
    initialDoc = new Document( (Element) doc.getRootElement().clone());
  }

  /**
   * Maps data Sources of loaded document to the  List model.<br>
   * The elements of the document are the data sources that are going to be configured.
   * These elements are not detached from the root of the document.
   *
   */
  private void initializeList() {
    //create list
    java.util.List dataSrcList = root.getChildren();
    Iterator i = dataSrcList.iterator();
    //put elements to list nodes
    while (i.hasNext()) {
      Element el = (Element) i.next();
      //create a ListNode with the element
      ListNode listNode = new ListNode(el);
      //add listnode to the list model
      listModel.addElement(listNode);
    }
  }

  /**
   * Set up GUI. <br>
   * Sets the components' properties and lays them out. <br>
   * Most of this code is automatically generated by JBuilder. Especially the
   * laying out of the components.<br>
   * This is the JBuilders method for setting up the GUI. It is necessary to
   * use this method it order to be able to modify the  GUI from JBuilder's
   * Design view.
   */
  private void jbInit() throws Exception {
    titledBorder1 = new TitledBorder("Data Source Title");
    panel1.setLayout(borderLayout1);
    jPanel1.setLayout(borderLayout2);
    jLabelDataSrc.setHorizontalTextPosition(SwingConstants.LEADING);
    jLabelDataSrc.setText("DataSources");
    jLabelDataSrc.setVerticalAlignment(javax.swing.SwingConstants.CENTER);
    jPanel2.setLayout(borderLayout3);
    jPanel3.setLayout(borderLayout4);
    jPanel5.setLayout(gridLayout1);
    jButtonAdd.setText("Add");
    jButtonAdd.addActionListener(new
                                 DataSourcesConfiguration_jButtonAdd_actionAdapter(this));
    jButtonAdd.setIcon(addDataSrcIcon);
    jButtonRemove.setText("Remove");
    jButtonRemove.addActionListener(new
                                    DataSourcesConfiguration_jButtonRemove_actionAdapter(this));
    jButtonRemove.setIcon(removeDataSrcIcon);
    jButtonProps.setToolTipText("");
    jButtonProps.setText("Properties");
    jButtonProps.addActionListener(new
                                   DataSourcesConfiguration_jButtonProps_actionAdapter(this));
```

```
  jButtonProps.setIcon(dataSrcPropsIcon);
  jButtonCancel.setText("Cancel");
  jButtonCancel.addActionListener(new
                                  DataSourcesConfiguration_jButtonCancel_actionAdapter(this));
  jButtonCancel.setPreferredSize(new Dimension(100, 25));
  jButtonCancel.setIcon(cancelIcon);
  jButtonOK.setText(okButtonName);
  jButtonOK.addActionListener(new
                              DataSourcesConfiguration_jButtonOK_actionAdapter(this));
  jButtonOK.setPreferredSize(new Dimension(100, 25));
  jButtonOK.setIcon(doneIcon);
  jPanel4.setLayout(flowLayout1);
  panel1.setPreferredSize(new Dimension(325, 150));
  panel1.setRequestFocusEnabled(true);
  this.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
  this.setModal(false);
  this.setResizable(false);
  jPanel3.setBorder(BorderFactory.createRaisedBevelBorder());
  jPanel3.setDoubleBuffered(true);
  gridLayout1.setColumns(1);
  gridLayout1.setHgap(5);
  gridLayout1.setRows(3);
  gridLayout1.setVgap(5);
  jPanel5.setMinimumSize(new Dimension(83, 95));
  jPanel5.setPreferredSize(new Dimension(120, 95));
  jPanel4.setPreferredSize(new Dimension(416, 40));
  jPanel2.setPreferredSize(new Dimension(63, 25));
  jPanel1.setPreferredSize(new Dimension(325, 164));
  jListDataSrc.setBorder(BorderFactory.createLoweredBevelBorder());
  jListDataSrc.setPreferredSize(new Dimension(200, 95));
  jListDataSrc.setModel(listModel);
  jListDataSrc.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
  jScrollPane1.setHorizontalScrollBarPolicy(JScrollPane.
                                            HORIZONTAL_SCROLLBAR_NEVER);
  getContentPane().add(panel1);
  panel1.add(jPanel1, BorderLayout.CENTER);
  jPanel1.add(jPanel2, BorderLayout.NORTH);
  jPanel1.add(jPanel3, BorderLayout.CENTER);
  jPanel2.add(jLabelDataSrc, BorderLayout.WEST);
  jPanel3.add(jPanel5, BorderLayout.EAST);
  jPanel5.add(jButtonAdd, null);
  jPanel5.add(jButtonRemove, null);
  jPanel5.add(jButtonProps, null);
  jPanel3.add(jScrollPane1, BorderLayout.CENTER);
  jScrollPane1.getViewport().add(jListDataSrc, null);
  jPanel4.add(jButtonOK, null);
  jPanel4.add(jButtonCancel, null);
  jPanel1.add(jPanel4, BorderLayout.SOUTH);
  super.setModal(true);
}

/**
 * Cancel button action performed. <br>
 * It restores the document to its inital state using the back up and so discards any
 * modifications.
 * @param e ActionEvent
 */
void jButtonCancel_actionPerformed(ActionEvent e) {
  //dispose changes
  parent.setDataSrcConfigDocument(initialDoc);
  //exit
  setVisible(false);
  dispose();
}

/**
 * Remove button action performed. <br>
 * Removes a Data Source configuration.<br>
 * First it changes the status of configuration into modified.
 * Gets the selected item of the JList. If user did not make a selection then
 * displays an error message. If there is a selection, retrieves the element and
 * removes it from the document. It also removes the selection from the list model of the
 * JList.
 *
 * @param e ActionEvent
 */
void jButtonRemove_actionPerformed(ActionEvent e) {
```

```
    //consfigurations are modified
    modified = true;
    //get index of selected item
    int index = jListDataSrc.getSelectedIndex();
    //check if user did make a selection
    if (index == -1) {
      JOptionPane.showMessageDialog(null,
                                    "Please first select a Data Source and then clikck
\"Remove\".",
                                    "No Data Source is selected",
                                    JOptionPane.WARNING_MESSAGE);

      return;
    }
    //remove element from doc
    //get Listnode form index
    ListNode listNode = (ListNode) listModel.getElementAt(index);
    //get element from listnode
    Element el = listNode.getDataSrc();
    //detach element from root of the document --> delete it
    el.detach();
    //check if remove action is succesfull
    if (el.getParent() != null) {
      System.err.println("ERROR - it stills have a parent!!!!");
    }
    //to be collected as garbage
    el = null;
    //remove it from list too
    listModel.remove(index);
  }

  /**
   * Add button action performed. <br>
   * Adds a new Data Source configuration.<br>
   * First it changes the status of configuration into modified.
   * Calls DataSourceProperties Dialog so user can create a new Data Source configuration.
   * Gets the new data source configuration from the called Dialog and adds it to
   * the document and to the JList.
   *
   * @param e ActionEvent
   * @see DataSourceProperties
   */
  void jButtonAdd_actionPerformed(ActionEvent e) {
    //consfigurations are modified
    modified = true;
    //Calls DataSourceProperties Dialog
    DataSourceProperties dsp = new DataSourceProperties(this, true, null,
        passwordMap);
    //get the new data source from DataSourceProperties Dialog
    Element newDataSrc = dsp.getDataSrc();

    //if it was created
    if (newDataSrc != null) {
      //add it to the document
      root.addContent(newDataSrc);
      //add it to the list!
      ListNode listNode = new ListNode(newDataSrc);
      listModel.addElement(listNode);
      //Caution!
      //Parent of newDataSrc must be null, because it is used as a flag when exiting the form
    }

  }

  /**
   * Properties button action performed.<br>
   * Displays and manipulates the Data Source configuration Properties.<br>
   * First it changes the status of configuration into modified. Check if user did make a
selection.
   * If user did make a selection, it takes the data source element and calls the
   * DataSourceProperties Dialog with it so user can modify it. The modifications
   * take effect directly to the dataSource element, because elements are not
   * detached from the root of he document. Finally, it updates the JList.
   *
   * @param e ActionEvent
   * @see DataSourceProperties
   */
```

```
  void jButtonProps_actionPerformed(ActionEvent e) {
    //consfigurations are modified
    modified = true;
    //get index of selection
    int index = jListDataSrc.getSelectedIndex();
    //check if user did make a selection
    if (index == -1) {

      JOptionPane.showMessageDialog(null,
                                    "You must first select a Data Source to view its
Properties!",
                                    "No Data source is selected",
                                    JOptionPane.WARNING_MESSAGE);
    }
    //user did make a selection
    else {
      //get listNode from index
      ListNode listNode = (ListNode) listModel.getElementAt(index);
      //get element from ListNode
      Element existingDataSrc = listNode.getDataSrc();
      //Call DataSourceProperties Dialog to display the data source
      DataSourceProperties dsp = new DataSourceProperties(this, true,
          existingDataSrc, passwordMap);

      //Modifacation take effect directly to the dataSource element of root
      //beacause element are not detached from the root of he doc
      //..so nothing to do concerning the doc

      //update list
      listNode = (ListNode) listModel.getElementAt(index);
      //Update the display of modified nodes of the list
      jListDataSrc.validate();
      jListDataSrc.repaint();
    }
  }

  /**
   * OK button action performed. <br>
   * If
   */
  void jButtonOK_actionPerformed(ActionEvent e) {
    //if data source configurations are modified
    if (modified) {
      //Ask to save
      Object[] options = {
          "Yes, Save changes",
          "No, Discard & Exit",
          "Cancel"};
      int n = JOptionPane.showOptionDialog(null,
                                           "Data Source Configuration may have been changed.
\nSave changes?",
                                           "Save?",
                                           JOptionPane.YES_NO_CANCEL_OPTION,
                                           JOptionPane.QUESTION_MESSAGE,
                                           null,
                                           options,
                                           options[2]);
      //if cancel,
      if (n == JOptionPane.CANCEL_OPTION) {
        //do nothing
        return;
      }
      //if no,
      else if (n == JOptionPane.NO_OPTION) {
        //call cancel button action perform to discard changes and exit
        jButtonCancel_actionPerformed(e);
      }
      //if yes
      else if (n == JOptionPane.YES_OPTION) {
        //apply changes to data source configurations of parent (AdminToolFrame)
        parent.updateDataStructure(doc);
        //exit
        setVisible(false);
        dispose();
      }
    }
  } //end jButtonOK_actionPerformed
```

```
  /**
   * The nodes of the list model that the JList use. The reason for creating this
   * class is for the desired description of a data source to be displayed in the JList.
   * <p>
   * The List model can hold any Object. If the list model was created to hold Element directly,
the
   * JList would display the string returned by the holding Elements toString method. Which is
   * not the desired decription for a data source. So this alternative is prefered.
   * </p>
   *
   */
  private class ListNode {
    /**The data source Element*/
    private Element dataSrc = null;

    /**
     * Constructor. Initializes the node with the passed element.
     * @param data the element of the node
     */
    public ListNode(Element data) {
      dataSrc = data;
    }

    /**
     * Return the discription of the data source element displayed in the list.
     * Overwrite toString method in order for the desired string to be
     * displayed in JList.
     * @return the discription of the data source element displayed in the list.
     */
    public String toString() {
      //create the desired description
      StringBuffer sb = new StringBuffer("");
      String type = dataSrc.getAttribute("type").getValue();
      sb.append(type.toUpperCase() + " source : ");
      sb.append("\"" +
                dataSrc.getChild("configuration").getAttribute("dsn").getValue() +
                "\"");
      //return so as to be diplayed in the list.
      return sb.toString();
    }

    /**
     * Getter method that returns the element of the node.
     * @return the element of the node
     */
    public Element getDataSrc() {
      return dataSrc;
    }

  }

//------------  Action Adapter classes ----------------------------------------
  /**
   * Helper inner class created automatically by development tool. <br>
   * I just calls the corresponding action performed method of AdminToolFrame.
   */

  class DataSourcesConfiguration_jButtonCancel_actionAdapter
      implements java.awt.event.ActionListener {
    DataSourcesConfiguration adaptee;

    DataSourcesConfiguration_jButtonCancel_actionAdapter(
        DataSourcesConfiguration
        adaptee) {
      this.adaptee = adaptee;
    }

    public void actionPerformed(ActionEvent e) {
      adaptee.jButtonCancel_actionPerformed(e);
    }
  }

  /**
   * Helper inner class created automatically by development tool. <br>
   * I just calls the corresponding action performed method of AdminToolFrame.
   */
```

```
  class DataSourcesConfiguration_jButtonRemove_actionAdapter
      implements java.awt.event.ActionListener {
    DataSourcesConfiguration adaptee;

    DataSourcesConfiguration_jButtonRemove_actionAdapter(
        DataSourcesConfiguration
        adaptee) {
      this.adaptee = adaptee;
    }

    public void actionPerformed(ActionEvent e) {
      adaptee.jButtonRemove_actionPerformed(e);
    }
  }

  /**
   * Helper inner class created automatically by development tool. <br>
   * I just calls the corresponding action performed method of AdminToolFrame.
   */

  class DataSourcesConfiguration_jButtonAdd_actionAdapter
      implements java.awt.event.ActionListener {
    DataSourcesConfiguration adaptee;

    DataSourcesConfiguration_jButtonAdd_actionAdapter(DataSourcesConfiguration
        adaptee) {
      this.adaptee = adaptee;
    }

    public void actionPerformed(ActionEvent e) {
      adaptee.jButtonAdd_actionPerformed(e);
    }
  }

  /**
   * Helper inner class created automatically by development tool. <br>
   * I just calls the corresponding action performed method of AdminToolFrame.
   */

  class DataSourcesConfiguration_jButtonProps_actionAdapter
      implements java.awt.event.ActionListener {
    DataSourcesConfiguration adaptee;

    DataSourcesConfiguration_jButtonProps_actionAdapter(
        DataSourcesConfiguration
        adaptee) {
      this.adaptee = adaptee;
    }

    public void actionPerformed(ActionEvent e) {
      adaptee.jButtonProps_actionPerformed(e);
    }
  }

  /**
   * Helper inner class created automatically by development tool. <br>
   * I just calls the corresponding action performed method of AdminToolFrame.
   */

  class DataSourcesConfiguration_jButtonOK_actionAdapter
      implements java.awt.event.ActionListener {
    DataSourcesConfiguration adaptee;

    DataSourcesConfiguration_jButtonOK_actionAdapter(DataSourcesConfiguration
        adaptee) {
      this.adaptee = adaptee;
    }

    public void actionPerformed(ActionEvent e) {
      adaptee.jButtonOK_actionPerformed(e);
    }
  }

//-----------------------------------------------------------------------------

} //end DataSourceConfiguration
```

admintool.FilesFilter.java

```
package admintool;

import java.io.File;
import javax.swing.*;
import javax.swing.filechooser.*;
/**
* <p>
* The File Filter is used from JFileChooser and supports xml, dsc, dst, is, xsl, asc files.<br>
* Dispite of implementing a seperate class for each file type extension, this class supports all
* the extensions that are needed by the application. Depending on the parameter
* passed on initialization the corresponding file filter type extension
* is created, eg. a "new FilesFilter("xml")" is an xml file filter.
* </p>
* <p>
 * It is necessary to remove the file filter from jfilechooser after use, because file filters
are
 * accomulated, that is the addition of a new file filter to jfilechooser does
 * not remove the previous one, but jfilechooser now displays both type of files.
 *</p>
 *<i>Tip: For more see The Java Tutorial:  Creating a GUI with JFC/Swing:
 * How to Use File Choosers </i>
 */
public class FilesFilter
    extends FileFilter {
  /**Information Structure file type extension.*/
  private final String is = "is";
  /**Data Source Configuration file type extension.*/
  private final String dsc = "dsc";
  /**Data Structure file type extension.*/
  private final String dst = "dst";
  /**XML file type extension.*/
  private final String xml = "xml";
  /**XSL file type extension.*/
  private final String xsl = "xsl";
  /**Association file type extension.*/
  private final String asc = "asc";
  /**Information Structure file description.*/
  private final String isDescription = "Information Structure files";
  /**Data Source Configuration file description.*/
  private final String dscDescription = "Data Source Configuration files";
   /**Data Structure file description.*/
  private final String dstDescription = "Data Structure files";
   /**XML file description.*/
  private final String xmlDescription = "XML files";
  /**XSL file description.*/
  private final String xslDescription = "XSL files";
  /**Association file description.*/
  private final String ascDescription = "Association files";
  /**The extension that the File Filter is initialezed, and thus the file type that the
   * object is filtering. */
  private String shownExtension;
  /**
   * Constructor, which initializes the extension of the file type that is going to be shown
   * on the file chooser.
   * @param ex String
   */
  public FilesFilter(String ex) {
    shownExtension = ex;
  }

  /**
   * Whether the given file is accepted by this filter.
   *
   * @param f the file that is tested  if it is accepted.
   * @return Yes if it is accepted
   */
  public boolean accept(File f) {
    //show directories
    if (f.isDirectory()) {
```

```
      return true;
    }
    //get extension
    String extension = getExtension(f);
    //check
    if (extension != null) {
      if (extension.equals(shownExtension)) {
        return true;
      }
      else {
        return false;
      }
    }
    return false;

  }

  /**
   * The description of this filter. For example: "JInformation Structure files"
   *
   * @return String
   */
  public String getDescription() {
    if (shownExtension.equals(is)) {
      return isDescription;
    }
    else if (shownExtension.equals(dsc)) {
      return dscDescription;
    }
    else if (shownExtension.equals(dst)) {
      return dstDescription;
    }
    else if (shownExtension.equals(xml)) {
      return xmlDescription;
    }
    else if (shownExtension.equals(xsl)) {
      return xslDescription;
    }
    else if (shownExtension.equals(asc)) {
      return ascDescription;
    }

    else { //Should not be excecuted
      System.out.println("ERROR in FilesFilter - unsupported extension");
    }
    return "";
  }

  /**
   * Gets the extension of a file.
   *
   * @param f the file
   * @return its extension
   */
  private String getExtension(File f) {
    String ext = null;
    String s = f.getName();
    int i = s.lastIndexOf('.');
    if (i > 0 && i < s.length() - 1) {
      ext = s.substring(i + 1).toLowerCase();
    }
    return ext;
  }
}
```

admintool.InformationFieldProperties.java

```
package admintool;

import java.util.*;

import java.awt.*;
```

```
import java.awt.event.*;
import javax.swing.*;
import javax.swing.tree.*;

import org.jdom.*;
import org.jdom.filter.*;
import admintool.jdomtreemodel.*;
import com.borland.jbcl.layout.*;
import javax.swing.event.DocumentListener;
import javax.swing.event.DocumentEvent;
import javax.swing.event.ChangeListener;
import javax.swing.event.ChangeEvent;

/**
 * A Dialog for displaying and manipulating an Information Field's Properties.
*
 *
 */

public class InformationFieldProperties
    extends JDialog {

  //data variables
  /**TreePath to the informationField element*/
  private TreePath path = null;
  /**JDOMTreeModel of the JTree*/
  private JDOMTreeModel treeModel = null;
  /**The Information Field. The "data" of this class.*/
  private Element el = null;
  /**The information Category of the Field.*/
  private String parentName = null;
  /**Name of the Field.*/
  private String name = null;
  /**Type of the Field*/
  private String type = null;
  /**Predefined values property of the Field*/
  private boolean predefined = false;
  /**Has value range property of the Field*/
  private boolean hasvaluerange = false;
  /**If it is not associatied, then it is true. */
  private boolean No = false;
  /**If it is associated, then true.*/
  private boolean Yes = false;
  //to track data variables' modifications
  /**The new value that the user entered in radio button No. The user input here
   * is kept in different variable in order to know if save is needed on exit.
   * @see modifyInformationField()*/
  private boolean inputNo = false;
  /**The new value that the user entered in radio button Yes. The user input here
   * is kept in different variable in order to know if save is needed on exit.
   * @see modifyInformationField()*/
  private boolean inputYes = false;
  /**Association Path of the Information Field , which is displayed in the the JTree.*/
  private String associationPath = null;
  /**XPath of the Information Field.*/
  private String xPath = null;
  /**The file or the Data Source where the XPath point to.*/
  private String source = null;
  /** In JDOM both Text and CDATA are considered to be Text content. So content is filtered to
be only Text.*/
  Filter textFilter = new ContentFilter(ContentFilter.TEXT);

  private ChangeListenerImpl changeListener = null;
  /**Status variable that indicates whether or not the field is modified by user and thus needs
to be saved.*/
  private boolean propertiesChanged = false;

  //GUI Components
  JPanel panel1 = null;
  /**Information Category of the Information Field.*/
  JTextField jTextFieldInfoCat = null;
  JLabel jLabelInfoCat = null;
  JLabel jLabelField = null;
  /**Name of the Information Field.*/
  JTextField jTextFieldName = null;
  JLabel jLabelType = null;
  /**Type of the Information Field.*/
```

