ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Π.Μ.Σ. "ΕΦΑΡΜΟΣΜΕΝΑ ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ"

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη διαδικτυακών εφαρμογών με χρήση ανοιχτού κώδικα πλαισίου Java

Χρήστος Κώτσας

Εισηγητής: Δρ Κωνσταντίνος Κουκουλέτσος, Καθηγητής

ΑΘΗΝΑ
ΙΟΥΝΙΟΣ 2019

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη διαδικτυακών εφαρμογών με χρήση ανοιχτού κώδικα πλαισίου Java

Χρήστος Κώτσας

A.M. : ais0111

Εισηγητής:

Κουκουλέτσος Κωνσταντίνος, Καθηγητης

Εξεταστική Επιτροπή:

....., Καθηγητής
....., Καθηγητής

Ημερομηνία εξέτασης: Ιούλιος 2019

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Κώτσας Χρήστος, του Ελευθερίου με αριθμό μητρώου ais0111 φοιτητής του Τμήματος Μηχανικών Η/Υ Συστημάτων Τ.Ε. του Α.Ε.Ι. Πειραιά Τ.Τ. πριν αναλάβω την εκπόνηση της Πτυχιακής Εργασίας μου, δηλώνω ότι ενημερώθηκα για τα παρακάτω:

«Η Πτυχιακή Εργασία (Π.Ε.) αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο του συγγραφέα, όσο και του Ιδρύματος και θα πρέπει να έχει μοναδικό χαρακτήρα και πρωτότυπο περιεχόμενο.

Απαγορεύεται αυστηρά οποιοδήποτε κομμάτι κειμένου της να εμφανίζεται αυτούσιο ή μεταφρασμένο από κάποια άλλη δημοσιευμένη πηγή. Κάθε τέτοια πράξη αποτελεί προϊόν λογοκλοπής και εγείρει θέμα Ηθικής Τάξης για τα πνευματικά δικαιώματα του άλλου συγγραφέα. Αποκλειστικός υπεύθυνος είναι ο συγγραφέας της Π.Ε., ο οποίος φέρει και την ευθύνη των συνεπειών, ποινικών και άλλων, αυτής της πράξης.

Πέραν των όποιων ποινικών ευθυνών του συγγραφέα σε περίπτωση που το Ίδρυμα του έχει απονείμει Πτυχίο, αυτό ανακαλείται με απόφαση της Συνέλευσης του Τμήματος. Η Συνέλευση του Τμήματος με νέα απόφασης της, μετά από αίτηση του ενδιαφερόμενου, του αναθέτει εκ νέου την εκπόνηση της Π.Ε. με άλλο θέμα και διαφορετικό επιβλέποντα καθηγητή. Η εκπόνηση της εν λόγω Π.Ε. πρέπει να ολοκληρωθεί εντός τουλάχιστον ενός ημερολογιακού 6μήνου από την ημερομηνία ανάθεσης της. Κατά τα λοιπά εφαρμόζονται τα προβλεπόμενα στο άρθρο 18, παρ. 5 του ισχύοντος Εσωτερικού Κανονισμού.»

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα εργασία αποτελεί διπλωματική εργασία στα πλαίσια του μεταπτυχιακού προγράμματος «Εφαρμοσμενα Πληροφοριακά Συστήματα» του τμήματος Μηχανικών Ηλεκτρονικών Υπολογιστικών Συστημάτων.

Πριν την παρουσίαση της διπλωματικής εργασίας,αισθάνομαι την υποχρέωση να ευχαριστήσω ορισμένους από τους ανθρώπους που γνώρισα, συνεργάστηκα μαζί τους και έπαιξαν πολύ σημαντικό ρόλο στην πραγματοποίησή της.

Πρώτα από όλους θέλω να ευχαριστήσω τον επιβλέποντα καθηγητή της διπλωματικής εργασίας, Καθηγητή Κώστα Κουκουλέτσο για την πολύτιμη καθοδήγηση του, την εμπιστοσύνη και εκτίμηση που μου έδειξε.

Στη συνέχεια θα ήθελα να ευχαριστήσω όλους τους καθηγητές του μεταπτυχιακού για τις πολύτιμες γνώσεις που μου πρόσφεραν.

Επίσης θα ήθελα να ευχαριστήσω την οικογένεια μου, που με υπομονή και κουράγιο πρόσφεραν την απαραίτητη ηθική συμπαράσταση για την ολοκλήρωση της μεταπτυχιακής μου εργασίας

Τέλος, ένα μεγάλο ευχαριστώ στους συμφοιτητές του μεταπτυχιακού προγράμματος για την συνεργασία που είχαμε όλα αυτό το διάστημα.

Περίληψη

Στο κεφάλαιο ένα δίνουμε τον ορισμό του framework Vaadin και στη συνέχεια γίνεται μία σύντομη ιστορική αναδρομή της πλατφόρμας ανοιχτού κώδικα. Η τελευταία έκδοση που χρησιμοποιείται είναι έκδοση 11, η οποία κυκλοφόρησε για πρώτη φορά το 2018.

Στο κεφάλαιο δύο περιγράφεται βήμα-βήμα η εγκατάσταση και η παραμετροποίηση όλων των πακέτων, που είναι απαραίτητα για την υλοποίηση του πρώτου project σε Vaadin 7. Όλα τα παραδείγματα καθώς επίσης και η εφαρμογή έχουν υλοποιηθεί με τη βοήθεια του IDE Eclipse.

Στο επόμενο κεφάλαιο περιγράφουμε αναλυτικά την κύρια συνάρτηση μέσα στην οποία υλοποιούμε τα αντικείμενα των components του Vaadin. Περιγράφουμε επίσης τα βασικότερα components χρησιμοποιώντας μικρά παραδείγματα. Μερικά από τα components είναι η εισαγωγή του Button, του Label, του TextField κ.α.

Στο τέταρτο κεφάλαιο συνεχίζουμε την περιγραφή των πιο περίπλοκων components που χρησιμοποιούνται στο Vaadin. Μερικά απ' αυτά είναι Check Box, Check Box Group, Twin Column Select, Date κ.α.

Στο πέμπτο κεφάλαιο περιγράφουμε τα είδη των Layouts και τις διαφορές τους. Τα Layouts είναι αόρατα components που μέσα σε αυτά μπορούμε να προσθέσουμε όλα τα άλλα Components. Θεωρούνται ως το βασικότερα component του Vaadin.

Στο προτελευταίο κεφάλαιο περιγράφουμε τους πίνακες, ένα από τα πλέον σημαντικά components του Vaadin. Επίσης περιγράφουμε τον τρόπο με τον οποίο εισάγουμε κεφαλίδες και υποσέλιδα σε έναν πίνακα, χρησιμοποιώντας απλά παραδείγματα. Ιδιαίτερη αναφορά κάνουμε για τον τρόπο με τον οποίο διαβάζουμε και εγγράφουμε δεδομένα σε ένα πίνακα.

Τέλος στο τελευταίο κεφάλαιο επεξηγούμε την εφαρμογή που υλοποιήθηκε και αφορά την καταχώρηση των πελατών μιας εταιρείας. Πιο συγκεκριμένα περιγράφουμε τον κώδικα της κάθε κλάσης ξεχωριστά και παρουσιάζουμε τα αποτελέσματα σε μορφή εικόνας.

ABSTRACT

This thesis consists of the following chapters.

Chapter 1: We explain the meaning of an open-source platform for web applications development. We give a brief history from the start, as an adapter built on top of Millstone UI framework in 2002, until nowadays.

Chapter 2: We describe step by step the installation of all applications that are necessary and explain how to set up the development environment in order to create our first project in Vaadin. The source code for all examples, including the "customer address book" application, have been developed in IDE eclipse.

Chapter 3: We explain basic usage of input components like "Button", "TextField" etc. and we also explain the main function in Vaadin Project.

Chapter 4: We cover more complicated components of Vaadin like Check Box, Check Box Group, Twin Column Select, Date etc. For all components we use simple examples and the results are depicted in pictures.

Chapter 5 covers the most important layout components. All the components have been arranged in "invisible" layout which are one the most valuable component in Vaadin.

Chapter 6 covers Using Tables, one of the most useful and powerful UI componets in Vaadin. After that we describe the way that we can add headers and footers in table using small examples. It also worth to cover the way that we can read and write data into tables.

Chapter 7: We develop a "Customer address book" application in Vaadin. The main purpose of this application is to store the clients of a company. Each source code is explained in detail and the results are shown in pictures.

ΕΠΙΣΤΗΜΟΝΙΚΗ ΠΕΡΙΟΧΗ: Δυαδικτυακές εφαρμογές

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: επεξεργαστής, λανθάνουσα μνήμη, διακοπές, assembly, DMA

Περιεχόμενα

Κεφάλαιο 1ο
Ιστορική αναδρομή του Vaadin

Ορισμός

Vaadin: είναι μια πλατφόρμα ανοιχτού κώδικα για την ανάπτυξη διαδικτυακών εφαρμογών . Η πλατφόρμα Vaadin περιλαμβάνει ένα σύνολο από web components, ένα framework της Java και ένα σύνολο εργαλείων και εφαρμογών. Το προϊόν της ναυαρχίδας, το Vaadin Flow (γνωστό και ως Vaadin Framework), επιτρέπει την υλοποίηση των διεπαφών χρήστη σε HTML5 χρησιμοποιώντας τη Γλώσσα Προγραμματισμού Java .

Ιστορία

Η ανάπτυξη ξεκίνησε για πρώτη φορά ως προσαρμογέας στην κορυφή του πλαισίου ιστού ανοιχτού κώδικα Millstone 3 που κυκλοφόρησε το έτος 2002. Εισήγαγε μια μηχανή επικοινωνίας που βασίζονταν σε Ajax . Κατά το 2006, η έννοια αυτή αναπτύχθηκε χωριστά ως εμπορικό προϊόν.

Στις αρχές του 2007 το όνομα του προϊόντος άλλαξε σε IT Mill Toolkit και η έκδοση 4 κυκλοφόρησε. Χρησιμοποίησε μια ιδιόκτητη υλοποίηση JavaScript Ajax για την απόδοση της πλευράς του πελάτη (client-side), γεγονός που καθιστούσε μάλλον περίπλοκο την υλοποίηση νέων widgets. Μέχρι το τέλος του έτους 2007 εγκαταλείφθηκε η ιδιόκτητη εφαρμογή από την πλευρά του πελάτη και η GWT ενσωματώθηκε στην κορυφή των συστατικών στοιχείων του διακομιστή. Ταυτόχρονα, η άδεια χρήσης του προϊόντος άλλαξε στην ανοιχτού κώδικα Apache License 2.0 . Η πρώτη απελευθέρωση του IT Mill Toolkit 5 έτοιμη για παραγωγή έγινε στις 4 Μαρτίου 2009, μετά από μία περίοδο δοκιμής για περισσότερο από ένα χρόνο.

Στις 11 Σεπτεμβρίου, 2008, ανακοινώθηκε δημόσια [1] [2] πως ο Michael Widenius - ο κύριος συντάκτης της αρχικής έκδοσης του MySQL –επένδυσε στην εταιρεία IT Mill ο οποίος είναι ο δημιουργός του Vaadin. Το μέγεθος της επένδυσης δεν γνωστοποιήθηκε.

