ΕΘΝΙΚΟ
ΜΕΤΣΟΒΙΟ
ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ
ΗΛΕΚΤΡΟΛΟΓΩΝ
ΜΗΧΑΝΙΚΩΝ ΚΑΙ
ΜΗΧΑΝΙΚΩΝ
ΥΠΟΛΟΓΙΣΤΩΝ

ΠΑΝΕΠΙΣΤΗΜΙΟ
ΠΕΙΡΑΙΑ
ΤΜΗΜΑ
ΒΙΟΜΗΧΑΝΙΚΗΣ
ΔΙΟΙΚΗΣΗΣ ΚΑΙ
ΤΕΧΝΟΛΟΓΙΑΣ

ΕΘΝΙΚΟ ΚΑΙ
ΚΑΠΟΔΙΣΤΡΙΑΚΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΑΘΗΝΩΝ
ΤΜΗΜΑ
ΟΙΚΟΝΟΜΙΚΩΝ
ΕΠΙΣΤΗΜΩΝ

ΔΙΕΠΙΣΤΗΜΟΝΙΚΟ – ΔΙΑΠΑΝΕΠΙΣΤΗΜΙΑΚΟ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
«ΤΕΧΝΟ-ΟΙΚΟΝΟΜΙΚΑ ΣΥΣΤΗΜΑΤΑ»

Χρήση Προτύπων Σχεδίασης για την Ανάπτυξη Διαδικτυακής Εφαρμογής Αποτίμησης Χρηματοοικονομικών Παραγώγων Προϊόντων

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Βασίλειος Κ. Παπακώστας

Επιβλέπων : Δημήτριος Ασκούνης
Επίκουρος Καθηγητής Ε.Μ.Π.

Αθήνα, Σεπτέμβριος 2005

ΕΘΝΙΚΟ
ΜΕΤΣΟΒΙΟ
ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ
ΗΛΕΚΤΡΟΛΟΓΩΝ
ΜΗΧΑΝΙΚΩΝ ΚΑΙ
ΜΗΧΑΝΙΚΩΝ
ΥΠΟΛΟΓΙΣΤΩΝ

ΠΑΝΕΠΙΣΤΗΜΙΟ
ΠΕΙΡΑΙΑ
ΤΜΗΜΑ
ΒΙΟΜΗΧΑΝΙΚΗΣ
ΔΙΟΙΚΗΣΗΣ ΚΑΙ
ΤΕΧΝΟΛΟΓΙΑΣ

ΕΘΝΙΚΟ ΚΑΙ
ΚΑΠΟΔΙΣΤΡΙΑΚΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΑΘΗΝΩΝ
ΤΜΗΜΑ
ΟΙΚΟΝΟΜΙΚΩΝ
ΕΠΙΣΤΗΜΩΝ

ΔΙΕΠΙΣΤΗΜΟΝΙΚΟ – ΔΙΑΠΑΝΕΠΙΣΤΗΜΙΑΚΟ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
«ΤΕΧΝΟ-ΟΙΚΟΝΟΜΙΚΑ ΣΥΣΤΗΜΑΤΑ»

Χρήση Προτύπων Σχεδίασης για την Ανάπτυξη Διαδικτυακής Εφαρμογής Αποτίμησης Χρηματοοικονομικών Παραγώγων Προϊόντων

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Βασίλειος Κ. Παπακώστας

Επιβλέπων : Δημήτριος Ασκούνης
Επίκουρος Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 29η Σεπτεμβρίου 2005.

....	....	....
Δ. Ασκούνης	Ι. Ε. Σαμουηλίδης	Ι. Ψαρράς
Επίκ. Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Αναπ. Καθηγητής Ε.Μ.Π.

Αθήνα, Σεπτέμβριος 2005

....
Βασίλειος Κ. Παπακώστας

Διπλωματούχος Μηχανικός Η/Υ και Πληροφορικής Πανεπιστημίου Πατρών

Πρόλογος

Η παρούσα διπλωματική εργασία εκπονήθηκε στα πλαίσια του διαπανεπιστημιακού μεταπτυχιακού προγράμματος σπουδών «Τεχνο-Οικονομικά Συστήματα» που διοργανώνεται από το Εθνικό Μετσόβιο Πολυτεχνείο και στο οποίο συμμετέχουν το Πανεπιστήμιο Πειραιά και το Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών.

Θα ήθελα να ευχαριστήσω τον υπεύθυνο καθηγητή κ. Δημήτριο Ασκούνη που μου έδωσε την ευκαιρία να ασχοληθώ με ένα εξαιρετικά ενδιαφέρον θέμα και ιδιαίτερα τον υποψήφιο διδάκτορα κ. Παναγιώτη Ξυδώνα, που ήταν ο επιβλέπων της εργασίας, για τις γνώσεις του και τη πολύτιμη βοήθειά του στο πεδίο της Χρηματοοικονομικής Μηχανικής και των χρηματοοικονομικών παραγώγων ειδικότερα, καθώς ήταν ένα καινούργιο αντικείμενο για μένα.

Σεπτέμβριος 2005

Βασίλειος Παπακώστας

Περίληψη

Τα χρηματοοικονομικά παράγωγα αποτελούν σήμερα ένα ισχυρό εργαλείο στα χέρια χρηματοπιστωτικών ιδρυμάτων και επενδυτών και συγκεντρώνουν όλο και μεγαλύτερο ενδιαφέρον. Χρησιμοποιούνται τόσο για την αποκόμιση κερδών όσο και για την αντιστάθμιση κινδύνου χαρτοφυλακίου. Πλήθος υπολογιστικών μοντέλων έχει αναπτυχθεί για την αποτίμησή τους και τη μελέτη της συμπεριφοράς τους. Αποτελούν, μεταξύ άλλων, αντικείμενο μελέτης ενός σχετικά νέου επιστημονικού κλάδου, της Χρηματοοικονομικής Μηχανικής (Financial Engineering).

Σκοπός της παρούσας διπλωματικής εργασίας είναι να συγκεράσει τη θεωρία και τα μαθηματικά μοντέλα που έχουν αναπτυχθεί γύρω από χρηματοοικονομικά παράγωγα με τις σύγχρονες τάσεις στο χώρο της Τεχνολογίας Λογισμικού. Αναφερόμαστε στην ανάπτυξη διαδικτυακών (web-based) εφαρμογών, την αρχιτεκτονική πολλαπλών επιπέδων (multi-tiered) και τη χρήση προτύπων σχεδίασης.

Προϊόν της παρούσας εργασίας είναι η σχεδίαση και ανάπτυξη μίας διαδικτυακής (web-based) εφαρμογής για την αποτίμηση χρηματιστηριακών παραγώγων και χαρτοφυλακίων σε πραγματικό χρόνο. Η σχεδίαση βασίζεται εξολοκλήρου στη χρήση προτύπων σχεδίασης, τόσο γενικών όσο και εξειδικευμένων για την ανάπτυξη διαδικτυακών εφαρμογών πολλαπλών επιπέδων. Υλοποιούνται τα κυριότερα μοντέλα (εξίσωση Black-Scholes, διωνυμικές μέθοδοι, προσομοίωση Monte Carlo) για την αποτίμηση παραγώγων και χαρτοφυλακίων. Η υλοποίηση βασίζεται στη δημοφιλή τεχνολογία J2EE και στη γλώσσα προγραμματισμού Java.

Τέλος παρέχονται τα αποτελέσματα χρήσης της εφαρμογής σε πραγματικά δεδομένα της Αγοράς Παραγώγων του Χρηματιστηρίου Αθηνών.

Λέξεις Κλειδιά: χρηματοοικονομικά παράγωγα, μοντέλα αποτίμησης, Black-Scholes, διωνυμικά δένδρα, Monte Carlo, πρότυπα σχεδίασης, Java, J2EE, EJB, JBoss, Unified Process

Abstract

Financial derivatives constitute today a powerful tool for financial institutions and investors and attract more and more interest. They are used both for gaining profit through speculation and for portfolio hedging. A number of computational models have been developed for pricing them and studying their behavior. They are, among others, the field of study of a relatively new discipline, Financial Engineering.

The purpose of the present thesis is to mingle the theory and mathematical models that have emerged around financial derivatives with the modern trends in Software Engineering. We refer to the development of web-based applications, the adoption of multi-tiered architectures and the use of design patterns.

The outcome of the present thesis is the design and development of a web-based real-time application for financial derivatives pricing. The design is entirely based on design patterns, both generic and specific for the development of web-based multi-tiered applications. The principal models (Black-Scholes, binomial trees, Monte Carlo simulation) for derivatives and portfolio pricing are implemented. Their implementation is based on popular J2EE technology and Java programming language.

Finally, we present the results of using the application on real data from the Athens Derivatives Exchange.

Keywords: financial derivatives, pricing models, Black-Scholes, binomial trees, Monte Carlo, design patterns, Java, J2EE, EJB, JBoss, Unified Process

Πίνακας περιεχομένων

1

Εισαγωγή

Το κεφάλαιο αυτό πραγματεύεται σε περίληψη το αντικείμενο της διπλωματικής εργασίας. Εξηγεί το σκεπτικό πίσω από την επιλογή του θέματος, της μεθοδολογίας και των προγραμματιστικών εργαλείων. Παραθέτει αναφορές σε βιβλία και δημοσιεύσεις σχετικές με το θέμα.

1.1 Αντικείμενο της Εργασίας

Τα χρηματοοικονομικά παράγωγα – χρηματοοικονομικά προϊόντα που η τιμή τους εξαρτάται από κάποιον υποκείμενο χρηματιστηριακό τίτλο ή προϊόν – έχουν καταστεί ιδιαίτερα δημοφιλή διεθνώς μεταξύ των επενδυτών για την αντιστάθμιση κινδύνου και την αποκόμιση κερδών. Η διαρκώς αυξανόμενη χρήση τους έχει οδηγήσει στην ανάπτυξη πλήθους μαθηματικών μοντέλων για την αποτίμησή τους και την εξεύρεση των χαρακτηριστικών τους. Η μελέτη τους αποτελεί μεταξύ άλλων αντικείμενο της Χρηματοοικονομικής Μηχανικής (Financial Engineering), ενός νέου κλάδου που συνδυάζει τα Εφαρμοσμένα Μαθηματικά με την Χρηματοοικονομική Θεωρία.

Για την αξιοποίηση των μοντέλων αυτών έχει αναπτυχθεί ένα πλήθος υπολογιστικών συστημάτων και εφαρμογών. Κάποιες αποτελούν εσωτερικές (in-house) εφαρμογές για μεγάλα χρηματοπιστωτικά ιδρύματα και επενδυτικούς οίκους ενώ άλλες πωλούνται σαν προϊόντα λογισμικού. Παρά την πληθώρα των υπολογιστικών μεθόδων και μοντέλων που εμφανίζονται στη σχετική βιβλιογραφία, η ύπαρξη βιβλίων και δημοσιεύσεων σχετικά με τη σχεδίαση και την υλοποίηση αλγορίθμων και υπολογιστικών συστημάτων για την εφαρμογή τους είναι περιορισμένη. Περιορίζονται στην παραδοσιακή αντικειμενοστραφή (object-oriented) σχεδίαση, την αποσπασματική χρήση κάποιων προτύπων σχεδίασης (design patterns) και τις γλώσσες προγραμματισμού C++ και Visual Basic.

Το αντικείμενο που η εργασία μας πραγματεύεται είναι η σχεδίαση και η υλοποίηση μίας διαδικτυακής (web-based) εφαρμογής για την αποτίμηση χρηματοοικονομικών παραγώγων και χαρτοφυλακίων σε πραγματικό χρόνο. Η εφαρμογή ανακτά τιμές παραγώγων και υποκείμενων τίτλων σε πραγματικό χρόνο από παρόχους χρηματιστηριακών δεδομένων και χρησιμοποιεί μοντέλα αποτίμησης για τον υπολογισμό της θεωρητικής τιμής και των χαρακτηριστικών (Greek letters) παραγώγων και χαρτοφυλακίων. Πέρα από την εμφάνιση των αποτελεσμάτων σε web οθόνες μέσω διαδικτύου και την αποθήκευσή τους για στατιστική επεξεργασία, έχει τη δυνατότητα αποστολής ειδοποιήσεων (μέσω ηλεκτρονικού ταχυδρομείου για παράδειγμα) όταν οι τιμές ή τα χαρακτηριστικά των παραγώγων ή χαρτοφυλακίων ικανοποιούν μία δεδομένη συνθήκη, π.χ. επιτρέπουν την εκτέλεση εξισορροπητικής κερδοσκοπίας (arbitrage) ή επιτάσσουν την αναδιάρθρωση ενός χαρτοφυλακίου.

Κεντρικό ρόλο στη σχεδίαση της εφαρμογής κατέχουν τα πρότυπα σχεδίασης (design patterns). Η σχεδίαση βασίζεται σχεδόν εξολοκλήρου σε αυτά. Εφαρμόζονται τόσο γενικευμένα πρότυπα σχεδίασης όσο και πρότυπα σχεδίασης για web-based εφαρμογές (J2EE design patterns).

Η σχεδίαση που παρουσιάζουμε αποσκοπεί στην εύκολη εισαγωγή νέων μοντέλων αποτίμησης, νέων παραγώγων προϊόντων και νέων παρόχων χρηματιστηριακών δεδομένων.

1.2 Υπόβαθρο

Ο M. Joshi [Joshi04] και ο D. Daffy [Daffy04] επιχειρούν να εφαρμόσουν τους κανόνες του αντικειμενοστραφούς προγραμματισμού και χρησιμοποιούν πρότυπα σχεδίασης για την αποτίμηση χρηματοοικονομικών παραγώγων. Ο J. London [London05] συγκεντρώνει ένα πλήθος μοντέλων αποτίμησης υλοποιημένων σε γλώσσα προγραμματισμού C++.

Οι J. Zhang και E. Sternbach [ZS96] μοντελοποιούν χρηματιστηριακά παράγωγα χρησιμοποιώντας πρότυπα σχεδίασης. Οι M. van der Meij, D. Schouten και A. Eliëns [MSE99] περιγράφουν τη χρήση προτύπων σχεδίασης σε μία εφαρμογή αποτίμησης παραγώγων. Η P. Marsura [Marsura98] περιγράφει μία ολοκληρωμένη εφαρμογή αποτίμησης παραγώγων και χαρτοφυλακίων με χρήση αντικειμένων και προτύπων σχεδίασης.

Οι T. Eggenschwiler, E. Gamma και A. Birrer [EG92, BE93] περιγράφουν τη χρήση αντικειμένων και πλαισίων εργασίας (frameworks) στη Χρηματοοικονομική Μηχανική. Οι

Μ. Κουλισιάνης, Γ. Τσώλης και Θ. Παπαθεοδώρου [KTP02] περιγράφουν μία web-based εφαρμογή για την αποτίμηση παραγώγων σε τεχνολογία PHP χρησιμοποιώντας τη μεθοδολογία «Περιβάλλον Επίλυσης Προβλημάτων» (Problem Solving Environment).

2

Στοιχεία Χρηματοοικονομικής Μηχανικής

Χρηματοοικονομική Μηχανική (Financial Engineering) είναι ο κλάδος των εφαρμοσμένων μαθηματικών που έχει ως αντικείμενο μελέτης τη συμπεριφορά των χρηματοοικονομικών εργαλείων. Στο κεφάλαιο αυτό θα αναφερθούμε στα Χρηματοοικονομικά Παράγωγα και την Αντιστάθμιση Κινδύνου.

2.1 Χρηματοοικονομικά Παράγωγα (Financial Derivatives)

Τα χρηματοοικονομικά παράγωγα (financial derivatives) είναι χρηματοοικονομικά προϊόντα που η αξία τους εξαρτάται («παράγεται») από έναν ή περισσότερους υποκείμενους τίτλους.

Τα παράγωγα έχουν καταστεί εξαιρετικά δημοφιλή μεταξύ των επενδυτών τα τελευταία χρόνια εξαιτίας των αυξημένων δυνατοτήτων που προσφέρουν για την αποκόμιση κερδών και την αντιστάθμιση κινδύνου. Διαπραγματεύονται σε ένα πλήθος χρηματιστηρίων διεθνώς. Στην Ελλάδα διαπραγματεύονται στην Αγορά Παραγώγων του Χρηματιστηρίου Αθηνών.

2.1.1 Συμβόλαια Μελλοντικής Εκπλήρωσης (Futures)

Ένα Συμβόλαιο Μελλοντικής Εκπλήρωσης (Futures Contract) είναι μία συμφωνία για αγορά ή πώληση ενός περιουσιακού στοιχείου σε μία δεδομένη τιμή σε μία συγκεκριμένη χρονική στιγμή στο μέλλον. Το περιουσιακό αυτό στοιχείο (υποκείμενος τίτλος) μπορεί να είναι εμπόρευμα, μετοχή, χρηματιστηριακός δείκτης, νόμισμα κτλ.

Το συμβόλαιο αφορά δύο αντισυμβαλλόμενους. Ο ένας αναλαμβάνει την υποχρέωση να αγοράσει τον υποκείμενο τίτλο (long position) και ο άλλος την υποχρέωση να πουλήσει τον ίδιο τίτλο (short position).

Τα Συμβόλαια Μελλοντικής Εκπλήρωσης (ΣΜΕ) διαπραγματεύονται σε χρηματιστήρια. Το μέγεθος του συμβολαίου, ο ακριβής χρόνος παράδοσης, οι όροι παράδοσης και η διαδικασία εξασφάλισης της αξιοπιστίας των αντισυμβαλλομένων καθορίζονται από το χρηματιστήριο.

2.1.2 Δικαιώματα Προαίρεσης (Options)

Ένα Δικαίωμα Προαίρεσης είναι ένα συμβόλαιο που δίνει στον αγοραστή του το δικαίωμα να αγοράσει ή να πουλήσει ένα περιουσιακό στοιχείο σε μία δεδομένη τιμή σε ή μέχρι μία συγκεκριμένη χρονική στιγμή στο μέλλον. Το περιουσιακό αυτό στοιχείο (υποκείμενος τίτλος) μπορεί να είναι εμπόρευμα, μετοχή, χρηματιστηριακός δείκτης, νόμισμα κτλ.