```
  JTextField jTextFieldType = null;
  /**Predefine values property of the Information Field.*/
  JCheckBox jCheckBoxPredefined = null;
  /**Value range property of the Information Field.*/
  JCheckBox jCheckBoxValueRange = null;
  /**OK button.*/
  JButton jButtonOK = null;
  /**Cancel button.*/
  JButton jButtonCancel = null;
  XYLayout xYLayout1 = null;
  ButtonGroup buttonGroup1 = null;
  /**Apply button.*/
  JButton jButtonApply = null;
  JPanel jPanel1 = new JPanel();
  /**Associated property of the Information Field.*/
  JRadioButton jRadioButtonNo = null;
  JLabel jLabelAssociated = null;
  /**Associated property of the Information Field.*/
  JRadioButton jRadioButtonYes = null;
  /**Association path of the Association. This is the path displayed in the
   * JTree. It also is refered as display path.*/
  JTextField jTextFieldAssocPath = null;
  JLabel jLabelXPath = null;
  /**XPath of the Association.*/
  JTextField jTextFieldXPath = null;
  JLabel jLabelAssocPath = null;
  XYLayout xYLayout2 = null;
  JLabel jLabelSource = null;
  /**Source where the XPath points.*/
  JTextField jTextFieldSource = null;

  //Coloring
  /**Color used.*/
  Color informationFieldNotAssociatedColor,
      informationFieldAssociationIncompleteColor,
      informationFieldAssociatedColor;
  //Imaging
  /**Icons used.*/
  ImageIcon CategoryIcon, FieldIcon, typeIcon, predefinedValuesIcon,
      valueRangeIcon, associatedIcon, associationPathIcon, xPathIcon, okIcon,
      applyIcon, cancelIcon;


  /**
   * Constructor.<br>
   * Does an initialization and then calls the jbInit method to set up the GUI.
   *
   * @param frame the parent frame, that is the AdminToolFrame
   * @param title the title
   * @param modal modal option
   * @param element the informationField element.
   * @param tm the JDOMTreeModel of the JTree
   * @param p the TreePath to the informationField element.
   */
  public InformationFieldProperties(Frame frame, String title, boolean modal,
                                    Element element, JDOMTreeModel tm,
                                    TreePath p) {
    //call super class (JDialog) constructor to create the Dialog
    super(frame, title, modal);
    //initialize data  variables
    el = element;
    treeModel = tm;
    path = p;
    //Initialize ChangeListener
    changeListener = new ChangeListenerImpl();
    //read Information Field Values
    readInformationFieldValues();

    //Initialization of Color and Imaging
    informationFieldNotAssociatedColor = Color.red;
    informationFieldAssociationIncompleteColor = new Color(200, 100, 0); //       dark orange
    informationFieldAssociatedColor = new Color(0, 128, 0); // dark green
    CategoryIcon = new
ImageIcon(InformationFieldProperties.class.getResource("informationCategoryExpanded.gif"));
    FieldIcon = new
ImageIcon(InformationFieldProperties.class.getResource("informationField.gif"));
    typeIcon = new ImageIcon(InformationFieldProperties.class.getResource("About16.gif"));
```

```
    predefinedValuesIcon = new
ImageIcon(InformationFieldProperties.class.getResource("predefinedValues.gif"));
    valueRangeIcon = new ImageIcon(InformationFieldProperties.class.getResource("Add16.gif"));
    associatedIcon = new
ImageIcon(InformationFieldProperties.class.getResource("associationProperties.gif"));
    associationPathIcon = new
ImageIcon(InformationFieldProperties.class.getResource("databaseField.gif"));
    xPathIcon = new
ImageIcon(InformationFieldProperties.class.getResource("ColumnInsertAfter16.gif"));
    okIcon = new ImageIcon(InformationFieldProperties.class.getResource("done.gif"));
    applyIcon = new ImageIcon(InformationFieldProperties.class.getResource("Save16.gif"));
    cancelIcon = new ImageIcon(InformationFieldProperties.class.getResource("cancel.gif"));

    //call jbInit to set up GUI
    try {
      jbInit();
      //pack and show
      pack();
      setVisible(true);
    }
    catch (Exception ex) {
      ex.printStackTrace();
    }
  }//end constructor

  /**
   * Reads Information Field Properties' values and initializes the Dialog's variables. <br>
   * When reading XPath the following must be considered: In JDOM both Text and
   * CDATA are considered to be Text content. So when filter content to be only Text.
   */
  private void readInformationFieldValues() {

    //information Category name
    parentName = el.getParentElement().getAttribute("name").getValue();
    //name
    name = el.getAttribute("name").getValue();
    //type
    type = el.getAttribute("type").getValue();
    //predefined
    String predef = el.getAttribute("haspredefinedvalues").getValue();
    if (predef.equals("false")) {
      predefined = false;
    }
    else {
      predefined = true;
    }
    //hasvaluerange
    String valrang = el.getAttribute("hasvaluerange").getValue();
    if (valrang.equals("false")) {
      hasvaluerange = false;
    }
    else {
      hasvaluerange = true;
    }
    //associated
    String associated = el.getAttribute("associated").getValue();
    if (associated.equals("No")) {
      No = true;
    }
    else if (associated.equals("Yes")) {
      Yes = true;
    }
    //if Yes, it is associated and so read the source, association path and XPath too.
    if (Yes == true) {
      //source
      source = el.getAttribute("source").getValue();
      //associationPath
      associationPath = el.getAttribute("associationPath").getValue();
      //getting XPath
      /* In JDOM both Text and CDATA are considered to be Text content. So  filter content
       * to be only Text.*/
      //get content with filtering
      java.util.List list = el.getContent(textFilter);
      Iterator i = list.iterator();
      Object o = null;
      while (i.hasNext()) {
        o = i.next();
```

```
      }
      //it will have only one Text node that is why the following if is outside while
      if (o instanceof Text) {
        Text text = (Text) o;
        xPath = text.getText();
      }
      else {
        System.err.println("ERROR in INformationFieldProperties.readInformationFieldValues
method - No Text node found!");
      }
      //end of getting XPath
    }
    //if No variables will be a empty
    else {
      source="";
      associationPath = "";
      xPath = "";
    }
  }//end readInformationFieldValues
  /**
   * Sets up the GUI. <br>
   * Sets the components' properties and lays them out. Change Listeners and Document Listeners
   * are added to the GUI componets, so as to know when data are modified. <br>
   * Most of this code is automatically generated by JBuilder. Especially the
   * laying out of the components.<br>
   * This is the JBuilders method for setting up the GUI. It is necessary to
   * use this method it order to be able to modify the  GUI from JBuilder's
   * Design view.
   *
   * @throws Exception
   */
  private void jbInit() throws Exception {

    panel1 = new JPanel();
    XYLayout xYLayout2 = new XYLayout();
    jLabelInfoCat = new JLabel();
    jTextFieldInfoCat = new JTextField();
    jLabelField = new JLabel();
    jTextFieldName = new JTextField();
    jLabelType = new JLabel();
    jLabelSource= new JLabel();
    jTextFieldType = new JTextField();
    jCheckBoxPredefined = new JCheckBox("Predefined values", predefined);
    jCheckBoxValueRange = new JCheckBox("Value range", hasvaluerange);
    buttonGroup1 = new ButtonGroup();
    jButtonOK = new JButton();
    jButtonCancel = new JButton();
    jButtonApply = new JButton();
    jRadioButtonNo = new JRadioButton("No", No);
    jLabelAssociated = new JLabel();
    jRadioButtonYes = new JRadioButton("Yes", Yes);
    jTextFieldAssocPath = new JTextField();
    jLabelXPath = new JLabel();
    jTextFieldXPath = new JTextField();
    jTextFieldSource= new JTextField();
    jLabelAssocPath = new JLabel();
    jButtonApply.setEnabled(false);
    xYLayout1 = new XYLayout();
    panel1.setLayout(xYLayout1);
    jTextFieldInfoCat.setText(parentName); //
    jTextFieldInfoCat.setEditable(false);
    jLabelInfoCat.setHorizontalAlignment(SwingConstants.RIGHT);
    jLabelInfoCat.setText("Information Category");
    jLabelInfoCat.setIcon(CategoryIcon);
    jLabelField.setHorizontalAlignment(SwingConstants.RIGHT);
    jLabelField.setIcon(FieldIcon);
    jLabelField.setText("Information Field");
    jTextFieldName.setToolTipText("");
    jTextFieldName.setText(name); //
    jLabelType.setHorizontalAlignment(SwingConstants.RIGHT);
    jLabelType.setIcon(typeIcon);
    jLabelType.setText("Type");
    jTextFieldType.setToolTipText("");
    jTextFieldType.setSelectionEnd(11);
    jTextFieldType.setSelectionStart(11);
    jTextFieldType.setText(type); //
    jButtonOK.setIcon(okIcon);
```

```
    jButtonOK.setText("OK");
    jButtonOK.addActionListener(new
                                InformationFieldProperties_jButtonOK_actionAdapter(this));
    jButtonCancel.setIcon(cancelIcon);
    jButtonCancel.setText("Cancel");
    jButtonCancel.addActionListener(new

InformationFieldProperties_jButtonCancel_actionAdapter(this));
    jTextFieldSource.setEditable(false);
    if (No) {
      jTextFieldAssocPath.setEditable(false);
      jTextFieldXPath.setEditable(false);

    }
    panel1.setMinimumSize(new Dimension(630, 290));
    panel1.setPreferredSize(new Dimension(515, 300));
    panel1.setRequestFocusEnabled(true);
    panel1.setToolTipText("");
    this.setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
    this.setResizable(false);
    jButtonApply.setIcon(applyIcon);
    jButtonApply.setText("Apply");
    jButtonApply.addActionListener(new
                                   InformationFieldProperties_jButtonApply_actionAdapter(this));
    jRadioButtonNo.setForeground(informationFieldNotAssociatedColor);
    jRadioButtonNo.addActionListener(new

InformationFieldProperties_jRadioButtonNo_actionAdapter(this));
    jRadioButtonNo.addChangeListener(changeListener);
    jLabelAssociated.setHorizontalAlignment(SwingConstants.RIGHT);
    jLabelAssociated.setIcon(associatedIcon);
    jLabelAssociated.setText("Associated");
    jRadioButtonYes.setForeground(informationFieldAssociatedColor);
    jRadioButtonYes.addActionListener(new

InformationFieldProperties_jRadioButtonYes_actionAdapter(this));
    jRadioButtonYes.addChangeListener(changeListener);
    jTextFieldAssocPath.setText(associationPath);
    jTextFieldSource.setText(source);
    jTextFieldAssocPath.setHorizontalAlignment(SwingConstants.RIGHT);
    jLabelXPath.setHorizontalAlignment(SwingConstants.RIGHT);
    jLabelXPath.setHorizontalTextPosition(SwingConstants.TRAILING);
    jLabelXPath.setIcon(xPathIcon);
    jLabelXPath.setText("XPath");
    jTextFieldXPath.setSelectionStart(11);
    jTextFieldXPath.setText(xPath);
    jTextFieldXPath.setHorizontalAlignment(SwingConstants.RIGHT);
    jLabelAssocPath.setHorizontalAlignment(SwingConstants.RIGHT);
    jLabelAssocPath.setHorizontalTextPosition(SwingConstants.TRAILING);
    jLabelAssocPath.setIcon(associationPathIcon);
    jLabelAssocPath.setText("Association Path");
    jPanel1.setLayout(xYLayout2);
    jPanel1.setBorder(BorderFactory.createEtchedBorder());
    jLabelSource.setRequestFocusEnabled(true);
    jLabelSource.setText("Source:");
    panel1.add(jLabelInfoCat, new XYConstraints(13, 15, 152, 19));
    panel1.add(jLabelField, new XYConstraints(13, 41, 152, 19));
    panel1.add(jLabelType,  new XYConstraints(13, 68, 152, 19));
    panel1.add(jCheckBoxPredefined, new XYConstraints(357, 41, 164, 22));
    panel1.add(jTextFieldName, new XYConstraints(172, 41, 173, 22));
    panel1.add(jCheckBoxValueRange, new XYConstraints(357, 68, 164, 22));
    panel1.add(jTextFieldInfoCat, new XYConstraints(172, 15, 173, 22));
    panel1.add(jTextFieldType, new XYConstraints(172, 68, 173, 22));
    jPanel1.add(jTextFieldXPath, new XYConstraints(3, 84, 473, -1));
    jPanel1.add(jLabelAssocPath, new XYConstraints(355, 29, 121, -1));
    jPanel1.add(jTextFieldAssocPath, new XYConstraints(3, 44, 473, -1));
    jPanel1.add(jLabelXPath, new XYConstraints(338, 68, 138, -1));
    jPanel1.add(jTextFieldSource,    new XYConstraints(329, 8, 147, 19));
    jPanel1.add(jLabelSource,  new XYConstraints(255, 8, 59, 19));
    panel1.add(jRadioButtonNo,    new XYConstraints(172, 96, 55, 15));
    panel1.add(jRadioButtonYes,        new XYConstraints(172, 116, 71, 15));
    panel1.add(jPanel1,  new XYConstraints(13, 137, 486, 115));
    panel1.add(jLabelAssociated,   new XYConstraints(75, 96, 90, -1));
    panel1.add(jButtonCancel, new XYConstraints(390, 266, 109, 22));
    panel1.add(jButtonOK, new XYConstraints(162, 266, 109, 22));
    panel1.add(jButtonApply, new XYConstraints(276, 266, 109, 22));
    jTextFieldName.getDocument().addDocumentListener(changeListener);
```

```
    jTextFieldType.getDocument().addDocumentListener(changeListener);
    jCheckBoxPredefined.addChangeListener(changeListener);
    jCheckBoxValueRange.addChangeListener(changeListener);
    jTextFieldAssocPath.getDocument().addDocumentListener(changeListener);
    jTextFieldSource.getDocument().addDocumentListener(changeListener);
    jTextFieldXPath.getDocument().addDocumentListener(changeListener);
    buttonGroup1.add(jRadioButtonYes);
    buttonGroup1.add(jRadioButtonNo);
    this.getContentPane().add(panel1, BorderLayout.WEST);
  }

  /**
   * Cancel button action performed.<br>
   * Close the Dialog window discarding changes.
   *
   * @param e ActionEvent
   */
  void jButtonCancel_actionPerformed(ActionEvent e) {
    //close window
    setVisible(false);
    dispose();
    //discard changes, that means do not save them back to the information field element. So do
nothing.

  }

  /**
   * Yes Radio button action performed. <br>
   * Enables Association path and XPath text field for user to manually add values.
   *
   * @param e ActionEvent
   */
  void jRadioButtonYes_actionPerformed(ActionEvent e) {
    //enable text fields
    jTextFieldAssocPath.setEditable(true);
    //    jTextFieldSource.setEditable(true);
    jTextFieldXPath.setEditable(true);
  }

  /**
   * No Radio button action performed. <br>
   * Disables Association path, XPath and Source text field for user to manually add values.
   *
   * @param e ActionEvent
   */
  void jRadioButtonNo_actionPerformed(ActionEvent e) {
    //disable text fields
    jTextFieldAssocPath.setEditable(false);
    jTextFieldSource.setEditable(false);
    jTextFieldXPath.setEditable(false);
  }

  /**
   * Apply button action performed.<br>
   * <ul>
   * <li>Asks whether to  save modifications to Information field or not.</li>
   * <li>If Yes: </li>
   * <ul>
   * <li>Read values that user entered </li>
   * <li>Modify field </li>
   * <li>Change status variable which indicates that field is modified </li>
   * </ul>
   * <li>if Cancel, do nothing. </li>
   * </ul>
   * @param e ActionEvent
   * @see readInputValues()
   * @see modifyInformationField()
   */
  void jButtonApply_actionPerformed(ActionEvent e) {
    //ask whether to  save or not
    Object[] options = {
        "Yes",
        "Cancel"};
    int n = JOptionPane.showOptionDialog(null,
                                         "Modify Information field?",
                                         "Verify",
                                         JOptionPane.YES_NO_OPTION,
```

```
                                        JOptionPane.QUESTION_MESSAGE,
                                        null,
                                        options,
                                        options[1]);

    //if yes
    if (n == JOptionPane.YES_OPTION) {
      //read values that user entered
      readInputValues();
      //modify field
      modifyInformationField();
      //change status variable which  indicates that field is modified
      propertiesChanged = false;
    }
    //if cancel, do nothing
  }//end jButtonApply_actionPerformed

  /**
   * Modifies the properties' values of the Information Field. <br>
   * There are four cases of modifing the Association of an Information Field.
   * <ul>
   * <li>Field was not associated and user let it as it was (No -> No): Do nothing  (case 0)
</li>
   * <li>Field was not associated and user associated it (No - Yes): Write new parameters (case
1)</li>
   * <li>Field was associated and user removed the association (Yes -> No): Erase parameters
(case 2)</li>
   * <li>Field was associated and whether user let it as it was or modified the association (Yes
-> Yes): Erase the old parameters, write new (case 3)</li>
   * </ul>
   *
   * where parameters are: associated, associationPath, XPath
   */
  private void modifyInformationField() {
    //set name
    el.getAttribute("name").setValue(name);
    //type
    el.getAttribute("type").setValue(type);
    //set predefined
    if (predefined) {
      el.getAttribute("haspredefinedvalues").setValue("true");
    }
    else {
      el.getAttribute("haspredefinedvalues").setValue("false");
    }
    //set hasvaluerange
    if (hasvaluerange) {
      el.getAttribute("hasvaluerange").setValue("true");
    }
    else {
      el.getAttribute("hasvaluerange").setValue("false");
    }

    //cases of associated
    //case 0
    if (No && inputNo) {
      //Do nothing
    }
    //case 1
    else if  (No && inputYes) {
      //Write new parameters
      writeParameters();
    }
    //case 2
    else if (Yes && inputNo) {
     //Erase parameters
     eraseParameters();
   }
   //case 3
   else if (Yes && inputYes) {
      eraseParameters();
      writeParameters();
    }
   //SHOULD NEVER BE EXECUTED
   else {
      System.err.println(
          "ERROR in Inforrmation FieldProperties.modyfyInformationField - Undefined case!");
```

```
    }
  }

  /**
   * Erases the XPath text node and the attributes that are related to it. <br>
   * Helper method used by modifyInformationField method. Finds the Text node and passes
   * it to the JDOMTreeModel.removeNodeFromParent method to do the removal.
   * @see JDOMTreeModel.removeNodeFromParent(Text, TreePath)
   */
  private void eraseParameters() {
    //get the Text node  that is going to be removed
    /*Remainder: In JDOM both Text and
     * CDATA are considered to be Text content. So when filter content to be only Text.*/
    Text text = null;
    java.util.List list = el.getContent(textFilter);
    Iterator i = list.iterator();
    while (i.hasNext()) {
      text = (Text) i.next();
    }
    //call removeNodeFromParent to remove XPath and attributes source and associationPath
    treeModel.removeNodeFromParent(text, path);
  }

  /**
   * Writes the XPath text node and the attributes that are related to it. <br>
   * Helper method used by modifyInformationField method. Creates a dummy Element
   * in which it puts the Text node and the relative attributes. Then it calls the
insertNodeInto
   * to do the insertion. The dummy node is used because insertNodeInto was design to
   * take an Element.
   * @see JDOMTreeModel.insertNodeInto(Element, Element, int, TreePath)
   */
  private void writeParameters() {
    //create a dataElement to put the values that are passed to insertNodeInto
    Element dataElement = new Element("dataElemet");
    dataElement.setAttribute("associationPath", associationPath);
    dataElement.setText(xPath);
    //call treeModel's method to do the modification
    treeModel.insertNodeInto(dataElement, el, 0, path);
  }

  /**
   * Reads the values entered by user and updates the class' data variables.
   * The association status of the Information Field is kept in inputNo and inputYes
   * variables in order to know if save is needed on exit.
   * @see modifyInformationField()
   */
  private void readInputValues() {

    //name
    name = jTextFieldName.getText();
    //type
    type = jTextFieldType.getText();
    //predefined
    predefined = jCheckBoxPredefined.isSelected();
    //valuerange
    hasvaluerange = jCheckBoxValueRange.isSelected();
    //associated & associationPath , xPath
    //if not associatied
    if (jRadioButtonNo.isSelected()) {
      inputNo = true;
      inputYes = false;
    }
    //if associated
    else if (jRadioButtonYes.isSelected()) {
      inputNo = false;
      inputYes = true;
      associationPath = jTextFieldAssocPath.getText();
      source =  jTextFieldSource.getText();
      xPath = jTextFieldXPath.getText();
    }
  }
  /**
   * OK button action performed. <br>
   * <ul>
   * <li>Infrmation Field Properties are changed, ask if whether to save or not:</li>
   * <ul>
```

```
   * <li>if yes: </li>
   * <ul>
   * <li>read values entered by user </li>
   * <li>update field </li>
   * <li>exit dialog </li>
   * </ul>
   * <li>if cancel: do nothing</li>
   * <li>if no: exit dialog without saving</li>
   * </ul>
   * <li>Infrmation Field Properties are not changed: just exit</li>
   * </ul>
   *
   * @param e ActionEvent
   */
  void jButtonOK_actionPerformed(ActionEvent e) {
    //if the properties of the information field are modified
    if (propertiesChanged) {
      //ask if whether to save or not
      Object[] options = {
          "Yes", "No",
          "Cancel"};
      int n = JOptionPane.showOptionDialog(this,
                                           "Modify Information field?",
                                           "Verify",
                                           JOptionPane.YES_NO_OPTION,
                                           JOptionPane.QUESTION_MESSAGE,
                                           null,
                                           options,
                                           options[2]);

      //if yes,
      if (n == JOptionPane.YES_OPTION) {
        //read values entered by user
        readInputValues();
        //update field
        modifyInformationField();
        //exit dialog
        jButtonCancel_actionPerformed(e);
      }
      //if no
      else if (n == JOptionPane.CANCEL_OPTION) {
        //do nothing
        return;
      }
      //if no
      else if (n == JOptionPane.NO_OPTION) {
        //just exit dialod, without saving
        jButtonCancel_actionPerformed(e);
      }