Στις 20 Μαΐου 2009, το IT Mill Toolkit μετονομάστηκε σε Vaadin Framework. Εκτός από την αλλαγή ονόματος, ξεκίνησε μια προ-έκδοση της έκδοσης 6 μαζί με έναν δικτυακό τόπο της κοινότητας. Αργότερα, η IT Mill Ltd, η εταιρεία πίσω από την ανοικτή πηγή Vaadin Framework, άλλαξε το όνομά της σε Vaadin Ltd.

Στις 30 Μαρτίου 2010 προστέθηκαν add-on Components στον πυρήνα του Vaadin Framework και τα οποία μπορούσαν να χρησιμοποιηθούν ελεύθερα ή για εμπορικούς σκοπούς.

Στις 22 Φεβρουαρίου 2017, κυκλοφόρησε η έκδοση Vaadin 8. Η έκδοση αυτή περιλάμβανε βελτιώσεις στην αρχικοποίηση των δεδομένων σύνδεσης API, σύγχρονα components της Java, και στοιχεία που διαχειρίζονταν πιο αποτελεσματικά την μνήμη και τη CPU.

Στις 25 Ιουνίου 2018, η έκδοση Vaadin 10 κυκλοφόρησε. Τα components του Vaadin 10 μπορούσαν να χρησιμοποιηθούν από οποιαδήποτε τεχνολογία η οποία υποστήριζε Web Components.

Στις 5 Σεπτεμβρίου 2018, κυκλοφόρησε το Vaadin 11 έχοντας μερικά νέα στοιχεία και τα διαγράμματα Vaadin 6.1.

[1] https://en.wikipedia.org/wiki/Vaadin

[2] https://vaadin.com/blog/vaadin-7-application-architecture

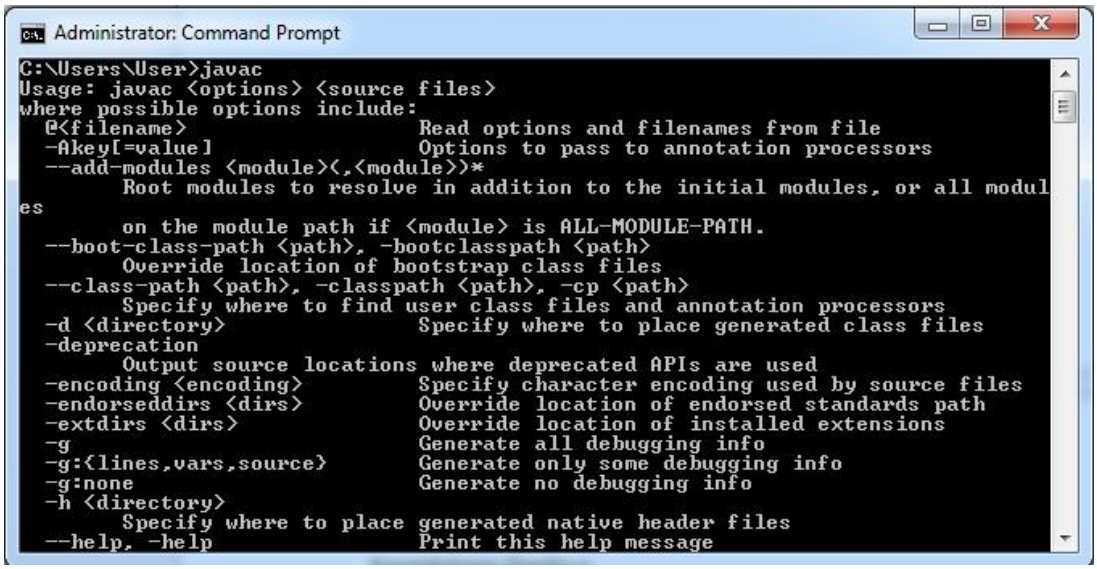
Κεφάλαιο 2ο

Εγκατάσταση Λογισμικών

Σκοπός του συγκεκριμένου κεφαλαίου είναι να δείξουμε την διαδικασία εγκατάστασης όλων των απαραίτητων πακέτων που χρειάζονται για να αναπτύξουμε ένα project σε vaadin.

1. Εγκατάσταση του JDK SE (Java Development Kit Standard Edition)

Πριν προχωρήσουμε στην εγκατάσταση του εν λόγω πακέτου θα πρέπει να ελέγξουμε εάν στο σύστημα μας υπάρχει εγκατεστημένο το πακέτο και εάν ναι θα πρέπει να ελέγξουμε την έκδοση του. Γι' αυτό το λόγο ανοίγουμε το Command Prompt (εάν χρησιμοποιούμε windows) και πληκτρολογούμε την εντολή javac:



(Εικόνα 1)

Το αποτέλεσμα της εντολής javac (Εικόνα 1) είναι να μας βγάλει όλες τις εντολές που είναι διαθέσιμες και χρησιμοποιούνται στο συγκεκριμένο πακέτο.

Σε περίπτωση που δεν έχουμε εγκαταστήσει το JDK ή δεν έχει παραμετροποιηθεί σωστά θα πρέπει, εκτελώντας την παραπάνω εντολή, να μας εμφανίζει το μήνυμα «javac is not recognized as internal or external command, operable program or batch file».

Το πρώτο βήμα είναι να κατεβάσουμε το JDK Tool kit από τον διαδικτυακό τόπο της Oracle (https://www.oracle.com/technetwork/java/javase/downloads/index.html) και να

αρχίσουμε την εγκατάσταση. Η σημερινή έκδοση του πακέτου είναι η Java SE 11.0.1, όμως εμείς θα δείξουμε την εγκατάσταση της πλατφόρμας με έκδοση 9.0.1. Η διαδικασία είναι παρόμοια και σε άλλες εκδόσεις.

(Εικόνα 2)

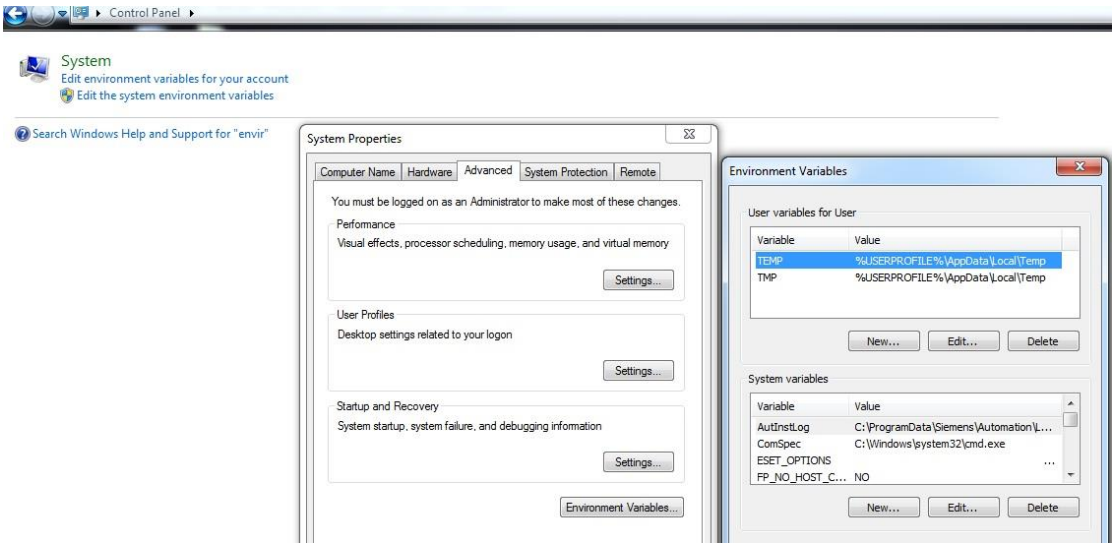
(Εικόνα 3)

(Εικόνα 4)

(Εικόνα 5)

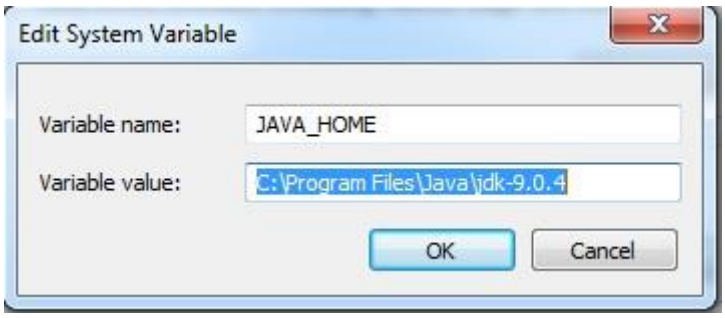
Μετά την επιτυχή εγκατάσταση του JDK πακέτου (βλ. Εικόνα 2, Εικόνα 3, Εικόνα 4, Εικόνα 5) πηγαίνουμε ξανά στο command prompt και ξαναπληκτρολογούμε την εντολή «javac», και παρατηρούμε ότι μας βγάζει το ίδιο μήνυμα λάθους «javac is not recognized as internal or external command, operable program or batch file». Αυτή τη φορά θα πρέπει να προχωρήσουμε στην παραμετροποίηση του Συστήματος μας.

Πηγαίνουμε στο Control Panel->Environment->Edit the system environment variables. Στο παράθυρο System properties επιλέγουμε την καρτέλα «advanced» και την επιλογή “Environment variables” η οποία μας εμφανίζει ένα καινούριο παράθυρο (βλ. εικ 6).



(Εικόνα 6)

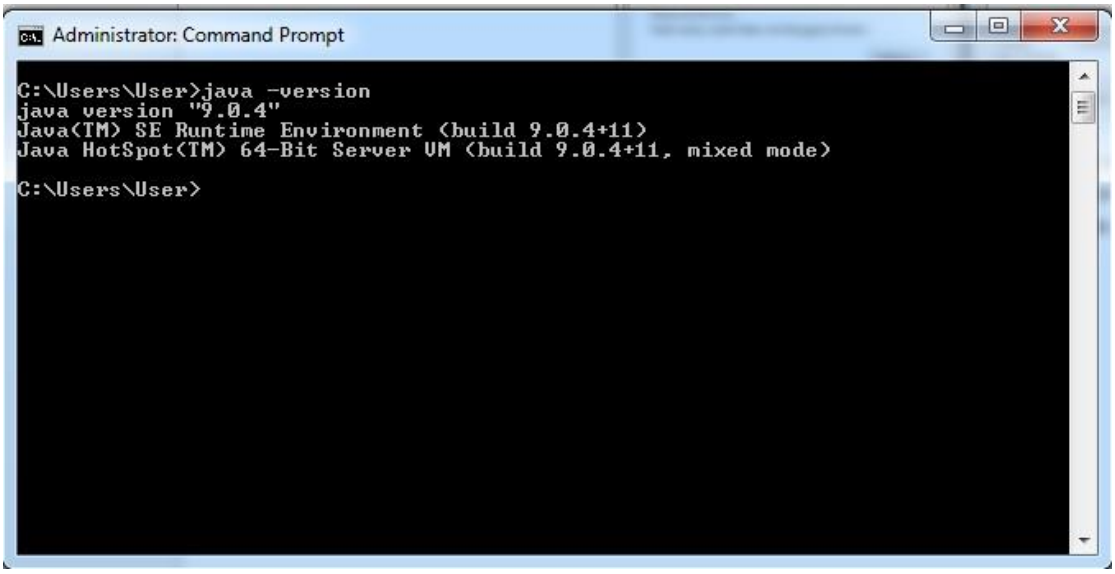
Στο παράθυρο επιλέγουμε την πλήκτρο New για να δημιουργήσουμε μία καινούρια μεταβλητή η οποία θα συνδεθεί με το JDK. Έτσι λοιπόν δίνουμε ένα όνομα στην μεταβλητή, που στην προκυμμένη περίπτωση την έχουμε ονομάσει «JAVA_HOME», και ως τιμή εισάγουμε το path στο οποίο έχουμε εγκαταστήσει το JDK (βλ. Εικόνα 7).