Το συμβόλαιο αφορά δύο αντισυμβαλλόμενους. Σε αντίθεση με ένα ΣΜΕ, ο αγοραστής ενός συμβολαίου δικαιώματος προαίρεσης (option) δεν είναι υποχρεωμένος να προχωρήσει στην αγοραπωλησία του υποκείμενου τίτλου. Θα το κάνει μόνο όταν η τιμή εξάσκησης (strike price) του συμβολαίου σε σχέση με την τρέχουσα τιμή (spot price) του υποκείμενου τίτλου τη στιγμή της εξάσκησης είναι τέτοια που να του εξασφαλίζει κέρδος. Ο πωλητής ωστόσο είναι υποχρεωμένος να τηρήσει τη συμφωνία εφόσον ο αγοραστής το αξιώσει.

Τα Δικαιώματα Προαίρεσης διαπραγματεύονται σε χρηματιστήρια. Το μέγεθος του συμβολαίου, ο ακριβής χρόνος παράδοσης, οι όροι παράδοσης και η διαδικασία εξασφάλισης της αξιοπιστίας των αντισυμβαλλομένων καθορίζονται από το χρηματιστήριο.

Ανάλογα με τον τύπο της συναλλαγής, τα συμβόλαια δικαιώματος προαίρεσης διακρίνονται σε δικαιώματα αγοράς και δικαιώματα πώλησης:

2.1.2.1 Δικαιώματα Αγοράς (Call Options)

Στα δικαιώματα αγοράς ο αγοραστής αποκτά το δικαίωμα να αγοράσει τον υποκείμενο τίτλο (long position). Ο πωλητής αποκτά την υποχρέωση να πουλήσει τον αντίστοιχο τίτλο (short position). Είναι στην ευχέρεια του αγοραστή να απαιτήσει ή όχι την εκτέλεση του συμβολαίου.

2.1.2.2 Δικαιώματα Πώλησης (Put Options)

Στα δικαιώματα πώλησης ο αγοραστής αποκτά το δικαίωμα να πουλήσει τον υποκείμενο τίτλο (long position). Ο πωλητής αποκτά την υποχρέωση να αγοράσει τον αντίστοιχο τίτλο (short position). Είναι στην ευχέρεια του αγοραστή να απαιτήσει ή όχι την εκτέλεση του συμβολαίου.

Ανάλογα με το χρόνο στον οποίο μπορεί να εκτελεστεί, ένα συμβόλαιο δικαιώματος προαίρεσης μπορεί να είναι ευρωπαϊκού ή αμερικανικού τύπου:

2.1.2.3 Δικαιώματα Ευρωπαϊκού Τύπου

Στα δικαιώματα προαίρεσης ευρωπαϊκού τύπου ο αγοραστής έχει το δικαίωμα να αξιώσει την εκτέλεση του συμβολαίου μόνο κατά την ημερομηνία που αναφέρεται στο συμβόλαιο (exercise date ή maturity).

2.1.2.4 Δικαιώματα Αμερικανικού Τύπου

Στα δικαιώματα προαίρεσης αμερικανικού τύπου ο αγοραστής έχει το δικαίωμα να αξιώσει την εκτέλεση του συμβολαίου οποιαδήποτε στιγμή κατά τη διάρκεια της περιόδου μέχρι την ημερομηνία που αναφέρεται στο συμβόλαιο (exercise date ή maturity).

2.1.2.5 Ισοτιμία Αγοράς-Πώλησης (Put-Call Parity)

Υπάρχει μία μαθηματική σχέση ανάμεσα στην τιμή c ενός δικαιώματος αγοράς ευρωπαϊκού τύπου και στην τιμή p του αντίστοιχου δικαιώματος πώλησης. Η σχέση αυτή είναι γνωστή ως Ισοτιμία Αγοράς-Πώλησης (Put-Call Parity). Για μία μετοχή που δεν αποδίδει μέρισμα, εκφράζεται από την εξίσωση:

$$c + Ke^{-rT} = p + S_0$$

όπου:

S_0: Τρέχουσα τιμή μετοχής

K: Τιμή εξάσκησης δικαιώματος

r: Επιτόκιο χωρίς κίνδυνο

T: Χρόνος μέχρι τη λήξη του δικαιώματος

2.2 Μοντέλα Αποτίμησης

Έχουμε αναφέρει ότι η αξία ενός χρηματοοικονομικού παραγώγου εξαρτάται από την αξία του υποκείμενου τίτλου και τα χαρακτηριστικά του συμβολαίου. Μπορούμε μελετώντας τα χαρακτηριστικά και την τρέχουσα τιμή του υποκείμενου τίτλου να προσεγγίσουμε με ακρίβεια την αξία που θεωρητικά θα έπρεπε να έχει ένα παράγωγο προϊόν σε μία δεδομένη χρονική στιγμή.

Αν η διαφορά ανάμεσα στην τρέχουσα και τη θεωρητική τιμή ενός παραγώγου είναι σημαντική, ένας επενδυτής έχει την ευκαιρία να πραγματοποιήσει εγγυημένο κέρδος (arbitrage). Για το λόγο αυτό η αποτίμηση χρηματοοικονομικών παραγώγων έχει γίνει αντικείμενο εκτεταμένης μελέτης τα τελευταία χρόνια. Ειδικότερα έχουν μελετηθεί τα δικαιώματα προαίρεσης, καθώς για τα ΣΜΕ η αποτίμηση είναι αρκετά απλή.

Ανώτερα μαθηματικά εργαλεία, όπως ο Στοχαστικός Λογισμός (Stochastic Calculus), έχουν χρησιμοποιηθεί για την ερμηνεία της συμπεριφοράς ενός χρηματοοικονομικού προϊόντος (μετοχής, χρηματιστηριακού δείκτη, νομίσματος). Πλήθος μοντέλων αποτίμησης (αναλυτικών και αριθμητικών μεθόδων) έχουν αναπτυχθεί και εφαρμοστεί για την αποτίμηση παραγώγων (derivatives pricing).

Θα παρουσιάσουμε εν συντομία τα πιο διαδεδομένα από αυτά:

2.2.1 Μοντέλο Black-Scholes

Οι F. Black και M. Scholes [BS73] οδήγησαν σε μία σημαντική καινοτομία στην αποτίμηση δικαιωμάτων προαίρεσης καταλήγοντας στην ομώνυμη εξίσωση:

2.2.1.1 Εξίσωση Black-Scholes

$$\frac{\partial f}{\partial t} + rS\frac{\partial f}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 f}{\partial S^2} = rf$$

όπου:

f: Τιμή δικαιώματος
S: Τιμή υποκείμενου τίτλου
r: Επιτόκιο χωρίς κίνδυνο
σ: Μεταβλητότητα υποκείμενου τίτλου

Από την εξίσωση Black-Scholes προκύπτουν αναλυτικές μαθηματικές εξισώσεις για τον υπολογισμό της θεωρητικής τιμής δικαιωμάτων αγοράς και πώλησης ευρωπαϊκού τύπου σε μετοχή που δεν αποδίδει μέρισμα:

$$c = S_0 N(d_1) - Ke^{-rT} N(d_2)$$

και

$$p = Ke^{-rT} N(-d_2) - S_0 N(-d_1)$$

όπου:

c: Τιμή δικαιώματος αγοράς
p: Τιμή δικαιώματος πώλησης
S_0: Τρέχουσα τιμή μετοχής
K: Τιμή εξάσκησης δικαιώματος
r: Επιτόκιο χωρίς κίνδυνο

T: Χρόνος μέχρι τη λήξη του δικαιώματος

και:

$$d_1 = \frac{\ln(S_0 / K) + (r + \sigma^2 / 2)T}{\sigma\sqrt{T}}$$

$$d_2 = \frac{\ln(S_0 / K) + (r - \sigma^2 / 2)T}{\sigma\sqrt{T}} = d_1 - \sigma\sqrt{T}$$

Η συνάρτηση N(x) είναι η αθροιστική συνάρτηση κατανομής πιθανότητας (cumulative probability distribution function) μίας τυποποιημένης κανονικής κατανομής (standardized normal distribution).

Οι ίδιες φόρμουλες, με μικρές τροποποιήσεις, μπορούν να χρησιμοποιηθούν για την αποτίμηση δικαιωμάτων όπου ο υποκείμενος τίτλος αποδίδει μερίσματα ή μερισματικές αποδόσεις καθώς και δικαιωμάτων αγοράς αμερικανικού τύπου.

2.2.2 Διωνυμικά Δένδρα (Binomial Trees)

Σε περιπτώσεις που δεν υπάρχει αναλυτική φόρμουλα για ένα συγκεκριμένο τύπο παραγώγου (π.χ. δικαίωμα πώλησης αμερικανικού τύπου), είμαστε υποχρεωμένοι να καταφύγουμε σε αριθμητικές μεθόδους για την αποτίμησή του. Τα διωνυμικά δένδρα είναι ιδιαίτερα χρήσιμα σε περιπτώσεις που ο κάτοχός τους έχει δυνατότητα πρώιμης εξάσκησης (early exercise) πριν από την ημερομηνία εξάσκησης (maturity).

Η μέθοδος αυτή αρχικά υποθέτει ότι η τιμή ενός προϊόντος (εμπορεύματος ή χρηματιστηριακού τίτλου) μπορεί σε ένα δεδομένο χρονικό διάστημα να κινηθεί είτε ανοδικά κατά ένα συγκεκριμένο ποσοστό είτε καθοδικά κατά ένα συγκεκριμένο ποσοστό. Χωρίζοντας την περίοδο μέχρι την ημερομηνία εξάσκησης σε έναν αριθμό χρονικών διαστημάτων και υπολογίζοντας όλες τις ενδιάμεσες τιμές για το προϊόν, η μέθοδος αυτή μπορεί να υπολογίζει τη θεωρητική τιμή ενός παραγώγου πάνω στο προϊόν αυτό με μεγάλη ακρίβεια.

2.2.3 Προσομοίωση Monte Carlo (Monte Carlo Simulation)

Η προσομοίωση Monte Carlo χρησιμοποιείται κυρίως όταν η τιμή του παραγώγου εξαρτάται από το ιστορικό ή μονοπάτι (history or path) της τιμής του υποκείμενου τίτλου ή από περισσότερες στοχαστικές μεταβλητές (stochastic variables).

Κατά την προσομοίωση Monte Carlo λαμβάνονται τυχαία μονοπάτια της στοχαστικής μεταβλητής (της τιμής του υποκείμενου τίτλου) σύμφωνα με μία κατανομή πιθανότητας (συνήθως κανονική). Υπολογίζεται η τιμή του παραγώγου για κάθε μονοπάτι και τελικά υπολογίζεται η μέση τιμή του, που αποτελεί και την εκτιμούμενη τιμή. Μπορεί να υπολογιστεί και το κανονικό σφάλμα (standard error) για τη συγκεκριμένη εκτίμηση.

2.3 Αντιστάθμιση Κινδύνου (Hedging)

Τα χρηματοοικονομικά παράγωγα αποτελούν σημαντικότατο εργαλείο για την αντιστάθμιση κινδύνου ενός χαρτοφυλακίου χρηματοοικονομικών προϊόντων.

Ένας επενδυτής μπορεί να περιορίσει ή και να εκμηδενίσει τον κίνδυνο που συνεπάγεται μία επένδυση σε χρηματοοικονομικά προϊόντα παίρνοντας κατάλληλες θέσεις αγοράς ή πώλησης σε παράγωγα προϊόντα. Για το σκοπό αυτό χρησιμοποιούνται κάποια χαρακτηριστικά των παραγώγων και χαρτοφυλακίων που είναι γνωστά ως Ελληνικά Γράμματα (Greek Letters ή απλά Greeks).

2.3.1 Ελληνικά Γράμματα (Greek Letters)

Κάθε ελληνικό γράμμα μετρά το μέγεθος του ρίσκου ενός παραγώγου ή χαρτοφυλακίου από μία διαφορετική διάσταση.

2.3.1.1 Δέλτα (Delta)

Το δέλτα ορίζεται ως ο ρυθμός μεταβολής της τιμής του παραγώγου ή χαρτοφυλακίου σε σχέση με την τιμή του υποκείμενου τίτλου:

$$\Delta = \frac{\partial \Pi}{\partial S}$$

2.3.1.2 Γάμμα (Gamma)

Το γάμμα ορίζεται ως ο ρυθμός μεταβολής του δέλτα του παραγώγου ή χαρτοφυλακίου σε σχέση με την τιμή του υποκείμενου τίτλου:

$$\Gamma = \frac{\partial^2 \Pi}{\partial S^2}$$

2.3.1.3 Θήτα (Theta)

Το θήτα ορίζεται ως ο ρυθμός μεταβολής της τιμής του παραγώγου ή χαρτοφυλακίου σε σχέση με το χρόνο:

$$\Theta = \frac{\partial \Pi}{\partial t}$$

2.3.1.4 Vega

Το vega ορίζεται ως ο ρυθμός μεταβολής της τιμής του παραγώγου ή χαρτοφυλακίου σε σχέση με τη μεταβλητότητα (volatility) της τιμής του υποκείμενου τίτλου:

$$V = \frac{\partial \Pi}{\partial \sigma}$$

2.3.1.5 Ρο (Rho)

Το ρο ορίζεται ως ο ρυθμός μεταβολής της τιμής του παραγώγου ή χαρτοφυλακίου σε σχέση με το επιτόκιο:

$$\rho = \frac{\partial \Pi}{\partial r}$$

3

Ανάλυση

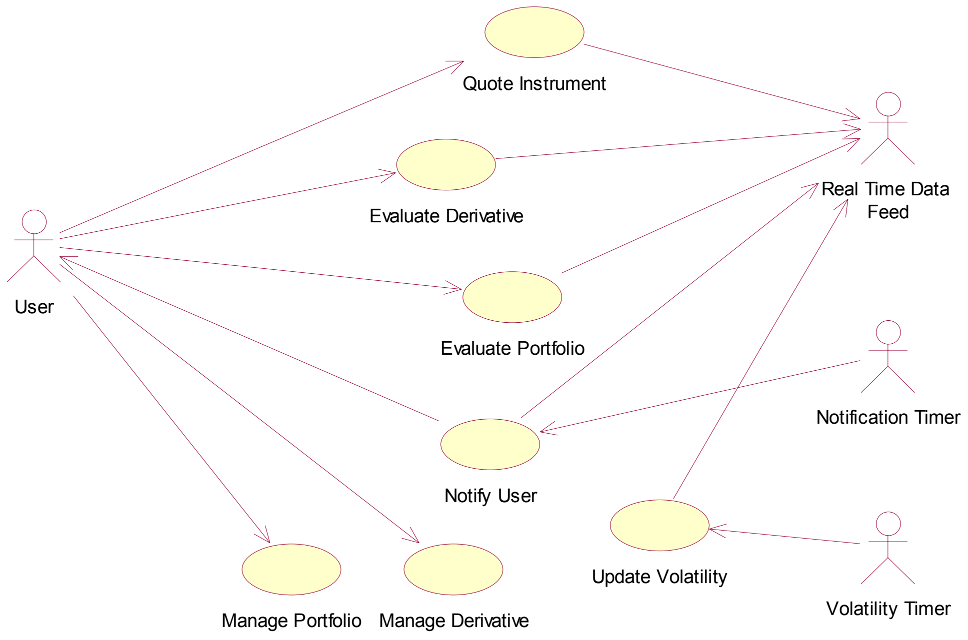
Η μεθοδολογία ανάπτυξης λογισμικού που χρησιμοποιήθηκε για τη σχεδίαση και υλοποίηση της εφαρμογής αποτίμησης χρηματοοικονομικών παραγώγων είναι η Unified Process [JBR99]. Πρόκειται για μία επαναληπτική (iterative) μέθοδο ανάπτυξης λογισμικού για τη δημιουργία αντικειμενοστραφών συστημάτων.

Η Unified Process υποστηρίζει την ακολουθιακή και προσθετική ανάπτυξη (iterative and incremental development). Κατά τη διάρκεια ζωής του έργου ορίζονται σύντομα υποέργα που ονομάζονται επαναλήψεις (iterations). Κάθε επανάληψη έχει τις δικές της δραστηριότητες σχετικές με ανάλυση απαιτήσεων, σχεδίαση, υλοποίηση και έλεγχο που προστίθενται σε αυτές της προηγούμενες επανάληψης. Το αποτέλεσμα κάθε επανάληψης είναι ένα ημιτελές εκτελέσιμο σύστημα. Το αποτέλεσμα της τελευταίας επανάληψης είναι το τελικό σύστημα.

Δε θα παρουσιάσουμε αναλυτικά τις επαναλήψεις και τα αποτελέσματα καθεμίας από αυτές καθώς κάτι τέτοιο είναι πέρα από το σκοπό της παρούσας εργασίας. Θα παρουσιάσουμε τα δημιουργήματα (artifacts) της ανάλυσης, σχεδίασης και υλοποίησης στην τελική τους μορφή. Στο κεφάλαιο αυτό περιγράφονται τα αποτελέσματα της Ανάλυσης Απαιτήσεων (Requirements Analysis) και της Αντικειμενοστραφούς Ανάλυσης (Object-Oriented Analysis).