    }
    //properties are not modified
    else {
      //so just exit
      jButtonCancel_actionPerformed(e);
    }
  }
  /**
   * An implementation of abstract class AbsractChangeListenerImpl.
   * See supperclass' API for more.
   */
  private class ChangeListenerImpl extends AbsractChangeListenerImpl {
    /**
     * Actions taken when data on GUI components are modified.<br>
     * Changes status of properties into modified. Enables the Apply button.
     *
     */
    void changesOccured()  {
      propertiesChanged = true;
      jButtonApply.setEnabled(true);
    }
  }

//----------------Action Adapter classes ---------------------------------------
  /**
  * Helper inner class created automatically by development tool. <br>
  * I just calls the corresponding action performed method of AdminToolFrame.
```

```
  */
class InformationFieldProperties_jButtonCancel_actionAdapter
    implements java.awt.event.ActionListener {
  InformationFieldProperties adaptee;

  InformationFieldProperties_jButtonCancel_actionAdapter(
      InformationFieldProperties adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jButtonCancel_actionPerformed(e);
  }
}
/**
  * Helper inner class created automatically by development tool. <br>
  * I just calls the corresponding action performed method of AdminToolFrame.
  */
class InformationFieldProperties_jButtonApply_actionAdapter
    implements java.awt.event.ActionListener {
  InformationFieldProperties adaptee;

  InformationFieldProperties_jButtonApply_actionAdapter(
      InformationFieldProperties adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jButtonApply_actionPerformed(e);
  }
}
/**
  * Helper inner class created automatically by development tool. <br>
  * I just calls the corresponding action performed method of AdminToolFrame.
  */
class InformationFieldProperties_jButtonOK_actionAdapter
    implements java.awt.event.ActionListener {
  InformationFieldProperties adaptee;

  InformationFieldProperties_jButtonOK_actionAdapter(InformationFieldProperties
      adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jButtonOK_actionPerformed(e);
  }
}
/**
  * Helper inner class created automatically by development tool. <br>
  * I just calls the corresponding action performed method of AdminToolFrame.
  */
class InformationFieldProperties_jRadioButtonYes_actionAdapter
    implements java.awt.event.ActionListener {
  InformationFieldProperties adaptee;

  InformationFieldProperties_jRadioButtonYes_actionAdapter(
      InformationFieldProperties adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jRadioButtonYes_actionPerformed(e);
  }
}
/**
  * Helper inner class created automatically by development tool. <br>
  * I just calls the corresponding action performed method of AdminToolFrame.
  */
class InformationFieldProperties_jRadioButtonNo_actionAdapter
    implements java.awt.event.ActionListener {
  InformationFieldProperties adaptee;

  InformationFieldProperties_jRadioButtonNo_actionAdapter(
      InformationFieldProperties adaptee) {
    this.adaptee = adaptee;
  }
```

```
  public void actionPerformed(ActionEvent e) {
    adaptee.jRadioButtonNo_actionPerformed(e);
  }
}
//-------------------------------------------------------------------------------

}//end InformationFiledProperties
```

admintool.ReportViewer.java

```
package admintool;

import java.io.*;
import java.net.*;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.border.*;

import org.jdom.*;
import org.jdom.transform.*;

/**
 * <p>
 * A Dialog for dispaying Report on Information Structure, XML of Information Structure
* and XML of a selection in Data Structure.
 * </p>
 * <p>
 * It is called on comands:
* <ul>
* <li>View Report on Information Structure</li>
* <li>View XML of Information Structure</li>
* <li>View XML of selection in Data Structure</li>
* </ul>
* The term <i>Report</i> is refered to the html generated when the XSL file is applied to the
* XML of the Information Structure.
 * </p>
 */
public class ReportViewer
    extends JDialog {
  //data variables
  /**The displayed document.*/
  private Document doc = null;
  /**The parent frame.*/
  private AdminToolFrame parent = null;
  /**The XSL file used to generate the html report from the Information Structure's XML.*/
  private File xslFile = null;
  /**Indicates whether or not it is a report. */
  private boolean report = false;
  /**The temp file used for displaying in the JEditorPane. JEditorPane displays a page by
loading a file.
   * So in order to display the documents, they are first outputted with XMLOutputter into
   * a file and then they are loaded into the JEditorPane as files.
 */
  private File file = null;

//GUI components
  JPanel panel1 = new JPanel();
  BorderLayout borderLayout1 = new BorderLayout();
  JPanel jPanel1 = new JPanel();
  JPanel jPanel2 = new JPanel();
  JPanel jPanel3 = new JPanel();
  BorderLayout borderLayout2 = new BorderLayout();
  /**OK button.*/
  JButton jButtonOK = new JButton();
  JLabel jLabel1 = new JLabel();
  /**Displaying Bar. Shows what is cirrently been displayed.*/
  JTextField jTextFieldDisplaying = new JTextField();
  BorderLayout borderLayout3 = new BorderLayout();
```

```
  //The Editor pane used to do the displaying.
  JEditorPane jEditorPane = new JEditorPane();
  /**Editor pane's font.*/
  Font editorFont = null;
  /**Icon used*/
  ImageIcon okIcon = null;
  TitledBorder titledBorder1;

  /**
   * Constructor.
   * <br> Does an initialization, then calls the jbInit method to set up the
   * GUI and finally calls the proper mrthod to do the displaying. Also, sets
   * the XSL file used to display the report.
   * <p>
   * The constructor takes only a document. So, when the "View xml of selection
   * in Data Structure" command is enetered, the element that was selected by the user is passed
   * to the constructor through a dummy document.
   * </p>
   *
   *
   * @param frame parent frame
   * @param title title
   * @param modal modal
   * @param doc the document that is going to be displayed.
   * @param report True if it is a report (html) and false if it is just XML.
   */
  public ReportViewer(Frame frame, String title, boolean modal, Document doc,
                      boolean report) {
    //call super class (JDialog) constructor to create the Dialog
    super(frame, title, modal);
    //initialize
    this.doc = doc;
    this.parent = (AdminToolFrame) frame;
    this.report = report;
    //XSL file used to display the report.
    xslFile = new File("./report.xsl");


    //font and image initialization
    editorFont = new Font("Tahoma", Font.PLAIN, 12);
    okIcon = new ImageIcon(ReportViewer.class.getResource("done.gif"));

    //set up GUI
    try {
      jbInit();
      pack();
      setVisible(true);
    }
    catch (Exception ex) {
      ex.printStackTrace();
    }

    //set the test of dislpay bar
    setDisplayingBar();
    //display document
    displayDocument();
  }

  /**
   * Sets the text of the Displaying Bar.<br>
   */
  private void setDisplayingBar() {
    if (doc.getRootElement().getName().equals("informationstructure")) {
      if (report){
        jTextFieldDisplaying.setText("Report on Information Strucure.");
      }
      else {
        jTextFieldDisplaying.setText("XML of Information Strucure.");
      }
    }
    else {
      jTextFieldDisplaying.setText("Selected element on Data Structure: " +
doc.getRootElement().getName());
    }
  }

  /**
```

```
   * Displays the document.<br>
   *JEditorPane displays a page loading a file.
   * So in order to display the documents, they are first outputted with XMLOutputter into
   * a file and then they are loaded into the JEditorPane as files.
   * When a report is to be displayed, the document is first transformed using createReport
method.
   * @see createReport()
   */
  private void displayDocument() {
    //used to display the file to JEditorPane
    URL url = null;

    //if it is a report
    if (report) {
      //create report
      createReport();
      //specify location of temp file
      file = new File("./tempfiles/report.html");
    }
    else {
      //specify location of temp file
      file = new File("./tempfiles/report.xml");
    }

    //output doc to  temp file
    parent.fileOutputXML(doc,file);

    //acquire file's url
    try {
      url = file.toURL();
    }
    catch (MalformedURLException ex1) {
      ex1.printStackTrace();
    }

    //load file using its url
    if (url != null) {
      try {
        jEditorPane.setPage(url);
      }
      catch (IOException e) {
        System.err.println("Attempted to read a bad URL: " + url);
      }
    }
    else {
      System.err.println("Couldn't find file");
    }
  }

  /**
   * Creates the *.html report document from the XML and the XSL files.
   * Loads the XSL file into a document and transforms the document using
   * an org.jdom.transform.XSLTransformer.
   * @see org.jdom.transform.XSLTransformer
   */
  private void createReport() {
    //load dcoument from xml file
    Document xslDoc = parent.loadXML(xslFile);
    //transform doc
    try {
      XSLTransformer transformer = new XSLTransformer(xslDoc);
      doc = transformer.transform(doc);
    }
    //catch errors - show error messages
    catch (XSLTransformException ex) {
      ex.printStackTrace();
      JOptionPane.showMessageDialog(null,
                                    "Error in XSL Transformation. Exception thrown... ",
                                    "Unable to Transform document",
                                    JOptionPane.ERROR_MESSAGE);
    }
  }
  /**
   * Set up GUI. <br>
   * Sets the components' properties and lays them out. <br>
   * Most of this code is automatically generated by JBuilder. Especially the
   * laying out of the components.<br>
```

```
   * This is the JBuilders method for setting up the GUI. It is necessary to
   * use this method it order to be able to modify the  GUI from JBuilder's
   * Design view.
   */
  private void jbInit() throws Exception {
    titledBorder1 = new TitledBorder("");
    jEditorPane.setFont(editorFont);
    jEditorPane.setEditable(false);
    JScrollPane jScrollPane1 = new JScrollPane(jEditorPane);

    panel1.setLayout(borderLayout1);
    jPanel2.setLayout(borderLayout2);
    jButtonOK.setText("Close");
    jButtonOK.addActionListener(new ReportViewer_jButtonOK_actionAdapter(this));
    jButtonOK.setIcon(okIcon);
    jLabel1.setText("Displaying:   ");
    jPanel1.setLayout(borderLayout3);
    jTextFieldDisplaying.setBackground(UIManager.getColor("window"));
    jTextFieldDisplaying.setFont(new java.awt.Font("Dialog", 0, 12));
    jTextFieldDisplaying.setBorder(titledBorder1);
    jTextFieldDisplaying.setEditable(false);
    getContentPane().add(panel1);
    panel1.add(jPanel1, BorderLayout.NORTH);
    jPanel1.add(jLabel1, BorderLayout.WEST);
    jPanel1.add(jTextFieldDisplaying, BorderLayout.CENTER);
    panel1.add(jPanel2, BorderLayout.CENTER);
    jPanel2.add(jScrollPane1, BorderLayout.CENTER);
    panel1.add(jPanel3, BorderLayout.SOUTH);
    jPanel3.add(jButtonOK, null);
    jScrollPane1.setVerticalScrollBarPolicy(
        JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);
    jScrollPane1.setPreferredSize(new Dimension(700, 500));
    jScrollPane1.setMinimumSize(new Dimension(10, 10));
  }

  /**
   * OK button action performed.<br>
   * Deletes the temp file and exits.
   * @param e ActionEvent
   */
  void jButtonOK_actionPerformed(ActionEvent e) {
    //delete temp file
    if(!file.delete()) {
      System.err.println("ERROR in ReportViewer - Temp file could not be deleted.");
    }
    //exit
    dispose();
    setVisible(false);
  }
//---------------------------Action Adapter class -----------------------------
  /**
    * Helper inner class created automatically by development tool. <br>
    * I just calls the corresponding action performed method of AdminToolFrame.
    */
  class ReportViewer_jButtonOK_actionAdapter
    implements java.awt.event.ActionListener {
  ReportViewer adaptee;

  ReportViewer_jButtonOK_actionAdapter(ReportViewer adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.jButtonOK_actionPerformed(e);
  }
}
//-----------------------------------------------------------------------------
}// end ReportViewer
```

admintool.TreeTransferHandler.java

```
package admintool;

import javax.swing.TransferHandler;
import org.jdom.Element;
import java.awt.event.*;
import java.awt.datatransfer.*;
import java.awt.dnd.*;
import javax.swing.JComponent;
import javax.swing.JTree;
import javax.swing.tree.TreeSelectionModel;
import javax.swing.tree.TreePath;
import java.io.*;

import admintool.XMLTransferable;
import admintool.jdomtreemodel.JDOMTreeModel;
import javax.swing.JOptionPane;
import admintool.jdomtreemodel.DisplayedPathEvaluator;
import admintool.jdomtreemodel.XPathEvaluator;
import org.jdom.Attribute;

/**
 * <p>
 * The TransferHanlder that provides the support for associating nodes in the JTree
 * through Drag and Drop (DnD).
 * </p>
 * <p>
 * This is the class that provides the DnD support of the application. After J2SE 1.4,
 * many Swing components provide out-of-the-box support for transferring data. The only thing
 * need to be done is to provide to the application a custom implementation of the
TransferHandler class.
 * Both JTree must have this class to have DnD support.
 * </p>
 * <p>
 * This class is a custom implementation of the TransferHandler for the AdminTool, and does the
following:
 * When dragging an node of the source JTree (Data Structure tree), its XPath and related
properties are tranfered
 * to the destination JTree (Informaion Structure tree). On drop, the XPath is appended
 * to the target Element a Text node. The rest of the tranfered properties are also appended to
the target.
 * </p>
 *<i>Tip: For more see The Java Tutorial:  Creating a GUI with JFC/Swing:
 * How to Use Drag and Drop and Data Transfer </i>
 */
public class TreeTransferHandler
    extends TransferHandler {
  /**Element Flavor, that is a DataFlavor for org.jdom.Element, used in transfering.*/
  DataFlavor elementFlavor;
  /**The string used to identify the MIME type for the Element flavor.*/
  String elementType = DataFlavor.javaJVMLocalObjectMimeType +
      ";class=org.jdom.Element";
  /**Source JTree. Where the Drag occured.*/
  JTree sourceTree = null;
  /**Target JTree. Where the Drop occured.*/
  JTree targetTree = null;
  /**Estimates the associaion path or displayed path of the dragged element.*/
  DisplayedPathEvaluator displayedPathEvaluator = null;
  /**Estimates the XPath of the dragged element.*/
  XPathEvaluator xPathEvaluator = null;

  /**
   * Constructor.
   * Does the initialization.
   */
  public TreeTransferHandler() {
    //create Element dataFlavor
    try {
      //create a DataFlavor for org.jdom.Element, that is an ElementFlavor
      elementFlavor = new DataFlavor(elementType);
    }
    catch (ClassNotFoundException e) {
      System.out.println(
          "TreeTransferHandler: unable to create data flavor");
    }
    //initialize
    displayedPathEvaluator = new DisplayedPathEvaluator();
    xPathEvaluator = new XPathEvaluator();
```

```
  } //end constructor

  /**
   * Returns the type of transfer action (COPY) supported by the source. The source JTree
   * does not need to modified  so a transfer operation of COPY  only should be advertised
   * in our case.
   *
   * @param c JComponent
   * @return COPY action only
   */
  public int getSourceActions(JComponent c) {
    return COPY;
  }

  /**
   * Causes the Swing drag support to be initiated. <br>
   * Just calls the superclass' method.
   * See TransferHandler API for more.
   *
   * @param comp the component holding the data to be transferred; this argument is provided to
enable sharing of TransferHandlers by multiple components
   * @param e  the event that triggered the transfer
   * @param action  the transfer action initially requested; this should be a value of either
COPY or MOVE; the value may be changed during the course of the drag operation
   * @see javax.swing.TransferHandler
   */
  public void exportAsDrag(JComponent comp, InputEvent e, int action) {
    super.exportAsDrag(comp, e, action);
  }

  /**
   * <p>
   * Creates the transfered data when a Drag occurs.
   * </p>
   * Creates a Transferable to use as the source for a data transfer. Returns
   * the representation of the data to be transferred, or null if the
   * component's property is null.<br>
   * The created Transferable is an instance of XMLTransferable for transfering
org.jdom.Element.
   * To make the application expandible, it was best considered that the Transferable
   * should transfer an Element. The reason is that although now only one Text node and two
   * attributes need to be transfered, in future an entire subtree might needed. So,
   * by transfering an Element you can transfer everything else this element is parent of.
   *
   *  <ul>
   * <li>Ensures that only JTrees can initiate a drag</li>
   * <li>Gets selected object. Cases of selected object:</li>
   * <ul>
   * <li>case of: Element</li>
   * <ul>
   * <li>Check if selected element is root of data structure. if yes then abort - user cannot
DnD root</li>
   * <li>create an Element to transfer the data</li>
   * <li>Estimates displayedPath and XPath of selected object.</li>
   * <li>Adds XPath to selected object</li>
   * <li>Adds association path to selected object</li>
   * <ul>
   * <li>if the selected object is a data source field, add association path</li>
   * <li>otherwise  the source tree displays a file structure then association path is the same
with the XPath</li>
   * </ul>
   * <li>Adds source, that is the data source name or the file name that XPath and Assocaion
Path refers to.</li>
   * <li>return an XMLTransferable containing the Element</li>
   * </ul>
   * <li>case of: Attribute</li>
   * <ul>
   * <li>same actions are performed</li>
   * </ul>
   * </ul>
   * </ul>
   *
   * @param c  the component holding the data to be transferred; this argument is provided to
enable sharing of TransferHandlers by multiple components
   * @return the representation of the data to be transferred (XMLTransferable), or null if the
property associated with c  is null
```

```
   * @see XMLTransferable
   */
  protected Transferable createTransferable(JComponent c) {
    Object selectedObject = null;
    //only JTrees can initiate a Drag
    if (c instanceof JTree) {
      sourceTree = (JTree) c;
      //get path of selection
      TreeSelectionModel selectionModel = sourceTree.getSelectionModel();
      TreePath path = selectionModel.getSelectionPath();
      if (path != null) {
        //get selected object
        selectedObject = (Object) path.getLastPathComponent();
      }
      //error case
      else {
        System.err.println("An node is not selected");
      }
      //error case
      if (selectedObject == null) {
        System.err.println("\tselectedObject = null");
        return null;
      }
      //cases of selected object
      //case of: Element
      if (selectedObject instanceof Element) {
        Element elSelection = (Element) selectedObject;
        String elSelectionName = elSelection.getName();
        //check if root of data structure
        if (elSelectionName.equals("dataStructure")) {
          //abort - user cannot DnD root
          return null;
        }
        //create an Element to transfer the data
        Element dataElement = new Element("dataElement");
        //Estimete displayedPath
        Object[] objects = path.getPath();
        String displayedPath = displayedPathEvaluator.getDisplayedPath(objects);
        //Estimate  XPath
        String xPath = xPathEvaluator.getXPath(selectedObject);

        //Add data to element which is going to be transfered
        //add XPath
        dataElement.setText(xPath);
        //Add association path
        //if the source tree displays a data source field, add association path
        if (elSelectionName.equals("dataStructureField")) {
          dataElement.setAttribute("associationPath", displayedPath);
        }
        //otherwise the source tree displays a file structure then associPath = XPath
        else {
          dataElement.setAttribute("associationPath", xPath);
        }
        //add source
        /*The object array contains the objects of the path from the root of the data structure
tree
         to the selection. So, the second object of the array is the data source or file loaded
to
         the tree. So pass it to getSourceName to get its name.*/
        dataElement.setAttribute("source", getSourceName(objects[1]));
        //return an XMLTransferable containing the element
        return new XMLTransferable(dataElement, elementFlavor);
      }
      //case of: Attribute
      else if (selectedObject instanceof Attribute) {
        Attribute attSelection = (Attribute) selectedObject;
        Element parent = attSelection.getParent();
        String parentName = parent.getName();

        //create an Element to transfer the data
        Element dataElement = new Element("dataElement");
        //Estimate displayedPath
        Object[] objects = path.getPath();
        String displayedPath = displayedPathEvaluator.getDisplayedPath(objects);

        //Estimate XPath
        String xPath = xPathEvaluator.getXPath(selectedObject);
```

```
        //Add data to element which is going to be transfered
        //Addd XPath
        dataElement.setText(xPath);
        //if the source tree displays a data source add association path
        if (parentName.equals("dataStructureField")) {
          dataElement.setAttribute("associationPath", displayedPath);
        }
        //otherwise the source tree displays a file structure then associPath = XPath
        else {
          dataElement.setAttribute("associationPath", xPath);
        }
        //add source
        /*The object array contains the objects of the path from the root of the data structure
tree
                to the selection. So, the second object of the array is the data source or file
loaded to
                the tree. So pass it to getSourceName to get its name.*/
        dataElement.setAttribute("source", getSourceName(objects[1]));
        //return an XMLTransferable containing the element
        return new XMLTransferable(dataElement, elementFlavor);
      }
      //error case
      else {
        JOptionPane.showMessageDialog(null,
                                      "Please choose an item that can be associated!",
                                      "This item cannot be associated",
                                      JOptionPane.ERROR_MESSAGE);
      }
    }
    return null;
  } //end of createTransferable

  /**
   * Gets the  data source or file loaded to data structure tree and returns its name.<br>
   * The name of the source is stored in the attribute of the corresponding element.
   *
   * @param object the  data source or file loaded to data structure tree
   * @return name of the source
   */
  private String getSourceName(Object object) {
    String srcString = null;
    Element srcElement = (Element) object;
    //get attribute that contains the source name
    try {
      srcString = srcElement.getAttribute("name").getValue();
    }
    catch (Exception ex) {
      ex.printStackTrace();
    }
    //return it
    return srcString;
  }