(Εικόνα 7)

Επίσης στην μεταβλητή path προσθέτουμε ως τιμή τον φάκελο bin του πακέτου JDK C:\Program Files\Java\jdk-9.0.4\bin. Η μεταβλητή path περιέχει τιμές οι οποίες χωρίζονται μεταξύ τους με το ελληνικό ερωτηματικό.

Μετά την ολοκλήρωση των παραμέτρων πηγαίνουμε ξανά στο command prompt και πληκτρολογούμε την εντολή "javac" και παρατηρούμε, αυτή τη φορά, ότι δεν εμφανίζει το μήνυμα λάθους. Πληκτρολογούμε την εντολή «java –version» και παίρνουμε τα ακόλουθα αποτελέσματα (βλ. Εικόνα 8):

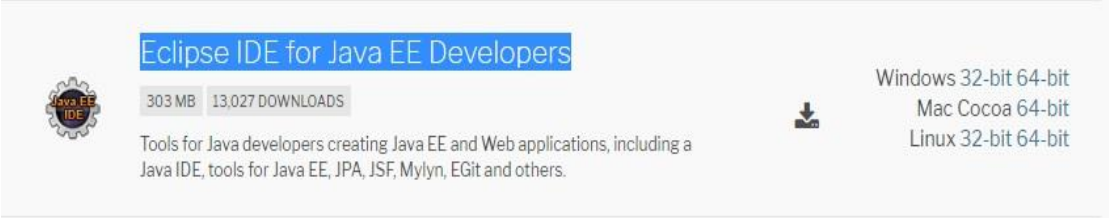


(Εικόνα 8)

2. Εγκατάσταση ενός IDE (Integrated Development Environment)

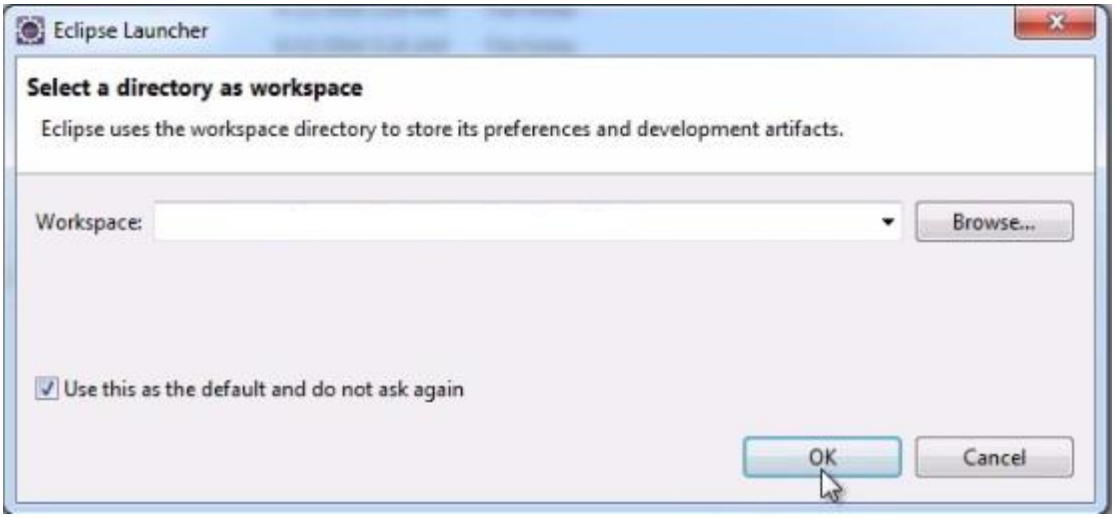
Για την ανάπτυξη μιας εφαρμογής σε Java το μόνο που χρειάζεται είναι το JDK πακέτο. Παρόλα αυτά υπάρχουν μερικά software τα οποία σε διευκολύνουν να αναπτύξεις και να πραγματοποιήσεις τις εφαρμογές πολή πιο εύκολα. Ένα απ' αυτά, που θα χρησιμοποιήσουμε, είναι το Eclipse Oxygen το οποίο ανήκει στο πακέτο «Eclipse IDE for Java EE Developers» και είναι κατάλληλο για ανάπτυξη διαδικτυακών εφαρμογών.

Έτσι λοιπόν πηγαίνουμε στο διαδικτυακό τόπο του Eclipse (https://www.eclipse.org/downloads/packages/release/oxygen/m2) και κατεβάζουμε το πακέτο «Eclipse IDE for Java EE Developers» (βλ. Εικόνα 9).



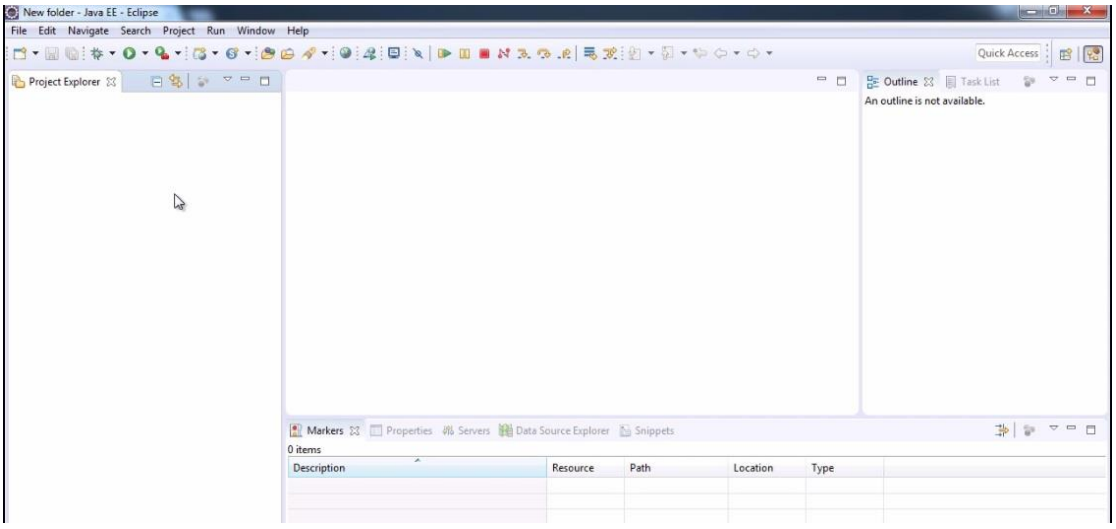
(Εικόνα 9)

Μετά την αποσυμπίεση του αρχείου και την εκτέλεση του exe αρχείου το πρώτο πράγμα που μας ζητάει είναι να του ορίσουμε ένα path στο οποίο θα αποθηκεύονται όλα τα projects (βλ. Εικόνα 10).



(Εικόνα 10)

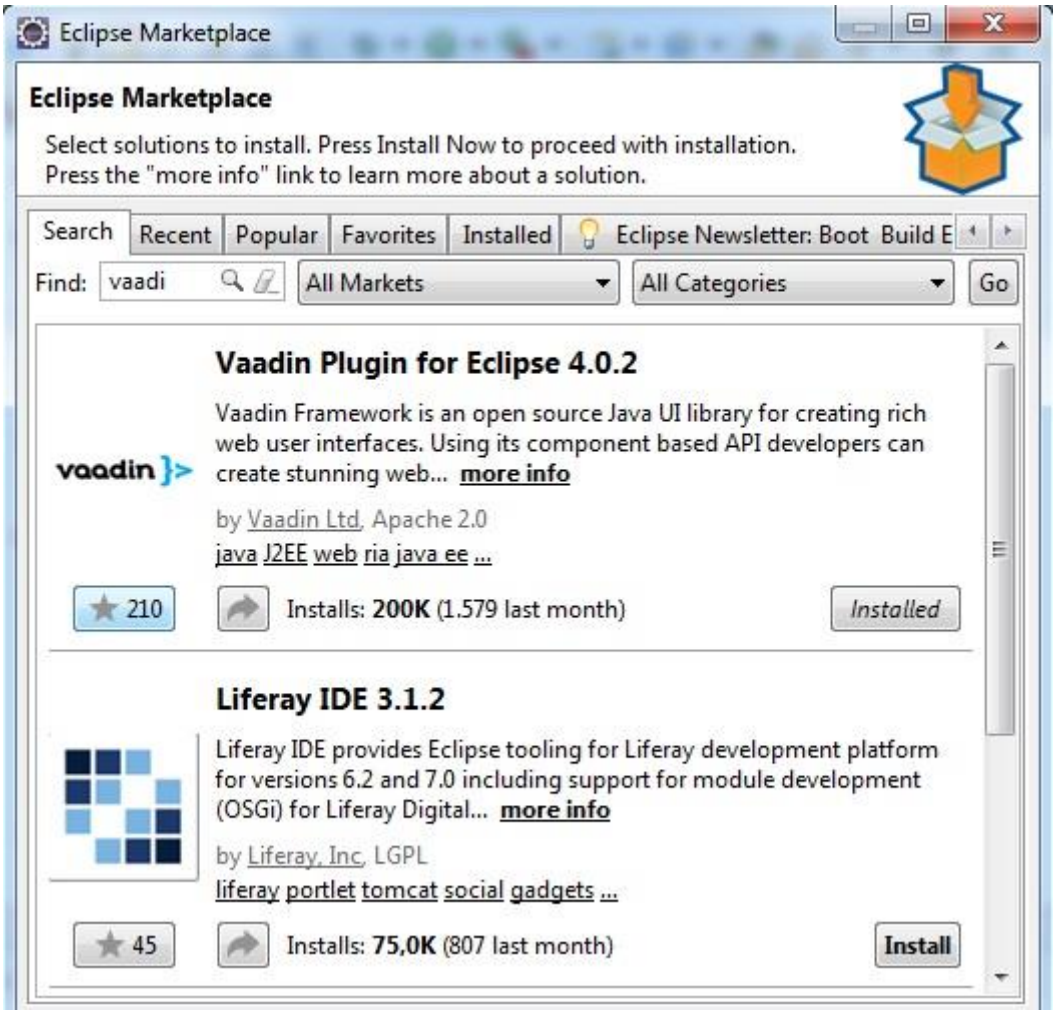
Μετά την επιτυχή εγκατάσταση του eclipse το κεντρικό παράθυρο φαίνεται στην παρακάτω Εικόνα 11.



(Εικόνα 11)

3. Εγκατάσταση του framework Vaadin στο περιβάλλον eclipse

Για να γίνει η εγκατάσταση του framework vaadin ή οποιουδήποτε άλλου framework απλά πηγαίνουμε στο μενού Help->Eclipse Marketplace. Στο καινούριο παράθυρο που μας εμφανίζεται πληκτρολογούμε τη λέξη κλειδί στο πεδίο αναζήτησης και πιέζουμε το πλήκτρο install (βλ. Εικόνα 12).