3.1 Σενάρια Χρήσης (Use Cases)

Ο ορισμός των Σεναρίων Χρήσης (Use Cases) είναι το αποτέλεσμα της Ανάλυσης Απαιτήσεων (Requirements Analysis). Τα σενάρια χρήσης που περιγράφουν τις απαιτήσεις για τη διαδικτυακή εφαρμογή αποτίμησης χρηματοοικονομικών παραγώγων απεικονίζονται στο Σχήμα 3.1:

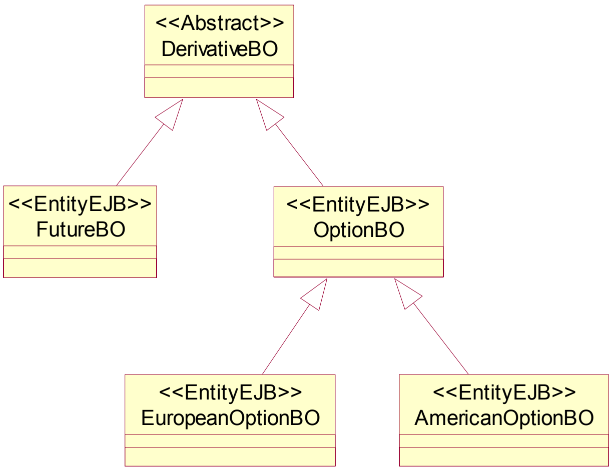


Σχήμα 3.1: Σενάρια Χρήσης

3.2 Μοντέλο Πεδίου (Domain Model)

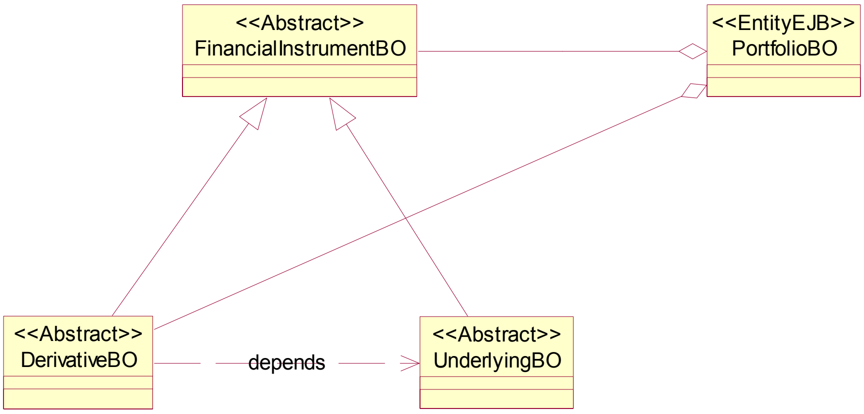
Το Μοντέλο Πεδίου (Domain Model) είναι δημιούργημα (artifact) της Αντικειμενοστραφούς Ανάλυσης (Object-Oriented Analysis). Πρόκειται για μία οπτική παρουσίαση των ιδεατών κλάσεων και χειροπιαστών αντικειμένων του πεδίου (domain) γύρω από το οποίο αναπτύσσεται μία εφαρμογή.

Το Domain Model για τα χρηματοοικονομικά παράγωγα εικονίζεται στο Σχήμα 3.2:



Σχήμα 3.2: Παράγωγα

Η σχέση χαρτοφυλακίων, παραγώγων και υποκείμενων τίτλων εικονίζεται στο Σχήμα 3.3:



Σχήμα 3.3: Χρηματοοικονομικά Εργαλεία

4

Σχεδίαση

Η σχεδίαση της διαδικτυακής εφαρμογής αποτίμησης χρηματοοικονομικών παραγώγων και χαρτοφυλακίων που αποτελεί αντικείμενο της παρούσας διπλωματικής εργασίας προέκυψε μέσα από μία σειρά επαναλήψεων (iterations) στα πλαίσια της μεθοδολογίας ανάπτυξης λογισμικού Unified Process που υιοθετήσαμε.

Στο κεφάλαιο αυτό περιγράφεται λεπτομερώς η σχεδίαση του συστήματος στην τελική της μορφή. Κάποια στοιχεία της σχεδίασης προϋποθέτουν τη χρήση της τεχνολογίας J2EE, για την οποία θα αναφέρουμε περισσότερα στο επόμενο κεφάλαιο.

4.1 Αρχιτεκτονική Πολλαπλών Επιπέδων (Multi-Tiered)

Η εφαρμογή αποτίμησης χρηματοοικονομικών παραγώγων που αποτελεί αντικείμενο της διπλωματικής εργασίας είναι μία διαδικτυακή (web-based) εφαρμογή. Η χρήση του διαδικτύου εισάγει κάποια πολυπλοκότητα στην αρχιτεκτονική της, καθώς ο χρήστης χρησιμοποιεί έναν περιηγητή (browser) για να αποκτήσει πρόσβαση στην εφαρμογή που τρέχει σε κάποιον εξυπηρετητή (server).

Η αρχιτεκτονική που επιλέχθηκε είναι η Αρχιτεκτονική Πολλαπλών Επιπέδων (Multi-Tiered Architecture). Κάθε επίπεδο (tier) είναι υπεύθυνο για ένα τμήμα της λειτουργικότητας του συστήματος. Είναι λογικά διαχωρισμένο από τα γειτονικά του επίπεδα και χαλαρά (loosely) συνδεδεμένο με αυτά.

Πρέπει να τονιστεί ότι η αρχιτεκτονική πολλαπλών επιπέδων είναι μία λογική και όχι φυσική αρχιτεκτονική. Αυτό σημαίνει ότι σε έναν υπολογιστή μπορεί να τρέχουν περισσότερα από ένα επίπεδα ή ένα επίπεδο μπορεί να τρέχει σε περισσότερους από έναν υπολογιστές.

Τα επίπεδα που μπορεί κανείς να διακρίνει σε μία πολυεπίπεδη αρχιτεκτονική είναι:

4.1.1 Επίπεδο Πελάτη (Client Tier)

Παρέχει στον τελικό χρήση το μέσο για να αποκτήσει πρόσβαση στην εφαρμογή. Για μία διαδικτυακή εφαρμογή σαν τη δική μας είναι ένας web browser, θα μπορούσε όμως να είναι κάτι πιο εξεζητημένο, όπως ένα κινητό τηλέφωνο ή PDA.

4.1.2 Επίπεδο Παρουσίασης (Presentation Tier)

Αναλαμβάνει τη λειτουργικότητα που απαιτείται για την παρουσίαση της εφαρμογής στο χρήστη. Δέχεται τα αιτήματα του χρήστη από το επίπεδο πελάτη, ελέγχει την πρόσβαση στο επίπεδο λογικής, κατασκευάζει και επιστρέφει τις απαντήσεις προς τα αιτήματα αυτά.

4.1.3 Επίπεδο Εργασίας (Business Tier)

Στο επίπεδο αυτό πραγματοποιείται η επεξεργασία των αιτημάτων του χρήστη και παράγονται τα αποτελέσματα.

4.1.4 Επίπεδο Ολοκλήρωσης (Integration Tier)

Ελέγχει τη διασύνδεση με τη βάση δεδομένων, με legacy συστήματα καθώς και με εξωτερικές εφαρμογές.

4.1.5 Επίπεδο Πηγών (Resource Tier)

Περιέχει τα λειτουργικά δεδομένα (business data). Αποτελείται από βάσεις δεδομένων, legacy συστήματα, δια-εταιρικά συστήματα διασύνδεσης (B2B) κτλ.

4.2 Πρότυπα Σχεδίασης (Design Patterns)

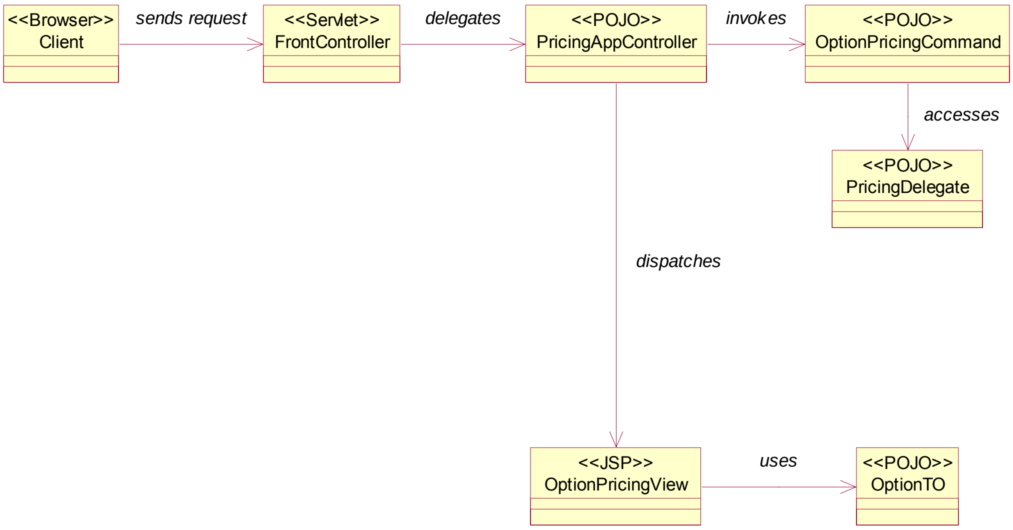
Τα πρότυπα σχεδίασης (design patterns) είναι επώνυμες και επακριβώς τεκμηριωμένες λύσεις σε προβλήματα που εμφανίζονται κατ' επανάληψη στη σχεδίαση προϊόντων λογισμικού.

Σημείο αναφοράς στην εξέλιξη των προτύπων σχεδίασης αποτέλεσε το βιβλίο "Design Patterns: Elements of Reusable Object-Oriented Software" [GHJV95] των E. Gamma, R. Helm, R. Johnson και J. Vlissides που εκδόθηκε το 1995. Μετά την έκδοση του βιβλίου αυτού τα πρότυπα σχεδίασης αποτέλεσαν κεντρική έννοια και κατέστησαν εξαιρετικά δημοφιλή μεταξύ των μηχανικών λογισμικού.

Η σχεδίαση της διαδικτυακής εφαρμογής αποτίμησης χρηματοοικονομικών παραγώγων που αποτελεί αντικείμενο της παρούσας διπλωματικής εργασίας βασίζεται σε μεγάλο βαθμό στα πρότυπα για την τεχνολογία J2EE που περιέχονται στο βιβλίο "Core J2EE Patterns: Best Practices and Design Strategies" [ACM03] των D. Alur, J. Crupi και D. Malks. Παρόλο που τα συγκεκριμένα πρότυπα έχουν προταθεί για την τεχνολογία J2EE, τα περισσότερα (είτε αυτούσια είτε με μικρές τροποποιήσεις) μπορούν να χρησιμοποιηθούν και για εναλλακτικές τεχνολογίες ανάπτυξης διαδικτυακού (web-based) λογισμικού, όπως η τεχνολογία .NET.

Θα κατηγοριοποιήσουμε τα πρότυπα που εφαρμόζουμε στη σχεδίασή μας με βάση το επίπεδο (tier) της αρχιτεκτονικής στο οποίο εμφανίζονται.

4.2.1 Επίπεδο Παρουσίασης (Presentation Tier)



Σχήμα 4.1: Πρότυπα Σχεδίασης Επιπέδου Παρουσίασης

4.2.1.1 Εμπρός Ελεγκτής (Front Controller)

Το πρότυπο Font Controller αποτελεί το αρχικό σημείο επικοινωνίας για το χειρισμό των αιτημάτων (requests) του χρήστη.

Σκοπός της χρήσης του είναι να μειωθούν οι εργασίες διαχείρισης (administration) και τοποθέτησης σε λειτουργία (deployment) της εφαρμογής. Η χρήση του επιτρέπει να εισάγουμε νέα λειτουργικότητα (νέα σενάρια χρήσης με νέες σελίδες) χωρίς να χρειαστεί να δηλώσουμε (register) τη νέα υπηρεσία στον εξυπηρετητή διαδικτύου (web server).

Η εφαρμογή μας χρησιμοποιεί έναν Front Controller για όλα τα αιτήματα του χρήστη. Αντιπροσωπεύεται από την κλάση FrontController και είναι servlet.

4.2.1.2 Αντικείμενο Συμφραζομένων (Context Object)

Το πρότυπο Context Object ενσωματώνει κατάσταση με τρόπο ανεξάρτητο από πρωτόκολλα με σκοπό να χρησιμοποιηθεί από διαφορετικά τμήματα της εφαρμογής.

Σκοπός της χρήσης του είναι να απομονώσει το πρωτόκολλο που χρησιμοποιείται για την αποστολή των αιτημάτων του χρήστη (που στην περίπτωσή μας είναι το HTTP πρωτόκολλο) από την υπόλοιπη εφαρμογή. Με τον τρόπο αυτό επιτυγχάνουμε μεγαλύτερη ευελιξία και προσαρμοστικότητα για την εφαρμογή μας. Είναι δυνατό να εισάγουμε ένα νέο πρωτόκολλο επικοινωνίας με το χρήστη (π.χ. WAP) χωρίς να επηρεαστεί το μεγαλύτερο τμήμα της εφαρμογής.

Χρησιμοποιούμε ένα Context Object για κάθε τύπο απαίτησης του χρήστη. Αντιπροσωπεύονται από κλάσεις όπως:

- FutureDefinitionRequestContext,
- FuturePricingRequestContext,
- OptionDefinitionRequestContext,
- OptionPricingRequestContext κτλ.

Για τη δημιουργία τους χρησιμοποιούμε ένα Εργοστάσιο (Factory).

4.2.1.3 Ελεγκτής Εφαρμογής (Application Controller)

Το πρότυπο Application Controller κεντρικοποιεί την ανάκτηση και επίκληση συστατικών της εφαρμογής σχετικών με την εξυπηρέτηση αιτημάτων, όπως Εντολών (Commands) και Όψεων (Views).

Επιτελεί δύο βασικούς ρόλους:

- Εντοπίζει και επικαλείται (invokes) ενέργειες για το χειρισμό αιτημάτων (action management)
- Κατευθύνει και αποστέλλει (dispatches) τα δεδομένα απόκρισης στην κατάλληλη όψη (view management)

Στη σχεδίασή μας χρησιμοποιούμε το interface ApplicationController και τις κλάσεις ManagementAppController και PricingAppController που υλοποιούν (implement) το interface. Για τη δημιουργία τους χρησιμοποιούμε ένα Εργοστάσιο (Factory).

4.2.1.4 Βοηθός Όψης (View Helper)

Το πρότυπο View Helper χρησιμοποιεί όψεις (views) για να ενσωματώσει τον κώδικα που μορφοποιεί την απόκριση στο αίτημα του χρήστη και βοηθούς (helpers) για να ενσωματώσει τη λογική που θα χρειαστεί η όψη για να εμφανίσει τα αποτελέσματα.

Στην εφαρμογή μας χρησιμοποιούμε ως όψεις ένα πλήθος σελίδων JSP. Ως βοηθούς χρησιμοποιούμε τους Αντιπροσώπους (Delegates), για τους οποίους θα αναφέρουμε περισσότερα στο επίπεδο εργασίας (business tier).

4.2.1.5 Εντολή (Command)

Το πρότυπο Command ενσωματώνει την ενέργεια που απαιτείται ως αποτέλεσμα ενός αιτήματος. Αυτό επιτυγχάνεται με την επίκληση (invocation) της αντίστοιχης λειτουργικότητας.

Στη σχεδίασή μας χρησιμοποιούμε μία Εντολή για κάθε τύπο απαίτησης του χρήστη. Αντιπροσωπεύονται από κλάσεις όπως:

- FutureDefinitionCommand,
- FuturePricingCommand,
- OptionDefinitionCommand,
- OptionPricingCommand κτλ.

Υπάρχει επομένως μία προς μία αντιστοίχηση ανάμεσα σε Context Objects και Commands. Για τη δημιουργία τους χρησιμοποιούμε ένα Εργοστάσιο (Factory).

4.2.1.6 Εργοστάσιο (Factory)

Το πρότυπο Factory είναι υπεύθυνο για τη δημιουργία αντικειμένων που υλοποιούν μία διεπαφή (interface) ή κληρονομούν μία αφηρημένη κλάση (abstract class).

Ο έλεγχος της δημιουργίας και η προσθήκη νέων τύπων αντικειμένων γίνεται με δηλωτικό τρόπο, με χρήση αρχείων XML. Περισσότερα θα αναφέρουμε στο επόμενο κεφάλαιο.

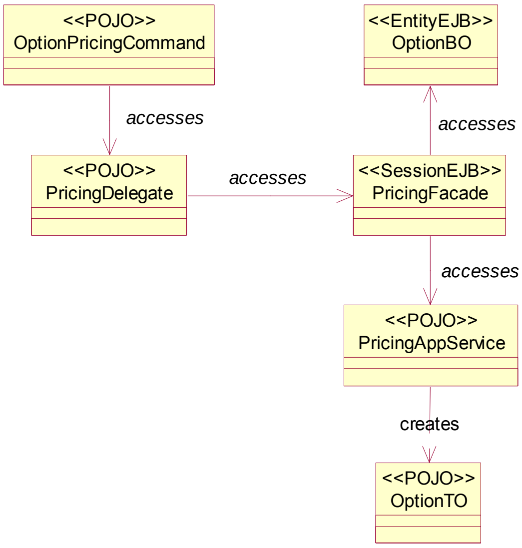
Στην εφαρμογή μας χρησιμοποιούμε ένα πλήθος κλάσεων που υλοποιούν αυτό το πρότυπο, όπως RequestContextFactory, ApplicationControllerFactory, CommandFactory κτλ.

4.2.1.7 Μοναδιαίο (Singleton)

Το πρότυπο Singleton ορίζει κλάσεις που μπορεί να έχουν μία μόνο υπόσταση (instance) σε μία εφαρμογή.

Οι κλάσεις της εφαρμογής μας που ακολουθούν το πρότυπο Factory, ακολουθούν και το πρότυπο Singleton.

4.2.2 Επίπεδο Εργασίας (Business Tier)



Σχήμα 4.2: Πρότυπα Σχεδίασης Επιπέδου Εργασίας

4.2.2.1 Απεσταλμένος Εργασίας (Business Delegate)

Το πρότυπο Business Delegate ενσωματώνει την πρόσβαση σε μία υπηρεσία εργασίας (business service).

Σκοπός της χρήσης του είναι να κρύψει της λεπτομέρειες υλοποίησης των υπηρεσιών εργασίας. Με τον τρόπο αυτό ελαττώνεται η αλληλεξάρτηση ανάμεσα στα συστατικά των επιπέδων παρουσίασης και εργασίας.