  /**
   * Invoked after data has been exported.
   * Dummy implementation. It is not needed for COPY. Only for MOVE to remove the
   * dragged data from source.
   *
   * @param source  the component that was the source of the data
   * @param data The data that was transferred or possibly null if the action is NONE.
   * @param action the actual action that was performed
   */
  protected void exportDone(JComponent source, Transferable data, int action) {
    super.exportDone(source, data, action);
  }

  /**
   * <p>
   * Imports the transfered data to the drop target.
   * </p>
   * Causes a transfer to a component from a DND drop operation.
   * The Transferable represents the data to be imported into the component.<br>
   * <ul>
   * <li>Checks if import can be performed, by calling canImport method.</li>
   * <li>Gets the transfered element from the Transferable</li>
```

```
   * <li>Gets selection path of target tree and then the target object of the target JTree
</li>
   * <li>If target object is a informationfield element, then delegate the insertion
   * of the XPath Text node and the rest of the data to tree model</li>
   * </ul>
   *
   * @param c he component to receive the transfer; this argument is provided to enable sharing
of TransferHandlers by multiple components
   * @param t he data to import
   * @return  if the data was inserted into the component, false otherwise
   *  @see XMLTransferable
   */
  public boolean importData(JComponent c, Transferable t) {
    //vars
    Object oTransfered = null;
    Element elTransfered = null;
    Object oTarget = null;
    Element elTarget = null;
    TreePath path = null;

    //if import cannot be performed
    if (!canImport(c, t.getTransferDataFlavors())) {
      //abort import
      return false;
    }
    //else import can be performed

    targetTree = (JTree) c;
    //get transfered Element
    try {
      oTransfered = (Element) t.getTransferData(elementFlavor);
    }
    catch (IOException ex) {
      System.err.println("ERROR in importData: I/O exception");
      return false;
    }
    catch (UnsupportedFlavorException ufe) {
      System.err.println("ERROR in importData: unsupported data flavor");
      return false;
    }

    if (oTransfered instanceof Element) {
      elTransfered = (Element) oTransfered;
    }
    else {
      System.err.println(
          "ERROR in importData: ERROR in casting transfered object");
    }
    //get selection path of target tree
    TreeSelectionModel selectionModel = targetTree.getSelectionModel();
    path = selectionModel.getSelectionPath();
    //get the target object of the target JTree
    if (path != null) {
      oTarget = (Object) path.getLastPathComponent();
    }
    //error cases
    else {
      System.err.println(
          "ERROR in importData: Failed to get selection path of drop.");
      return false;
    }
    if (oTarget == null) {
      System.err.println("ERROR in importData: Failed to get target object.");
      return false;
    }
    //cases of target object
    //case of: Element
    if (oTarget instanceof Element) {
      elTarget = (Element) oTarget;
      //get name of target element
      String elTargetName = elTarget.getName();
      //error case
      if (!elTargetName.equals("informationfield")) {
        JOptionPane.showMessageDialog(null,
                                      "You can only drop it in a Information Field!",
                                      "Error in dropping",
                                      JOptionPane.ERROR_MESSAGE);
```

```
        return false;
      }
    }
    //no other target object supported
    else {
      System.err.println("ERROR in importData: Not an Element");
      return false;
    }

    //check if source and target tree is the same tree
    if (sourceTree.equals(targetTree)) {
      System.out.println("ERROR - Tree modification is forbitten... :-b ");
      return false;
    }

    //delegate the insertion of the XPath Text node and the rest of the data to tree model
    JDOMTreeModel treeModel = (JDOMTreeModel) targetTree.getModel();
    treeModel.insertNodeInto(elTransfered, elTarget, 0, path);

    return true;
  } //end of importData

  /**
   * Indicates whether the JTree that had the drop,  would accept an import of the given set of
   * data flavors prior to actually attempting to import it.<br>
   * When a JTree has a drop on it, calls this method to find out that the
   * data flavors of the Transferable object matches the data flavor that it supports.<br>
   * This method answers the question:<i> Has the target JTree support for this kind of dropped
data?</i>
   *
   * @param c the component to receive the transfer
   * @param flavors the data formats available
   * @return true if the data can be inserted into the component, false otherwise
   */
  public boolean canImport(JComponent c, DataFlavor[] flavors) {
    //find out if transferable has at least one DataFlavor that matches the target JTree Data
Flavor
    for (int i = 0; i < flavors.length; i++) {
      if (flavors[i].equals(elementFlavor)) {
        return true;
      }
    }
    return false;
  }

} //end of class
```

admintool.XMLTransferable.java

```
package admintool;

import org.jdom.Element;
import java.awt.datatransfer.Transferable;
import java.awt.datatransfer.DataFlavor;
import java.awt.datatransfer.UnsupportedFlavorException;

/**
 * <p>
 * Transferable interface implementation that supports the transfer of org.jdom.Element
objects.<br>
 * Every custom TransferHandler implementation needs an implementation of Transferable
 * interface to wrap the transfered object. This class is an implementation that
 * supports the transfer of org.jdom.Element objects.
 * </p>
 *
 */
public class XMLTransferable
    implements Transferable {
  /**Element that is transfered.*/
  Element dataElement = null;
  /**Element Flavor. Flavor of the transfered data.*/
  DataFlavor elementFlavor = null;
```

```
  /**
   * Constructor. Does the initialization. DataFlavor is also passed to
   * constructor in order to be easier to add support of more flavors in the
   * future .
   *
   * @param o the transfered data.
   * @param elFl the DataFlavor of the transfered data
   */
  public XMLTransferable(Object o, DataFlavor elFl) {
    dataElement = (Element) o;
    elementFlavor = elFl;
  }

  /**
   * Returns an array with the Element Flavor.
   *
   * @return array with the Element Flavor.[]
   */
  public DataFlavor[] getTransferDataFlavors() {
    return new DataFlavor[] {
        elementFlavor};
  }

  /**
   * Returns whether or not the Element flavor is supported for this object.
   *
   * @param flavor the DataFlavor of the object.
   * @return whether or not the Element flavor is supported for this object.
   */
  public boolean isDataFlavorSupported(DataFlavor flavor) {
    if (elementFlavor.equals(flavor)) {
      return true;
    }
    return false;
  }

  /**
   * Returns the transfered Element. The class of the object returned is
   * defined by the representation class of the flavor.
   *
   * @param flavor DataFlavor
   * @return the transfered object
   */
  public Object getTransferData(DataFlavor flavor) throws
      UnsupportedFlavorException {
    if (!isDataFlavorSupported(flavor)) {
      throw new UnsupportedFlavorException(flavor);
    }
    return dataElement;
  }
}
```

admintool.XMLTransformer.java

```
package admintool;

import org.jdom.Document;

/**
 *  This is the interface that XML_JDOMTransformLoader class needs a class to implement, in
order for
 *  that class to be considered a Transformation class.<br>
 *  It is used becasue there is not a priory knowledge of what transformer classes might be
available in the
 *  future.
 *
 */
public interface XMLTransformer {
  /**
   * Transforms a document into another document.
   * Call that method to do the transformation.
```

```
   * @param inputDoc document that is going to be transformed
   * @return the transformed document
   */
  public Document transform(Document inputDoc);

  /**
   * Returns the name of the class displayed the XML_JDOMTransformer's comboBox.
   * @return The transformation class name
   */
  public String toString();

  /**
   * Sets the name of the class returned with toString method.
   * @param displayName the name of the class
   */
  public void setName(String displayName);

}
```

admintool.XML_JDOMTransformLoader.java

```
package admintool;

import java.io.*;
import java.util.*;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

import org.jdom.*;
import com.borland.jbcl.layout.*;
import org.jdom.output.XMLOutputter;
import org.jdom.output.Format;

/**
 * <p>
 * Load generic XML file through JDOM transformation Dialog.<br>
 * Prompts the user to choose a a generic XML and a transformation class  Xto transform the
 * generic XML in the appropriate structure, and then does the transform. The transformed
 * the document is returned to the AdminToolFrame with a getter method.
 * </p>
 */

public class XML_JDOMTransformLoader
    extends JDialog {
  //data variables
  /**Parent frame.*/
  private AdminToolFrame parent = null;

  /**Generic XML file.*/
  private Document xmlDoc = null;

  /**The transformed document.*/
  private Document transformedDoc = null;

  /**The transformation classes list that the combo box displays.*/
  private Vector xmlTransformersList = new Vector();

  /**The Properties file which indicates which classes must be loaded.*/
  private File propertiesFile = null;

  JPanel panel1 = new JPanel();
  BorderLayout borderLayout1 = new BorderLayout();
  JPanel jPanelMain = new JPanel();
  JPanel jPanelTitle = new JPanel();
  BorderLayout borderLayout2 = new BorderLayout();
  JLabel jLabelTitle = new JLabel();
  JPanel jPanelGrid = new JPanel();
  GridLayout gridLayout1 = new GridLayout();
  JPanel jPanelButtons = new JPanel();
  /**OK button.*/
```

```
JButton jButtonOK = new JButton();
/**Cancel button.*/
JButton jButtonCancel = new JButton();
JPanel jPanelXML = new JPanel();
JPanel jPanelJDOM = new JPanel();
XYLayout xYLayout2 = new XYLayout();
JLabel jLabelXML = new JLabel();
/**File path of XML file.*/
JTextField jTextFieldPath1 = new JTextField();
/**Choose a generic XML file button.*/
JButton jButtonXML = new JButton();
XYLayout xYLayout3 = new XYLayout();
JLabel jLabelPath1 = new JLabel();
FlowLayout flowLayout1 = new FlowLayout();
JLabel jLabelJDOMTransformClass = new JLabel();
FlowLayout flowLayout2 = new FlowLayout();
/**Combo box with the transformation classes*/
JComboBox jComboBoxTransformers = new JComboBox(xmlTransformersList);
/**The file chooser used in this dialog.*/
JFileChooser jFileChooserXMLXSLT = new JFileChooser();
/**File filter for xml files *.xml used in JFileChooser*/
FilesFilter xmlFileFilter = new FilesFilter("xml"); //data struct
/**Icons used.*/
ImageIcon xmlIcon, jdomIcon, folderIcon, okIcon, cancelIcon;

/**
 * Constructor.<br>
 * Does a general initialization, initializes the Transformation Classes List and
 * then calls the jbInit method to set up the GUI.<br>
 * Here is set the ile path of Proprties file.
 *
 * @param frame parent frame
 * @param title the title
 * @param modal modal
 */
public XML_JDOMTransformLoader(Frame frame, String title, boolean modal) {
  super(frame, title, modal);
  //iitialize
  parent = (AdminToolFrame) frame;
  //file path of Proprties file
  propertiesFile = new File("./XMLTransformers/transformers.properties");
  //Initializes the Transformation Classes List.
  initializeXmlTransformers();

  //loading imaeges
  xmlIcon = new ImageIcon(XML_JDOMTransformLoader.class.getResource(
      "xml.gif"));
  jdomIcon = new ImageIcon(XML_JDOMTransformLoader.class.getResource(
      "code20.gif"));
  folderIcon = new ImageIcon(XML_JDOMTransformLoader.class.getResource(
      "Open16.gif"));
  okIcon = new ImageIcon(XML_JDOMTransformLoader.class.getResource(
      "done.gif"));
  cancelIcon = new ImageIcon(XML_JDOMTransformLoader.class.getResource(
      "cancel.gif"));

  //set up GUI.
  try {
    jbInit();
    pack();
    setVisible(true);
  }
  catch (Exception ex) {
    ex.printStackTrace();
  }
}

/**
 * <p>
 * Initializes the Transformation Classes List.
 * Loads the Properties file that contains the names of the classes that must be loaded. Loads
 * each class at runtime, instanciates an object, cast it to be an XMLTransformer and
 * adds it to the list.<br>
 * The directory of the *.class files that are loaded must be in the CLASSPATH.
 * </p>
 * <p>
 * Future support for new transformers must be added here.
```

```
 * Transformation class MUST implement interface XMLTransformer.
 * </p>
 * @see XMLTransformer
 */
private void initializeXmlTransformers() {
  String className;
  XMLTransformer transformer = null;

  //load Properties file
  Properties transformers = new Properties();
  try {
    FileInputStream input = new FileInputStream(propertiesFile);
    transformers.load(input);
    input.close();
  }
  catch (IOException ioException) {
    ioException.printStackTrace();
  }
  catch (IllegalArgumentException ilex) {
    ilex.printStackTrace();
  }

  //Get Emumeration to parse Properties
  Enumeration enumeration = transformers.propertyNames();
  while (enumeration.hasMoreElements()) {
    //get class name from enumeration
    className = transformers.getProperty(enumeration.nextElement().toString());
    //load class at RUNTIME
    try {
      //load class at runtime
      Class transformerClass = Class.forName(className);
      //create object from class that was loaded at runtime
      Object transformerObject = transformerClass.newInstance();
      //cast it to XMLTransformer Interface
      transformer = (XMLTransformer) transformerObject;
      //set name to be displayed in combo box
      transformer.setName(className);
      //add it on the list
      xmlTransformersList.add(transformer);
    }
    catch (Exception e) {
      e.printStackTrace();
    }
  } //end while
}

/**
 * Set up GUI. <br>
 * Sets the components' properties and lays them out. <br>
 * Initializes the combo box with the Transformation Classes List.<br>
 * Most of this code is automatically generated by JBuilder. Especially the
 * laying out of the components.<br>
 * This is the JBuilders method for setting up the GUI. It is necessary to
 * use this method it order to be able to modify the  GUI from JBuilder's
 * Design view.
 */
private void jbInit() throws Exception {
  panel1.setLayout(borderLayout1);
  jPanelMain.setLayout(borderLayout2);
  jLabelTitle.setPreferredSize(new Dimension(250, 34));
  jLabelTitle.setToolTipText("");
  jLabelTitle.setText("...using a JDOM class for transformation");
  jPanelTitle.setLayout(flowLayout1);
  jPanelGrid.setLayout(gridLayout1);
  gridLayout1.setColumns(1);
  gridLayout1.setHgap(10);
  gridLayout1.setRows(2);
  gridLayout1.setVgap(5);
  jPanelButtons.setLayout(flowLayout2);
  jPanelButtons.setPreferredSize(new Dimension(30, 30));
  jButtonOK.setText("Load");
  jButtonOK.addActionListener(new
                              XML_JDOMTransformLoader_jButtonOK_actionAdapter(this));
  jButtonOK.setMaximumSize(new Dimension(100, 25));
  jButtonOK.setMinimumSize(new Dimension(100, 25));
  jButtonOK.setPreferredSize(new Dimension(100, 25));
  jButtonOK.setIcon(okIcon);
```

```
  jButtonOK.setEnabled(false);
  jButtonCancel.setText("Cancel");
  jButtonCancel.addActionListener(new
                                  XML_JDOMTransformLoader_jButtonCancel_actionAdapter(this));
  jButtonCancel.setMaximumSize(new Dimension(100, 25));
  jButtonCancel.setMinimumSize(new Dimension(100, 25));
  jButtonCancel.setPreferredSize(new Dimension(100, 25));
  jButtonCancel.setIcon(cancelIcon);
  borderLayout2.setHgap(5);
  borderLayout2.setVgap(5);
  jPanelGrid.setBorder(BorderFactory.createEtchedBorder());
  jPanelXML.setBorder(BorderFactory.createEtchedBorder());
  jPanelXML.setPreferredSize(new Dimension(190, 66));
  jPanelXML.setLayout(xYLayout2);
  jPanelJDOM.setBorder(BorderFactory.createEtchedBorder());
  jPanelJDOM.setLayout(xYLayout3);
  jPanelMain.setPreferredSize(new Dimension(270, 250));
  jLabelXML.setText("Choose a generic XML file");
  jLabelXML.setIcon(xmlIcon);
  jTextFieldPath1.setBackground(SystemColor.window);
  jTextFieldPath1.setFont(new java.awt.Font("Dialog", 1, 11));
  jTextFieldPath1.setBorder(BorderFactory.createLoweredBevelBorder());
  jTextFieldPath1.setEditable(false);
  jTextFieldPath1.setText("");
  jButtonXML.setBorder(BorderFactory.createRaisedBevelBorder());
  jButtonXML.setToolTipText("Choose a generic XML file");
  jButtonXML.setIcon(folderIcon);
  jButtonXML.setSelectedIcon(null);
  jButtonXML.setText("");
  jButtonXML.addActionListener(new
                               XML_JDOMTransformLoader_jButtonXML_actionAdapter(this));
  xYLayout3.setHeight(0);
  jLabelPath1.setFont(new java.awt.Font("Dialog", 0, 11));
  jLabelPath1.setHorizontalAlignment(SwingConstants.LEADING);
  jLabelPath1.setHorizontalTextPosition(SwingConstants.TRAILING);
  jLabelPath1.setText("Path:");
  panel1.setPreferredSize(new Dimension(270, 250));
  jLabelJDOMTransformClass.setIcon(jdomIcon);
  jLabelJDOMTransformClass.setText("Choose a JDOM Transformation class");
  flowLayout2.setHgap(1);
  flowLayout2.setVgap(1);
  this.setResizable(false);
  getContentPane().add(panel1);
  panel1.add(jPanelMain, BorderLayout.CENTER);
  jPanelMain.add(jPanelTitle, BorderLayout.NORTH);
  jPanelTitle.add(jLabelTitle, null);
  jPanelMain.add(jPanelGrid, BorderLayout.CENTER);
  jPanelMain.add(jPanelButtons, BorderLayout.SOUTH);
  jPanelGrid.add(jPanelXML, null);
  jPanelXML.add(jLabelXML, new XYConstraints(8, 6, 183, 27));
  jPanelXML.add(jLabelPath1, new XYConstraints(39, 40, 34, 22));
  jPanelXML.add(jTextFieldPath1, new XYConstraints(75, 41, 167, -1));
  jPanelXML.add(jButtonXML, new XYConstraints(190, 6, 52, 27));
  jPanelGrid.add(jPanelJDOM, null);
  jPanelJDOM.add(jLabelJDOMTransformClass, new XYConstraints(10, 7, 248, 27));
  jPanelJDOM.add(jComboBoxTransformers, new XYConstraints(35, 39, 208, 25));
  jPanelButtons.add(jButtonOK, null);
  jPanelButtons.add(jButtonCancel, null);
  jComboBoxTransformers.setSelectedItem(xmlTransformersList.get(0));
}

/**
 * Cancel button action performed. <br>
 * Exits the Dialog.
 * @param e ActionEvent
 */
void jButtonCancel_actionPerformed(ActionEvent e) {
  setVisible(false);
  dispose();
}

/**
 * Choose a generic XML file action performed. <br>
 * Use a file chooser to chooser a file. Set up file chooser with filefilter and
 * title, get file, load it by calling loadXML method, show file path in corresponding
 * text field and reset filechooser.
 *
```

```
   * @param e ActionEvent
   * @see AdminToolFrame.loadXML(File)
   */
  void jButtonXML_actionPerformed(ActionEvent e) {
    //Use a fileChooser to choose the file
    File file = null;
    //set up file chooser
    jFileChooserXMLXSLT.setDialogTitle("Choose a XML file");
    jFileChooserXMLXSLT.addChoosableFileFilter(xmlFileFilter);
    if (JFileChooser.APPROVE_OPTION == jFileChooserXMLXSLT.showOpenDialog(this)) {
      // get file.
      file = jFileChooserXMLXSLT.getSelectedFile();
      //test if it is the proper file type
      if (jFileChooserXMLXSLT.accept(file)) {
        //load it as a JDOM document
        xmlDoc = parent.loadXML(file);
        //show file path in corresponding text field
        jTextFieldPath1.setText(file.getAbsolutePath());
        //enable OK button
        jButtonOK.setEnabled(true);
      }
      else {
        JOptionPane.showMessageDialog(null,
                                      "Please choose a XML (*.xml) file.",
                                      "Wrong file type",
                                      JOptionPane.ERROR_MESSAGE);
      }
    }
    //reset file chooser
    jFileChooserXMLXSLT.addChoosableFileFilter(xmlFileFilter);

  }

  /**
   * OK button action performed. <br>
   * Transformes the document using the selected Transformation class and exits.
   * @param e ActionEvent
   */
  void jButtonOK_actionPerformed(ActionEvent e) {
    //get selected transformation class
    XMLTransformer transformer = (XMLTransformer) jComboBoxTransformers.
        getSelectedItem();
    //transform
    transformedDoc = transformer.transform(xmlDoc);
    //exit
    setVisible(false);
    dispose();
  }

  /**
   * Gets the transformed document. <br>
   * Used by AdminToolFrame to get the document that was transformed.
   */
  public Document getTransformedDoc() {
    return transformedDoc;
  }

//----------------------Action Adapter classes---------------------------------
  /**
   * Helper inner class created automatically by development tool. <br>
   * I just calls the corresponding action performed method of AdminToolFrame.
   */

  class XML_JDOMTransformLoader_jButtonCancel_actionAdapter
      implements java.awt.event.ActionListener {
    XML_JDOMTransformLoader adaptee;

    XML_JDOMTransformLoader_jButtonCancel_actionAdapter(XML_JDOMTransformLoader
        adaptee) {
      this.adaptee = adaptee;
    }

    public void actionPerformed(ActionEvent e) {
      adaptee.jButtonCancel_actionPerformed(e);
    }
  }
```