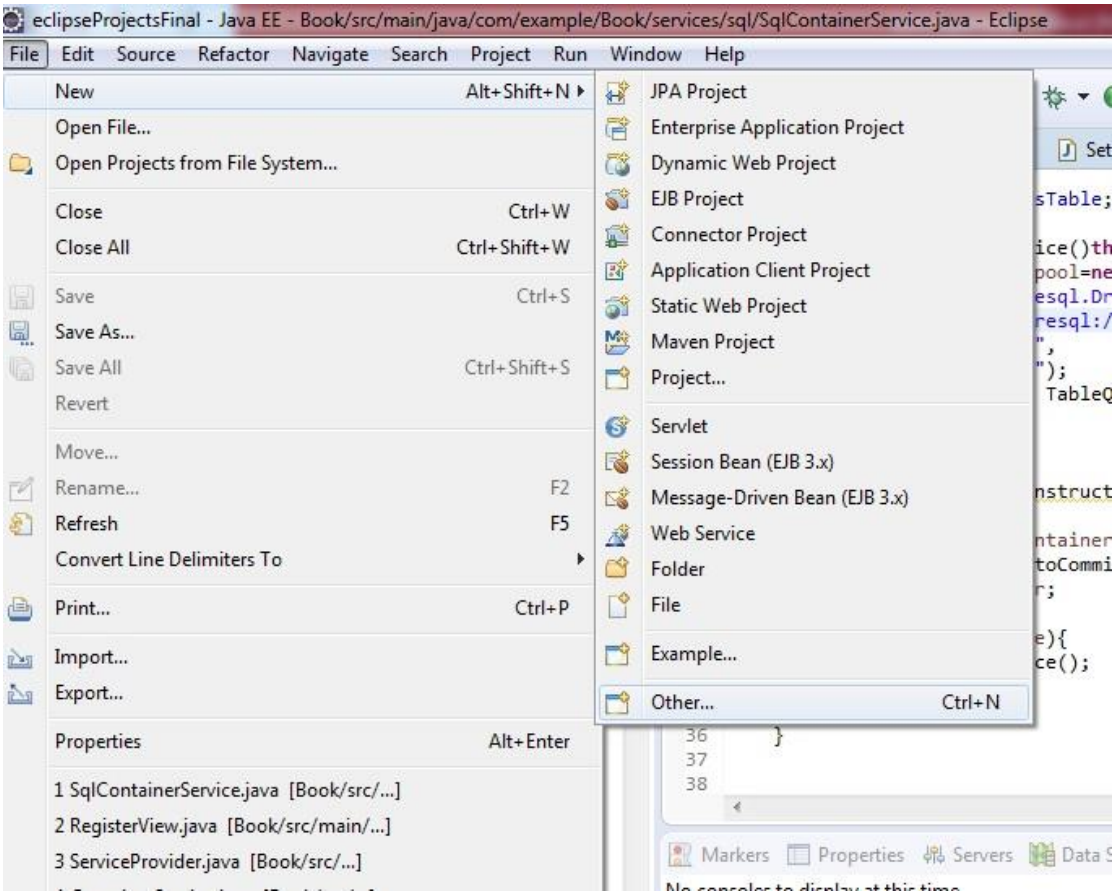
(Εικόνα 12)

Στη συνέχεια ακολουθούμε τις οδηγίες και όταν ολοκληρωθεί η διαδικασία κάνουμε επανεκκίνηση στο πρόγραμμα Eclipse.

Θα πρέπει να επισημάνουμε ότι μπορούμε να εγκαταστήσουμε περισσότερα από ένα frameworks στο περιβάλλον eclipse.

4. Δημιουργία του πρώτου Project

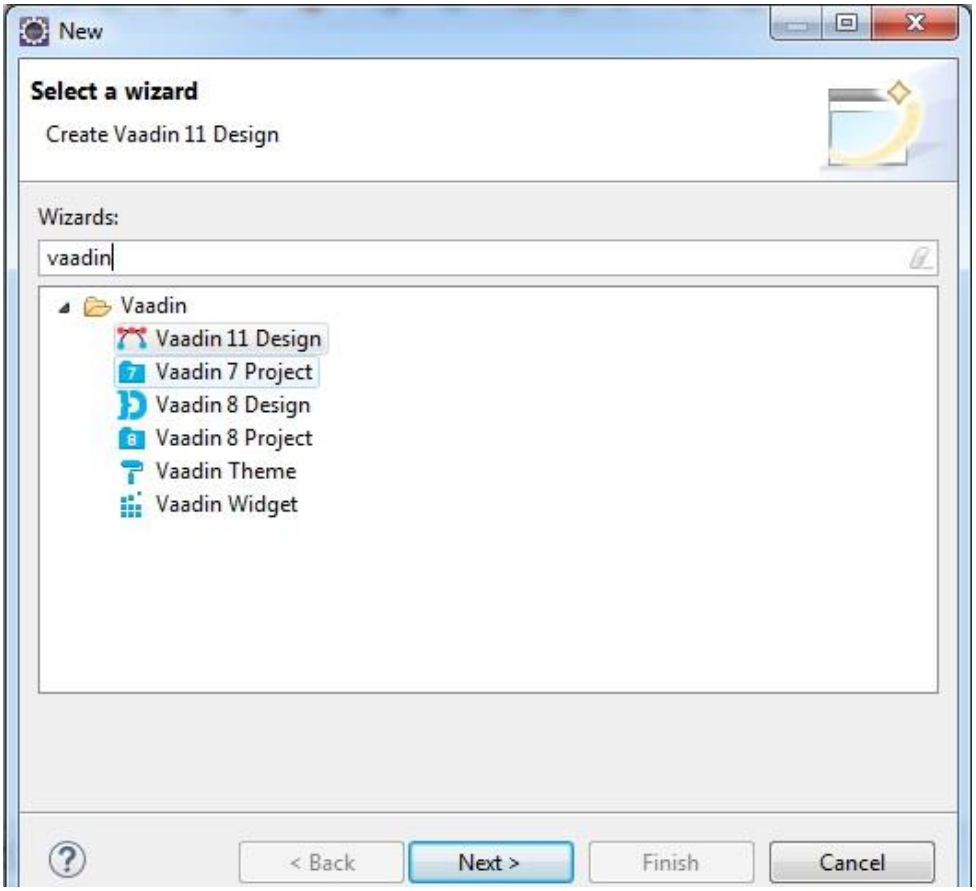
Για να δημιουργήσουμε ένα project, χρησιμοποιώντας το Vaadin, πηγαίνουμε στο μενού File->New->other (βλ. Εικόνα 13).



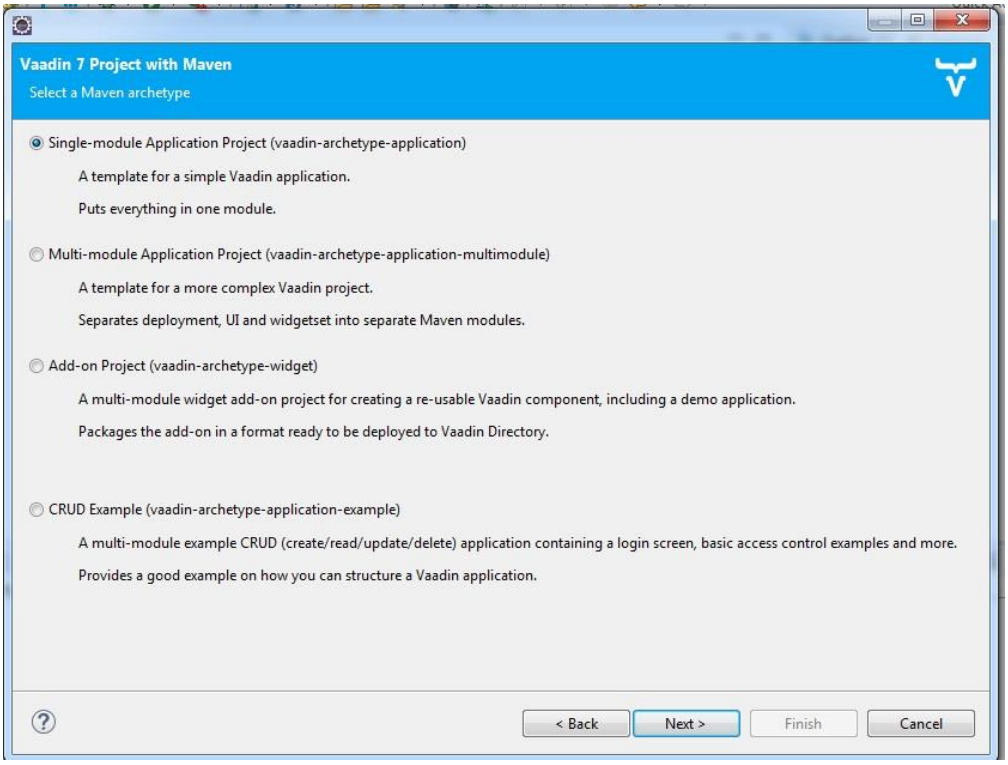
(Εικόνα 13)

Στο παράθυρο που εμφανίζεται επιλέγουμε την έκδοση του framework που θα χρησιμοποιήσουμε (βλ. Εικόνα 14). Εμείς για την ανάπτυξη της εφαρμογής θα χρησιμοποιήσουμε την έκδοση 7. Μπορούμε να εγκαταστήσουμε και να χρησιμοποιήσουμε περισσότερες από μία εκδόσεις του Vaadin.

Στη συνέχεια πιέζουμε το πλήκτρο επόμενο (βλ. Εικόνα 14).

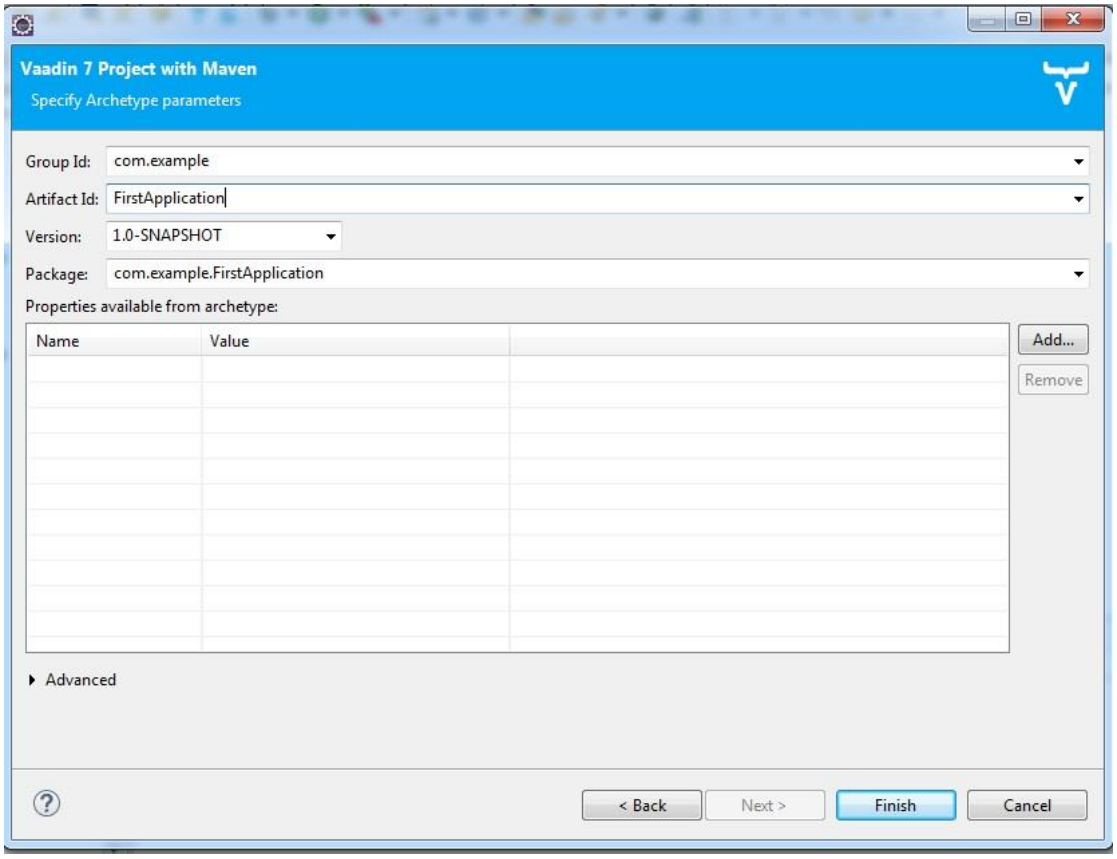


(Εικόνα 14)



(Εικόνα 15)

Ανάλογα με την πολυπλοκότητα του Project επιλέγουμε μία από τις 4 διαθέσιμες επιλογές (βλ. Εικόνα 15). Για την δημιουργία της εφαρμογής μας αρκεί να επιλέξουμε την επιλογή “Single Module Application Project”.