Παρόλο που ένας Business Delegate τρέχει στο επίπεδο παρουσίασης, το κατατάσσουμε στο επίπεδο εργασίας γιατί είναι μία λογική προέκταση του επιπέδου αυτού.

Στη σχεδίασή μας χρησιμοποιούμε ένα Business Delegate για κάθε Session Façade, στα οποία θα αναφερθούμε στη συνέχεια. Οι Αντιπρόσωποι Εργασίας αντιπροσωπεύονται από τις κλάσεις ManagementDelegate, PricingDelegate και NotificationDelegate.

4.2.2.2 Εντοπιστής Υπηρεσίας (Service Locator)

Το πρότυπο Service Locator ενσωματώνει την αναζήτηση υπηρεσιών (services) και συστατικών (components).

Σκοπός της χρήσης του είναι να κρύψει τις λεπτομέρειες υλοποίησης του μηχανισμού αναζήτησης και να αυξήσει την απόδοση αποθηκεύοντας προσωρινά (cache) τα αποτελέσματα προηγούμενων αναζητήσεων.

Στην εφαρμογή μας χρησιμοποιούμε έναν Εντοπιστή Υπηρεσίας που αντιπροσωπεύεται από την κλάση ServiceLocator. Η κλάση αυτή ακολουθεί και το πρότυπο Singleton.

4.2.2.3 Πρόσοψη Συνεδρίασης (Session Façade)

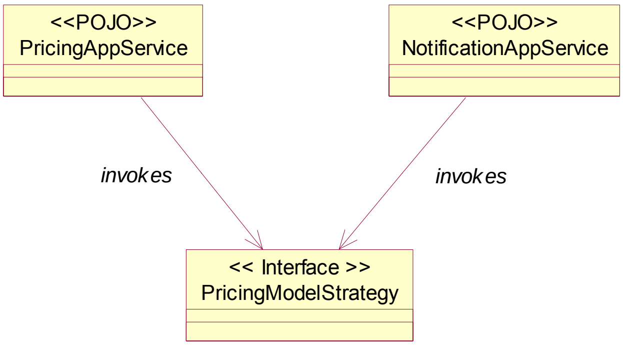
Το πρότυπο Session Façade ενσωματώνει συστατικά του επιπέδου εργασίας και αποκαλύπτει (exposes) μία «χονδρο-κοσκινισμένη» (coarse-grained) υπηρεσία προς απόμακρους πελάτες.

Μία «χονδρο-κοσκινισμένη» (coarse-grained) υπηρεσία συγκεντρώνει και επιστρέφει έναν όγκο πληροφορίας για τον οποίο μπορεί να απαιτούνται περισσότερα ενδιάμεσα βήματα, σε αντίθεση με μία «ψιλο-κοσκινισμένη» (fine-grained) υπηρεσία που επιστρέφει τα αποτελέσματα μίας απλής διεργασίας.

Σκοπός της χρήσης του προτύπου αυτού είναι να ελαττώσει τον αριθμό των απομακρυσμένων επικλήσεων μεθόδων (remote method invocations) ανάμεσα σε συστατικά του επιπέδου παρουσίασης και του επιπέδου εργασίας, μειώνοντας τη χρήση του δικτύου και αυξάνοντας την απόδοση.

Οι κλάσεις ManagementFacade, PricingFacade και NotificationFacade της σχεδίασής μας ακολουθούν το πρότυπο Session Façade.

4.2.2.4 Υπηρεσία Εφαρμογής (Application Service)



Σχήμα 4.3: Διαστρωμάτωση Υπηρεσιών Εφαρμογής

Το πρότυπο Application Service αποτελεί το συστατικό για την ενσωμάτωση της λογικής εργασίας (business logic).

Σκοπός του είναι να ενσωματώσει την υψηλότερου επιπέδου λογική εργασίας σε ένα ξεχωριστό συστατικό που χρησιμοποιεί τα υποκείμενα Αντικείμενα Εργασίας (Business Objects), για τα οποία θα μιλήσουμε στη συνέχεια, και υπηρεσίες. Με τον τρόπο αυτό μειώνεται η αλληλεπίδραση ανάμεσα (low coupling) ανάμεσα στα Αντικείμενα Εργασίας.

Η σχεδίασή μας χρησιμοποιεί τη στρατηγική στρωμάτων (layer strategy) [ACM03] σε ό,τι αφορά τη χρήση των Application Services. Στο υψηλότερου επιπέδου στρώμα βρίσκονται οι κλάσεις PricingAppService και NotificationAppService. Στο χαμηλότερου επιπέδου στρώμα βρίσκονται η διεπαφή PricingModelStrategy και οι κλάσεις BlackScholesAppService, BinomialTreeAppService και MonteCarloAppService που την υλοποιούν. Θα αναφέρουμε περισσότερα για τις κλάσεις του στρώματος αυτού στην ενότητα για το πρότυπο Στρατηγική (Strategy).

Η μεταβλητότητα (volatility) των χρηματοοικονομικών προϊόντων υπολογίζεται σε καθημερινή βάση από την κλάση VolatilityAppService.

4.2.2.5 Αντικείμενο Εργασίας (Business Object)

Το πρότυπο Business Object ενσωματώνει και διαχειρίζεται δεδομένα εργασίας (business data), συμπεριφορά και αποθήκευση (persistence).

Η χρήση Business Objects σε συνδυασμό με τη χρήση Application Services βοηθά στο διαχωρισμό της λογικής αποθήκευσης (persistence logic) από τη λογική εργασίας (business logic). Με τον τρόπο αυτό επιτυγχάνουμε τη δημιουργία αντικειμένων με υψηλή συνάφεια (high cohesion).

Στη εφαρμογή μας εφαρμόζουμε τη στρατηγική της χρήσης entity beans για το μηχανισμό αποθήκευσης. Θα ασχοληθούμε με το μηχανισμό αυτό εκτενέστερα στο επόμενο κεφάλαιο.

Η σχεδίασή μας περιλαμβάνει τα ακόλουθα Business Objects:

- FinancialInstrumentBO
- StockBO
- IndexBO
- CurrencyBO
- DerivativeBO
- FutureBO
- OptionBO
- EuropeanOptionBO
- AmericanOptionBO
- PortfolioBO

4.2.2.6 Σύνθετη Οντότητα (Composite Entity)

Το πρότυπο Composite Entity συγκεντρώνει ένα σύνολο από συσχετιζόμενα Business Objects σε ένα «χονδρο-κοσκινισμένο» (coarse-grained) entity bean.

Η χρήση του επιτρέπει την υλοποίηση αντικειμένων-κηδεμόνων (parent objects) που διαχειρίζονται εξαρτημένα αντικείμενα (dependent objects). Ο κύκλος ζωής ενός εξαρτημένου αντικειμένου είναι στενά συνδεδεμένος με τον κύκλο ζωής του αντικειμένου-κηδεμόνα και διαχειρίζεται από αυτό.

Στη σχεδίασή μας ορίζουμε την κλάση PortfolioBO που αντιπροσωπεύει ένα χαρτοφυλάκιο ως αντικείμενο-κηδεμόνα και την κλάση PortfolioInstrument που αντιπροσωπεύει τα περιεχόμενα του χαρτοφυλακίου ως εξαρτημένο αντικείμενο.

4.2.2.7 Αντικείμενο Μεταφοράς (Transfer Object)

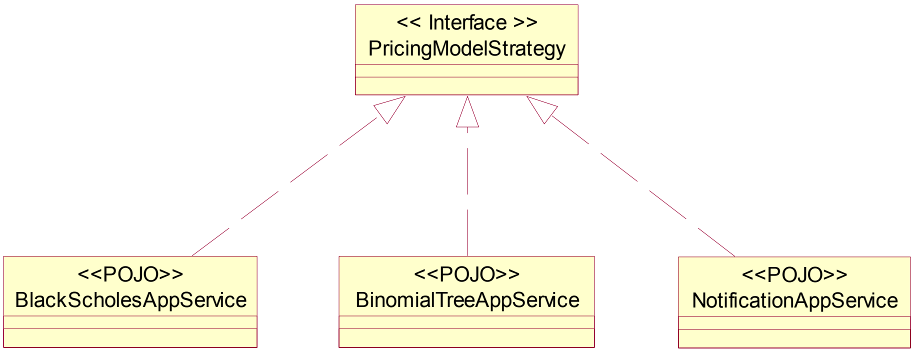
Το πρότυπο Transfer Object μεταφέρει πολλαπλά δεδομένα μεταξύ των επιπέδων (tiers) μίας εφαρμογής. Στη βιβλιογραφία συχνά αναφέρεται και ως Data Transfer Object.

Σκοπός της χρήσης του είναι να βελτιστοποιήσει τη μεταφορά δεδομένων μεταξύ των επιπέδων. Αντί να αποστέλλονται ή να παραλαμβάνονται ξεχωριστές τιμές, τα δεδομένα συγκεντρώνονται σε μία δομή που απαιτείται για την εντολή ή την απάντηση.

Στη σχεδίασή μας χρησιμοποιούμε τη στρατηγική πολλαπλών αντικειμένων μεταφοράς (multiple transfer objects strategy) [ACM03]. Ορίζουμε ένα Transfer Object για κάθε Business Object της εφαρμογής:

- FinancialInstrumentTO
- StockTO
- IndexTO
- CurrencyTO κτλ.

4.2.2.8 Στρατηγική (Strategy)



Σχήμα 4.4: Στρατηγική (Strategy)

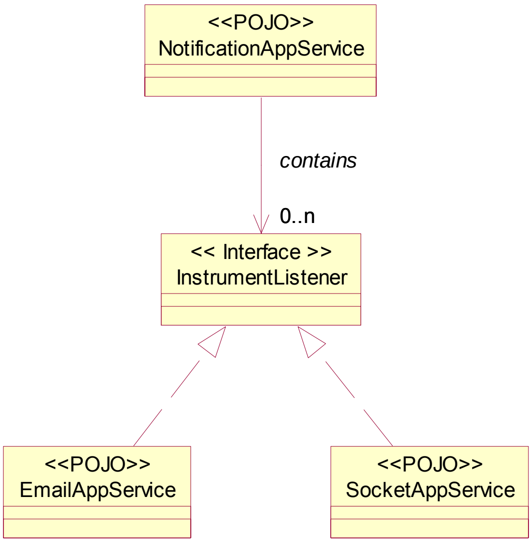
Το πρότυπο σχεδίασης Strategy ενσωματώνει μία οικογένεια αλγορίθμων και τους κάνει εναλλάξιμους.

Με τον τρόπο αυτό απομονώνουμε τους αλγορίθμους από τα αντικείμενα που τους χρησιμοποιούν, επιτυγχάνοντας υψηλή συνάφεια (high cohesion) για τα αντικείμενα αυτά.

Στην περίπτωση της εφαρμογής μας, οι αλγόριθμοι είναι τα μοντέλα αποτίμησης χρηματοοικονομικών παραγώγων και τα αντικείμενα είναι οι κλάσεις των διαφόρων παραγώγων. Με τον τρόπο αυτό μπορούμε να εισάγουμε νέα μοντέλα αποτίμησης με ελάχιστες τροποποιήσεις.

Η διεπαφή PricingModelStrategy και οι κλάσεις BlackScholesAppService, BinomialTreeAppService και MonteCarloAppService ακολουθούν το πρότυπο αυτό.

4.2.2.9 Παρατηρητής (Observer)



Σχήμα 4.5: Παρατηρητής (Observer)

Το πρότυπο Observer ορίζει μία ένα-προς-πολλά εξάρτηση ανάμεσα σε ένα παρατηρούμενο αντικείμενο (observable ή publisher) και ένα ή περισσότερα αντικείμενα-παρατηρητές (observers ή subscribers). Όταν το παρατηρούμενο αντικείμενο αλλάζει κατάσταση, όλα τα αντικείμενα-παρατηρητές ειδοποιούνται αυτόματα.

Εφαρμόζουμε το πρότυπο Observer για μία πολύ σημαντική λειτουργία της εφαρμογής μας. Το σύστημά μας στέλνει ειδοποιήσεις όταν η κατάσταση ενός παραγώγου ή χαρτοφυλακίου ανταποκρίνεται σε κάποιους προκαθορισμένους κανόνες. Για παράδειγμα, όταν η διαφορά μεταξύ της τρέχουσας και της θεωρητικής τιμής ενός παραγώγου γίνει αρκετά μεγάλη ή όταν η τιμή του δέλτα ενός χαρτοφυλακίου ως προς έναν από τους υποκείμενους τίτλους του ξεπεράσει μία τιμή. Στην περίπτωση αυτή είναι πρωταρχικής σημασίας για το χρήστη να ειδοποιηθεί έγκαιρα, ώστε να εκμεταλλευτεί την ευκαιρία εξισορροπητικής κερδοσκοπίας (arbitrage) ή να φροντίσει για την αναδιάρθρωση του χαρτοφυλακίου αντίστοιχα.

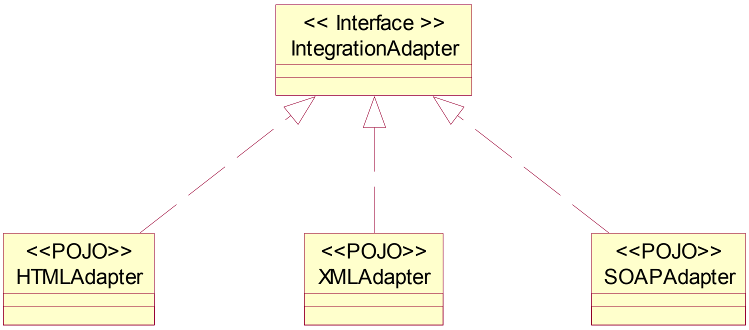
Στη σχεδίασή μας για λόγους απλότητας ορίζουμε ως παρατηρούμενο αντικείμενο την κλάση NotificationAppService και όχι κάθε Business Object ξεχωριστά. Η κλάση NotificationAppService θα ενεργοποιείται σε τακτά χρονικά διαστήματα (π.χ. κάθε 10 λεπτά) από κάποιον χρονομετρητή (timer) και θα ελέγχει την κατάσταση παραγώγων και χαρτοφυλακίων, αποστέλλοντας ειδοποιήσεις προς τα αντικείμενα-παρατηρητές.

Τα αντικείμενα-παρατηρητές υλοποιούν τη διεπαφή InstrumentListener. Στην εφαρμογή μας έχουμε ορίσει ως observers τις κλάσεις EmailAppService και SocketAppService που αποστέλλουν ειδοποιήσεις μέσω e-mail και socket αντίστοιχα. Μπορούν να προστεθούν νέοι τρόποι ειδοποίησης (π.χ. μέσω SMS) με ελάχιστες τροποποιήσεις.

4.2.3 Επίπεδο Ολοκλήρωσης (Integration Tier)

Σχήμα 4.6: Πρότυπο Σχεδίασης Επιπέδου Ολοκλήρωσης

4.2.3.1 Προσαρμογέας Ολοκλήρωσης (Integration Adapter)



Σχήμα 4.7: Προσαρμογέας Ολοκλήρωσης (Integration Adapter)

Το πρότυπο Προσαρμογέας (Adapter) μετατρέπει τη διεπαφή (interface) μίας κλάσης ή συστήματος σε μία άλλη διεπαφή που ένα αντικείμενο-πελάτης μπορεί να χρησιμοποιήσει.

Το πρότυπο Integration Adapter είναι μία ειδική περίπτωση Adapter που αναφέρεται στην ολοκλήρωση με τρίτα συστήματα. Το κάθε ένα από τα συστήματα αυτά επιτελεί παρόμοια λειτουργία αλλά έχει διαφορετική διεπαφή. Το πρότυπο αυτό μετατρέπει κάθε ξεχωριστή διεπαφή σε μία προκαθορισμένη διεπαφή, ώστε ένα αντικείμενο-πελάτης να μπορεί να χρησιμοποιήσει οποιοδήποτε από τα συστήματα στα οποία οι διεπαφές αντιστοιχούν.

Υπάρχει ένας μεγάλος αριθμός από παρόχους πληροφορίας σχετικά με χρηματοοικονομικά προϊόντα, όμως ο καθένας παρέχει το δική του διεπαφή. Η χρήση του IntegrationAdapter επιτρέπει στην εφαρμογή μας να χρησιμοποιεί πληροφορία προερχόμενη από διαφορετικές πηγές και μέσω διαφορετικών πρωτοκόλλων, π.χ. HTML, XML, SOAP κτλ.

5

Υλοποίηση

Στο κεφάλαιο αυτό περιγράφονται τα χαρακτηριστικά της υλοποίησης της διαδικτυακής εφαρμογής αποτίμησης χρηματοοικονομικών παραγώγων που αποτελεί αντικείμενο της παρούσας διπλωματικής εργασίας.

Αρχικά θα αναφερθούμε στις τεχνολογίες, τις πλατφόρμες και τα προγραμματιστικά εργαλεία που επιλέξαμε για την υλοποίηση. Στη συνέχεια θα περιγράψουμε λεπτομέρειες της υλοποίησης που είτε παρουσιάζουν κάποια καινοτομία είτε είναι άξιες ενδιαφέροντος.

5.1 Τεχνολογίες και Προγραμματιστικά Εργαλεία

5.1.1 Γλώσσα Προγραμματισμού Java

Η γλώσσα προγραμματισμού που επιλέξαμε για να αναπτύξουμε τον κώδικα της εφαρμογής είναι η Java. Ο λόγος είναι ότι χρειαζόμαστε μία γλώσσα κατάλληλη για την ανάπτυξη διαδικτυακών εφαρμογών και η Java είναι η πιο ευρέως διαδεδομένη γλώσσα για το σκοπό αυτό. Επιπλέον υποστηρίζεται από μία πολύ δυνατή τεχνολογία, την J2EE, στην οποία θα αναφερθούμε εκτενέστερα στη συνέχεια.