```
  /**
   * Helper inner class created automatically by development tool. <br>
   * I just calls the corresponding action performed method of AdminToolFrame.
   */

  class XML_JDOMTransformLoader_jButtonXML_actionAdapter
      implements java.awt.event.ActionListener {
    XML_JDOMTransformLoader adaptee;

    XML_JDOMTransformLoader_jButtonXML_actionAdapter(XML_JDOMTransformLoader
        adaptee) {
      this.adaptee = adaptee;
    }

    public void actionPerformed(ActionEvent e) {
      adaptee.jButtonXML_actionPerformed(e);
    }
  }

  /**
   * Helper inner class created automatically by development tool. <br>
   * I just calls the corresponding action performed method of AdminToolFrame.
   */

  class XML_JDOMTransformLoader_jButtonOK_actionAdapter
      implements java.awt.event.ActionListener {
    XML_JDOMTransformLoader adaptee;

    XML_JDOMTransformLoader_jButtonOK_actionAdapter(XML_JDOMTransformLoader
        adaptee) {
      this.adaptee = adaptee;
    }

    public void actionPerformed(ActionEvent e) {
      adaptee.jButtonOK_actionPerformed(e);
    }
  }

//-----------------------------------------------------------------------------

} //end XML_JDOMTransfoemLoader
```

admintool.XML_XSLTLoader.java

```
package admintool;

import java.awt.*;
import javax.swing.*;
import com.borland.jbcl.layout.*;
import java.awt.event.*;
import java.io.File;
import org.jdom.Document;
import org.jdom.output.XMLOutputter;
import org.jdom.output.Format;
import org.jdom.transform.XSLTransformer;
import org.jdom.transform.XSLTransformException;
import java.io.IOException;

/**
 * <p>
 * Load generic XML file through XSLT transformation Dialog.<br>
 * Prompts the user to choose a a generic XML and a XSL file to transform the
 * generic XML in the appropriate structure, and then does the transform. The transformed
 *  the document is returned to the AdminToolFrame with a getter method.
 * </p>
 *
 *
 */
public class XML_XSLTLoader
    extends JDialog {
  //data variables
  /**Parent frame.*/
```

```
private AdminToolFrame parent = null;

/**Generic XML file.*/
private Document xmlDoc = null;

/**XSL file.*/
private Document xslDoc = null;

/**The transformed document.*/
private Document transformedDoc = null;

JPanel panel1 = new JPanel();
BorderLayout borderLayout1 = new BorderLayout();
JPanel jPanelMain = new JPanel();
JPanel jPanelTitle = new JPanel();
BorderLayout borderLayout2 = new BorderLayout();
JLabel jLabelTitle = new JLabel();
JPanel jPanelGrid = new JPanel();
GridLayout gridLayout1 = new GridLayout();
JPanel jPanelButtons = new JPanel();
/**OK button.*/
JButton jButtonOK = new JButton();
/**Cancel button.*/
JButton jButtonCancel = new JButton();
JPanel jPanelXML = new JPanel();
JPanel jPanelXSLT = new JPanel();
XYLayout xYLayout2 = new XYLayout();
JLabel jLabelXML = new JLabel();
/**File path of XML file.*/
JTextField jTextFieldPath1 = new JTextField();
/**Choose a generic XML file button.*/
JButton jButtonXML = new JButton();
XYLayout xYLayout3 = new XYLayout();
JLabel jLabelPath1 = new JLabel();
FlowLayout flowLayout1 = new FlowLayout();
/**Choose a XSL file button.*/
JButton jButtonXSLT = new JButton();
/**File path of XSL file.*/
JTextField jTextFieldPath2 = new JTextField();
JLabel jLabelPath2 = new JLabel();
JLabel jLabelXSLT = new JLabel();
FlowLayout flowLayout2 = new FlowLayout();
/**The file chooser used in this dialog.*/
JFileChooser jFileChooserXMLXSLT = new JFileChooser();
/**File filter for xml files *.xml used in JFileChooser*/
FilesFilter xmlFileFilter = new FilesFilter("xml");
/**File filter for xsl files *.xsl used in JFileChooser*/
FilesFilter xslFileFilter = new FilesFilter("xsl");
/**Icons used.*/
ImageIcon xmlIcon, xsltIcon, folderIcon, okIcon, cancelIcon;
/**
 * Constructor. Does an initialization and then calls the jbInit method to set up the GUI.
 *
 * @param frame parent frame
 * @param title the title
 * @param modal modal
 */
public XML_XSLTLoader(Frame frame, String title, boolean modal) {
  //call super class (JDialog) constructor to create the Dialog
  super(frame, title, modal);

  //initialization
  parent = (AdminToolFrame) frame;

  //loading imaeges
  xmlIcon = new ImageIcon(XML_XSLTLoader.class.getResource(
      "xml.gif"));
  xsltIcon = new ImageIcon(XML_XSLTLoader.class.getResource(
      "xslt.gif"));
  folderIcon = new ImageIcon(XML_XSLTLoader.class.getResource(
      "Open16.gif"));
  okIcon = new ImageIcon(XML_XSLTLoader.class.getResource(
      "done.gif"));
  cancelIcon = new ImageIcon(XML_XSLTLoader.class.getResource(
      "cancel.gif"));
  //set up GUI
  try {
```

```
    jbInit();
    pack();
    setVisible(true);
  }
  catch (Exception ex) {
    ex.printStackTrace();
  }
} //end constructor

/**
 * Set up GUI. <br>
 * Sets the components' properties and lays them out. <br>
 * Most of this code is automatically generated by JBuilder. Especially the
 * laying out of the components.<br>
 * This is the JBuilders method for setting up the GUI. It is necessary to
 * use this method it order to be able to modify the  GUI from JBuilder's
 * Design view.
 */
private void jbInit() throws Exception {
  panel1.setLayout(borderLayout1);
  jPanelMain.setLayout(borderLayout2);
  jLabelTitle.setPreferredSize(new Dimension(250, 34));
  jLabelTitle.setToolTipText("");
  jLabelTitle.setText("...through XSLT transformation");
  jPanelTitle.setLayout(flowLayout1);
  jPanelGrid.setLayout(gridLayout1);
  gridLayout1.setColumns(1);
  gridLayout1.setHgap(10);
  gridLayout1.setRows(2);
  gridLayout1.setVgap(5);
  jPanelButtons.setLayout(flowLayout2);
  jPanelButtons.setPreferredSize(new Dimension(30, 30));
  jButtonOK.setText("Load");
  jButtonOK.addActionListener(new XML_XSLTLoader_jButtonOK_actionAdapter(this));
  jButtonOK.setMaximumSize(new Dimension(100, 25));
  jButtonOK.setMinimumSize(new Dimension(100, 25));
  jButtonOK.setPreferredSize(new Dimension(100, 25));
  jButtonOK.setIcon(okIcon);
  jButtonOK.setEnabled(false);
  jButtonCancel.setText("Cancel");
  jButtonCancel.addActionListener(new
                                  XML_XSLTLoader_jButtonCancel_actionAdapter(this));
  jButtonCancel.setMaximumSize(new Dimension(100, 25));
  jButtonCancel.setMinimumSize(new Dimension(100, 25));
  jButtonCancel.setPreferredSize(new Dimension(100, 25));
  jButtonCancel.setIcon(cancelIcon);
  borderLayout2.setHgap(5);
  borderLayout2.setVgap(5);
  jPanelGrid.setBorder(BorderFactory.createEtchedBorder());
  jPanelXML.setBorder(BorderFactory.createEtchedBorder());
  jPanelXML.setPreferredSize(new Dimension(190, 66));
  jPanelXML.setLayout(xYLayout2);
  jPanelXSLT.setBorder(BorderFactory.createEtchedBorder());
  jPanelXSLT.setLayout(xYLayout3);
  jPanelMain.setPreferredSize(new Dimension(270, 250));
  jLabelXML.setText("Choose a generic XML file");
  jLabelXML.setIcon(xmlIcon);
  jTextFieldPath1.setBackground(SystemColor.window);
  jTextFieldPath1.setFont(new java.awt.Font("Dialog", 1, 11));
  jTextFieldPath1.setBorder(BorderFactory.createLoweredBevelBorder());
  jTextFieldPath1.setEditable(false);
  jTextFieldPath1.setText("");
  jButtonXML.setBorder(BorderFactory.createRaisedBevelBorder());
  jButtonXML.setToolTipText("Choose a generic XML file");
  jButtonXML.setIcon(folderIcon);
  jButtonXML.setSelectedIcon(null);
  jButtonXML.setText("");
  jButtonXML.addActionListener(new XML_XSLTLoader_jButtonXML_actionAdapter(this));
  xYLayout3.setHeight(0);
  jLabelPath1.setFont(new java.awt.Font("Dialog", 0, 11));
  jLabelPath1.setHorizontalAlignment(SwingConstants.LEADING);
  jLabelPath1.setHorizontalTextPosition(SwingConstants.TRAILING);
  jLabelPath1.setText("Path:");
  panel1.setPreferredSize(new Dimension(270, 250));
  jButtonXSLT.setText("");
  jButtonXSLT.addActionListener(new XML_XSLTLoader_jButtonXSLT_actionAdapter(this));
  jButtonXSLT.setSelectedIcon(null);
```

```
  jButtonXSLT.setIcon(folderIcon);
  jButtonXSLT.setToolTipText("Choose a XSL file");
  jButtonXSLT.setBorder(BorderFactory.createRaisedBevelBorder());
  jButtonXSLT.setEnabled(false);
  jTextFieldPath2.setText("");
  jTextFieldPath2.setEditable(false);
  jTextFieldPath2.setBackground(SystemColor.window);
  jTextFieldPath2.setFont(new java.awt.Font("Dialog", 1, 11));
  jTextFieldPath2.setBorder(BorderFactory.createLoweredBevelBorder());
  jLabelPath2.setText("Path:");
  jLabelPath2.setHorizontalTextPosition(SwingConstants.TRAILING);
  jLabelPath2.setHorizontalAlignment(SwingConstants.LEADING);
  jLabelPath2.setFont(new java.awt.Font("Dialog", 0, 11));
  jLabelXSLT.setIcon(xsltIcon);
  jLabelXSLT.setText("Choose a XSLT file");
  flowLayout2.setHgap(1);
  flowLayout2.setVgap(1);
  this.setResizable(false);
  getContentPane().add(panel1);
  panel1.add(jPanelMain, BorderLayout.CENTER);
  jPanelMain.add(jPanelTitle, BorderLayout.NORTH);
  jPanelTitle.add(jLabelTitle, null);
  jPanelMain.add(jPanelGrid, BorderLayout.CENTER);
  jPanelMain.add(jPanelButtons, BorderLayout.SOUTH);
  jPanelGrid.add(jPanelXML, null);
  jPanelXML.add(jLabelXML, new XYConstraints(8, 6, 183, 27));
  jPanelXML.add(jLabelPath1, new XYConstraints(39, 40, 34, 22));
  jPanelXML.add(jTextFieldPath1, new XYConstraints(75, 41, 167, -1));
  jPanelXML.add(jButtonXML, new XYConstraints(190, 6, 52, 27));
  jPanelGrid.add(jPanelXSLT, null);
  jPanelXSLT.add(jTextFieldPath2, new XYConstraints(72, 40, 167, -1));
  jPanelXSLT.add(jLabelXSLT, new XYConstraints(10, 7, 180, 27));
  jPanelXSLT.add(jLabelPath2, new XYConstraints(36, 39, 34, 22));
  jPanelXSLT.add(jButtonXSLT, new XYConstraints(187, 7, 52, 27));
  jPanelButtons.add(jButtonOK, null);
  jPanelButtons.add(jButtonCancel, null);
}

/**
 * Cancel button action performed. <br>
 * Exits the Dialog.
 * @param e ActionEvent
 */
void jButtonCancel_actionPerformed(ActionEvent e) {
  setVisible(false);
  dispose();
}

/**
 * Choose a generic XML file action performed. <br>
 * Use a file chooser to chooser a file. Set up file chooser with filefilter and
 * title, get file, load it by calling loadXML method, show file path in corresponding
 * text field and reset filechooser.
 *
 * @param e ActionEvent
 * @see AdminToolFrame.loadXML(File)
 */
void jButtonXML_actionPerformed(ActionEvent e) {
  //Use a fileChooser to choose the file
  File file = null;
  //set up file chooser
  jFileChooserXMLXSLT.setDialogTitle("Choose a XML file");
  jFileChooserXMLXSLT.addChoosableFileFilter(xmlFileFilter);
  if (JFileChooser.APPROVE_OPTION == jFileChooserXMLXSLT.showOpenDialog(this)) {
    // get file
    file = jFileChooserXMLXSLT.getSelectedFile();
    //test if it is the proper file type
    if (jFileChooserXMLXSLT.accept(file)) {
      //load it as a JDOM document
      xmlDoc = parent.loadXML(file);
      //show file path in corresponding text field
      jTextFieldPath1.setText(file.getAbsolutePath());
      //enable XSL button
      jButtonXSLT.setEnabled(true);
    }
    else {
      JOptionPane.showMessageDialog(null,
```

```
                                    "Please choose a XML (*.xml) file.",
                                    "Wrong file type",
                                    JOptionPane.ERROR_MESSAGE);
    }
  }
  //reset file chooser
  jFileChooserXMLXSLT.removeChoosableFileFilter(xmlFileFilter);
}

/**
 * Choose a generic XSL file action performed. <br>
 * Use a file chooser to chooser a file. Set up file chooser with filefilter and
 * title, get file, load it by calling loadXML method, show file path in corresponding
 * text field and reset filechooser.
 *
 * @param e ActionEvent
 * @see AdminToolFrame.loadXML(File)
 */
void jButtonXSLT_actionPerformed(ActionEvent e) {
  //Use a fileChooser to choose the file
  File file = null;
  //set up file chooser
  jFileChooserXMLXSLT.setDialogTitle("Choose a XSL file");
  jFileChooserXMLXSLT.addChoosableFileFilter(xslFileFilter);
  if (JFileChooser.APPROVE_OPTION == jFileChooserXMLXSLT.showOpenDialog(this)) {
    // get file
    file = jFileChooserXMLXSLT.getSelectedFile();
    //test if it is the proper file type
    if (jFileChooserXMLXSLT.accept(file)) {
      //load it as a JDOM document
      xslDoc = parent.loadXML(file);
      //show file path in corresponding text field
      jTextFieldPath2.setText(file.getAbsolutePath());
      //enable OK button
      jButtonOK.setEnabled(true);
    }
    else {
      JOptionPane.showMessageDialog(null,
                                    "Please choose a XSL (*.xsl) file.",
                                    "Wrong file type",
                                    JOptionPane.ERROR_MESSAGE);
    }
  }
  //reset filechooser
  jFileChooserXMLXSLT.removeChoosableFileFilter(xslFileFilter);
}

/**
 * OK button action performed. <br>
 * Transformes the document using JDOM's XSLTransformer and exits.
 * @param e ActionEvent
 * @see org.jdom.transform.XSLTransformer
 */
void jButtonOK_actionPerformed(ActionEvent e) {

  //transform the document
  Document transforedDoc = null;
  try {
    XSLTransformer transformer = new XSLTransformer(xslDoc);
    transformedDoc = transformer.transform(xmlDoc);
  }
  //catch error and show error messages
  catch (XSLTransformException ex) {
    ex.printStackTrace();
    JOptionPane.showMessageDialog(null,
                                  "Error in XSL Transformation. Exception thrown... ",
                                  "Unable to Transform document",
                                  JOptionPane.ERROR_MESSAGE);
  }
  //exit
  setVisible(false);
  dispose();
} //end jButtonOK_actionPerformed

/**
 * Gets the transformed document. <br>
 * Used by AdminToolFrame to get the document that was transformed.
```

```
   */
  public Document getTransformedDoc() {
    return transformedDoc;
  }

//-------------Action Adapter classes ----------------------------------------
  /**
   * Helper inner class created automatically by development tool. <br>
   * I just calls the corresponding action performed method of AdminToolFrame.
   */
  class XML_XSLTLoader_jButtonCancel_actionAdapter
      implements java.awt.event.ActionListener {
    XML_XSLTLoader adaptee;

    XML_XSLTLoader_jButtonCancel_actionAdapter(XML_XSLTLoader adaptee) {
      this.adaptee = adaptee;
    }

    public void actionPerformed(ActionEvent e) {
      adaptee.jButtonCancel_actionPerformed(e);
    }
  }

  /**
   * Helper inner class created automatically by development tool. <br>
   * I just calls the corresponding action performed method of AdminToolFrame.
   */
  class XML_XSLTLoader_jButtonXML_actionAdapter
      implements java.awt.event.ActionListener {
    XML_XSLTLoader adaptee;

    XML_XSLTLoader_jButtonXML_actionAdapter(XML_XSLTLoader adaptee) {
      this.adaptee = adaptee;
    }

    public void actionPerformed(ActionEvent e) {
      adaptee.jButtonXML_actionPerformed(e);
    }
  }

  /**
   * Helper inner class created automatically by development tool. <br>
   * I just calls the corresponding action performed method of AdminToolFrame.
   */
  class XML_XSLTLoader_jButtonXSLT_actionAdapter
      implements java.awt.event.ActionListener {
    XML_XSLTLoader adaptee;

    XML_XSLTLoader_jButtonXSLT_actionAdapter(XML_XSLTLoader adaptee) {
      this.adaptee = adaptee;
    }

    public void actionPerformed(ActionEvent e) {
      adaptee.jButtonXSLT_actionPerformed(e);
    }
  }

  /**
   * Helper inner class created automatically by development tool. <br>
   * I just calls the corresponding action performed method of AdminToolFrame.
   */
  class XML_XSLTLoader_jButtonOK_actionAdapter
      implements java.awt.event.ActionListener {
    XML_XSLTLoader adaptee;

    XML_XSLTLoader_jButtonOK_actionAdapter(XML_XSLTLoader adaptee) {
      this.adaptee = adaptee;
    }

    public void actionPerformed(ActionEvent e) {
      adaptee.jButtonOK_actionPerformed(e);
    }
  }

//-----------------------------------------------------------------------------

} //end XML_XSLTLoader
```

Package: admintool.jdomtreemodel:

admintool.jdomtreemodel.DataSrcMapper.java

```
package admintool.jdomtreemodel;

import java.sql.*;

import javax.swing.*;

import org.jdom.*;

/**
 * Retrives metadata from a datasources and maps them to a Data Source Document.
 */
public class DataSrcMapper {
  // JDBC driver name and database URL
  /**JDBC driver.*/
  private String JDBC_DRIVER = null;
  /**Data source name.*/
  private String DATA_SRC_NAME = null;
  /**Data source type. Type can be: ODBC, Oracle, SQL server etc.*/
  private String DATA_SRC_TYPE = null;
   /**Connection to the database.*/
  private Connection connection;
  /** Statement for accessing and querying database.*/
  private Statement statement;
  /**Meta data results retrieved from database.*/
  private ResultSetMetaData metaData = null;
  /**Data Source Document where the retrieved data source is mapped.*/
  Document doc = null;
  /**Name attribute variable.*/
  Attribute attName = null;
  /**Type attribute variable.*/
  Attribute attType = null;
  /**
   * Creates the Data source document based
   * on the meta data it retrieves from the corresponding data source.
   *
   * @param driver the JDBC driver
   * @param dataSrcName data source name
   * @param dataSrcType data source type.
   */
  public DataSrcMapper(String driver, String dataSrcName, String dataSrcType) {
    //Initialoze data variables
    JDBC_DRIVER = driver;
    DATA_SRC_NAME = dataSrcName.toLowerCase();
    DATA_SRC_TYPE = dataSrcType.toLowerCase();
        String jdbcString = new String("jdbc:" +
                                  DATA_SRC_TYPE + ":" +
                                  DATA_SRC_NAME);
   //create the document
    doc = new Document(new Element("dataStructure"));
    //document's root.
    Element elDataStruct = doc.getRootElement();
   //Start mapping
    try {
      //initiallize the JDOM  tree
      //Constructin the JDOM structure

      //construct data source
```

```
      Element elDataSrc = new Element("dataStructureSource");
      attName = new Attribute("name", DATA_SRC_NAME);
      elDataSrc.setAttribute(attName);
      attType = new Attribute("type", new String(DATA_SRC_TYPE).toUpperCase());
      elDataSrc.setAttribute(attType);
      elDataStruct.addContent(elDataSrc);

      //Add a comment
      elDataSrc.addContent(new Comment("Comment: Sample Data Source Structure"));

      // load database driver class
      /*You do not need to create an instance of a driver and register it
       with the DriverManager  because calling Class.forName  will do that for you
automatically.*/
      Class.forName(JDBC_DRIVER);
      // establish connection to database
      connection = DriverManager.getConnection("jdbc:" + DATA_SRC_TYPE + ":" +
                                               DATA_SRC_NAME); //jdbc:odbc:customers
      //get metadata
      DatabaseMetaData databaseMetaData = connection.getMetaData();
      ResultSet catalogResultSet = databaseMetaData.getCatalogs();
      ResultSetMetaData catalogMetaData = catalogResultSet.getMetaData();

      int catalogNumberOfColumns = catalogMetaData.getColumnCount();