(Εικόνα 16)

Στο τελευταίο παράθυρο προσδιορίζουμε το όνομα της εφαρμογής μας (Artifact Id), το όνομα του πακέτου και στη συνέχεια πληκτρολογούμε το πλήκτρο Finish για να δημιουργηθεί το Project (βλ. Εικόνα 16).

Κεφάλαιο 3ο

Παρουσίαση των βασικών Components του Vaadin

1) Δημιουργία & επεξήγηση του πρώτου Project

Σε κάθε project που δημιουργείται, η διαδικασία του οποίου εξηγείται αναλυτικά στο προηγούμενο κεφάλαιο, παράγεται μία κεντρική κλάση με όνομα συνήθως το "MyUI". Το όνομα της κλάσης μπορεί να αλλάξει κατά τη διαδικασία δημιουργίας του Project.

Η κλάση αυτή κληρονομεί την κλάση του συστήματος UI το οποίο είναι το πρώτο component στην Ιεραρχία του vaadin. Στο ακόλουθο παράδειγμα η Κεντρική κλάση εμφανίζεται με κόκκινο χρώμα.

```
package com.example.myapplication;
import javax.servlet.annotation.WebServlet;
import com.vaadin.annotations.Theme;
import com.vaadin.annotations.VaadinServletConfiguration;
import com.vaadin.server.VaadinRequest;
import com.vaadin.server.VaadinServlet;
import com.vaadin.ui.Button;
import com.vaadin.ui.Label;
import com.vaadin.ui.TextField;
import com.vaadin.ui.UI;
import com.vaadin.ui.VerticalLayout;
@Theme("mytheme")
public class MyUI extends UI {
    @Override
    protected void init(VaadinRequest vaadinRequest) {

    }

    @WebServlet(urlPatterns = "/*", name = "MyUIServlet", asyncSupported = true)
    @VaadinServletConfiguration(ui = MyUI.class, productionMode = false)
    public static class MyUIServlet extends VaadinServlet {

    }

}
```

Μέσα σε κάθε "MyUI" κλάση υπάρχει μία και μοναδική μέθοδο void με το όνομα init και όρισμα το component VaadinRequest. Η χρησιμότητα αυτής της μεθόδου είναι να καλείται κάθε φορά που ο χρήστης δίνει του URL της εφαρμογής. Άρα όταν πληκτρολογούμε το URL της εφαρμογής που θέλουμε να εμφανίσουμε στο φυλλομετρητή μας (Browser) εκτελείτε η συγκεκριμένη μέθοδος. Η μέθοδος αυτή στην ουσία περιέχει όλο τον κώδικά μας και στο παραπάνω παράδειγμα απεικονίζεται με μπλε χρώμα.

Στις πρώτες γραμμές του παραπάνω παραδείγματος, με εξαίρεση την πρώτη γραμμή που περιγράφει το πακέτο στο οποίο ανήκει η εφαρμογή μας, απεικονίζονται όλες οι βιβλιοθήκες που εισάγονται για να αναγνωριστούν όλα τα components του Vaadin.

2) Vertical Layout

Το δεύτερο στην ιεραρχία Component στο framework Vaadin είναι το layout το οποίο είναι ένα αόρατο πλαίσιο, μέσα στο οποίο θα τοποθετούνται όλα τα άλλα components και συνήθως είναι η πρώτη εντολή που γράφουμε στη μέθοδο init.

Για παράδειγμα η εντολή **VerticalLayout layout = new VerticalLayout();** δημιουργεί ένα αντικείμενο με το όνομα layout τύπου VerticalLayout και σκοπό έχει τα επόμενα components να είναι στοιχισμένα κάθετα με τη σειρά που θα τοποθετηθούν.

```
public class MyUI extends UI {

    @Override
    protected void init(VaadinRequest VaadinRequest) {
    VerticalLayout layout=new VerticalLayout();

    setContent(layout);

}
```

Η εντολή **set Content(layout);** θέτει το layout που δημιουργήσαμε στην κλάση «MyUI». Με άλλα λόγια καλώντας την συγκεκριμένη εντολή λέμε ότι το αντικείμενο layout είναι το «παιδί» της κλάσης «MyUI».

3) Buttons

Το Button είναι ένα component το οποίο εμφανίζεται στον Browser και όταν το πατήσουμε, με το δείκτη του ποντικιού, κάνει μία ενέργεια που του έχουμε ορίσει εμείς. Για παράδειγμα θα προσπαθήσουμε να δημιουργήσουμε μία εφαρμογή που θα έχει ένα button και κάθε φορά που κάνουμε κλικ επάνω του θα μας τυπώνει ένα default μήνυμα.

Άρα μέσα στο αντικείμενο layout που ορίσαμε προηγουμένως δημιουργούμε το ακόλουθο αντικείμενο Button button=new Button("Click me"). Έτσι το πρόγραμμα έχει ως εξής:

```
protected void init(VaadinRequest VaadinRequest) {
    final Vertical Layout layout = new VerticalLayout();


    Button button = new Button("Click Me");
    layout.addComponents(button);

    setContent(layout);
}
```

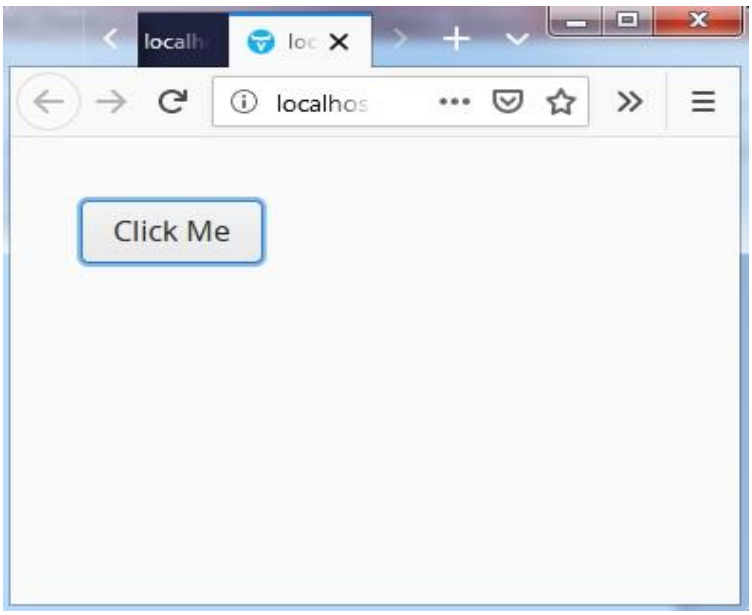
Στη συνέχεια προσθέτουμε το component Button στο layout component για να μπορέσει να εμφανιστεί στο φυλλομετρητή.

Μέχρι στιγμής δημιουργήσαμε τον κώδικα για να εμφανιστεί στον Browser μόνο ένα κουμπί το οποίο θα περιέχει μέσα του το κείμενο "Click me" (βλ. Εικόνα 17).

Για να εκτελέσει μία ενέργεια θα πρέπει να καλέσουμε την μέθοδο addClickListener η οποία θα παρακολουθεί το συγκεκριμένο πλήκτρο και κάθε φορά που θα κάνουμε click επάνω του θα εκτελείται ένα κομμάτι κώδικα. Άρα ο κώδικας γίνεται:

```
public class MyUI extends UI {

    @Override
    protected void init(VaadinRequest VaadinRequest) {
    final VerticalLayout layout = new VerticalLayout();


    Button button = new Button("Click Me");
    layout.addComponents(button);
    button.addClickListener (new Button.ClickListener() {
```

```
                @Override
        public void button Click (ClickEvent event) {
                        // TODO Auto-generated method stub
Notification.show("You press the button"+ " '"+
event.getButton().getCaption().toString()+"'") ;
                }
            });
setContent(layout);
  }
```



(Εικόνα 17)

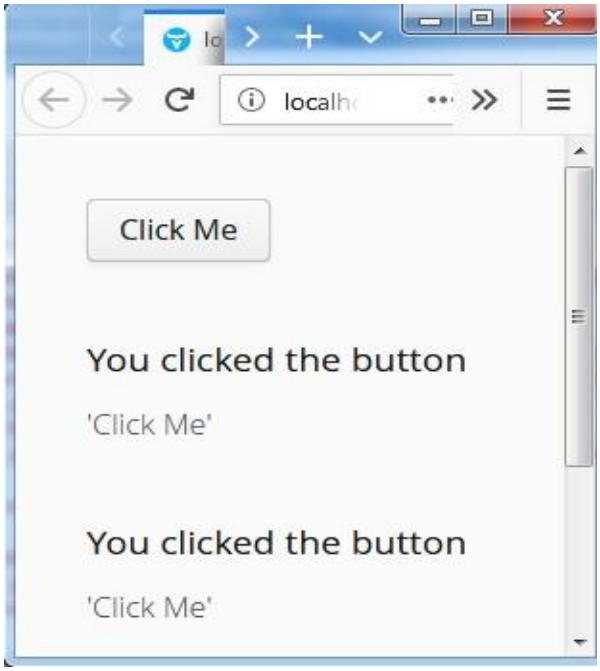
4) Labels

Labels είναι τα components που χρησιμοποιούνται από το Vaadin για να τυπωθεί ένα κείμενο στον Browser. Το μόνο που χρειάζεται είναι να δημιουργηθεί ένα αντικείμενο τύπου Label και στη συνέχεια να ενταχθεί στο layout το οποίο έχουμε δημιουργήσει.

```
  public class MyUI extends UI {

      @Override
      protected void init(VaadinRequest vaadinRequest) {
    final VerticalLayout layout = new VerticalLayout();
    Button button = new Button("Click Me");
```

```
        layout.addComponents(button);

        button.addClickListener(new Button.ClickListener() {

                        @Override
                        public void buttonClick(ClickEvent event) {
                                // TODO Auto-generated method stub
Label label=new Label("<b><h3>You clicked the button
</b></h3>"+"'"+event.getButton().getCaption().toString()+"'");
            label.setContentMode(ContentMode.HTML);
layout.addComponent(label);                                }
                    });
                setContent(layout);
    }
```

Άρα στο προηγούμενο πρόγραμμα έχουμε εισάγει μία ετικέτα και κάθε φορά που κάνουμε click στο button τυπώνουμε το κείμενο που έχουμε ορίσει, στην προκειμένη περίπτωση θα τυπώσει το κείμενο «You clicked the button ‘Click Me’», το ένα κάτω από το άλλο. Τα αποτελέσματα φαίνονται στην παρακάτω Εικόνα 18.