Επιλέξαμε να κάνουμε ακόμα και τη συγγραφή των αλγορίθμων των μοντέλων αποτίμησης χρηματοοικονομικών παραγώγων σε Java, παρόλο που οι αλγόριθμοι που μπορεί κανείς να βρει στη βιβλιογραφία είναι συνήθως υλοποιημένοι σε C++ και παρόλο που μπορεί κάποιοι να ισχυριστούν ότι η C++ είναι πιο γρήγορη γλώσσα στην εκτέλεσή της. Ο λόγος είναι ότι η συμπαγής αρχιτεκτονική σχεδίαση που βασίζεται στην τεχνολογία J2EE δίνει τη δυνατότητα να τοποθετήσουμε συστατικά που απαιτούν μεγάλη υπολογιστική δύναμη σε έναν ξεχωριστό ισχυρό εξυπηρετητή ή και σε περισσότερους εξυπηρετητές. Περισσότερα θα αναφέρουμε στην αμέσως επόμενη ενότητα.

5.1.2 Τεχνολογία Ανάπτυξης Διαδικτυακών Εφαρμογών J2EE

Η τεχνολογία που επιλέξαμε για την ανάπτυξη της διαδικτυακής εφαρμογής αποτίμησης χρηματοοικονομικών παραγώγων είναι η Java 2 Enterprise Edition (J2EE 1.4). Ο λόγος είναι ότι πρόκειται για την πιο διαδεδομένη τεχνολογία ανάπτυξης διαδικτυακών (web-based) εφαρμογών.

Η συγκεκριμένη τεχνολογία υποστηρίζεται από δυνατές ομάδες εργασίας, από πλήθος προγραμματιστικών εργαλείων και εξυπηρετητών εφαρμογών και – το σημαντικότερο ίσως – από εκτεταμένη βιβλιογραφία, ιδιαίτερα σε ό,τι αφορά τη χρήση προτύπων σχεδίασης (design patterns).

Η τεχνολογία J2EE υιοθετεί την ανάπτυξη αρχιτεκτονικών πολλαπλών επιπέδων (multi-tiered). Αυτό αποτελεί και το σημαντικότερο χαρακτηριστικό της.

Όπως αναφέραμε και στο κεφάλαιο της σχεδίασης, η αρχιτεκτονική πολλαπλών επιπέδων είναι μία λογική αρχιτεκτονική. Μπορεί σε έναν υπολογιστή να είναι τοποθετημένα περισσότερα από ένα επίπεδα ή ένα επίπεδο να μοιράζεται σε περισσότερους από έναν υπολογιστές. Αυτό επιτρέπει ευελιξία σε ό,τι αφορά τον αρχική φυσική αρχιτεκτονική, αλλά και δυνατότητα εύκολου ανασχεδιασμού (refactoring). Για παράδειγμα, αν αποδειχθεί ότι ένα συστατικό της εφαρμογής είναι εξαιρετικά απαιτητικό σε υπολογιστική ισχύ (π.χ. οι αλγόριθμοι για την προσομοίωση Monte Carlo), μπορεί το συστατικό αυτό να μεταφερθεί σε έναν ισχυρότερο υπολογιστή ή και σε περισσότερους χωρίς να αλλάξει η λογική αρχιτεκτονική.

5.1.3 Εξυπηρετητής Εφαρμογών JBoss

Για την τοποθέτηση σε λειτουργία (deployment) της διαδικτυακής εφαρμογής αποτίμησης χρηματοοικονομικών παραγώγων επιλέξαμε τον εξυπηρετητή διαδικτυακών εφαρμογών (application server) JBoss.

Ο JBoss είναι εξυπηρετητής εφαρμογών (application server) συμβατός με την τεχνολογία J2EE. Είναι λογισμικό ανοιχτής πηγής (open source) και έχει καταστεί δημοφιλής όχι μόνο για την ανάπτυξη ερευνητικών εφαρμογών αλλά και για την ανάπτυξη πλήθους εμπορικών εφαρμογών.

5.1.4 Βάση Δεδομένων MySQL

Η βάση δεδομένων που χρησιμοποιούμε για την αποθήκευση των δεδομένων της εφαρμογής αποτίμησης χρηματοοικονομικών παραγώγων είναι η ανοιχτής πηγής (open source) βάση δεδομένων MySQL.

Ο κυριότερος λόγος που επιλέξαμε τη MySQL είναι η ευκολία με την οποία μπορεί να συνδεθεί με τον εξυπηρετητή εφαρμογών JBoss.

5.2 Λεπτομέρειες Υλοποίησης

5.2.1 Επίπεδο Παρουσίασης (Presentation Tier)

5.2.1.1 Servlets

Η κλάση FrontController που υιοθετεί το πρότυπο Front Controller υλοποιείται ως servlet. Είναι το μοναδικό συστατικό της εφαρμογής που δηλώνεται στον περιγραφέα τοποθέτησης (deployment descriptor), το αρχείο web.xml.

5.2.1.2 JavaServer Pages (JSPs)

Για τη δημιουργία των όψεων (views), τη χρήση των οποίων αναλύσαμε στο κεφάλαιο της σχεδίασης, χρησιμοποιούμε την τεχνολογία JavaServer Pages (JSP 2.0) που εμπεριέχεται στην τεχνολογία J2EE.

Για να αποφύγουμε τη χρήση κώδικα γραμμένου σε Java μέσα στις σελίδες JSP, χρησιμοποιούμε τη βιβλιοθήκη προσαρμόσιμων ετικετών (custom tags) JSP Standard Tag Library (JSTL 1.1).

5.2.1.3 Σύνδεση Εντολών και Όψεων

Στο κεφάλαιο της σχεδίασης ορίσαμε ένα πλήθος κλάσεων που υιοθετούν πρότυπα σχεδίασης, δεν ορίσαμε όμως με ποιο τρόπο οι κλάσεις αυτές συνδέονται μεταξύ τους. Οι εντολές που προκύπτουν από τα αιτήματα των χρηστών και οι όψεις που αντιστοιχούν στις σελίδες που οι χρήστες παίρνουν ως απάντηση, όπως και η πληροφορία που αυτές οι σελίδες περιέχουν, θα πρέπει να συνδέονται με κάποιο δηλωτικό τρόπο.

Το πλαίσιο εργασίας (framework) Struts θα μπορούσε να χρησιμοποιηθεί για το σκοπό αυτό. Επιλέξαμε ωστόσο να μη χρησιμοποιήσουμε το Struts, γιατί θέλαμε να έχουμε όσο το δυνατό μεγαλύτερη ευελιξία στη σχεδίασή μας, καθώς το Struts υιοθετεί τα δικά του πρότυπα σχεδίασης.

Για το λόγο αυτό επιλέξαμε μία εξατομικευμένη (custom) λύση. Χρησιμοποιούμε αρχεία XML για να εξομοιώσουμε κατά κάποιο τρόπο τη λειτουργικότητα του Struts. Με τον τρόπο αυτό συνδέουμε τα συμφραζόμενα απαιτήσεων (request context), τους ελεγκτές εφαρμογής (application controllers), τις εντολές (commands), τις όψεις (views) και την πληροφορία που αυτές εμφανίζουν μέσω των αντικειμένων μεταφοράς (transfer objects).

Η πληροφορία που περιέχεται στα αρχεία αυτά διαβάζεται σε χάρτες (maps) μέσω της βοηθητικής κλάσης MapLoader. Τα εργοστάσια (factories) χρησιμοποιούν αυτούς τους χάρτες για να δημιουργήσουν τα κατάλληλα αντικείμενα που θα χειριστούν το αίτημα.

Ενδεικτικά, η πληροφορία που ακολουθεί είναι τμήμα του αρχείου CommandMap.xml:

```
<command-map>
  <command-class>
    <command>select_stock</command>
    <type>com.derimon.command.StockQuoteCommand</type>
    <logical-view>stock_quote</logical-view>
  </command-class>
  <command-class>
    <command>quote_stock</command>
    <type>com.derimon.command.StockQuoteCommand</type>
    <logical-view>stock_quote_results</logical-view>
  </command-class>
  <command-class>
    <command>select_index</command>
    <type>com.derimon.command.IndexQuoteCommand</type>
    <logical-view>index_quote</logical-view>
  </command-class>
  <command-class>
    <command>quote_index</command>
    <type>com.derimon.command.IndexQuoteCommand</type>
    <logical-view>index_quote_results</logical-view>
  </command-class>
  <command-class>
    <command>select_future</command>
    <type>com.derimon.command.FuturePricingCommand</type>
    <logical-view>future_pricing</logical-view>
  </command-class>
  <command-class>
    <command>price_future</command>
    <type>com.derimon.command.FuturePricingCommand</type>
    <logical-view>future_pricing_results</logical-view>
  </command-class>
  <command-class>
    <command>select_option</command>
    <type>com.derimon.command.OptionPricingCommand</type>
    <logical-view>option_pricing</logical-view>
  </command-class>
  <command-class>
    <command>define_option_pricing</command>
    <type>com.derimon.command.OptionPricingCommand</type>
    <logical-view>option_pricing_process</logical-view>
  </command-class>
  <command-class>
    <command>price_option</command>
    <type>com.derimon.command.OptionPricingCommand</type>
    <logical-view>option_pricing_results</logical-view>
  </command-class>
  <command-class>
    <command>select_portfolio</command>
    <type>com.derimon.command.PortfolioPricingCommand</type>
    <logical-view>portfolio_pricing</logical-view>
  </command-class>
  <command-class>
    <command>define_portfolio_pricing</command>
    <type>com.derimon.command.PortfolioPricingCommand</type>
    <logical-view>portfolio_pricing_process</logical-view>
  </command-class>
  <command-class>
```

```
    <command>price_portfolio</command>
    <type>com.derimon.command.PortfolioPricingCommand</type>
    <logical-view>portfolio_pricing_results</logical-view>
  </command-class>
</command-map>
```

Η πληροφορία που ακολουθεί είναι τμήμα του αρχείου ViewMap.xml:

```
<view-map>
  <view-file>
    <logical-view>stock_quote</logical-view>
    <location>/StockQuote.jsp</location>
    <view-attribute>stock_list</view-attribute>
  </view-file>
  <view-file>
    <logical-view>stock_quote_results</logical-view>
    <location>/StockQuoteResults.jsp</location>
    <view-attribute>stock</view-attribute>
  </view-file>
  <view-file>
    <logical-view>index_quote</logical-view>
    <location>/IndexQuote.jsp</location>
    <view-attribute>index_list</view-attribute>
  </view-file>
  <view-file>
    <logical-view>index_quote_results</logical-view>
    <location>/IndexQuoteResults.jsp</location>
    <view-attribute>index</view-attribute>
  </view-file>
  <view-file>
    <logical-view>future_pricing</logical-view>
    <location>/FuturePricing.jsp</location>
    <view-attribute>future_list</view-attribute>
  </view-file>
  <view-file>
    <logical-view>future_pricing_results</logical-view>
    <location>/FuturePricingResults.jsp</location>
    <view-attribute>future</view-attribute>
  </view-file>
  <view-file>
    <logical-view>option_pricing</logical-view>
    <location>/OptionPricing.jsp</location>
    <view-attribute>option_list</view-attribute>
  </view-file>
  <view-file>
    <logical-view>option_pricing_process</logical-view>
    <location>/OptionPricingProcess.jsp</location>
    <view-attribute>option</view-attribute>
  </view-file>
  <view-file>
    <logical-view>option_pricing_results</logical-view>
    <location>/OptionPricingResults.jsp</location>
    <view-attribute>option</view-attribute>
  </view-file>
  <view-file>
    <logical-view>portfolio_pricing</logical-view>
    <location>/PortfolioPricing.jsp</location>
    <view-attribute>portfolio_map</view-attribute>
  </view-file>
  <view-file>
    <logical-view>portfolio_pricing_process</logical-view>
    <location>/PortfolioPricingProcess.jsp</location>
    <view-attribute>portfolio_map</view-attribute>
  </view-file>
  <view-file>
    <logical-view>portfolio_pricing_results</logical-view>
    <location>/PortfolioPricingResults.jsp</location>
    <view-attribute>portfolio</view-attribute>
  </view-file>
</view-map>
```

5.2.2 Επίπεδο Λογικής (Business Tier)

Για την υλοποίηση των συστατικών του επιπέδου λογικής για τα οποία απαιτείται αυξημένη λειτουργικότητα, όπως εντοπισμός, συναλλαγές, αποθήκευση κτλ. χρησιμοποιούμε την τεχνολογία Enterprise Java Beans (EJB 3.0) που εμπεριέχεται στην τεχνολογία J2EE.

5.2.2.1 Stateless Session Beans

Οι κλάσεις που υιοθετούν το πρότυπο Session Façade υλοποιούνται ως Stateless Session Beans. Για τη διαχείριση συναλλαγών (transaction management) χρησιμοποιούμε Container Managed Transactions (CMT).

5.2.2.2 Entity Beans

Οι κλάσεις που υιοθετούν το πρότυπο Business Object υλοποιούνται ως Entity Beans. Οι κλάσεις αυτές δεν διαχειρίζονται οι ίδιες την αποθήκευσή τους (persistence) στη βάση δεδομένων αλλά αναθέτουμε αυτή τη λειτουργία στον υποδοχέα (container), δηλαδή τον εξυπηρετητή εφαρμογών JBoss, καθώς χρησιμοποιούμε Container Managed Persistence (CMP).

Στα κεφάλαια της ανάλυσης και της σχεδίασης είδαμε ότι οι κλάσεις που υλοποιούν τα χρηματοοικονομικά παράγωγα έχουν σχέσεις κληρονομικότητας μεταξύ τους. Ένα πρόβλημα που έπρεπε να ξεπεράσουμε έχει να κάνει με το γεγονός ότι η τεχνολογία EJB δεν υποστηρίζει εγγενώς (inherently) την κληρονομικότητα (inheritance) ανάμεσα σε entity beans.

Για να το επιτύχουμε αυτό είχαμε τις εξής επιλογές: Είτε να υλοποιήσουμε τα Business Objects σαν Plain Old Java Objects (POJOs) και να χρησιμοποιήσουμε το πρότυπο σχεδίασης Data Access Object (DAO) για την αποθήκευσή τους (persistence) είτε να αναζητήσουμε κάποιο διαφορετικό μονοπάτι (workaround) σε σχέση με τη συνήθη χρήση τους.

Ο E. Proulx σε μία σειρά άρθρων με τίτλο EJB Inheritance [Proulx02] προτείνει μία τεχνική για τη χρήση κληρονομικότητας σε Entity Beans. Το κλειδί για να το επιτύχει αυτό είναι ο ορισμός δύο νέων πολυμορφικών μεθόδων αναζήτησης. Οι μέθοδοι αυτές, τις οποίες ονομάζει locate() και locateAll(), λειτουργούν όπως οι findByPrimaryKey() και findAll() αντίστοιχα, αλλά με πολυμορφικό τρόπο. Όταν χρησιμοποιηθεί η μέθοδος locate() ενός entity bean και της δοθεί κάποιο κλειδί που αντιστοιχεί σε entity bean απόγονο αυτού, το αντικείμενο που θα επιστρέψει θα υλοποιεί το local ή remote interface του απογόνου. Το ίδιο ισχύει και για τα αντικείμενα που επιστρέφει η μέθοδος locateAll().

Για παράδειγμα, αν χρησιμοποιήσουμε τη μέθοδο findByPrimaryKey() της κλάσης DerivativeBO με παράμετρο τον κωδικό ενός δικαιώματος προαίρεσης, το αντικείμενο που θα επιστραφεί θα υλοποιεί τη διεπαφή DerivativeBOLocal. Αυτό όμως μας εμποδίζει να χρησιμοποιήσουμε μεθόδους που έχουν οριστεί ειδικά για δικαιώματα προαίρεσης στην κλάση OptionBO, η οποία κληρονομεί τη DerivativeBO. Αν όμως χρησιμοποιήσουμε τη μέθοδο locate() της κλάσης DerivativeBO με παράμετρο τον κωδικό του ίδιου δικαιώματος προαίρεσης, το αντικείμενο που θα επιστραφεί θα υλοποιεί τη διεπαφή OptionBOLocal, επιτρέποντάς μας να χρησιμοποιήσουμε τις εξειδικευμένες μεθόδους για δικαιώματα προαίρεσης.

Ένα μειονέκτημα της τεχνικής του Proulx είναι ότι η υλοποίηση των δύο αυτών μεθόδων στον πρόγονο πρέπει τροποποιείται κάθε φορά που προστίθεται ένας νέος απόγονος, κάτι που μειώνει την πρακτικότητά της. Για το λόγο επεκτείνουμε τη λύση του Proulx ώστε ο πρόγονος να μη χρειάζεται να τροποποιηθεί στο ελάχιστο. Αυτό είναι δυνατό να επιτευχθεί με δηλωτικό τρόπο.