      //Retrieving table names
      String[] tableTypesString = new String[1];
      tableTypesString[0] = "TABLE";
      ResultSet tables = databaseMetaData.getTables(null, null, null,
          tableTypesString);
      //for all the tables of the data source
      while (tables.next()) {
        //     Retrieving the database table name
        String tableName = tables.getString("TABLE_NAME");
        // create Statement for querying database
        statement = connection.createStatement();
        // query database
        ResultSet resultSet = statement.executeQuery("SELECT * FROM " +
            tableName);
        // process query results
        metaData = resultSet.getMetaData();
        int numberOfColumns = metaData.getColumnCount();
        //construct Tables elements of data source
        Element elDataSrcTable = new Element("dataStructureTable");
        attName = new Attribute("name", metaData.getTableName(1));
        elDataSrcTable.setAttribute(attName);
        Attribute attAbstract = new Attribute("abstract", "false");
        elDataSrcTable.setAttribute(attAbstract);
        elDataSrc.addContent(elDataSrcTable);
        //construct Field elements of data source
        int numberOfFields = metaData.getColumnCount();
        for (int i = 1; i <= numberOfFields; i++) {
          Element elDataSrcField = new Element("dataStructureField");
          attName = new Attribute("name", metaData.getColumnName(i));
          elDataSrcField.setAttribute(attName);
          attType = new Attribute("type", metaData.getColumnTypeName(i));
          elDataSrcField.setAttribute(attType);
          Attribute attClass = new Attribute("class",
                                             metaData.getColumnClassName(i));
          elDataSrcField.setAttribute(attClass);
          Attribute attHasPredefinedValues = new Attribute(
              "haspredefinedvalues", "false");
          elDataSrcField.setAttribute(attHasPredefinedValues);
          Attribute attHasValueRange = new Attribute("hasvaluerange", "false");
          elDataSrcField.setAttribute(attHasValueRange);
          elDataSrcTable.addContent(elDataSrcField);
        }
      } //end while
    } // end try
     // detect problems interacting with the database
     catch (SQLException sqlException) {
        JOptionPane.showMessageDialog(null,"Data Source Mapper:\n" +
                                      "Could not load DriverManager for " + jdbcString +"!"+
"\nPropable cause: Data Source was not found on local host." + "\nClassNotFoundException
thrown..."  ,
                                      "Failure in connecting to Data Source",
                                      JOptionPane.ERROR_MESSAGE);
```

```
    }
    // detect problems loading database driver
    catch (ClassNotFoundException classNotFound) {
      JOptionPane.showMessageDialog(null,"Data Source Mapper:\n" +
                                    "Could not establish connection for " + jdbcString + "!
\nSQLException thrown...",
                                    "Failure in connecting to Data Source",
                                    JOptionPane.ERROR_MESSAGE);
    }
    // ensure statement and connection are closed properly
    finally {
      try {
        if (statement!=null){
          statement.close();
        }
        if (connection!=null) {
        connection.close();
        }

      }
      // handle exceptions closing statement and connection
      catch (SQLException sqlException) {
        JOptionPane.showMessageDialog(null,"Data Source Mapper:\n" +
                                    "Could not close connection for " + jdbcString + "!
\nSQLException thrown...",
                                    "Failure in closing connection",
                                    JOptionPane.ERROR_MESSAGE);
      }
    } //end of finally
  } // end constructor

  /**
   * Gets the Data source document that the Data Source is retrieved and mapped on.
   *
   * @return Document
   */
  public Document getMappedDataSrc() {
    return doc;
  }
}
```

admintool.jdomtreemodel.DataSrcTreeCellRenderer.java

```
package admintool.jdomtreemodel;

import org.jdom.Element;
import javax.swing.tree.DefaultTreeCellRenderer;
import javax.swing.ImageIcon;
import javax.swing.JTree;
import java.awt.Component;
import org.jdom.Content;
import java.awt.Font;
import java.awt.Color;
import org.jdom.Attribute;
import org.jdom.Text;
import org.jdom.CDATA;

/**
 *
 * <p>Tree Cell Renderer of the Data Sstructure JTree.</p>
 * @see javax.swing.tree.DefaultTreeCellRenderer
 */
public class DataSrcTreeCellRenderer
    extends DefaultTreeCellRenderer {
  /**Font used.*/
  Font dbFont, dbTableFont, dbFieldFont;
  /**Color used.*/
  Color dbColor, databaseTableColor, databaseFieldColor;
  /**Image used.*/
  ImageIcon databasesNotLoadedIcon, databasesLoadedIcon,
      databaseLoadedIcon, dataFileIcon, databaseNotLoadedIcon,
      databaseTableNotExpandedIcon, databaseTableExpandedIcon,
```

```
      databaseFieldIcon, conditionedField, attIcon, textIcon, cdataIcon,
      elementIcon, xmlIcon, noIcon;

  /**
   * Constructor. Initializes fonts, colors and loads images. When loading an image,
   * if image is not found then the application won't start. The way that the images are
   * loaded ensures that are all loaded if the appplication starts.
   */
  public DataSrcTreeCellRenderer() {
    //initialize fonts
    dbFont = new Font("Tahoma", Font.BOLD, 14);
    dbTableFont = new Font("Tahoma", Font.BOLD, 12);
    dbFieldFont = new Font("Tahoma", Font.BOLD, 12);
    //initialize colors
    dbColor = Color.blue;
    databaseTableColor = Color.DARK_GRAY;
    databaseFieldColor = Color.black;
    //Loading tree icons
    databasesNotLoadedIcon = new ImageIcon(DataSrcTreeCellRenderer.class.
                                           getResource("databasesNotLoaded.gif"));
    databasesLoadedIcon = new ImageIcon(DataSrcTreeCellRenderer.class.
                                        getResource("databasesLoaded.gif"));
    databaseLoadedIcon = new ImageIcon(DataSrcTreeCellRenderer.class.
                                       getResource("databaseLoaded.gif"));
    databaseNotLoadedIcon = new ImageIcon(DataSrcTreeCellRenderer.class.
                                          getResource("databaseNotLoaded.gif"));
    databaseTableNotExpandedIcon = new ImageIcon(
        DataSrcTreeCellRenderer.class.getResource(
        "databaseTableNotExpanded.gif"));
    databaseTableExpandedIcon = new ImageIcon(
        DataSrcTreeCellRenderer.class.getResource("databaseTableExpanded.gif"));
    databaseFieldIcon = new ImageIcon(DataSrcTreeCellRenderer.class.getResource(
        "databaseField.gif"));
    conditionedField = new ImageIcon(DataSrcTreeCellRenderer.class.getResource(
        "done.gif"));
    dataFileIcon = new ImageIcon(DataSrcTreeCellRenderer.class.getResource(
        "dst.gif"));
    attIcon = new ImageIcon(DataSrcTreeCellRenderer.class.getResource(
        "attr.gif"));
    textIcon = new ImageIcon(DataSrcTreeCellRenderer.class.getResource(
        "text.gif"));
    cdataIcon = new ImageIcon(DataSrcTreeCellRenderer.class.getResource(
        "cdata.gif"));
    elementIcon = new ImageIcon(DataSrcTreeCellRenderer.class.getResource(
        "element.gif"));
    xmlIcon = new ImageIcon(DataSrcTreeCellRenderer.class.getResource(
        "xml.gif"));
      noIcon = new ImageIcon(DataSrcTreeCellRenderer.class.getResource(
        "noIcon.gif"));
  } //end constructor

  /**
   *
   * <p>
   * The method that does the rendering.
   * </p>
   * Cases of rendered object:
   * <ul>
   * <li>Case of: Element. <br> Element can be:</li>
   * <ul>
   * <li>case of: Data structure</li>
   * <ul>
   * <li>is empty</li>
   * <li>or not empty</li>
   * </ul>
   * <li>case of: loaded data structures. Load structures can be:</li>
   * <ul>
   * <li>case of: data source structure, that is a data source which is retrived and loaded in
the tree</li>
   * <li>case of: data structure file, that is a data structure loaded from file</li>
   * </ul>
   * <li>case of: Data structure category</li>
   * <li>case of: Data structure field</li>
   * <li>case of: any other element, that is the case when a generic XML is loaded. It has the
following subcases: </li>
   * <ul>
   * <li>case of: the root of a generic xml</li>
```

```
   * <li>case of: the rest elements of a generic xml</li>
   * </ul>
   * </ul>
   * <li>case of: Attribute</li>
   * <li>case of: CDATA</li>
   * <li>Case of: Text.</li>
   * <li>Case of: any other node</li>
   * </ul>
   * In each case the method sets the appropriate color, font and text that the
   * rendered objecy would be displayed. CDATA test must preceed Text test! Otherwise it
mistakes CDATA nodes to be Text nodes
   *
   * @param tree JTree
   * @param value the Object that is going to be renderered
   * @param sel boolean
   * @param expanded boolean
   * @param leaf boolean
   * @param row int
   * @param hasFocus boolean
   * @return Component
   */
  public Component getTreeCellRendererComponent(JTree tree, Object value,
                                                boolean sel, boolean expanded,
                                                boolean leaf, int row,
                                                boolean hasFocus) {
    //call superclass method to initialize the parameters
    super.getTreeCellRendererComponent(tree, value, sel, expanded, leaf, row,
                                       hasFocus);

    //create a string buffer to store the description that will be displayed
    StringBuffer sb = new StringBuffer("");
    //cases of rendered object
    //case of: Element
    if (value instanceof Element) {
      Element el = (Element) value;
      String elName = el.getName();
      //cases of element
      //case of: Data structure
      if (elName.equals("dataStructure")) {
        //if data strucure is empty
        if (el.getChildren().size() == 0) { //not loaded
          setFont(dbFont);
          setForeground(Color.GRAY);
          setIcon(databasesNotLoadedIcon);
          setText("No Data Structures loaded");
        }
        //if it is not empty
        else {
          setFont(dbFont);
          setForeground(Color.DARK_GRAY);
          setIcon(databasesLoadedIcon);
          sb.append("Available Data Structures");
          //if type is not None then display it
          String s = null;
          try {
            s = el.getAttribute("type").getValue();
            if (!s.equals("None")) {
              sb.append(" (Type: " + s + ")");
            }
          }
          catch (Exception ex) {
          }
          setText(sb.toString());
        }
      }
      //case of: data source structure, that is a data source which is retrived and loaded in
the tree
      else if (elName.equals("dataStructureSource")) {
        setFont(dbFont);
        setForeground(dbColor);
        setText(el.getAttribute("type").getValue() + " source: '" +
                el.getAttribute("name").getValue() + "'");
        setIcon(databaseLoadedIcon);
      }
      //case of: data structure file, that is a data structure loaded from file
      else if (elName.equals("dataStructureFile")) {
        setFont(dbFont);
```

```
    setForeground(new Color(4, 4, 175)); //setForeground(dbColor);
    String type = null;
    String typeAtt = el.getAttribute("type").getValue();
    if (typeAtt.equals("dst")) {
      type = new String("Data Structured");
    }
    setText(type + " file: '" +
            el.getAttribute("name").getValue() + "'");
    setIcon(dataFileIcon);
  }
  //case of: data structure table
  else if (elName.equals("dataStructureTable")) {
    setFont(dbTableFont);
    setForeground(databaseTableColor);
    setText("Table: " + el.getAttribute("name").getValue());
    setClosedIcon(databaseTableNotExpandedIcon);
    setOpenIcon(databaseTableExpandedIcon);
  }
  //case of: data structure field
  else if (elName.equals("dataStructureField")) {
    setFont(dbFieldFont);
    sb.append("Field");
    sb.append(": \"" + el.getAttribute("name").getValue() + "\"  (Type: " +
              el.getAttribute("type").getValue() + ")");
    setText(sb.toString());
    setIcon(databaseFieldIcon);
  }
  //case of any other element, that is the case when a generic XML is loaded
  else { //any other element

    //case of: the root of a generic xml
    /*This must be after "dataStructure" test.
    otherwise it will throw an Exception*/
    String parentName = el.getParentElement().getName();
    if (parentName.equals("dataStructure")) {
      setFont(dbFont);
      setForeground(new Color(4, 4, 175)); //setForeground(dbColor);
      //show type
      String type = null;
      String typeAtt = el.getAttribute("type").getValue();
      if (typeAtt.equals("xml")) {
        type = new String("XML:");
      }
      else {
        type = new String("Undefined");
      }
      setText(type + "'" +
              el.getAttribute("name").getValue() + "'" + ". Root name: " +
              el.getName());
      setIcon(xmlIcon);
    }
    //case of: the rest elements of a generic xml
    else {
      setFont(dbFieldFont);
      sb.append(elName);
      setText(sb.toString());
      setIcon(elementIcon); //create an empty icon
    }
  }
}
//case of: attribute
else if (value instanceof Attribute) {
  Attribute att = (Attribute) value;
  setFont(new Font("Tahoma", Font.PLAIN, 10));
  setForeground(Color.GRAY);
  sb.append(att.getName() + "= '" + att.getValue() + "'");
  setText(sb.toString());
  setIcon(attIcon);
}
//case of: CDATA, Warning: It must preceed Text test!
else if (value instanceof CDATA) {
  setFont(new Font("Tahoma", Font.PLAIN, 11));
  setForeground(Color.DARK_GRAY);
  setText(value.toString());
  setIcon(cdataIcon);
}
//case of: Text
```

```
    else if (value instanceof Text) {
      Text text = (Text) value;
      setFont(new Font("Tahoma", Font.PLAIN, 11));
      setForeground(Color.DARK_GRAY);
      setText(text.getText());
      setIcon(textIcon);
    }
    //case of: any other content
    else if (value instanceof Content) {
      setFont(new Font("Tahoma", Font.PLAIN, 11));
      setForeground(Color.DARK_GRAY);
      setText(value.toString());
      setIcon(noIcon);
    }
    return this;
  }//end metohod
}//end class
```

admintool.jdomtreemodel.DisplayedPathEvaluator.java

```
package admintool.jdomtreemodel;

import org.jdom.Element;
import org.jdom.Attribute;

/**
* <p>
* Utility class that encloses the mechanism of evaluting the assocoation path given data source
field of
* a data source that is loaded in Data Structure tree. In fact, that is the path to the
* data source field.</p>
* <i>Tip: The <code>displayedPath</code> is also refered and as <code>associationPath</code> in
this application code.</i>
* @see admintool.TreeTransferHandler
 */
public class DisplayedPathEvaluator {
  /**
   * Default constructor.
   */
  public DisplayedPathEvaluator() {
  }

  /**
   * <p>
   * Returns the displayed path (or association path).<br>
   * It parses the objects that represent a path from selection to root, and for each objects
   * test the following cases:
   * <ul>
   * <li>Case of: Element:</li>
   * <ul>
   * <li>if it is not a field append only its name</li>
   * <li>if it is a field and this is the last object in the path, append type too.</li>
   * </ul>
   * <li>Case of: Attribute</li>
   * <ul>
   * <li>append attribute name</li>
   * </ul>
   * </ul>
   * </p>
   *
   * @param objects the objects that represent a path from selection to root.
   * @return the displayed path.
   */
  public String getDisplayedPath(Object[] objects) {
    //create a string buffer to store the displayedpath
    StringBuffer sb = new StringBuffer("");
    //parse objects  that represent a path from selection to root one by one
    for (int i = 0; i < objects.length; i++) {
      Element el = null;
      //cases of selected object
      //case of: element
      if (objects[i] instanceof Element) {
```

```
        el = (Element) objects[i];
        String elName = el.getName();
        //if it is not a field append only its name
        if ( (elName.equals("dataStructureSource")) ||
            (elName.equals("dataStructureFile")) ||
            (elName.equals("dataStructureTable"))
            ) {
          sb.append(el.getAttribute("name").getValue() + ":");

        }
        //if it is a field and this is the last object in the path, append type too.
        else if (elName.equals("dataStructureField")) {
          sb.append(el.getAttribute("name").getValue() );
          //if this is the last object in path (that means no Attribute is selected)
          if (i==(objects.length-1) ) {
            sb.append(  " (Type: " + el.getAttribute("type").getValue() + ")");
          }
        }

      }
      //case of: attribute
      else if (objects[i] instanceof Attribute) {
        //append attribute name
        Attribute att = (Attribute) objects[i] ;
        sb.append(":@"+ att.getName());
      }
      //error case
      else {
        return "";
      }
    } //end for
    return sb.toString();
  }//end method
}
```

admintool.jdomtreemodel.InfoStructTreeCellRenderer.java

```
package admintool.jdomtreemodel;

import org.jdom.Element;
import javax.swing.tree.DefaultTreeCellRenderer;
import javax.swing.ImageIcon;
import javax.swing.JTree;
import java.awt.Component;
import org.jdom.Content;
import org.jdom.Text;
import java.awt.Font;
import java.awt.Color;
import org.jdom.Attribute;
import java.util.List;
import java.util.Iterator;

/**
 * <p>Tree Cell Renderer  of the Information Structure JTree.</p>
 *
 * @see javax.swing.tree.DefaultTreeCellRenderer
 */
public class InfoStructTreeCellRenderer
    extends DefaultTreeCellRenderer {
  /**Font used.*/
  Font informationStructureFont, informationCategoryFont, informationFieldFont,
      associationPathFont;
  /**Color used.*/
  Color informationStructureColor, informationCategoryColor,
      informationFieldNotAssociatedColor,
      informationFieldAssociationIncompleteColor,
      informationFieldAssociatedColor;
  /**Image used.*/
  ImageIcon informationStructureNotLoadedIcon,
      informationStructureLoadedIcon,
      informationCategoryNotExpandedIcon,
      informationCategoryExpandedIcon,
```

```
      informationFieldNotAssociatedIcon,
      informationFieldAssociationIncompleteIcon,
      informationFieldAssociatedIcon,
      associationPathIcon;
  /**
   * Constructor. Initializes fonts, colors and loads images. When loading an image,
   * if image is not found then the application won't start. The way that the images are
   * loaded ensures that are all loaded if the appplication starts.
   */
  public InfoStructTreeCellRenderer() {
    //initialize fonts
    informationStructureFont = new Font("Tahoma", Font.BOLD, 14);
    informationCategoryFont = new Font("Tahoma", Font.BOLD, 12);
    informationFieldFont = new Font("Tahoma", Font.BOLD, 12);
    associationPathFont = new Font("Tahoma", Font.BOLD, 11);
    //initialize colors
    informationStructureColor = Color.blue;
    informationCategoryColor = Color.DARK_GRAY;
    informationFieldNotAssociatedColor = Color.red;
    informationFieldAssociationIncompleteColor = new Color(200, 100, 0); //       dark orange
    informationFieldAssociatedColor = new Color(0, 128, 0); // dark green
    //Loading tree icons
    informationStructureNotLoadedIcon = new ImageIcon(
        InfoStructTreeCellRenderer.class.getResource(
        "informationStructureNotLoaded.gif"));
    informationStructureLoadedIcon = new ImageIcon(
        InfoStructTreeCellRenderer.class.getResource(
        "informationStructureLoaded.gif"));
    informationCategoryNotExpandedIcon = new ImageIcon(
        InfoStructTreeCellRenderer.class.getResource(
        "informationCategoryNotExpanded.gif"));
    informationCategoryExpandedIcon = new ImageIcon(
        InfoStructTreeCellRenderer.class.getResource(
        "informationCategoryExpanded.gif"));
    informationFieldNotAssociatedIcon = new ImageIcon(
        InfoStructTreeCellRenderer.class.getResource(
        "informationFieldNotAssociated.gif"));
    informationFieldAssociationIncompleteIcon = new ImageIcon(
        InfoStructTreeCellRenderer.class.getResource(
        "informationFieldAssociationIncomplete.gif"));
    informationFieldAssociatedIcon = new ImageIcon(
        InfoStructTreeCellRenderer.class.getResource(
        "informationFieldAssociated.gif"));
    associationPathIcon = new ImageIcon(InfoStructTreeCellRenderer.class.
                                        getResource("databaseField.gif"));
  } //end constructor

  /**
   * <p>
   * The method that does the rendering.
   * </p>
   * Cases of rendered object:
   * <ul>
   * <li>Case of: Element. <br> Element can be:</li>
   * <ul>
   * <li>case of: Information structure</li>
   * <ul>
   * <li>is empty</li>
   * <li>or not empty</li>
   * </ul>
   * <li>case of: Information category</li>
   * <li>case of: information field</li>
   * <ul>
   * <li>Not associated</li>
   * <li>Associated</li>
   * </ul>
   * </ul>
   * <li>Case of: Text. The text node carries the XPath string. But, the text node would be
dislpayed
      with the association path attribute of its parent element.</li>
   * <li>Case of: any other node</li>
   * </ul>
   * In each case the method sets the appropriate color, font and text that the
   * rendered objecy would be displayed.
   *
   * @param tree JTree
   * @param value the Object that is going to be renderered
```

```
 * @param sel boolean
 * @param expanded boolean
 * @param leaf boolean
 * @param row int
 * @param hasFocus boolean
 * @return Component
 */
public Component getTreeCellRendererComponent(JTree tree, Object value,
                                               boolean sel, boolean expanded,
                                               boolean leaf, int row,
                                               boolean hasFocus) {
  //call superclass method to initialize the parameters
  super.getTreeCellRendererComponent(tree, value, sel, expanded, leaf, row,
                                     hasFocus);