(Εικόνα 18)

Η εντολή **"label.set Content Mode (ContentMode.HTML);"** μας λέει οτι το κείμενο που θα συμπεριληφθεί στην ετικέτα μπορεί να δεχθεί επιπλέον και εντολές της HTML για την μορφοποίηση του.

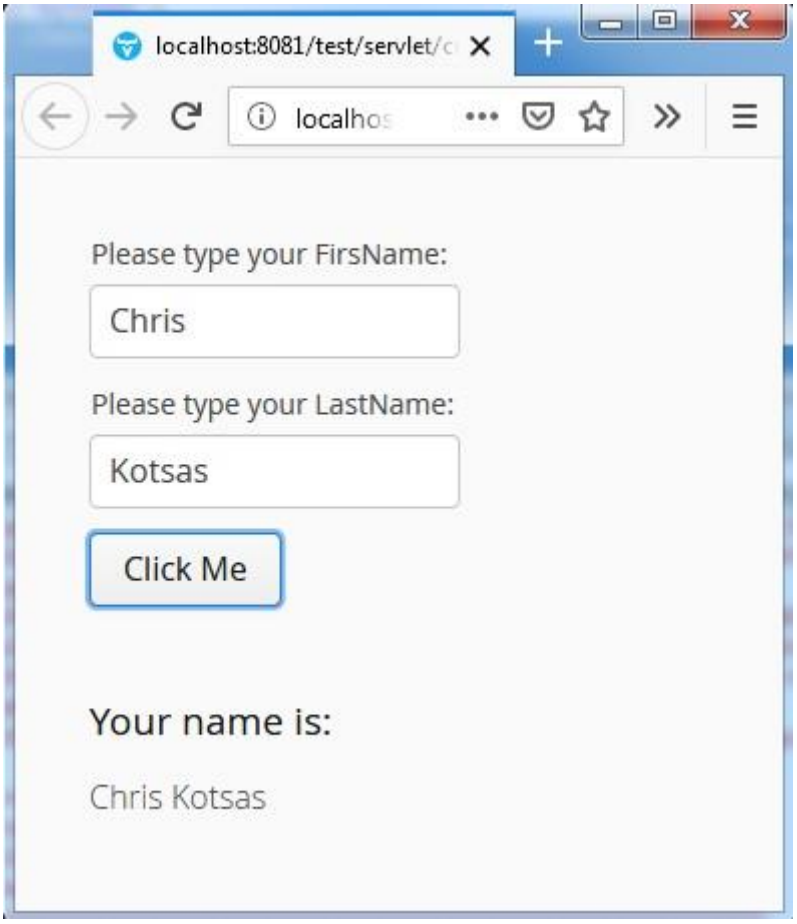
5) TextField

Το συγκεκριμένο component είναι ένα πεδίο στο οποίο μπορούμε να πληκτρολογήσουμε ένα κείμενο, θεωρείται σαν μία μονάδα εισόδου στην οποία μπορούμε να εισάγουμε πληροφορίες.

```
public class MyUI extends UI {

   @Override
   protected void init(VaadinRequest vaadinRequest) {
      final VerticalLayout layout = new VerticalLayout();

     TextField firstname=new TextField("Please type your FirstName:");
     TextField lastname=new TextField("Please type your LastName:");

      Button button = new Button("Click Me");

      layout.addComponent(firstname);
      layout.addComponent(lastname);
      layout.addComponents(button);

      button.addClickListener(new Button.ClickListener() {

                        @Override
         public void buttonClick(ClickEvent event) {
                              // TODO Auto-generated method stub
         Label label=new Label("<b><h3>Your name is:</b></h3>"+
firstname.getValue().toString()+" "+lastname.getValue().toString());
         label.setContentMode(ContentMode.HTML);
         layout.addComponent(label);
                        }
               });
```

```
        setContent(layout);
    }
```

Στο παραπάνω παράδειγμα έχουμε εισάγει δύο text fields στα οποία ζητάμε από το χρήστη να πληκτρολογήσει το όνομα και το επώνυμό του και με το πάτημα του πλήκτρου “Click Me” τα τυπώνει σε μία ετικέτα (Label). Τα αποτελέσματα φαίνονται στην ακόλουθη Εικόνα 19:



(Εικόνα 19)

Κεφάλαιο 4ο

Components εισόδου & φόρμες

1) COMBOBOXES

Το ComboBox είναι ένα component εισόδου που επιτρέπει στον χρήστη να επιλέξει μία επιλογή από μία λίστα επιλογών.

Για να μπορέσουμε να εισάγουμε τις επιλογές που θα εμφανίζονται στο ComboBox, αυτό γίνεται με δύο τρόπους:

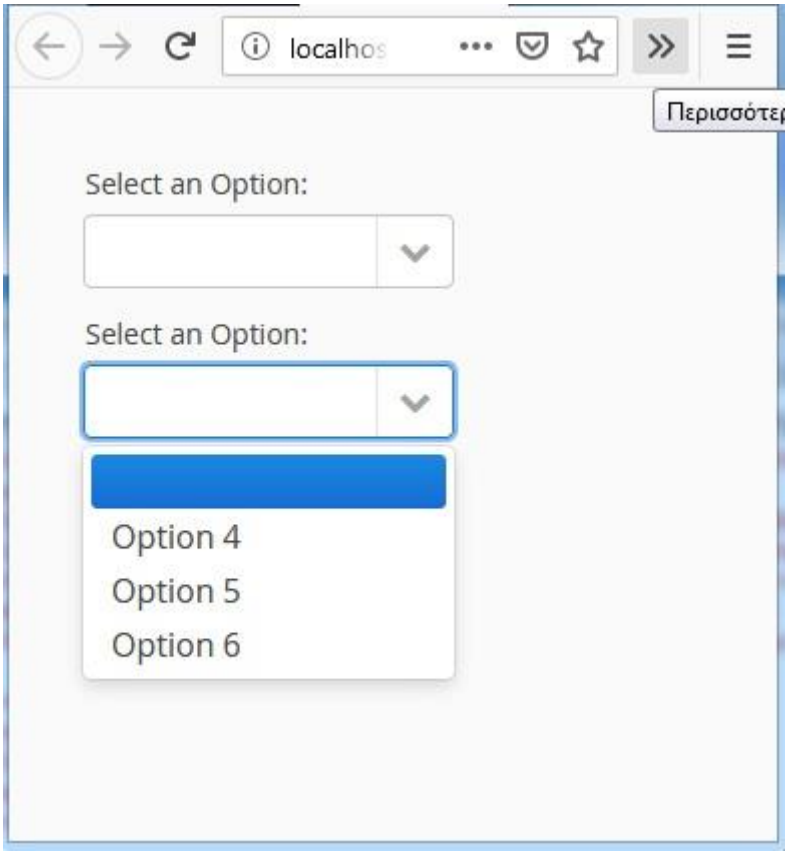
a. Με την εντολή set Items ("Επιλογή 1", "Επιλογή 2", "Επιλογή 3".....);

b. Με τη δημιουργία μιας Array List η οποία θα περιέχει όλες τις επιλογές που θέλουμε να εμφανίζεται στο ComboBox. Πιο αναλυτικά δημιουργούμε ένα αντικείμενο τύπου ArrayList (ArrayList<String> options=new ArrayList<>();) και στη συνέχεια με την εντολή add γεμίζουμε την λίστα.

Και οι δύο τρόποι εμφανίζονται στο ακόλουθο πρόγραμμα:

```
public class MyUI extends UI {

    @Override
    protected void init(VaadinRequest vaadinRequest) {
        final VerticalLayout layout = new VerticalLayout();

        ComboBox combo1=new ComboBox("Select an Option: ");
        combo1.setItems("Option 1", "Option 2", "Option 3");

        ArrayList<String> options=new ArrayList<>();
        options.add("Option 4");
        options.add("Option 5");
        options.add("Option 6");
        ComboBox combo2=new ComboBox("Select an Option: ",options);

        layout.addComponents(combo1);
        layout.addComponent(combo2);

        setContent(layout);
            }
```

Τα αποτελέσματα φαίνονται στην παρακάτω Εικόνα 20:



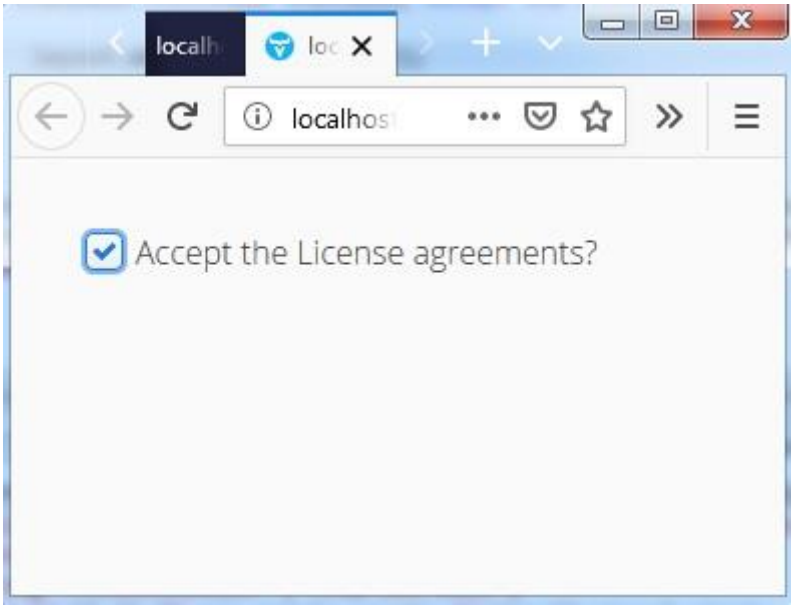
(Εικόνα 20)

2) CHECKBOX

Το CheckBox είναι ευρέως διαδεδομένο και γνωστό σε όλους. Όλοι το έχουμε δει κυρίως κατά την εγκατάσταση ενός Software που μας εμφανίζεται μία επιλογή και μας ενημερώνει εάν δεχόμαστε τους όρους εγκατάστασης του προγράμματος. Για να συνεχίσουμε την εγκατάσταση θα πρέπει οπωσδήποτε να «τσεκάρουμε» το checkbox (βλ. Εικόνα 21).

```
public class MyUI extends UI {

    @Override
    protected void init(VaadinRequest vaadinRequest) {
        final VerticalLayout layout = new VerticalLayout();

        CheckBox check=new CheckBox("Accept the License agreements?");
```

```
layout.addComponents(check);

setContent(layout);
        }
```



(Εικόνα 21)

3) CHECK BOX GROUP

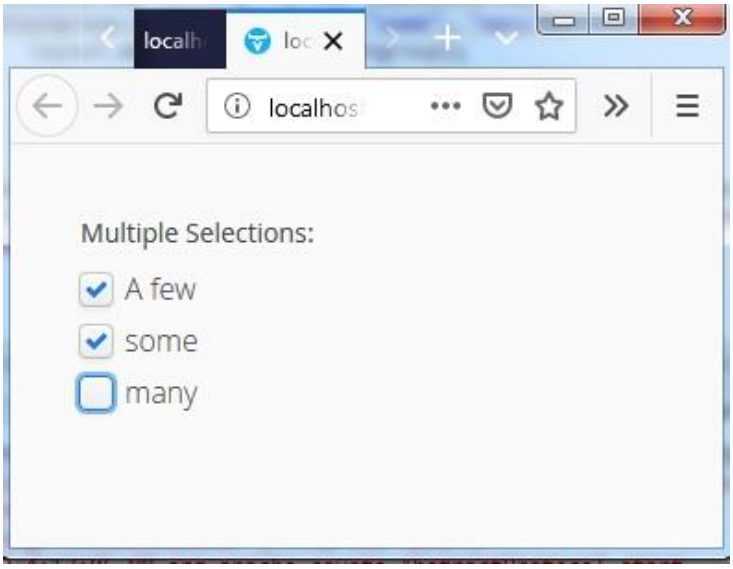
Το Option Group μοιάζει πολύ με το ComboBox component με τη μόνη διαφορά ότι οι επιλογές δεν εμφανίζονται σε μία drop down list, αλλά είναι ορατές στον browser και μπορώ να «τσεκάρω» περισσότερες από μία επιλογές.

```
public class MyUI extends UI {

@Override
protected void init(VaadinRequest vaadinRequest) {
final VerticalLayout layout = new VerticalLayout();

CheckBoxGroup optiongroup=new CheckBoxGroup("Multiple Selections:");
optiongroup.setItems("A few", "some", "many");
layout.addComponents(optiongroup);

setContent(layout);
```

```
}
```

Οι τρόποι με τους οποίους θέτω τις επιλογές που θα εμφανίζονται είναι ίδιοι με αυτές που έχουμε αναφέρει στο componenet ComboBox.