Χρησιμοποιούμε ένα XML αρχείο στο οποίο δηλώνουμε τις σχέσεις κληρονομικότητας ανάμεσα στα entity beans:

```
<inheritance-map>
  <business-object>
    <name>local/FinancialInstrumentBO</name>
    <class>com.derimon.business.FinancialInstrumentBOLocalHome</class>
  </business-object>
  <business-object>
    <name>local/DerivativeBO</name>
    <class>com.derimon.business.DerivativeBOLocalHome</class>
  </business-object>
  <business-object>
    <name>local/StockBO</name>
    <class>com.derimon.business.StockBOLocalHome</class>
  </business-object>
  <business-object>
    <name>local/IndexBO</name>
    <class>com.derimon.business.IndexBOLocalHome</class>
  </business-object>
  <business-object>
    <name>local/CurrencyBO</name>
    <class>com.derimon.business.CurrencyBOLocalHome</class>
  </business-object>
  <business-object>
    <name>local/FutureBO</name>
    <class>com.derimon.business.FutureBOLocalHome</class>
  </business-object>
  <business-object>
    <name>local/OptionBO</name>
    <class>com.derimon.business.OptionBOLocalHome</class>
  </business-object>
  <business-object>
    <name>local/EuropeanOptionBO</name>
    <class>com.derimon.business.EuropeanOptionBOLocalHome</class>
  </business-object>
  <business-object>
    <name>local/AmericanOptionBO</name>
    <class>com.derimon.business.AmericanOptionBOLocalHome</class>
  </business-object>
  <inheritance>
    <ancestor>local/FinancialInstrumentBO</ancestor>
    <descendant>local/StockBO</descendant>
    <descendant>local/IndexBO</descendant>
    <descendant>local/CurrencyBO</descendant>
    <descendant>local/DerivativeBO</descendant>
  </inheritance>
  <inheritance>
    <ancestor>local/DerivativeBO</ancestor>
```

```
    <descendant>local/FutureBO</descendant>
    <descendant>local/OptionBO</descendant>
  </inheritance>
  <inheritance>
    <ancestor>local/OptionBO</ancestor>
    <descendant>local/EuropeanOptionBO</descendant>
    <descendant>local/AmericanOptionBO</descendant>
  </inheritance>
</inheritance-map>
```

Με τον τρόπο αυτό κάθε entity bean γνωρίζει τους απογόνους του χωρίς να απαιτείται καμία αλλαγή στην υλοποίησή του. Παραθέτουμε την υλοποίηση των μεθόδων ejbHomeLocate() και ejbHomeLocateAll() που αντιστοιχούν στις πολυμορφικές μεθόδους locate() και locateAll():

```
public FinancialInstrumentBOLocal ejbHomeLocate(String ric) throws FinderException {
  Context jndiCtx;
  Object homeObject = null;

  //At this point, all base EJB classes must know their EJB subclasses...
  if (inheritanceMap == null || inheritanceList == null) {
    initializeInheritance(name);
  }

  if (inheritanceList.size() > 0) {
    Iterator iterator = inheritanceList.iterator();
    while (iterator.hasNext()) {
      String jndiRef = (String) iterator.next();
      try {
        jndiCtx = new InitialContext();
        homeObject = jndiCtx.lookup(jndiRef);
      } catch (NamingException e) {
        e.printStackTrace();
        throw new FinderException("Cannot obtain JNDI initial context.");
      }
      try {
        System.out.println("Locating BOs for class " + (String)
inheritanceMap.get(jndiRef));
        // Dynamically load the class
        Class c = Class.forName((String) inheritanceMap.get(jndiRef));
        java.lang.reflect.Method method = c.getMethod("locate", new Class[] {
String.class });
        FinancialInstrumentBOLocal d = (FinancialInstrumentBOLocal)
method.invoke(homeObject, new Object[] { ric });
        return d;
      } catch (ClassNotFoundException e) {
        e.printStackTrace();
      } catch (NoSuchMethodException e) {
        e.printStackTrace();
      } catch (IllegalAccessException e) {
        e.printStackTrace();
      } catch (java.lang.reflect.InvocationTargetException ite) {
        System.out.println(ite);
      }
    }
  }
  else {
    String jndiRef = name;
    try {
      jndiCtx = new InitialContext();
      homeObject = jndiCtx.lookup(jndiRef);
    } catch (NamingException e) {
      e.printStackTrace();
      throw new FinderException("Cannot obtain JNDI initial context.");
    }
    try {
      System.out.println("Locating BOs for class " + (String)
inheritanceMap.get(jndiRef));
      // Dynamically load the class
      Class c = Class.forName((String) inheritanceMap.get(jndiRef));
      java.lang.reflect.Method method = c.getMethod("findLocateByPrimaryKey", new
Class[] { String.class });
```

```
      Collection collection = (Collection) method.invoke(homeObject, new Object[] {
ric });
      return (FinancialInstrumentBOLocal) collection.iterator().next();
    } catch (ClassNotFoundException e) {
      e.printStackTrace();
    } catch (NoSuchMethodException e) {
      System.out.println(e);
    } catch (IllegalAccessException e) {
      e.printStackTrace();
    } catch (java.lang.reflect.InvocationTargetException ite) {
      System.out.println(ite);
    } catch (NoSuchElementException nsee) {
      System.out.println(nsee);
    }
    try {
      System.out.println("Locating BOs for class " + (String)
inheritanceMap.get(jndiRef));
      // Dynamically load the class
      Class c = Class.forName((String) inheritanceMap.get(jndiRef));
      java.lang.reflect.Method method = c.getMethod("findByPrimaryKey", new Class[] {
String.class });
      FinancialInstrumentBOLocal d = (FinancialInstrumentBOLocal)
method.invoke(homeObject, new Object[] { ric });
      return d;
    } catch (ClassNotFoundException e) {
      e.printStackTrace();
    } catch (NoSuchMethodException e) {
      e.printStackTrace();
    } catch (IllegalAccessException e) {
      e.printStackTrace();
    } catch (java.lang.reflect.InvocationTargetException ite) {
      System.out.println(ite);
    }
  }
  throw new FinderException("Cannot find any " + this.getClass().getName() + " with
key " + ric);
}

public Collection ejbHomeLocateAll() throws FinderException
{
  Context jndiCtx;
  Object homeObject = null;
  ArrayList list = new ArrayList();
  //At this point, all base EJB classes must know their EJB subclasses...
  if (inheritanceMap == null || inheritanceList == null) {
    initializeInheritance(name);
  }

  if (inheritanceList.size() > 0) {
    Iterator iterator = inheritanceList.iterator();
    while (iterator.hasNext()) {
      String jndiRef = (String) iterator.next();
      try {
        jndiCtx = new InitialContext();
        homeObject = jndiCtx.lookup(jndiRef);
      } catch (NamingException e) {
        e.printStackTrace();
        throw new FinderException("Cannot obtain JNDI initial context.");
      }
      try {
        System.out.println("Locating BOs for class " + (String)
inheritanceMap.get(jndiRef));
        // Dynamically load the class
        Class c = Class.forName((String) inheritanceMap.get(jndiRef));
        java.lang.reflect.Method method = c.getMethod("locateAll", null);
        list.addAll((Collection) method.invoke(homeObject, null));
      } catch (ClassNotFoundException e) {
        e.printStackTrace();
      } catch (NoSuchMethodException e) {
        e.printStackTrace();
      } catch (IllegalAccessException e) {
        e.printStackTrace();
      } catch (java.lang.reflect.InvocationTargetException ite) {
        System.out.println(ite);
      }
    }
  }
```

```
  else {
    String jndiRef = name;
    try {
      jndiCtx = new InitialContext();
      homeObject = jndiCtx.lookup(jndiRef);
    } catch (NamingException e) {
      e.printStackTrace();
      throw new FinderException("Cannot obtain JNDI initial context.");
    }
    try {
      System.out.println("Locating BOs for class " + (String)
inheritanceMap.get(jndiRef));
      // Dynamically load the class
      Class c = Class.forName((String) inheritanceMap.get(jndiRef));
      java.lang.reflect.Method method = c.getMethod("findAll", null);
      list.addAll((Collection) method.invoke(homeObject, null));
    } catch (ClassNotFoundException e) {
      e.printStackTrace();
    } catch (NoSuchMethodException e) {
      e.printStackTrace();
    } catch (IllegalAccessException e) {
      e.printStackTrace();
    } catch (java.lang.reflect.InvocationTargetException ite) {
      System.out.println(ite);
    }
  }

  return list;
}
```

5.2.2.3 EJB Timer

Για δρομολογήσουμε (schedule) την κλήση σε τακτά χρονικά διαστήματα της υπηρεσίας ειδοποίησης της εφαρμογής αποτίμησης χρηματοοικονομικών παραγώγων, που υλοποιείται από την κλάση NotificationAppService, κάνουμε χρήση της υπηρεσίας χρονομετρητή για EJB (EJB Timer Service) που περιέχεται στην έκδοση EJB 2.1.

5.2.2.4 Σύνδεση Αντικειμένων Εργασίας και Μοντέλων Αποτίμησης

Ένα σημαντικό χαρακτηριστικό της εφαρμογής μας είναι η ευελιξία στον ορισμό νέων χρηματοοικονομικών προϊόντων και νέων μοντέλων αποτίμησης. Εξίσου σημαντικό όμως είναι και ο ορισμός των σχέσεων ανάμεσα σε παράγωγα και μοντέλα αποτίμησης να μπορεί να γίνει εύκολα, καθώς κάθε μοντέλο αποτίμησης είναι κατάλληλο για ορισμένους μόνο τύπους παραγώγων.

Για το σκοπό αυτό χρησιμοποιούμε το deployment descriptor αρχείο ejb-jar.xml. Εισάγουμε μία εγγραφή περιβάλλοντος (environment entry) σε κάθε business object που περιέχει τα μοντέλα που μπορούν να χρησιμοποιηθούν για την αποτίμησή του.

Για παράδειγμα, για το EuropeanOptionBO, εισάγουμε την εγγραφή:

```
<env-entry>
  <env-entry-name>methods</env-entry-name>
  <env-entry-type>java.lang.String</env-entry-type>
  <env-entry-value>black,binomial,monte</env-entry-value>
</env-entry>
```

5.2.3 Επίπεδο Διασύνδεσης (Integration Tier)

Για να αντλήσουμε δεδομένα για τις τιμές παραγώγων και μετοχών από τον πάροχο πληροφορίας Reuters, υλοποιούμε την κλάση ReutersAdapter που διαβάζει τις αντίστοιχες τιμές από την απάντηση που έρχεται σε μορφή XML.

Αντίστοιχη πληροφορία μπορούμε να πάρουμε όταν η πηγή της πληροφορίας είναι σε μορφή HTML, υλοποιώντας μία αντίστοιχη κλάση.

6

Μελέτη Περίπτωσης (Case Study)

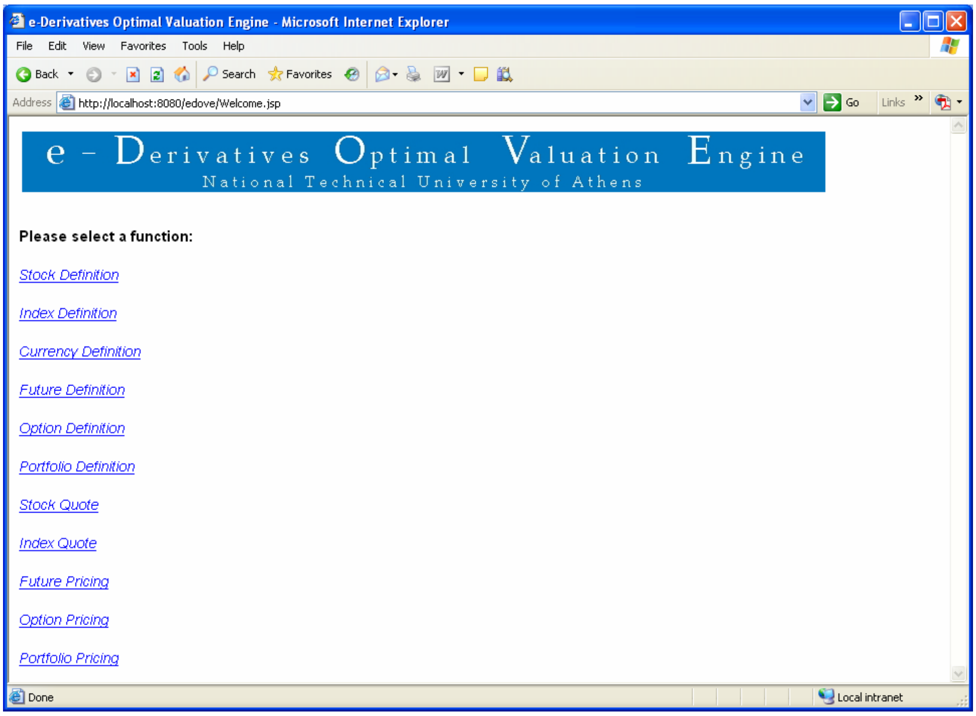
Στο κεφάλαιο αυτό δίνουμε ένα παράδειγμα των δυνατοτήτων της διαδικτυακής εφαρμογής αποτίμησης χρηματοοικονομικών παραγώγων που αποτελεί το προϊόν της παρούσας διπλωματικής εργασίας. Παρουσιάζουμε τα αποτελέσματα μίας μελέτης περίπτωσης πάνω σε πραγματικά δεδομένα από την Αγορά Παραγώγων του Χρηματιστηρίου Αθηνών.

6.1 Χρηματιστηριακά Δεδομένα

Για τη μελέτη μας χρησιμοποιούμε δεδομένα που η διαδικτυακή εφαρμογή αποτίμησης συλλέγει σε πραγματικό χρόνο από την Αγορά Παραγώγων του Χρηματιστηρίου Αθηνών. Επιλέγουμε τα δικαιώματα προαίρεσης με τιμή εξάσκησης 1800 για τους μήνες Οκτώβριο και Νοέμβριο του 2005 στο δείκτη “FTSE/ASE 20” του Χρηματιστηρίου Αθηνών. Θα μελετήσουμε τη συμπεριφορά τους κατά τη διάρκεια μίας συνεδρίασης του χρηματιστηρίου.

Ο πάροχος από τον οποίο θα αντλήσουμε τα χρηματιστηριακά δεδομένα για τη μελέτη περίπτωσης είναι το Reuters. Για το δείκτη και τα παράγωγά του χρησιμοποιούμε τους αντίστοιχους κωδικούς Reuters Instrument Codes (RICs).

Αρχικά πρέπει να ορίσουμε το δείκτη “FTSE/ASE 20” στην εφαρμογή μας. Από την αρχική σελίδα επιλέγουμε «Ορισμός Δεικτών».



e - Derivatives Optimal Valuation Engine
National Technical University of Athens

Please select a function:

Stock Definition

Index Definition

Currency Definition

Future Definition

Option Definition

Portfolio Definition

Stock Quote

Index Quote

Future Pricing

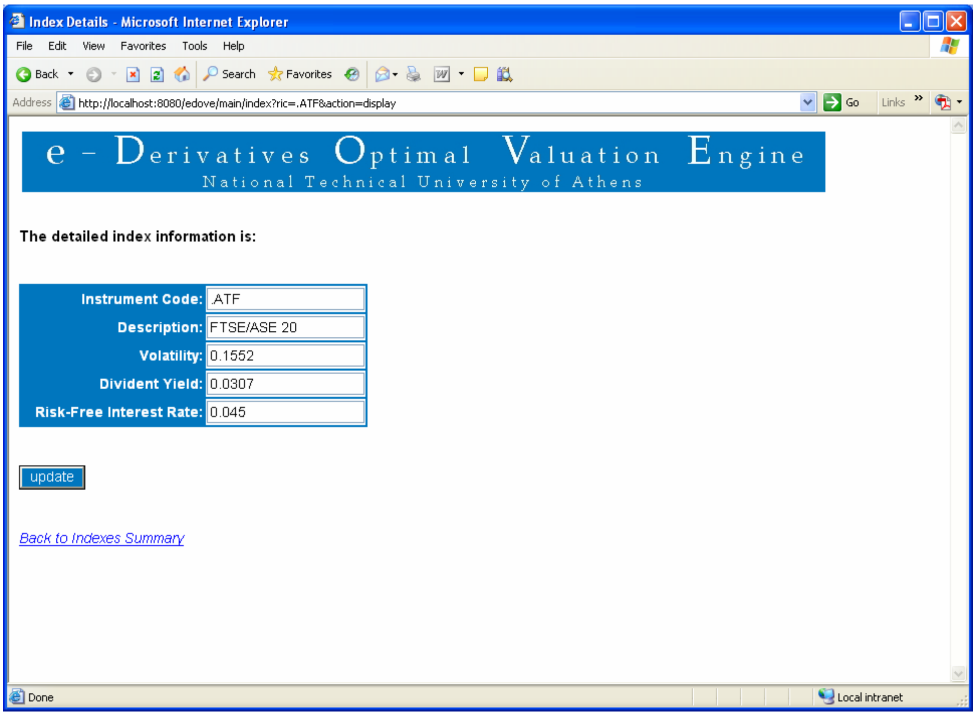
Option Pricing

Portfolio Pricing

Σχήμα 6.1: Αρχική σελίδα

Τα χαρακτηριστικά που εισάγουμε για το δείκτη “FTSE/ASE 20” είναι:

Κωδικός	.ATF
Περιγραφή	FTSE/ASE 20
Μεταβλητότητα	(0,1552)
Μερισματική Απόδοση	0,0307
Επιτόκιο Χωρίς Κίνδυνο	0,045



e - Derivatives Optimal Valuation Engine
National Technical University of Athens

The detailed index information is:

Instrument Code: .ATF
Description: FTSE/ASE 20
Volatility: 0.1552
Divident Yield: 0.0307
Risk-Free Interest Rate: 0.045

Back to Indexes Summary

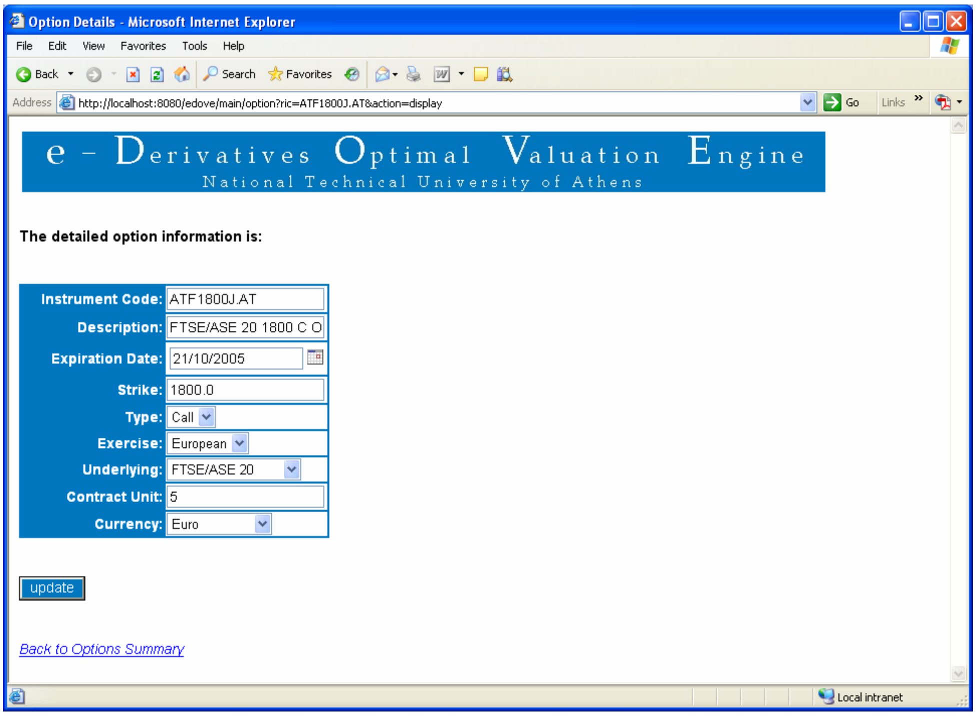
Σχήμα 6.2: Ορισμός Δείκτη “FTSE/ASE 20”

Θέτουμε το Επιτόκιο Χωρίς Κίνδυνο (Risk Free Interest Rate) στην τιμή 4,5% και υπολογίζουμε τη τρέχουσα μερισματική απόδοση του δείκτη “FTSE/ASE 20” σε 3,07%. Η μεταβλητότητα το δείκτη τους τελευταίους δώδεκα μήνες υπολογίζεται αυτόματα από την εφαρμογή μας.