  //create a string buffer to store the description that will be displayed
  StringBuffer sb = new StringBuffer("");
  //cases of rendered object
  //case of: Element
  if (value instanceof Element) {
    Element el = (Element) value;
    String elName = el.getName();
    //cases of element
    //case of: Information structure
    if (elName.equals("informationstructure")) {
      //if information strucure is empty
      if (el.getChildren().size() == 0) { //not loaded
        setFont(informationStructureFont);
        setForeground(Color.GRAY);
        setIcon(informationStructureNotLoadedIcon);
        setText("No Information Structure loaded");
      }
      //if it is not empty
      else {
        setFont(informationStructureFont);
        setForeground(informationStructureColor);
        setIcon(informationStructureLoadedIcon);
        sb.append("Information Structure");
        //if type si not None the display it
        String s = null;
        try {
          s = el.getAttribute("type").getValue();
          if (!s.equals("None")) {
            sb.append(" (Type: " + s + ")");
          }
        }
        catch (Exception ex) {
        }
        setText(sb.toString());
      }
    }
    //case of: Information category
    else if (elName.equals("informationcategory")) {
      setFont(informationCategoryFont);
      setForeground(informationCategoryColor);
      setText(el.getAttribute("name").getValue());
      setClosedIcon(informationCategoryNotExpandedIcon);
      setOpenIcon(informationCategoryExpandedIcon);
    }
    //case of: information field
    else if (elName.equals("informationfield")) {
      setFont(informationFieldFont);
      String assState = el.getAttribute("associated").getValue();
      //case of field is not associated
      if (assState.equals("No")) {
        setForeground(informationFieldNotAssociatedColor);
        setIcon(informationFieldNotAssociatedIcon);
      }
      //case of field is associated
      else if (assState.equals("Yes")) {
        setForeground(informationFieldAssociatedColor);
        setIcon(informationFieldAssociatedIcon);
      }
      //error case
      else {
        System.err.println(
            "ERROR in TreeCellRendrer - cannot define color or image");
```

```
          setForeground(Color.GRAY);
        }
        sb.append("Information Field");
        sb.append(": \"" + el.getAttribute("name").getValue() + "\"  (Type: " +
                  el.getAttribute("type").getValue() + ")");
        setText(sb.toString());
      }
      //error case
      else {
        System.out.println(
            "ERROR in infoStructTreeCellRenderer - Unexpected Element: <" +
            el.getName() + ">");
      }
    }
    //case of: Text
    else if (value instanceof org.jdom.Text) {
      String s = null;
      Text text = (Text) value;
      //The text node carries the XPath string. But, the text node would be dislpayed
      //with the association path attribute of its parent element.
      Element parent = text.getParentElement();
      Attribute assocPathAttr = parent.getAttribute("associationPath");
      if (assocPathAttr == null) {
        s = "ERROR null attribute found";
      }
      else {
        s = assocPathAttr.getValue();
      }
      setFont(associationPathFont);
      setIcon(associationPathIcon);
      setForeground(Color.DARK_GRAY);
      setText(s);
    }
    //case of: any other node
    else if (value instanceof Content) {
      setText("Content" + value.toString());
    }
    return this;
  } //end method
} //end class
```

admintool.jdomtreemodel.JDOMTreeModel.java

```
package admintool.jdomtreemodel;

import java.util.*;

import javax.swing.*;
import javax.swing.event.*;
import javax.swing.tree.*;

// JDOM imports
import org.jdom.*;
import org.jdom.filter.*;
import admintool.*;

//import org.xml.sax.XMLFilter;

/**
 *
 * <p>
 * Tree Model implementation to display a JDOM Document
 *  in a JTree,  with insert and remove support but without displaying attributes.
 * </p>
 * <p>
 * The structure shown by the
 * JTrees using this Tree Model is the structure of the XML document of the Tree Model.
 * The JTree is displayed by calling the methods of the Interface TreeModel that this
 * class implements. And these methods are implemented in order to outline the structure of the
XML
 * document of the this class.
 * </p>
```

```
 * <p>
 * This class implements the TreeModel Interface, instead of extending the DefaultTreeModel
 * class. As a result the data structure of the XML document is directly mapped to the
 * structure shown by JTree. If a DefaultTreeModel class was used, the current structure shown
by
 * JTree would be the Structure of the DefaultTreeModel and not of the XML document.
 *  </p>
 * <p>
 * This class does not provide support for displaying Attributes as leaves of the
 * tree. Attributes are completely ignored. The subclass JDOMTrreModelAttr extends
 * this functionality.
 * </p>
 * <p>
 * Used by Information Structure JTree.
 *  </p>
 *  <p>
 *  <i>  Note: <code>Content</code> is refered to org.jdom.Content.</i>
 * </p>
 *
 *
 */
public class JDOMTreeModel
    implements TreeModel {

  /**The JDOM Document that this Tree Model is based.*/
  Document doc = null;
  /**The root of the Tree Model, which is the root of the document too.*/
  Element root = null;
  /**Event Listeners List. Listeners that listens to the Tree Model's changes.*/
  Vector listeners = null;
  /**Through this reference the GUIUpdater of AdminToolFrame is notified, when a modification
   * on TreeModel is made. The information structure tree holds the <i>data</i> of the
   * application. So, when a modification is done upon it, the GUIUpdater must be notified
   * to change the GUI (for example to enable the Save Button).*/
  DataModifier dataModifier = null;

  /**
   * Constructor.
   * <p>
   * The AdminToolFrame passed to the constructor is casted to a DataModifier reference.
   * See Interface DataModifier for more.
   * </p>
   *
   * @param document - the document of the TreeModel.
   * @param dataModifier - the reference through which GUIUpdater will be notified.
   * @see DataModifier
   */
  public JDOMTreeModel(Document document,
                       DataModifier dataModifier) {
    listeners = new Vector();
    doc = document;
    root = doc.getRootElement();
    this.dataModifier = dataModifier;
  }

  /**
   * Returns the root of the tree.
   *
   *
   * @return Object - the root of the tree
   */
  public Object getRoot() {
    return root;
  }

  /**
   * Returns the number of children of parent. Returns 0 if the node is a leaf
   * or if it has no children. Parent must be a node previously obtained from this data source.
   *
   * @param parent - a node in the tree, obtained from this data source
   * @return the number of children of the node parent
   * @see org.jdom.Element.getContentSize()
   * @see org.jdom.Document.getContentSize()
   */
  public int getChildCount(Object parent) {
    //Only Element and Document have Children!
    //Parent case:
```

```
  if (parent instanceof Document) {
    Document docParent = (Document) parent; //cast it
    //delegate the calls to proper JDOM function
    return docParent.getContentSize();
  }
  //Element case:
  else if (parent instanceof Element) {
    Element elParent = (Element) parent; //cast it
    //delegate the calls to proper JDOM function
    return elParent.getContentSize();
  }
  //No Document or Element --> it is a Leaf -->so it has no children
  else {
    return 0;
  }
}

/**
 * Returns true if node is a leaf.<br>
 * Only Element and Document have Children.
 * If a Document or an Element has no Content, then it is considered to
 * be a leaf and thus the method returns true.
 *
 * @param node - a node in the tree, obtained from this data source
 * @return true if node is a leaf
 */
public boolean isLeaf(Object node) {
  //Only Element and Document have Children
  //Check document
  if (node instanceof Document) {
    Document docNode = (Document) node;
    //if it has no content then it is a leaf
    if (docNode.getContentSize() == 0) {
      return true;
    }
    else {
      return false;
    }
  }
  //Check Element
  else if (node instanceof Element) {
    Element elNode = (Element) node; //cast it
    //if it has no content then it is a leaf
    if (elNode.getContentSize() == 0) {
      return true;
    }
    else {
      return false;
    }
  }
  //It sure is a leaf
  else {
    return true;
  }
} //end is Leaf

/**
 * Adds a listener for the TreeModelEvent  posted after the tree changes.
 *
 * @param l - the listener to add
 * @see removeTreeModelListener(TreeModelListener)
 */
public void addTreeModelListener(TreeModelListener l) {
  listeners.add(l);
}

/**
 * Removes a listener previously added with addTreeModelListener.
 *
 * @param l - the listener to remove
 * @see addTreeModelListener(TreeModelListener)
 */
public void removeTreeModelListener(TreeModelListener l) {
  listeners.remove(l);
}

/**
```

```
   * Returns the child of parent at "index" index  in the parent's child array.
   * parent must be a node previously obtained from this data source.
   * This should not return null  if index  is a valid index for parent
   * (that is index >= 0 &&  index < getChildCount(parent)).
   *
   * @param parent  a node in the tree, obtained from this data source
   * @param index the index
   * @return the child of parent at index index
   */

  public Object getChild(Object parent, int index) {
    //Only Element and Document have Children
    //check document
    if (parent instanceof Document) {
      Document docParent = (Document) parent; //cast it
      return docParent.getContent(index);
    }
    //check Element
    else if (parent instanceof Element) {
      Element elParent = (Element) parent; //cast it
      //inside bounds
      if ( (index >= 0) && (index < elParent.getContentSize())) {
        return elParent.getContent(index);
      }
      //out of bounds
      else if (index >= (elParent.getContentSize()) ||
               (index < 0)) {
        System.err.println("*** ERROR in getChild! -- > index out of bounds");
        return null;
      }
      //error case
      else {
        System.out.println("*** ERROR in getChild! unknown error "); //Method called in a node
with no children
        return null;
      }
    }
    //THIS SHOULD NEVER BE EXECUTED
    else {
      System.out.println(
          "*** ERROR in getChild! No Document or Element parent faound! "); //Method called in a
node with no children
      return null;
    }
  } //end getChild

  /**
   * Returns the index of child in parent. If parent  is null or child is null, returns -1.<br>
   *
   * Parent can be: a Document or an Element
   * Child can be: an Element or Content.
   *
   * @param parent a node in the tree, obtained from this data source
   * @param child the node we are interested in
   * @return the index of the child in the parent, or -1 if either child or parent are null
   */
  public int getIndexOfChild(Object parent, Object child) {

    //First satisfy: "If parent  is null or child is null, returns -1."
    if (parent == null) {
      return -1;
    }
    if (child == null) {
      return -1;
    }

    //Taking care of parent
    //check parent = Document
    if (parent instanceof Document) {
      Document docParent = (Document) parent; //cast it
      return docParent.indexOf( (Content) child); //cast child to Content type
    }
    //check parent is Element
    else if (parent instanceof Element) {
      Element elParent = (Element) parent; //cast it
      //Taking care of child, child can be any object of org.jdom.Content type
      if (child instanceof Content) {
```

```
        Content conChild = (Content) child;
        return (elParent.indexOf(conChild)); //cast child to Content type
      }
      //error case
      else {
        System.err.println("*** ERROR in getIndexOfChild: Incompatible types--> Child is not of
Content type. Child: " +
                           child.toString());
        //return an big number to crash tree
        return -100;
      }
    }
    //THIS SHOULD NEVER BE EXECUTED
    else {
      System.err.println("*** ERROR in getIndexOfChild!"); //Unspecified Error
      return -100;
    }
  } //end getIndexOfChild

  /**
   * Not implemented. Because it is not been used.
   *
   * @param path TreePath
   * @param newValue Object
   */
  public void valueForPathChanged(TreePath path, Object newValue) {
  }

  /**
   * Invoked this to insert newChild at location index in parents children.
   * <br> This will then message nodesWereInserted to create the appropriate
   * event. This is the preferred way to add children as it will create the
   * appropriate TreeModelEvent.
   *
   * <p> It supports three cases of insertion. Case where insertion occurs:
   * <ul>
   * <li>on an Information Field, when a Assocition was created by a DnD. Cases
   * of Information Filed:</li>
   * <ul>
   * <li>not associated yet, so it can be associated </li>
   * <li>already associaited, show error message</li>
   * </ul>
   * <li>on the root of Data Structrue tree, when a data structure is loaded
   * </li>
   * <li>on the root of Inforation Structrue tree, when a information structure
   * is loaded</li>
   * </ul> </p>
   *
   * <p> It was written based on DefaultTreeModel's insertNodeInto method. So,
   * after an insertion occurs, nodesWereInserted method is called to fire the
   * TreeModelEvents in order for JTree to be notified that its Tree Model is
   * changed, and update its appearance. </p>
   *
   * <p> After insertion, it notifies that data is modified in all cases (also
   * when an insertion occures in the Data Structure tree, which is not needed).
   * Nevertheless, this way was prefered because it would be easier to extend
   * the program (in future release when data structure might be considered to
   * be data). </p>
   *
   * @param dataElement the new child
   * @param parent the parent
   * @param index the index
   * @param path the path in the tree where the new node will be inserted, used for firrong the
TreeModelEvent
   * @see nodesWereInserted(Text , Element, int , TreePath)
   * @see nodesWereInserted(Element , Element , int , TreePath )
   */
  public void insertNodeInto(Element dataElement, Element parent,
                             int index,
                             TreePath path) {

    String parentName = parent.getName();
    //case where insertion occurs on an Information Field,  when a Assocition
    //was created by a DnD.
    if (parentName.equals("informationfield")) {
      //variable initialization
      Element isFieldElement = parent;
```

```
      String xPath = null;
      String displayedPath = null;
      String source = null;
      Text textNode = null;
      Filter textfilter = new ContentFilter(ContentFilter.TEXT);

      //Insert the data
      //get the data. Data is: XPath, displayPath, source atrribute
      xPath = dataElement.getText();
      displayedPath = dataElement.getAttribute("associationPath").getValue();
      //try catch because attribute might be missing
      try {
        source = dataElement.getAttribute("source").getValue();
      }
      catch (Exception e) {
        source = "";

      }
      String associated = isFieldElement.getAttribute("associated").getValue();
      //got data
      //cases of Information Field:
      //not associated yet, so it can be associated
      if (associated.equals("No")) {
        //do the actual insertion and everything else required for an information
        //field to be considered associated.
        //insert XPath
        textNode = new Text(xPath);
        isFieldElement.addContent(textNode);
        //set association path attribute
        isFieldElement.setAttribute("associationPath", displayedPath);
        //set source attribute
        isFieldElement.setAttribute("source", source);
        //change status of Information Field into "associated"
        isFieldElement.setAttribute("associated", "Yes");
      }
      //already associaited, show error message
      else if (associated.equals("Yes")) {
        JOptionPane.showMessageDialog(null,
                                      "Association already defined. \nIf you want to add a new
association, first delete the existing.",
                                      "Association already defined",
                                      JOptionPane.WARNING_MESSAGE);
      }
      //SHOULD NEVER BE EXECUTED
      else {
        JOptionPane.showMessageDialog(null,
                                      "Unexpected association stated! Coding ERROR!",
                                      "Unexpected association stated",
                                      JOptionPane.ERROR_MESSAGE);

      }
      //fire treemodel events, in order for JTree to now that Tree Model is modified
      nodesWereInserted(textNode, isFieldElement, index, path);
    }
    //case where insertion occurs on the root of Data Structrue tree,
    //when a data structure is loaded
    else if (parentName.equals("dataStructure")) {
      //insert
      Element dataStructRoot = parent;
      dataStructRoot.addContent(dataElement);
      //fire treemodel events, in order for JTree to now that Tree Model is modified
      nodesWereInserted(dataElement, dataStructRoot, index, path);

    }
    //case where insertion occurs on the root of Inforation Structrue tree,
    //when a information structure is loaded
    else if (parentName.equals("informationstructure")) {
      //insert
      Element infoStructRoot = parent;
      infoStructRoot.addContent(dataElement);
      //fire treemodel events, in order for JTree to now that Tree Model is modified
      nodesWereInserted(dataElement, infoStructRoot, index, path);
    }

    //On all cases
    //Notify that data is modified
    notifyDataModification();
```

```
}

/**
 *
 * Notifies AdminToolFrame that its data (that is the Information Structure)
 * is modified (and thus needs to be saved).
 * <p>
 * It checks to see which tree it is and only calls dataModified method in AdminToolFrame when
 * it founds that it is a information structure tree.<br>
 * Data Structure  is not considered to be data, so we do not need to do any
 * notificationwhen an isertion or deletion is made. <br>
 * The test, whether or not the notification should occured, was prefered to
 * be done here for making the application easier to be extend in the future.
 * </p>
 * @see DataModifier.dataModified()
 */
private void notifyDataModification() {
  //get root of tree
  Element root = (Element) getRoot();
  //check root name
  if (root.getName().equals("informationstructure")) {
    //notify
    dataModifier.dataModified();
  }
  //else its a data Structure, so do nothing
}

/**
 * Invoke this method after you've inserted some Elements into Element.
 * Gathers parameters for fireTreeNodesInserted method  and makes the call to
 * it. The fireTreeNodesInserted method is the one that actually fires the Tree Model Events.
 * <br>
 * It is called by insertNodeInto.
 * @param newChild Element child
 * @param parent parent
 * @param index the position in parent where the new node was inserted
 * @param path the path in the tree where the new node was inserted

 * @see insertNodeInto(Element , Element , int ,  TreePath )
 * @see fireTreeNodesInserted(TreePath , int[] , Object[] )
 */
public void nodesWereInserted(Element newChild, Element parent, int index,
                              TreePath path) {

  // create int array with the indeces of of new chidren
  int temp = getIndexOfChild(parent, newChild);
  int[] childIndices = {
      temp};

  //check that all parameters are OK
  if (listeners != null && parent != null && childIndices != null
      && childIndices.length > 0) {
    //create an array with new children
    Object[] newChildren = {
        newChild};
    //Fire events (TreeModelEvents)
    fireTreeNodesInserted(path, childIndices, newChildren);
  }
}

/**
 * Invoke this method after you've inserted some Text into Element. Gathers
 * parameters for fireTreeNodesInserted method and makes the call to it. The
 * fireTreeNodesInserted method is the one that actually fires the Tree Model
 * Events.
 * <br> It is called by insertNodeInto.
 *
 * @see insertNodeInto(Element , Element , int , TreePath )
 * @see fireTreeNodesInserted(TreePath , int[] , Object[] )
 * @param newChild Text child
 * @param parent parent
 * @param index the position in parent where the new node was inserted
 * @param path the path in the tree where the new node was inserted
 */
public void nodesWereInserted(Text newChild, Element parent, int index,
                              TreePath path) {
```

```
  // create int array with the indeces of of new chidren
  int temp = getIndexOfChild(parent, newChild);
  int[] childIndices = {
      temp};
  //check that all parameters are OK
  if (listeners != null && parent != null && childIndices != null
      && childIndices.length > 0) {
    //create an array with new children
    Object[] newChildren = {
        newChild};

    //Fire events (TreeModelEvents)
    fireTreeNodesInserted(path, childIndices, newChildren);
  }
}

/**
 * Notifies all listeners that have registered interest for notification on
 * this event type (Tree Model Event).
 * <br> The event instance is lazily created using the parameters passed into
 * the fire method. The TreePath parameter which is initially passed to
 * insertNodeInto method is used to create the event.
 * <br> It is necessary to fire events (TreeModelEvents) in order for the
 * JTree to be notified and update its apperance. That is upon initialization
 * JTree adds listeners to its Tree Model, to listen for events that change
 * the structure of the tree (Tree Model Events).
 *
 * @see nodesWereInserted(Text , Element, int , TreePath)
 * @see nodesWereInserted(Element , Element , int , TreePath )
 * @param path the path in the tree where the new node was inserted
 * @param indices indices of new nodes
 * @param children new nodes
 */
private void fireTreeNodesInserted(TreePath path,
                                   int[] indices, Object[] children) {
  //Take passed parameters and create TreeModelEvent to indicate node change
  TreeModelEvent event = new TreeModelEvent(this,
                                            path, indices, children);
  //Notify listeners
  Iterator iterator = listeners.iterator();
  TreeModelListener listener = null;
  // send TreeModelEvent to each listener
  while (iterator.hasNext()) {
    listener = (TreeModelListener) iterator.next();
    listener.treeNodesInserted(event);
  }
}

/**
 * Invoke this to remove a Text node from its parent Element.<br>
 * This will then message nodesWereRemoved to create the appropriate
 * event. This is the preferred way to remove children as it will create the
 * appropriate TreeModelEvent.
 * <p> It was written based on DefaultTreeModel's removeNodeFromPArent method. So,
 * after an removal occurs, nodesWereRemoved method is called to fire the
 * TreeModelEvents in order for JTree to be notified that its Tree Model is
 * changed, and update its appearance. </p>
 *
 * <p> After removal, it notifies that data is modified in all cases (also
 * when a removal occures in the Data Structure tree, which is not needed).
 * Nevertheless, this way was prefered because it would be easier to extend
 * the program (in future release when data structure might be considered to
 * be data). </p>
 *
 * @param removedEl the Text node to be removed
 * @param parentPath the path to the parent of the node which is going to be removed
 * @see nodesWereRemoved(Element , int[] , Object[] , TreePath )
 */

public void removeNodeFromParent(Text text, TreePath path) {
  //create array with the removed nodes and array with the indices of these nodes
  int[] childIndex = new int[1];
  Object[] removedArray = new Object[1];
  Element parent = text.getParentElement();
  childIndex[0] = parent.indexOf(text);
  removedArray[0] = text;
  //remove Text node of XPath
```

```
    Filter textFilter = new ContentFilter(ContentFilter.TEXT);
    parent.removeContent(textFilter);

    //set associated attribute
    parent.setAttribute("associated", "No");
    //remove associationPath attribute
    if (!parent.removeAttribute("associationPath")) {
      System.err.println("Error in removing attribute: associationPath");
    }
    if (!parent.removeAttribute("source")) {
      System.err.println("Error in removing attribute: source");
    }



    //fire treemodel events, in order for JTree to now that Tree Model is modified
    nodesWereRemoved(parent, childIndex, removedArray, path);
    //Notify that data is modified
    notifyDataModification();