Στην ακόλουθη Εικόνα 22 εμφανίζεται ένα CheckBoxGroup με τις επιλογές που του έχουμε θέσει:



(Εικόνα 22)

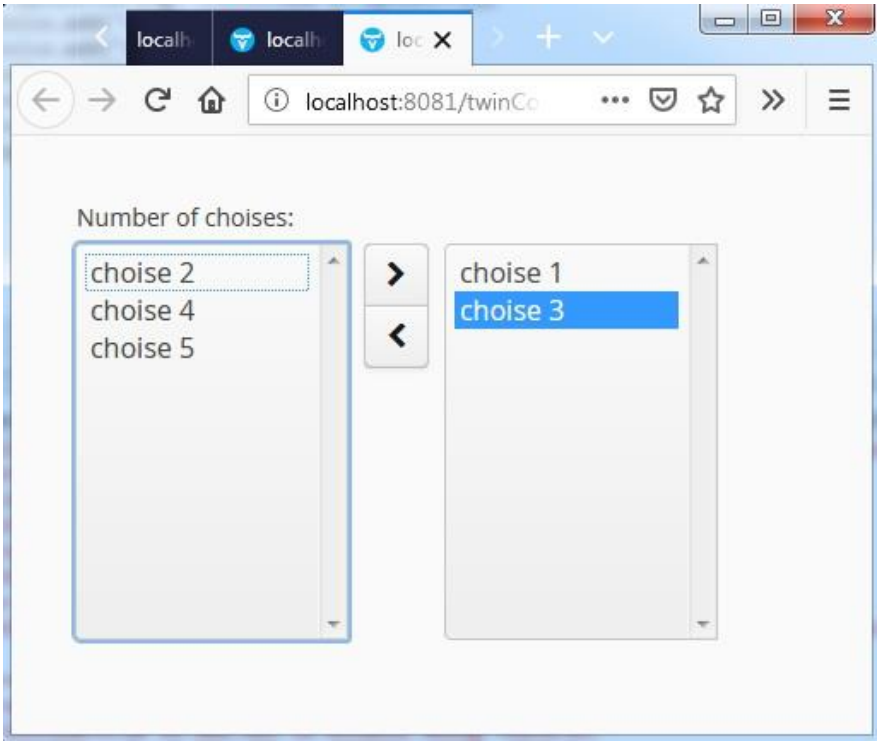
4) <u>TWIN COLUMN SELECT</u>

Αντί για το CheckBoxGroup μπορούμε να χρησιμοποιήσουμε το component TwinColSelect το οποίο εμφανίζει δύο στήλες και μπορούμε μία ή περισσότερες επιλογές να τις μεταφέρουμε στη δεύτερη στήλη (βλ. Εικόνα 23).

```
public class MyUI extends UI {


    @Override
    protected void init(VaadinRequest vaadinRequest) {
        final VerticalLayout layout = new VerticalLayout();

    ArrayList<String> choise=new ArrayList<>();
    choise.add("choise 1");
    choise.add("choise 2");
    choise.add("choise 3");
```

```
choise.add("choise 4");
choise.add("choise 5");
TwinColSelect list=new TwinColSelect("Number of choises:", choise);

   layout.addComponents(list);

   setContent(layout);
                }
```

Οι επιλογές που θα εμφανίζονται στις στήλες εισάγονται με την εντολή add.

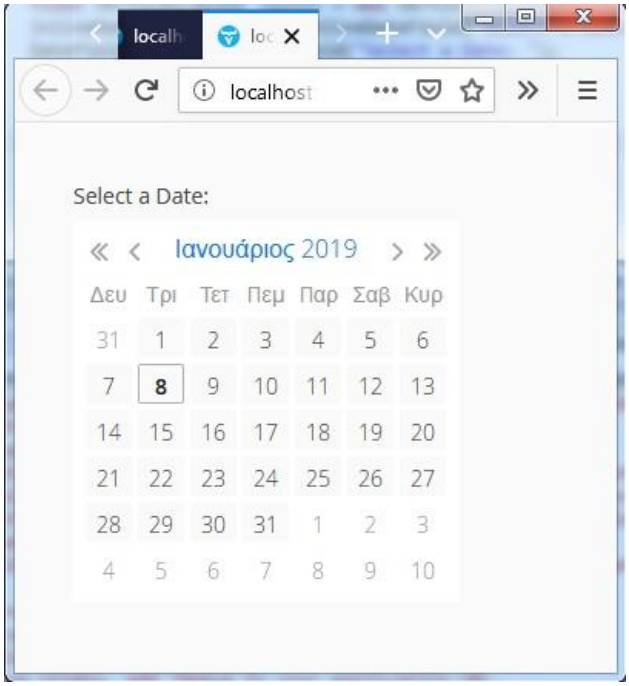


(Εικόνα 23)

5) <u>DATE</u>

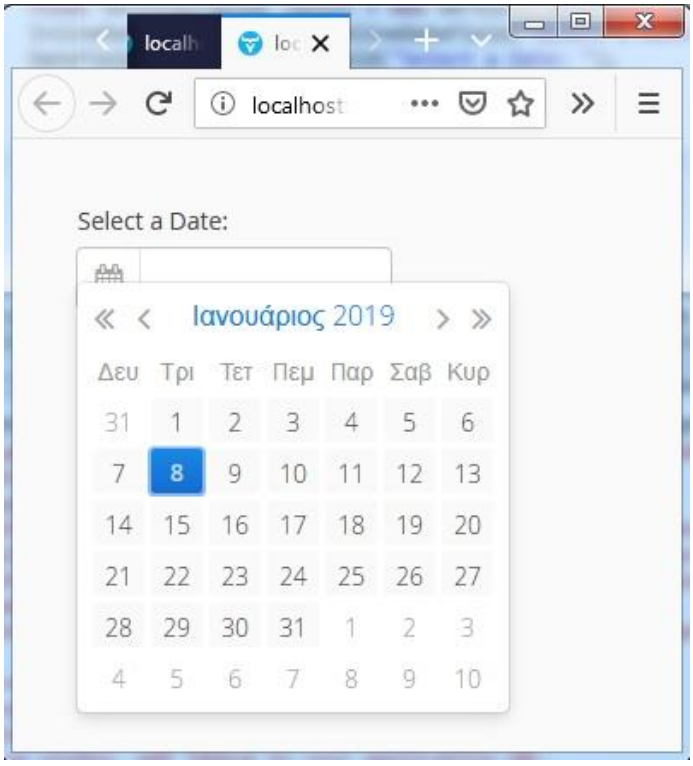
Το framework Vaadin προσφέρει ένα πολύ καλό Date Component με αρκετές παραλλαγές.

Εάν θέλουμε να εμφανίσουμε ένα ημερολόγιο τυπωμένο επάνω στο Browser δεν έχουμε παρά να δημιουργήσουμε ένα αντικείμενο τύπου InlineDateField (βλ. Εικόνα 24).



(Εικόνα 24)

Εάν όμως θέλουμε να εμφανίζεται ένα πεδίο στο οποίο, είτε εισάγουμε χειροκίνητα την ημερομηνία, είτε επιλέγουμε την ημερομηνία από το ημερολόγιο που εμφανίζεται, δημιουργούμε ένα αντικείμενο τύπου DateField (βλ. Εικόνα 25).



(Εικόνα 25)

Ο κώδικας και για τις δύο παραπάνω περιπτώσεις είναι ο ακόλουθος:

```
public class MyUI extends UI {


 @Override
protected void init(VaadinRequest vaadinRequest) {
final VerticalLayout layout = new VerticalLayout();
InlineDateField date=new InlineDateField("Select a Date: ");
DateField date1=new DateField("Select a Date: ");
layout.addComponents(date);
layout.addComponent(date1);

setContent(layout);
                }
```

Κεφάλαιο 5ο

Γνωριμία με διάφορα layouts του Vaadin

1) **Vertical & Horizontal Layouts**

Μέχρι τώρα έχουμε εισάγει components μόνο σε Vertical Layout. Σε αυτό το κεφάλαιο θα γνωρίσουμε και μερικά ακόμα layouts που χρησιμοποιούνται ευρέως στο Vaadin.

Ο καλύτερος τρόπος για να παρουσιαστούν τα εν λόγω components είναι μέσο παραδειγμάτων, γι' αυτό το λόγο θα δημιουργήσουμε κώδικα ο οποίος θα απεικονίζει οριζόντια και κάθετα layouts (Vertical & Horizontal Layouts).

Η διαφορά των δύο layouts είναι στον τρόπο που στοιχίζονται τα componets, στο Horizontal Layout η στοίχιση γίνεται οριζόντια ενώ στο Vertical Layout η στοίχιση γίνεται κάθετα.

```
package com.example.layouts;

import com.vaadin.ui.Button;
import com.vaadin.ui.HorizontalLayout;
import com.vaadin.ui.HorizontalSplitPanel;
import com.vaadin.ui.Label;
import com.vaadin.ui.TextField;
import com.vaadin.ui.VerticalLayout;

public class LayoutExample extends VerticalLayout {
        HorizontalSplitPanel main=new HorizontalSplitPanel();
        VerticalLayout menu=new VerticalLayout();
        HorizontalLayout maindisplay=new HorizontalLayout();

        public LayoutExample(){
                addComponent(main);
                main.addComponent(menu);
                main.addComponent(maindisplay);
                showLines();
```

```
            //main.setSizeFull();
            menu.setSizeFull();
            maindisplay.setSizeFull();

            menu.addComponent(new Label("Menu Label"));
            menu.addComponent(new Button("Click me"));
            menu.addComponent(new TextField("Type your name"));
            maindisplay.addComponent(new Label("MainDisplay label"));
            maindisplay.addComponent(new Button("Button 1"));
            maindisplay.addComponent(new Button("Button 2"));

            setExpandRatio(main,1);

    }

    private void showLines(){
            String style="border";
            setStyleName(style);
            main.setStyleName(style);
            menu.setStyleName(style);
            maindisplay.setStyleName(style);

    }

            }
```

Σε μεγάλα προγράμματα ομαδοποιούμε τον κώδικά μας σε υποπρογράμματα δημιουργώντας κλάσεις. Έτσι λοιπόν στο συγκεκριμένο παράδειγμα δημιουργήσαμε την κλάση Layout Example η οποία θα δημιουργεί ένα Vertical και ένα Horizontal Layout.