Στη συνέχεια πρέπει να ορίσουμε τα δικαιώματα προαίρεσης στο δείκτη “FTSE/ASE 20”. Από την αρχική σελίδα επιλέγουμε «Ορισμός Δικαιωμάτων».

Τα χαρακτηριστικά που εισάγουμε για τα δικαιώματα προαίρεσης είναι:

Κωδικός	ATF1800J.AT	ATF1800V.AT	ATF1800K.AT	ATF1800W.AT
Περιγραφή	FTSE/ASE 20 1800 C Oct05	FTSE/ASE 20 1800 P Oct05	FTSE/ASE 20 1800 C Nov05	FTSE/ASE 20 1800 C Nov05
Ημερομηνία Λήξης	21/10/2005	21/10/2005	18/11/2005	18/11/2005
Τιμή Εξάσκησης	1800	1800	1800	1800
Τύπος Δικαιώματος	Αγοράς	Πώλησης	Αγοράς	Πώλησης
Τύπος Εξάσκησης	Ευρωπαϊκός	Ευρωπαϊκός	Ευρωπαϊκός	Ευρωπαϊκός
Υποκείμενος Τίτλος	.ATF	.ATF	.ATF	.ATF
Μονάδα Συμβολαίου	5	5	5	5
Νόμισμα	Ευρώ	Ευρώ	Ευρώ	Ευρώ



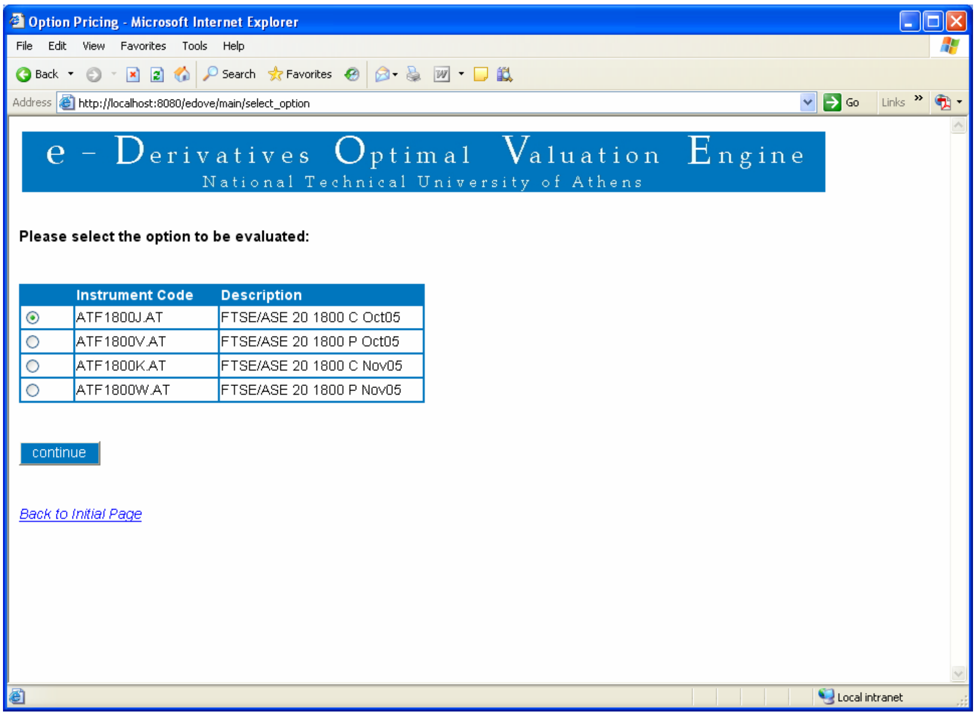
http://localhost:8080/edove/main/option?ric=ATF1800J.AT&action=display

The detailed option information is:

Instrument Code:	ATF1800J.AT
Description:	FTSE/ASE 20 1800 C O
Expiration Date:	21/10/2005
Strike:	1800.0
Type:	Call
Exercise:	European
Underlying:	FTSE/ASE 20
Contract Unit:	5
Currency:	Euro

Σχήμα 6.3: Ορισμός Δικαιώματος Προαίρεσης “FTSE/ASE 20 1800 C Oct05”

Αφού ορίσαμε το δείκτη και τα παράγωγά του, θα δείξουμε πώς μπορούμε να χρησιμοποιήσουμε τη διαδικτυακή εφαρμογή αποτίμησης χρηματοοικονομικών παραγώγων για να υπολογίσουμε την τρέχουσα και τη θεωρητική τους τιμή. Από την αρχική σελίδα επιλέγουμε «Αποτίμηση Δικαιωμάτων»:



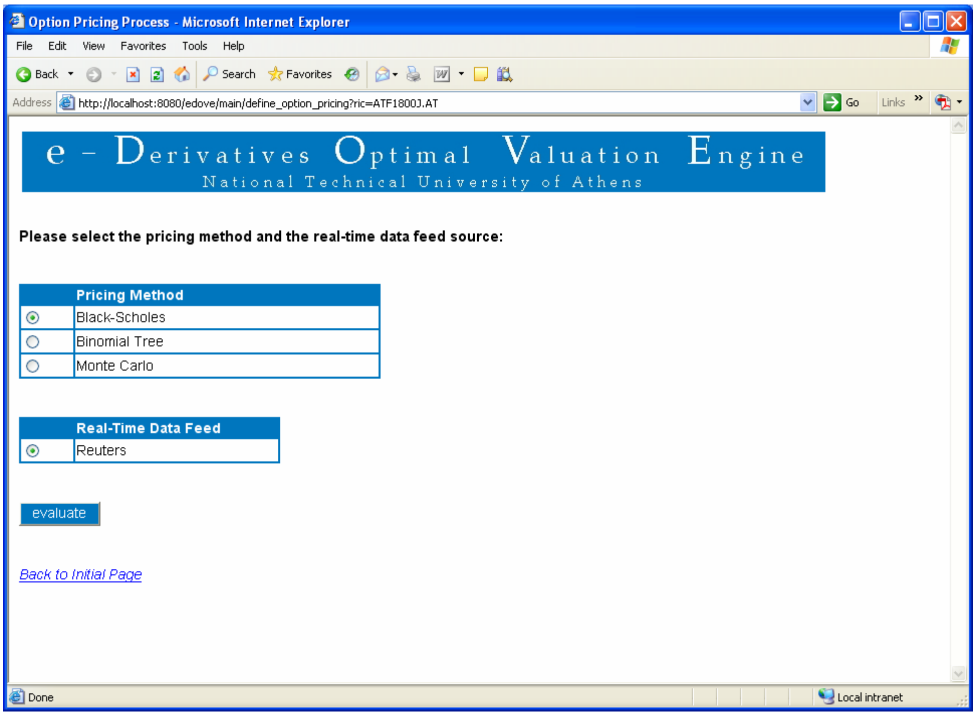
http://localhost:8080/edove/main/select_option

Please select the option to be evaluated:

	Instrument Code	Description
●	ATF1800J.AT	FTSE/ASE 20 1800 C Oct05
○	ATF1800V.AT	FTSE/ASE 20 1800 P Oct05
○	ATF1800K.AT	FTSE/ASE 20 1800 C Nov05
○	ATF1800W.AT	FTSE/ASE 20 1800 P Nov05

Σχήμα 6.4: Αποτίμηση Δικαιωμάτων Προαίρεσης

Στη συνέχεια επιλέγουμε το μοντέλο αποτίμησης:



http://localhost:8080/edove/main/define_option_pricing?ric=ATF1800J.AT

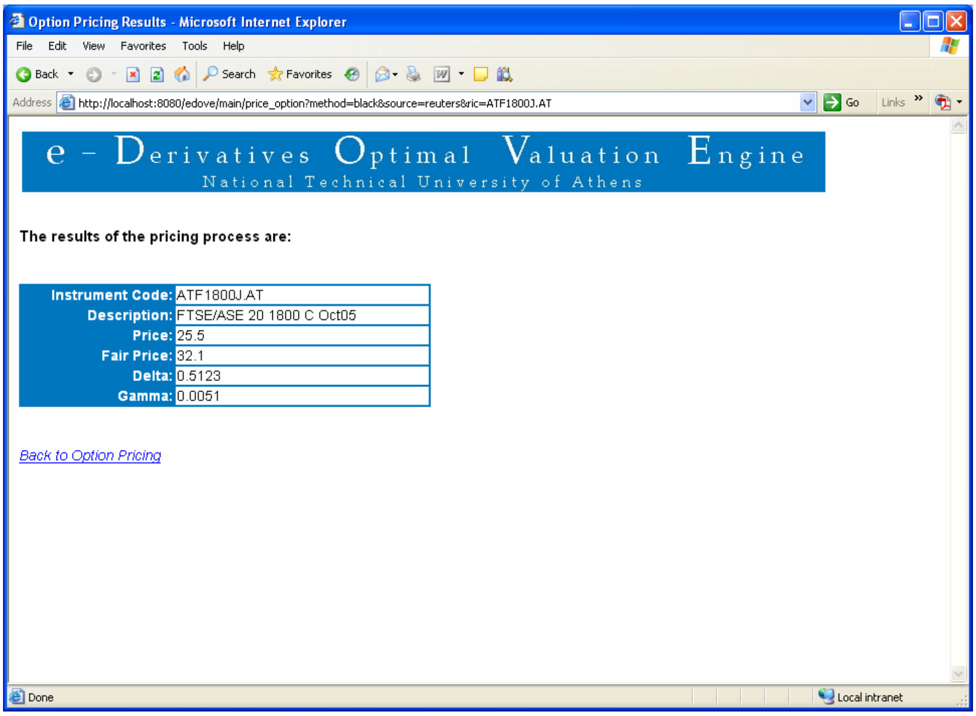
Please select the pricing method and the real-time data feed source:

	Pricing Method
●	Black-Scholes
○	Binomial Tree
○	Monte Carlo

	Real-Time Data Feed
●	Reuters

Σχήμα 6.5: Επιλογή Μοντέλου Αποτίμησης

Το αποτέλεσμα της διαδικασίας αποτίμησης είναι εικονίζεται στο σχήμα 6.6. Η τρέχουσα τιμή του δικαιώματος προαίρεσης συλλέγεται από τον πάροχο χρηματιστηριακών δεδομένων. Εκτός από τη θεωρητική τιμή, υπολογίζονται τα Ελληνικά Γράμματα (Greek Letters) «δέλτα» και «γάμμα».



http://localhost:8080/edove/main/price_option?method=black&source=reuters&ric=ATF1800J.AT

e - Derivatives Optimal Valuation Engine
National Technical University of Athens

The results of the pricing process are:

Instrument Code:	ATF1800J.AT
Description:	FTSE/ASE 20 1800 C Oct05
Price:	25.5
Fair Price:	32.1
Delta:	0.5123
Gamma:	0.0051

Back to Option Pricing

Σχήμα 6.6: Αποτελέσματα Αποτίμησης Δικαιώματος Προαίρεσης “FTSE/ASE 20 1800 C Oct05”

6.2 Απόκλιση από Ισοτιμία Αγοράς-Πώλησης (Put-Call Parity)

Θα χρησιμοποιήσουμε το μηχανισμό ειδοποίησης της εφαρμογής αποτίμησης χρηματοοικονομικών παραγώγων για να ελέγξουμε κατά πόσο τηρείται η Ισοτιμία Αγοράς-Πώλησης (Put-Call Parity) για τα δικαιώματα προαίρεσης στον δείκτη “FTSE/ASE 20” που ορίσαμε στην προηγούμενη ενότητα.

Αν και ο μηχανισμός ειδοποίησης προορίζεται κατά κύριο λόγο να μας ενημερώνει για ευκαιρίες αποκόμισης κέρδους (arbitrage) ή ανάγκη αναδιάρθρωσης χαρτοφυλακίου (hedging), στην προκειμένη περίπτωση θα χρησιμοποιηθεί για να συλλέξουμε χρηματιστηριακά δεδομένα για ολόκληρη τη διάρκεια μίας συνεδρίασης της Αγοράς Παραγώγων του Χρηματιστηρίου Αθηνών.

Για τα ζεύγη δικαιωμάτων προαίρεσης αγοράς-πώλησης που ορίσαμε στην προηγούμενη ενότητα, υπολογίζουμε την απόκλιση από την Ισοτιμία Αγοράς-Πώλησης (Put-Call Parity), όπως προκύπτει από τη διαφορά:

$$D = c + Ke^{-(r-r_f)T} - p - I_0$$

όπου:

c: Τιμή δικαιώματος αγοράς

p: Τιμή δικαιώματος πώλησης

I_0: Τρέχουσα τιμή δείκτη

K: Τιμή εξάσκησης δικαιώματος

r: Επιτόκιο χωρίς κίνδυνο

r_f: Μερισματική απόδοση δείκτη

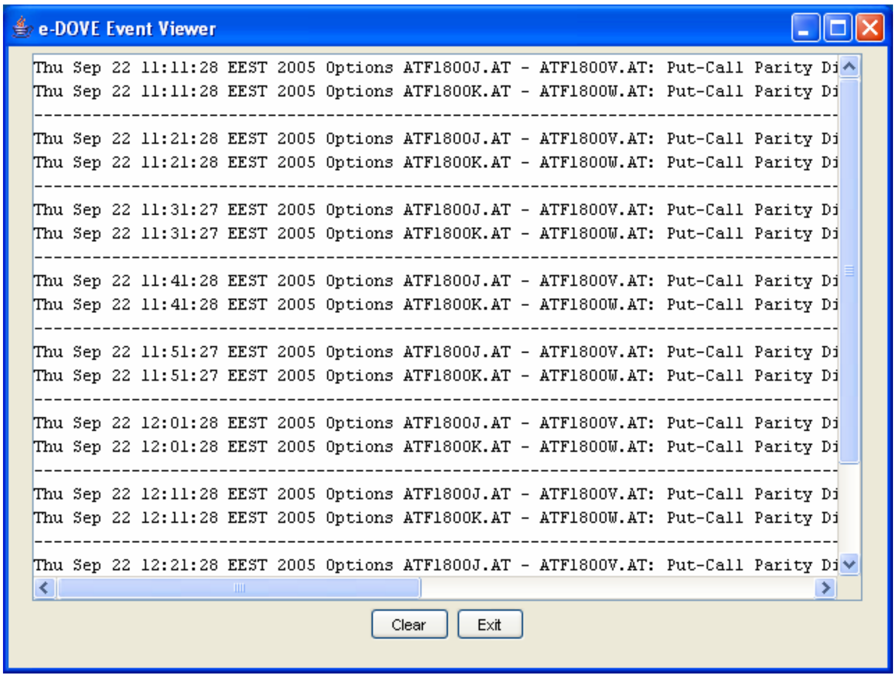
T: Χρόνος μέχρι τη λήξη του δικαιώματος

Αν η απόκλιση από την Ισοτιμία Αγοράς-Πώλησης είναι σημαντική, δημιουργούνται ευκαιρίες για εξισορροπητική κερδοσκοπία (arbitrage).

Εισάγουμε κώδικα που υπολογίζει την παραπάνω διαφορά στη μέθοδο detectEvent() του EuropeanOptionBO, όπως ορίζει το πρότυπο σχεδίασης Observer που περιγράψαμε αναλυτικά στο κεφάλαιο της σχεδίασης.