  }

  /**
   * Invoke this to remove an Element node from its parent Element.<br>
   * This will then message nodesWereRemoved to create the appropriate
   * event. This is the preferred way to remove children as it will create the
   * appropriate TreeModelEvent.
   * <p> It was written based on DefaultTreeModel's removeNodeFromPArent method. So,
   * after an removal occurs, nodesWereRemoved method is called to fire the
   * TreeModelEvents in order for JTree to be notified that its Tree Model is
   * changed, and update its appearance. </p>
   *
   * <p> After removal, it notifies that data is modified in all cases (also
   * when a removal occures in the Data Structure tree, which is not needed).
   * Nevertheless, this way was prefered because it would be easier to extend
   * the program (in future release when data structure might be considered to
   * be data). </p>
   *
   * @param removedEl the Text node to be removed
   * @param parentPath the path to the parent of the node which is going to be removed
   * @see nodesWereRemoved(Element , int[] , Object[] , TreePath )
   */
  public void removeNodeFromParent(Element removedEl, TreePath parentPath) {
    //create array with the removed nodes and array with the indices of these nodes in ascending
order
    int[] childIndex = new int[1];
    Object[] removedArray = new Object[1];
    Element parent = removedEl.getParentElement();
    childIndex[0] = parent.indexOf(removedEl);
    removedArray[0] = removedEl;

    //remove element
    parent.removeContent(removedEl);

    //fire treemodel events, in order for JTree to now that Tree Model is modified
    nodesWereRemoved(parent, childIndex, removedArray, parentPath);
    //Notify that data is modified
    notifyDataModification();
  }

  /**
   * Invoke this method after you've removed some nodes from an Element nodes.
   * Gathers parameters for fireTreeNodesRemoved method and makes the call to it. The
   * fireTreeNodesRemoved method is the one that actually fires the Tree Model
   * Events.
   * <br> It is called by reomoveNodeFromParent.
   *
   * @param node the parent
   * @param childIndices array with the indices of these nodes in ascending order
   * @param removedChildren  array with the removed nodes
   * @param path the path to the parent of the node which was removed
   */
  public void nodesWereRemoved(Element node, int[] childIndices,
                               Object[] removedChildren, TreePath path) {
    //check that all parameters are OK
    if (node != null && childIndices != null) {
      //Fire events (TreeModelEvents)
```

```
      fireTreeNodesRemoved(this, path, childIndices,
                           removedChildren);
    }
  }

  /**
   * Notifies all listeners that have registered interest for notification on
   * this event type (Tree Model Event).
   * <br> The event instance is lazily created using the parameters passed into
   * the fire method. The TreePath parameter which is initially passed to
   * removeNodeFromParent method is used to create the event.
   * <br> It is necessary to fire events (TreeModelEvents) in order for the
   * JTree to be notified and update its apperance. That is upon initialization
   * JTree adds listeners to its Tree Model, to listen for events that change
   * the structure of the tree (Tree Model Events).
   *
   * @param source the node where elements are being removed
   * @param path the path to the parent of the node which was removed
   * @param childIndices the indices of the removed elements
   * @param children the removed elements
   *
   */
  private void fireTreeNodesRemoved(Object source, TreePath path,
                                    int[] childIndices,
                                    Object[] children) {
    //Take passed parameters and create TreeModelEvent to indicate node change
    TreeModelEvent event = new TreeModelEvent(this,
                                              path, childIndices, children);
    //Notify listeners
    Iterator iterator = listeners.iterator();
    TreeModelListener listener = null;
    // send TreeModelEvent to each listener
    while (iterator.hasNext()) {
      listener = (TreeModelListener) iterator.next();
      listener.treeNodesRemoved(event);
    }
  }

  /**
   * Removes all children of name <code>childrenName</code> or all the children with any name,
if childrenName=="all"
   * is entered, from the root of the Tree Model. <br>
   * The remove action is delegate to the removeNodeFromParent method.
   *
   * Used in AdminToolFrame to clear the trees.
   *
   * @param childrenName the name of the children of hte root that must be renoved. If "all" the
everything is removed
   * @see removeNodeFromParent(Element , TreePath )
   */
  public void removeAll(String childrenName) {
    //get root
    Element root = (Element) getRoot();
    //Build root path needed to call removeNodeFromParent method
    Element[] pathToRoot = new Element[1];
    pathToRoot[0] = root;
    TreePath rootPath = new TreePath(pathToRoot);

    //the error must be that I iterate the content list here and modifying it in
    //treeModel1.removeNodeFromParent(el, dataStructRootPath);

    List list = null;
    //if "all" get all childdren
    if (childrenName.equals("all")) {
      list = root.getChildren();
    }
    //else get children with name children name
    else {
      list = root.getChildren(childrenName);
    }
    //copy the list to avoid concurrent modification exceprion
    List copyList = new Vector();
    Iterator i = list.iterator();
    while (i.hasNext()) {
      Element el = (Element) i.next();
      copyList.add(el);
    }
```

```
    //used copy list to remove the elements.
    //the removal is delegated to removeNodeFromParent
    Iterator copyI = copyList.iterator();
    while (copyI.hasNext()) {
      Element el = (Element) copyI.next();
      removeNodeFromParent(el, rootPath);
    }
  }
}
```

admintool.jdomtreemodel.JDOMTreeModelAttr.java

```
package admintool.jdomtreemodel;

import org.jdom.Document;
import org.jdom.Element;
import org.jdom.Attribute;
import java.util.List;
import org.jdom.Content;
import org.jdom.Comment;
import org.jdom.CDATA;
import org.jdom.Text;
import org.jdom.EntityRef;
import org.jdom.ProcessingInstruction;
import org.jdom.DocType;
import admintool.DataModifier;

/**
 *  <p>
 * Tree Model implementation to display a JDOM Document
 *  in a JTree,  with insert and remove support and displaying attributes.
 * </p>
 * <p>
* Extends the functionality of its superclass by overiding the display tree structure
* methods in order to  put Atribute display support.
* </p>
* <p>
 * Must be used only in a NON mutable JTree. No mutation support is added for attributes yet.
 * </p>
* <p>
*  Used by Data Structure tree were Attributes need to be displayed.
*  </p>
* <p>
* <i>Before reading this class' API, first read its superclass' API. </i>
* </p>
* @see JDOMTreeModel
 */
public class JDOMTreeModelAttr
    extends JDOMTreeModel {
  /**
   * Constructor. Just calls the superclass constructor.
   * @param document  the document of the TreeModel.
   * @param dataModifier  the reference through which GUIUpdater will be notified.
   */
  public JDOMTreeModelAttr(Document document,
                           DataModifier admintoolFrameGUIUpdater) {
    super(document, admintoolFrameGUIUpdater);
  }

  /**
   * Returns the child of parent at "index" index  in the parent's child array.
   * See superclass' method description for more.
   * <p>
   * It manipulates the two
   * list (the attribute list and the Content list) as one list. This list begins
   * with the attribute list and continues with the content list. So, there are the following
   * index cases when parent is an Element:
   * <ul>
   * <li>index <  number of attributes, so index indicates an attribute. The index stays the
same.</li>
   * <li>number of attributes < index < number of attributes and Content, so index indicates a
Content. Thus Index =
```

```
   * index - number of attributes, and use new index to get the content.  </li>
   * <li>index > number of attributes and Content OR index < 0, out of bounds </li>
   * </ul>
   * </p>
   * @param parent  a node in the tree, obtained from this data source
   * @param index the index
   * @return the child of parent at index index
   */
  public Object getChild(Object parent, int index) {
    //Only Element and Document have Children
    //check document
    if (parent instanceof Document) {
      Document docParent = (Document) parent; //cast it
      return docParent.getContent(index);
    }
    //check element
    else if (parent instanceof Element) {
      Element elParent = (Element) parent; //cast it
      List attrs = elParent.getAttributes();
      //index <  number of attributes, so index indicates an attribute
      if (index < attrs.size()) {
        return (Attribute) attrs.get(index);
      }
      //number of attributes < index < number of attributes and Content, so index indicates a
Content
      else if ( (index >= attrs.size()) &&
               (index < (elParent.getContentSize() + attrs.size()))) {
        return elParent.getContent(index - attrs.size());
      }
      //index > number of attributes and Content OR index < 0, out of bounds
      else if ( (index > (elParent.getContentSize() + attrs.size())) ||
               (index < 0)) {
        System.out.println("*** ERROR in getChild! -- > index out of bounds");
        return null;
      }//THIS SHOULD NEVER BE EXECUTED
      else {
        System.err.println("*** ERROR in getChild! unknown error "); //Method called in a node
with no children
        return null;
      }
    }
    else { //THIS SHOULD NEVER BE EXECUTED
      System.err.println(
          "*** ERROR in getChild! No Document or Element parent faound! "); //Method called in a
node with no children
      return null;
    }
  }

  /**
   * Returns the number of children of parent. Returns 0 if the node is a leaf
   * or if it has no children. Parent must be a node previously obtained from this data source.
   * See superclass' method description for more. <br>
   * When the parent is an Element, it returns the sum of the number of elements and the number
of attributes
   *
   * @param parent - a node in the tree, obtained from this data source
   * @return the number of children of the node parent
   */
  public int getChildCount(Object parent) {
    //Only Element and Document have Children
    //check documwnt
    if (parent instanceof Document) {
      Document docParent = (Document) parent; //cast it
      return docParent.getContentSize();
    }
    //check element
    else if (parent instanceof Element) {
      Element elParent = (Element) parent; //cast it
      //return the sum of the number of elements and the number of attributes
      return (elParent.getContentSize() + elParent.getAttributes().size());
    }
    else { //No Document or Element -->Leaf --> no children
      return 0;
    }
  }
```

```
  /**
   * Returns true if node is a leaf.
   * See superclass' method description for more.
   * @param node - a node in the tree, obtained from this data source
   * @return true if node is a leaf
   */
  public boolean isLeaf(Object node) {
    //Only Element and Document have Children
    //check document
    if (node instanceof Document) {
      Document docNode = (Document) node;
      if (docNode.getContentSize() == 0) {
        return true;
      }
      else {
        return false;
      }
    }
    //check element
    else if (node instanceof Element) {
      Element elNode = (Element) node; //cast it
      //check if element has attributes too
      if ( (elNode.getAttributes().size() == 0) &&
          (elNode.getContentSize() == 0)) {
        return true;
      }
      else {
        return false;
      }
    }
    else { //It sure is a leaf
     return true; //*** Do not erase! it is necessary for the TreeModel implementaion
    }
 }

  /**
   * Returns the index of child in parent. If parent  is null or child is null, returns -1.
    *See superclass' method description for more.<br>
    * Parent can be: a Document or an Element
    * Child can be: an Element, a Content or an Attribute.<br>
   * In the case of an Element or a Content child, it returns the sum of the number of
attributes that
    * the parent Element has, plus the index of the  Element or Content child.
    * @param parent a node in the tree, obtained from this data source
  * @param child the node we are interested in
  * @return the index of the child in the parent, or -1 if either child or parent are null
  */
  public int getIndexOfChild(Object parent, Object child) {
    /* To begin with, first satisfy:
     * "If parent  is null or child is null, returns -1."*/
    if (parent == null) {
      return -1;
    }
    if (child == null) {
      return -1;
    }

    //Taking care of parent
    //check if parent is a document
    if (parent instanceof Document) {
      Document docParent = (Document) parent; //cast it
      return docParent.indexOf( (Content) child); //cast child to Content type
    }
    //check if parent is an Element
    else if (parent instanceof Element) {
      Element elParent = (Element) parent; //cast it
      //Taking care of child
      //check if child is an Attribute,
      if (child instanceof Attribute) {
        Attribute attrChild = (Attribute) child;
        return elParent.getAttributes().indexOf(attrChild);
      }
      //check if child is an Element,
      else if (child instanceof Element) { //child is an Element
        Element elChild = (Element) child; //cast it
        //make sure it is child of its parent
        if (elParent.indexOf( (Content) child) == -1) {
```

```
          System.err.println(
              "*** ERROR in getIndexOfChild: not a child of this parent.");
          return -100;
        }
        else { //it surely is child of its parent, return the sum as described in API.
          return (elParent.indexOf(elChild) + elParent.getAttributes().size()); //cast child to
Content type
        }
      }
      ////child is a Content, return the sum as described in API.
      else if (child instanceof Content) {
        Content conChild = (Content) child;
        return (elParent.indexOf(conChild) + elParent.getAttributes().size()); //cast child to
Content type
      }//THIS SHOULD NEVER BE EXECUTED
      else {
        System.err.println("*** ERROR in getIndexOfChild: Incompatible types--> Child is not of
Content type. Child: " +
                           child.toString());
        return -100;
      }
    }
    else { //THIS SHOULD NEVER BE EXECUTED
      System.err.println("*** ERROR in getIndexOfChild!"); //Unspecified Error
      return -100;
    }
  }

}//end class
```

admintool.jdomtreemodel.XMLWhitespaceFilter.java

```
package admintool.jdomtreemodel;

import org.xml.sax.helpers.XMLFilterImpl;
import org.xml.sax.SAXException;
import org.xml.sax.XMLReader;
/**
* <p>
  *  XML filter used in order to avoid loading ignorable whitespace as Text nodes.
* </p>
* <p>
   * The application should be able to load XML files without their corresponinding DTD ot XSD
for
  * validation. So, if an XML filter is not used while loading the XML file with the SAXBuilder,
and
  * SAXBuilder would mistakely consider the whitespace Text nodes, as Text nodes with chatacter
data. That is
  * the SAXBuilder cannot distinguish which Text node is useful and which has unwanted
  * whitespace text. <br>
  * The only way to specify to SAXBuilder which are the useful Text nodes is to use
  * a XML filter.
  * </p>
 */
public class XMLWhitespaceFilter
    extends XMLFilterImpl {

  /**
   * Default constructor.
 */
  public XMLWhitespaceFilter() {
  }

  /**
   * Constructor passing a XMLReader in superclass constructor - not used in this version of the
application.
   * @param parent XMLReader
   */
  public XMLWhitespaceFilter(XMLReader parent) {
    super(parent);
  }
```

```
  /**
   * Overriding characters method of of SAXBuilder to filter character data the
   * create the Text nodes.<br>
   * Creates a string from character data and trims its whitespace with String.trim method.
   * If trimmed string is not an empty string, then the character data are not whitspace data,
thus
   * pass them to the superclass' method to create the Text node. Else, do nothing.
   * @param ch character data
   * @param start start index
   * @param length length
   * @throws SAXException
   */
    public void characters(char[] ch, int start, int length) throws SAXException {
    //create a string from character data
    String string = null;
    try {
      //public String(char[] value,int offset,int count)
      string = new String(ch, start, length);
    }
    catch (IndexOutOfBoundsException ioobe) {
      System.err.println("*** Exception in filter: " + ioobe);
      ioobe.printStackTrace();
    }
    //trim whit space
    String trimmedString = string.trim();
    //if trim string != empty string, then it is not a whitespace
    if (trimmedString.compareTo("") != 0) {
      //pass it to superclass method
      super.characters(ch, start, length);
    }
    //else, trim string == empty, ignore...
  } //end method characters
} //end class
```

admintool.jdomtreemodel.XPathEvaluator.java

```
package admintool.jdomtreemodel;

import java.util.*;
import org.jdom.*;

/**
 *
 * <p>Utility class that encloses the mechanism of
 * evaluting the xpath of a given node in a JDOM document. A <i> node </i> can be an
* Element, an Attribute, CDATA or Text.</p>
* <p>
* The XPath is return by the getXPath method. For example to estimate the following XPath: <br>
*  <code>/dataStructureSource/dataStructureTable[3]/dataStructureField[5]</code><br>
* the XPath is divided into 3 parts. One part for each element. These parts are estimated
* by the getElementXPath method.
* </p>
* @see admintool.TreeTransferHandler
 */
public class XPathEvaluator {
  /** XPath of this node is evaluted.*/
  private Object object;

  /**
   * Default constructor.
   */
  public XPathEvaluator() {
  }

  /**
   * <p>Estimates the part of XPath of the object passed. <br>
   * </p>
   * <p>
   * <ul>
   * <li>Creates a string buffer to store the string:</li>
   * <li>adds the element's name</li>
```

```
   * <li>adds the selector ([x]) that this element has in its parent child list. if it's the
only child,  no selector is added</li>
   * </ul>
   * </p>
   *
   *
   * @param currentNode the element that its part of the XPath expression is estimated
   * @return String the part of XPath expression of this element
   */
  private String getElementXPath(Object currentNode) {

    if (currentNode instanceof Element) {
      Element current = (Element) currentNode;
      //create string buffer to store the XPath part and add the current element name
      StringBuffer buf = new StringBuffer("/").append(current.getName());
      //get parent
      Element parent = current.getParentElement();

      // if we're at the root element, do not append [x] in the dataSourceStructure
      if (parent.getParent().getParent()==null){
        return buf.toString();
      }

      // Check for other siblings of the same name and namespace
      Namespace ns = current.getNamespace();
      List siblings = parent.getChildren(current.getName(), ns);
      //count siblings
      int total = 0;
      Iterator i = siblings.iterator();
      while (i.hasNext()) {
        total++;
        if (current == i.next()) { //obviously do not count current!
          break;
        }
      }

      // No selector needed if this is the only element
      if ( (total == 1) && (!i.hasNext())) {
        return buf.toString();
      }
      //else add number of this element
      return buf.append("[")
          .append(String.valueOf(total))
          .append("]").toString();
    }
    //error case
    else {
      System.err.println("ERROR in XPAthEvaluator -  getElementXpath: Cannot evaluate! No
Element found *** \n\t" +
          currentNode.toString());
      return "*** No Element ***";
    }
  }

  /**
   * <p> Returns the complete XPath expression for this node. A <i> node </i>
   * can be an Element or an Attribute. </p>
   * <p>
   * Firstly, it initializes the object variable. There are several cases of the object:
   * <ul>
   * <li>Case of: Element</li>
   * <ul>
   * <li>if element is root element return error message</li>
   * <li>store in a filo queue  the Elements from selection to root</li>
   * <li>create a string buffer to store the XPath string and for every element
   *    from root to selection, append to buffer the XPath part of each element. The XPath part
is estimated
   *    from getElementXPath method.</li>
   * </ul>
   * <li>Case of: Attribute</li>
   * <ul>
   * <li>Creates a new instance of XPathEvaluator to estimate the XPath of its parent
element.</li>
   * <li>Appends the XPath part of the Attribute</li>
   * </ul>
   * <li>Case of: CDATA. XPath does not support CDATA the way it supports Text,
        so only the parent element XPath is returned.</li>
```

```
   * <ul>
   * <li>Creates a new instance of XPathEvaluator to estimate the XPath of its parent
element.</li>
   * </ul>
   * <li>Case of: Text</li>
   * <ul>
   * <li>Creates a new instance of XPathEvaluator to estimate the XPath of its parent
element.</li>
   * <li>Appends the XPath part of the Text</li>
   * </ul>
   * </ul>
   * </p>
   *
   * @return complete XPath.
   * @param o the object that we want to estimate its XPath
   * @see  getElementXPath(Object )
   */
  public String getXPath(Object o) {
    //initialize variable with passed object
    this.object = o;
    //Cases of object:
    // Case of: element
    if (object instanceof Element) {
      Element element = (Element) object;
      Element parent = element.getParentElement();

      // If this is null, we're at the root
      //for root (=dataStructure) return an empty string
      if (parent == null) {
        return "\"Please selected a descendant to display its Xpath\"" ;//return "/" +
element.getName();
      }

      // Otherwise, build a path back to the root
      //create a filo queue to store the Elements from selection to root
      Stack stack = new Stack();
      stack.add(element);
      do {
        //do not put root to stack, because root is not part of the data strusture
        if (!parent.getName().equals("dataStructure")) {
          stack.add(parent);
        }
        parent = parent.getParentElement();
      } while (parent != null);

      // Build the path
      //create a string buffer to store the XPath string
      StringBuffer xpath = new StringBuffer();
      //for every element from root to selection
      while (!stack.isEmpty()) {
        //append to buffer the XPath part of each element
        xpath.append(getElementXPath( (Object) stack.pop()));
      }
      return xpath.toString();
    }

    //Case of: attribute
    if (object instanceof Attribute) {
      Attribute attribute = (Attribute) object;
      //estimate the XPath of the Element that this attribute belongs
      Element parent = attribute.getParent();
      XPathEvaluator xpe = new XPathEvaluator();
      StringBuffer xpath = new StringBuffer(xpe.getXPath(parent))
          //add attribute part of XPath
          .append("/")
          .append("@")
          .append(attribute.getName())
          .append("");
      return xpath.toString();
    }

    //Case of: CDATA
    //XPath does not support CDATA the way it supports text,
    //so only the parent element XPath is returned in this case
    if (object instanceof CDATA) {
      CDATA cdata = (CDATA) object;
      //estimate the XPath of the Element that this CDATA belongs
```

```
      XPathEvaluator xpe = new XPathEvaluator();
      StringBuffer xpath = new StringBuffer(
          xpe.getXPath(cdata.getParentElement()));
      return xpath.toString();
    }

    //Case of: Text
    if (object instanceof Text) {
      Text text = (Text) object;
      //estimate the XPath of the Element that this Text belongs
      XPathEvaluator xpe = new XPathEvaluator();
      StringBuffer xpath = new StringBuffer(
          xpe.getXPath(text.getParentElement()))
           //add Text part of Xpath
          .append("[child::text()]");
      return xpath.toString();
    }

    // Other node types could follow here, but for now return ...
    return "Node type not supported yet.";
  }
}
```