6.3 Αποτελέσματα Μελέτης

Συλλέγουμε δεδομένα για την απόκλιση από την Ισοτιμία Αγοράς-Πώλησης (Put-Call Parity) των δικαιωμάτων προαίρεσης στο δείκτη “FTSE/ASE 20” καθόλη τη διάρκεια της συνεδρίασης της Αγοράς Παραγώγων του Χρηματιστηρίου Αθηνών της 22^{ης} Σεπτεμβρίου 2005. Η αποτίμηση λαμβάνει χώρα κάθε 10 λεπτά. Τα δεδομένα αποστέλλονται μέσω TCP/IP στη βοηθητική εφαρμογή Event Viewer, η οποία τα καταγράφει σε αρχείο και τα εμφανίζει στο παράθυρο ειδοποίησης που εικονίζεται στο σχήμα 6.7:

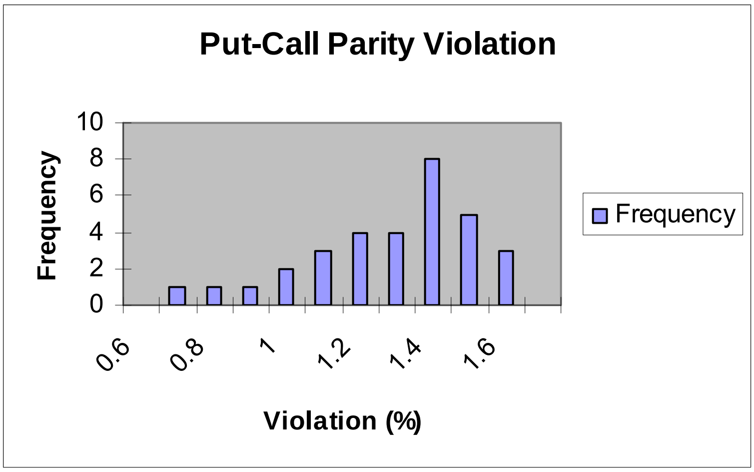


```
Thu Sep 22 11:11:28 EEST 2005 Options ATF1800J.AT - ATF1800V.AT: Put-Call Parity Di
Thu Sep 22 11:11:28 EEST 2005 Options ATF1800K.AT - ATF1800W.AT: Put-Call Parity Di
-----------------------------------------------------------------------------------
Thu Sep 22 11:21:28 EEST 2005 Options ATF1800J.AT - ATF1800V.AT: Put-Call Parity Di
Thu Sep 22 11:21:28 EEST 2005 Options ATF1800K.AT - ATF1800W.AT: Put-Call Parity Di
-----------------------------------------------------------------------------------
Thu Sep 22 11:31:27 EEST 2005 Options ATF1800J.AT - ATF1800V.AT: Put-Call Parity Di
Thu Sep 22 11:31:27 EEST 2005 Options ATF1800K.AT - ATF1800W.AT: Put-Call Parity Di
-----------------------------------------------------------------------------------
Thu Sep 22 11:41:28 EEST 2005 Options ATF1800J.AT - ATF1800V.AT: Put-Call Parity Di
Thu Sep 22 11:41:28 EEST 2005 Options ATF1800K.AT - ATF1800W.AT: Put-Call Parity Di
-----------------------------------------------------------------------------------
Thu Sep 22 11:51:27 EEST 2005 Options ATF1800J.AT - ATF1800V.AT: Put-Call Parity Di
Thu Sep 22 11:51:27 EEST 2005 Options ATF1800K.AT - ATF1800W.AT: Put-Call Parity Di
-----------------------------------------------------------------------------------
Thu Sep 22 12:01:28 EEST 2005 Options ATF1800J.AT - ATF1800V.AT: Put-Call Parity Di
Thu Sep 22 12:01:28 EEST 2005 Options ATF1800K.AT - ATF1800W.AT: Put-Call Parity Di
-----------------------------------------------------------------------------------
Thu Sep 22 12:11:28 EEST 2005 Options ATF1800J.AT - ATF1800V.AT: Put-Call Parity Di
Thu Sep 22 12:11:28 EEST 2005 Options ATF1800K.AT - ATF1800W.AT: Put-Call Parity Di
-----------------------------------------------------------------------------------
Thu Sep 22 12:21:28 EEST 2005 Options ATF1800J.AT - ATF1800V.AT: Put-Call Parity Di
```

Σχήμα 6.7: Δεδομένα Απόκλισης από Ισοτιμία Αγοράς-Πώλησης

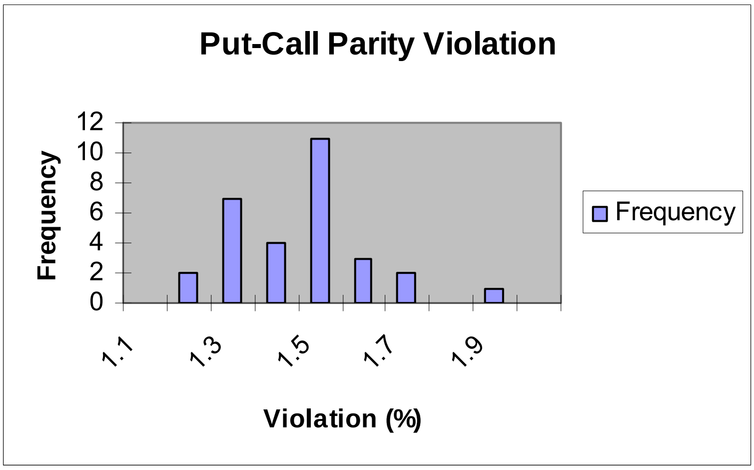
Από τα δεδομένα αυτά προκύπτει ένας αριθμός δειγμάτων της ποσοστιαίας απόκλισης από την Ισοτιμία Αγοράς-Πώλησης (Put-Call Parity) κατά τη διάρκεια της συγκεκριμένης συνεδρίασης.

Το ιστόγραμμα με την κατανομή των δειγμάτων απόκλισης για τα δικαιώματα προαίρεσης στο δείκτη “FTSE/ASE 20” με τιμή εξάσκησης 1800 για το μήνα Οκτώβριο του 2005 εικονίζεται στο σχήμα 6.8:



Σχήμα 6.8: Ιστόγραμμα Απόκλισης από Ισοτιμία Αγοράς-Πώλησης κατά την 22η Σεπτεμβρίου 2005 για τα Δικαιώματα Οκτωβρίου 2005 στο δείκτη “FTSE/ASE 20” με τιμή εξάσκησης 1800

Το ιστόγραμμα με την κατανομή των δειγμάτων απόκλισης για τα δικαιώματα προαίρεσης στο δείκτη “FTSE/ASE 20” με τιμή εξάσκησης 1800 για το μήνα Νοέμβριο του 2005 εικονίζεται στο σχήμα 6.9:



Σχήμα 6.9: Ιστόγραμμα Απόκλισης από Ισοτιμία Αγοράς-Πώλησης κατά την 22η Σεπτεμβρίου 2005 για τα Δικαιώματα Νοεμβρίου 2005 στο δείκτη “FTSE/ASE 20” με τιμή εξάσκησης 1800

7

Επίλογος

Στο κεφάλαιο αυτό συνοψίζονται τα αποτελέσματα της διπλωματικής εργασίας. Εξετάζεται σε πιο βαθμοί οι στόχοι που είχαν τεθεί αρχικά έχουν ικανοποιηθεί. Τέλος γίνεται μία αναφορά σε σημεία στα οποία μπορούν να προκύψουν βελτιώσεις και επεκτάσεις.

7.1 Σύνοψη και Συμπεράσματα

Σκοπός της παρούσας διπλωματικής εργασίας, όπως τον είχαμε ορίσει, ήταν ο συγκερασμός της θεωρίας και των μαθηματικών μοντέλων που έχουν αναπτυχθεί για την αποτίμηση χρηματοοικονομικών παραγώγων με τις σύγχρονες τάσεις στο χώρο της Τεχνολογίας Λογισμικού, όπως η ανάπτυξη διαδικτυακών (web-based) εφαρμογών, η αρχιτεκτονική πολλαπλών επιπέδων (multi-tiered) και η χρήση προτύπων σχεδίασης (design patterns).

Προϊόν του εγχειρήματος αυτού είναι η σχεδίαση και ανάπτυξη μίας διαδικτυακής (web-based) εφαρμογής για την αποτίμηση χρηματιστηριακών παραγώγων και χαρτοφυλακίων σε πραγματικό χρόνο. Η σχεδίαση βασίζεται εξολοκλήρου στη χρήση προτύπων σχεδίασης. Κάνουμε χρήση τόσο γενικευμένων προτύπων σχεδίασης όσο και εξειδικευμένων για την τεχνολογία J2EE.

Υλοποιούμε τα κυριότερα μοντέλα για την αποτίμηση χρηματοοικονομικών παραγώγων (εξίσωση Black-Scholes, διωνυμικές μέθοδοι, προσομοίωση Monte Carlo).

Τέλος επιδεικνύουμε τη χρήση της εφαρμογής πάνω σε πραγματικά δεδομένα από την Αγορά Παραγώγων του Χρηματιστηρίου Αθηνών.

7.2 Μελλοντικές Επεκτάσεις

Μία προφανής βελτίωση είναι η προσθήκη νέων τύπων χρηματοοικονομικών προϊόντων και νέων μοντέλων αποτίμησης.

Μπορεί να εξεταστεί η χρήση κανόνων αποφάσεων (decision rules) για τον ορισμό των κανόνων που θα ενεργοποιούν το μηχανισμό ειδοποίησης. Οι κανόνες αυτοί θα μπορούν να ορίζονται με δηλωτικό τρόπο και όχι προγραμματιστικά, όπως στην τρέχουσα υλοποίηση.

Μπορεί τέλος να εξεταστεί η προσθήκη περισσότερων στοιχείων Χρηματοοικονομικής Μηχανικής (Financial Engineering), όπως η Αξία σε Κίνδυνο (Value at Risk), ο Πιστωτικός Κίνδυνος (Credit Risk) και η προσομοίωση ακραίων σεναρίων.

8

Βιβλιογραφία

ACM03	Alur, D., Crupi, J., and Malks, D., “Core J2EE Patterns: Best Practices and Design Strategies”, Second Edition, Prentice Hall, 2003
BE93	Birrer, A., and Eggenschwiler, T., “Frameworks in the Financial Engineering Domain: An Experience Report”, In Proceedings ECOOP '93 (LNCS 707). Berlin: Springer-Verlag, 1993, 21-35.
BF00	Bertsimas, D., and Freund, R., “Data, Models and Decisions: The Fundamentals of Management Science”, South-Western, 2000
BS73	Black, F., and Scholes, M., “The Pricing of Options and Corporate Liabilities”, Journal of Political Economy, 81 (May/June 1973), 637-659.
BSB04	Basham, B., Sierra K., and Bates, B., “Head First Servlets & JSP”, O’Reilly, 2004
DAKW03	Dudney, B., Asbury, S., Krozak, J., and Wittkopf, K., “J2EE Antipatterns”, Wiley, 2003
Duffy04	Duffy, D., “Financial Instrument Pricing Using C++”, Wiley, 2004
EG92	Eggenschwiler, T., and Gamma, E., “ET++SwapsManager: Using Object Technology in the Financial Engineering Domain”, In Proceedings OOPSLA '92, ACM SIGPLAN Notices 27, 10 (October 1992), 166-177.
FFSB04	Freeman, Eric, Freeman, Elizabeth, Sierra K., and Bates, B., “Head First Design Patterns”, O’Reilly, 2004
GHJV95	Gamma, E., Helm, R., Johnson, R., and Vlissides, J., “Design Patterns: Elements of Reusable Object-Oriented Software”, Addison-Wesley, 1995
HB04	Hall, M., and Brown, L., “Core Servlets and JavaServer Pages”, 2nd Edition, Prentice Hall, 2004
Hull03	Hull, J., “Options, Futures and Other Derivatives”, Fifth Edition, Prentice Hall, 2003

JBR99 Jacobson, I., Booch, G., and Rumbaugh, J., “The Unified Software Development Process”, Addison-Wesley, 1999

Joshi04 Joshi, M., “C++ Design Patterns and Derivatives Pricing”, Cambridge, 2004

KTP02 Koulisianis, M., Tsolis, G., and Papatheodorou, T., “A Web-Based Problem Solving Environment for Solution of Option Pricing Problems and Comparison of Methods”, Proceedings of the International Conference on Computational Science-Part I, p.673-682, April 21-24, 2002

Larman05 Larman, C., “Applying UML and Patterns”, Third Edition, Prentice Hall, 2005

London05 London, J., “Modeling Derivatives in C++”, Wiley, 2005

Marinescu02 Marinescu, F., “EJB Design Patterns: Advanced Patterns Processes, and Idioms”, Wiley, 2002

Marsura98 Marsura P., “A Risk Management Framework for Derivative Instruments”, M.Sc. Thesis, University of Illinois, Chicago, 1998.

Mikosch98 Mikosch, T., “Elementary Stochastic Calculus with Finance in View”, World Scientific, 1998

Mittnik00 Mittnik, S., and Rieken, S., "Put-Call Parity and the Informational Efficiency of the German DAX-Index Options Market", International Review of Financial Analysis, Volume 9, Issue 3, Pages 259-279, Autumn 2000

MSE99 van der Meij, M., Schouten, D., and Eliëns, A., “Design Patterns for Derivatives Software”, ICT Architecture in the BeNeLux 1999, November 18-19, 1999, Amsterdam, The Netherlands

Proulx02 Proulx, E., “EJB Inheritance, Part 1”, ONJava.com, 2004
Available on-line at:
http://www.onjava.com/pub/a/onjava/2002/09/04/ejbinherit.html
RSB05 Rodman, E., Sriganesh, R. P., and Brose, G., “Mastering Enterprise JavaBeans”, Third Edition, Wiley, 2005

SB03 Sierra K., and Bates, B., “Head First EJB”, O’Reilly, 2003

Wilmott01 Wilmott, P., “Paul Wilmott Introduces Quantitative Finance”, Wiley, 2001

WP97 Watsham, T. J., and Parramore K., “Quantitative Methods in Finance”, Thomson, 1997

ZS96 Zhang, J. Q., and Sternbach, E., “Financial Software Design Patterns”, Journal of Object-Oriented Programming 8, 9 (February 1996), 6-12.

Παράρτημα Α

Δεδομένα απόκλισης από την Ισοτιμία Αγοράς-Πώλησης για τα Δικαιώματα Οκτωβρίου 2005 στο δείκτη “FTSE/ASE 20” του Χρηματιστηρίου Αθηνών με τιμή εξάσκησης 1800 κατά τη συνεδρίαση της 22ης Σεπτεμβρίου 2005:

Time	.ATF	Call Price	Put Price	PCP Violation	PCP Violation %
11:11:28	1779	15.5	61.5	27.01	1.518269
11:21:28	1783.76	16.5	55.5	24.78	1.3892
11:31:27	1779.42	15.5	56.5	22.43	1.260523
11:41:28	1773.66	14	58	19.66	1.108442
11:51:27	1777.25	15	58	22.26	1.252497
12:01:28	1780.37	16.5	58	23.88	1.341294
12:11:28	1776.74	15.5	60	23.24	1.308014
12:21:28	1776.71	15.5	60	23.21	1.306347
12:31:27	1776.76	15.5	57.5	20.77	1.168982
12:41:28	1777.2	15.5	57.5	21.2	1.192888
12:51:28	1777	15.5	57.5	21.01	1.18233
13:01:28	1777.56	15.5	55	19.07	1.072819
13:11:28	1776.5	15.5	55	18	1.013228
13:21:29	1775.76	15	55	17.77	1.000698
13:31:28	1774.68	15	55	16.68	0.939888
13:41:27	1773.61	14.5	55	16.12	0.908881
13:51:29	1773.64	14.5	64	25.14	1.417424
14:01:28	1772.12	14.5	64	23.63	1.333431
14:11:28	1772.15	14.25	64	23.9	1.348644
14:21:28	1772.42	14	64	24.42	1.377777
14:31:28	1771.52	16	64	21.53	1.21534
14:41:27	1771.27	16	64	21.28	1.201398
14:51:28	1773.16	16	64	23.16	1.306143
15:01:27	1775.28	16.25	64	25.03	1.409918
15:11:28	1775.71	16.25	64	25.45	1.43323
15:21:28	1775.03	15	64	26.03	1.466454
15:31:28	1775.87	15	64	26.87	1.513061
15:41:27	1775.54	16	64	25.54	1.438436
15:51:28	1779.34	17.25	64	28.09	1.578675
16:01:28	1779.96	18	50.5	14.45	0.811816
16:11:28	1779.96	19	50.5	13.45	0.755635
16:21:57	1779.96	19	49	11.95	0.671363

Πηγή: Reuters

Παράρτημα Β

Δεδομένα απόκλισης από την Ισοτιμία Αγοράς-Πώλησης για τα Δικαιώματα Νοεμβρίου 2005 στο δείκτη “FTSE/ASE 20” του Χρηματιστηρίου Αθηνών με τιμή εξάσκησης 1800 κατά τη συνεδρίαση της 22ης Σεπτεμβρίου 2005:

Time	.ATF	Call Price	Put Price	PCP Violation	PCP Violation %
11:11:28	1779	29	74	27.98	1.572794
11:21:28	1783.76	29	74	32.75	1.836009
11:31:27	1779.42	29	74	28.4	1.596026
11:41:28	1773.66	29	74	22.64	1.276457
11:51:27	1777.25	29	74	26.24	1.476438
12:01:28	1780.37	29	74	29.36	1.649095
12:11:28	1776.74	29	74	25.72	1.447595
12:21:28	1776.71	29	74	25.69	1.445931
12:31:27	1776.76	29	74	25.75	1.449267
12:41:28	1777.2	29	74	26.18	1.473104
12:51:28	1777	29	74	25.99	1.462577
13:01:28	1777.56	29	74	26.54	1.493058
13:11:28	1776.5	29	74	25.47	1.433718
13:21:29	1775.76	29	74	24.74	1.393206
13:31:28	1774.68	29	74	23.65	1.332635
13:41:27	1773.61	29	74	22.59	1.273673
13:51:29	1773.64	29	74	22.61	1.27478
14:01:28	1772.12	29	74.5	21.6	1.218879
14:11:28	1772.15	29	74.5	21.62	1.219987
14:21:28	1772.42	29	74.5	21.89	1.235035
14:31:28	1771.52	29	74.5	21	1.185423
14:41:28	1771.27	29	74.5	20.75	1.171476
14:51:28	1773.16	29	74.5	22.63	1.276253
15:01:27	1775.28	29	74.5	24.76	1.39471
15:11:28	1775.71	29	74.5	25.18	1.418024
15:21:28	1775.03	29	74.5	24.51	1.380822
15:31:28	1775.87	29	74.5	25.35	1.427469
15:41:27	1775.54	29	74.5	25.02	1.409149
15:51:28	1779.34	29.75	74.5	28.07	1.577551
16:01:28	1779.96	29.75	74.5	28.68	1.611272

Πηγή: Reuters