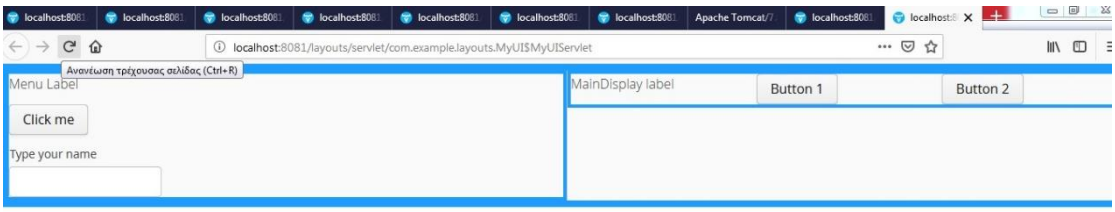
Για να μπορέσει να υλοποιηθεί η συγκεκριμένη κλάση θα πρέπει στην συνάρτηση init να δημιουργηθεί ένα αντικείμενο τύπου Layout Example. Έτσι ο κώδικας που προσθέτουμε είναι:

Layout Example layout=new Layout Example();

Set Content (layout);

Όλα τα layouts είναι “αόρατα”, γι’ αυτό το λόγο έχουμε προσθέσει την μέθοδο show Lines η οποία μας προσθέτει περίγραμμα γύρω από τα layouts.

Τα αποτελέσματα απεικονίζονται στην παρακάτω Εικόνα 26:



(Εικόνα 26)

2) Split Panels

Τα split Panels χωρίζουν ένα layout σε δύο μέρη και προσθέτουν ένα διαχωριστικό στη μέση. Το διαχωριστικό επιτρέπει την προσαρμογή των δύο επιμέρους Layouts χρησιμοποιώντας τη μέθοδο drag and drop.

Η εντολή εισαγωγής ενός SplitPanel είναι:

HorizontalSplitPanel main=new HorizontalSplitPanel ();

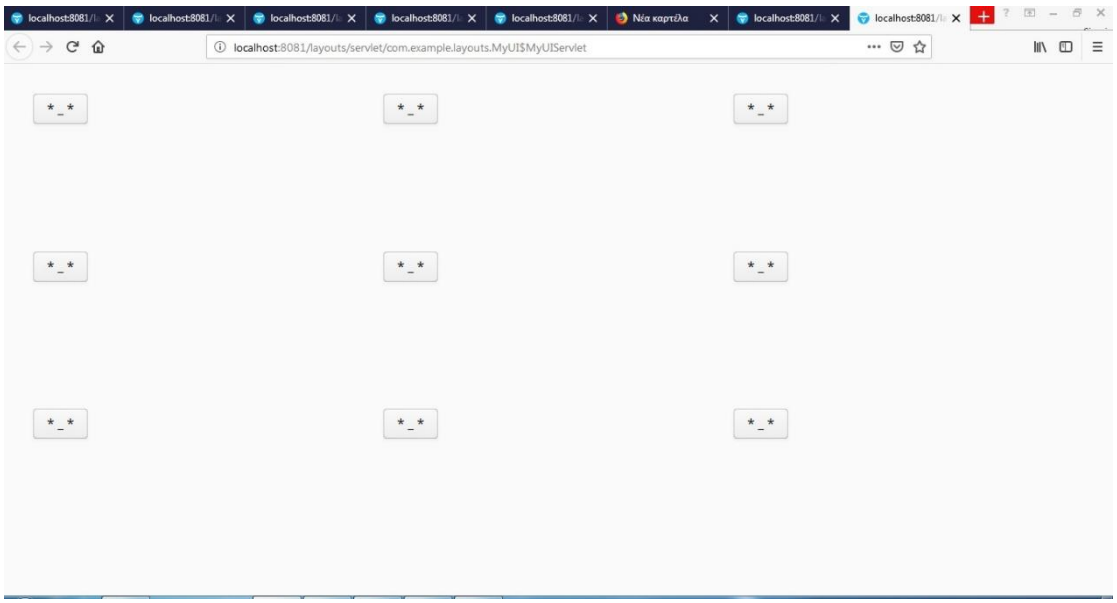
(Εικόνα 27)

3) Grid Layouts

Το Grid Layout τοποθετεί τα Components στοιχισμένα σε γραμμές και στήλες. Ο κώδικας και τα αποτελέσματα του φαίνονται παρακάτω (βλ. Εικόνα 28):

```
protected void init(VaadinRequest vaadinRequest) {
        int rows=3, columns=3;
        GridLayout grid=new GridLayout(columns, rows);
        grid.setMargin(true);
        grid.setSizeFull();
        setContent(grid);
```

```
for (int column=0;column<3;column++){
                for(int row=0;row<3;row++){
                        Button button=new Button("* _ *");
                        grid.addComponent(button, column, row);

                }
        }

          }
```



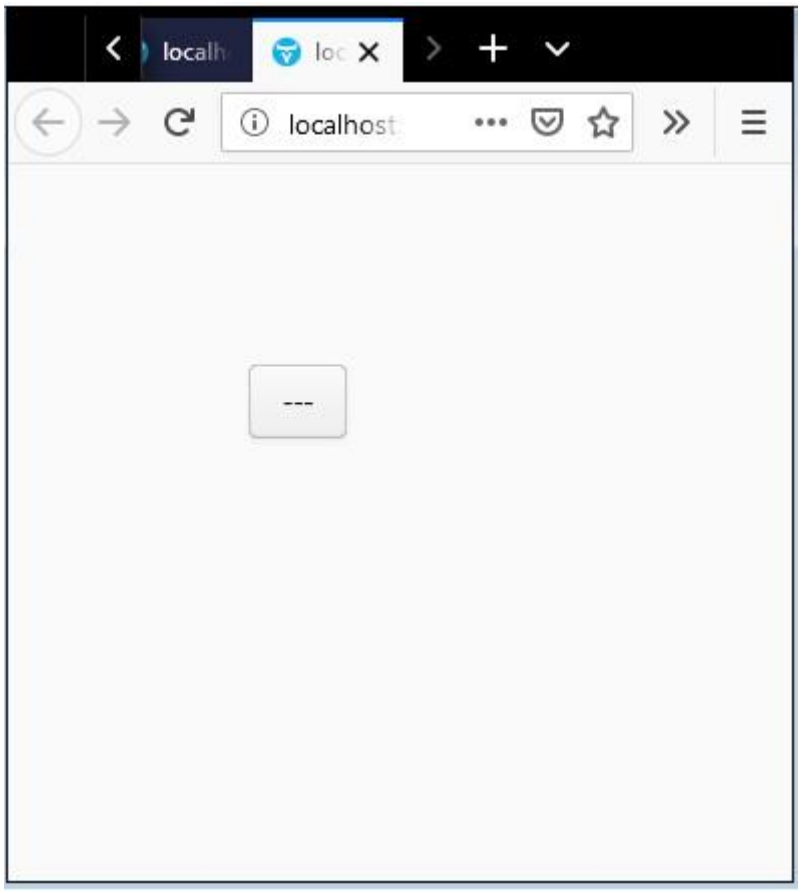
(Εικόνα 28)

4) <u>Absolute Layout</u>

Χρησιμοποιούμε το εν λόγω Layout για να τοποθετήσουμε ένα component σε συγκεκριμένη θέση της σελίδας. Οι συντεταγμένες για την τοποθέτηση του component δίνεται σε CSS format. Για παράδειγμα εάν θέλουμε να τοποθετήσουμε ένα button 120px από τα αριστερά της σελίδας και 100px απο το επάνω μέρος της σελίδας γράφουμε τον εξής κώδικα:

```
        protected void init(VaadinRequest vaadinRequest) {

         AbsoluteLayout absolute=new AbsoluteLayout();
         absolute.addComponent((new Button("---")), "left:120px;top:100px");
   setContent(absolute);

   }
```

Το αποτέλεσμα φαίνεται στην ακόλουθη Εικόνα 29:

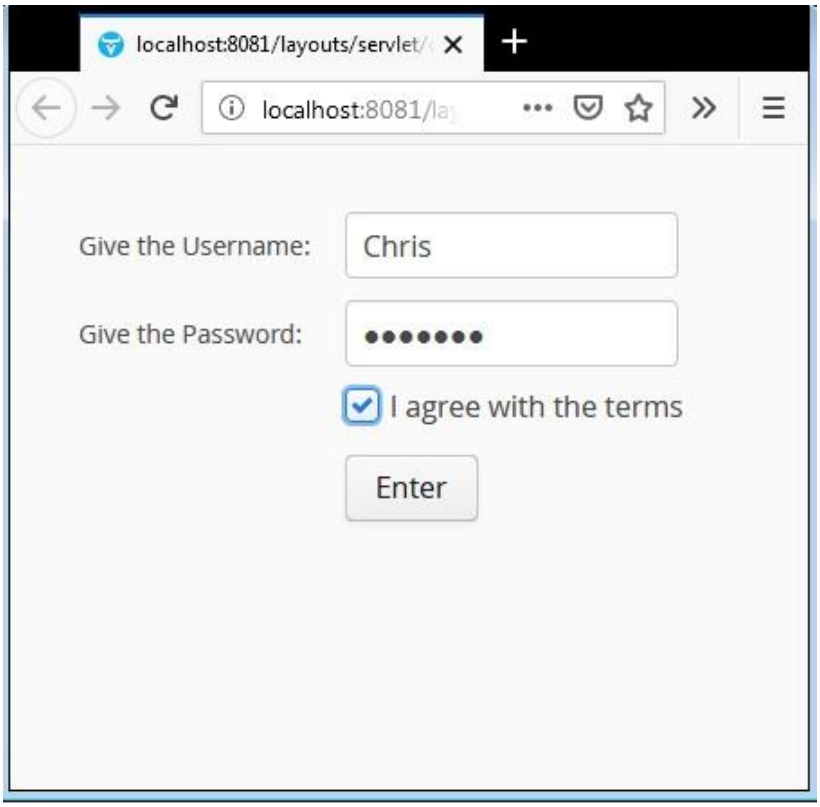


(Εικόνα 29)

5) <u>Form Layout</u>

Μέχρι τώρα τα components εισόδου που έχουμε συναντήσει η ετικέτα ήταν στο επάνω μέρος τους. Αυτό όμως αλλάζει στο Form Layout (βλ. Εικόνα 30) εμφανίζοντας την ετικέτα στα αριστερά και στη συνέχεια τα components.

```
protected void init(VaadinRequest vaadinRequest) {

 FormLayout form=new FormLayout();

 TextField text1=new TextField("Give the Username:");
 PasswordField text2=new PasswordField("Give the Password:");
 CheckBox check=new CheckBox("I agree with the terms");
 Button button=new Button("Enter");
 form.addComponent(text1);
 form.addComponent(text2);
 form.addComponent(check);
 form.addComponent(button);
 form.setSpacing(true);
 form.setMargin(true);

 setContent(form);
 }
```



(Εικόνα 30)

Όπως φαίνεται στην Εικόνα όλα τα components, εκτός από το button, έχουν την ετικέτα στα αριστερά. Φαίνεται σαν να υπάρχουν δύο στήλες, η μία για τις ετικέτες και η άλλη για τα components (βλ. Εικόνα 30).

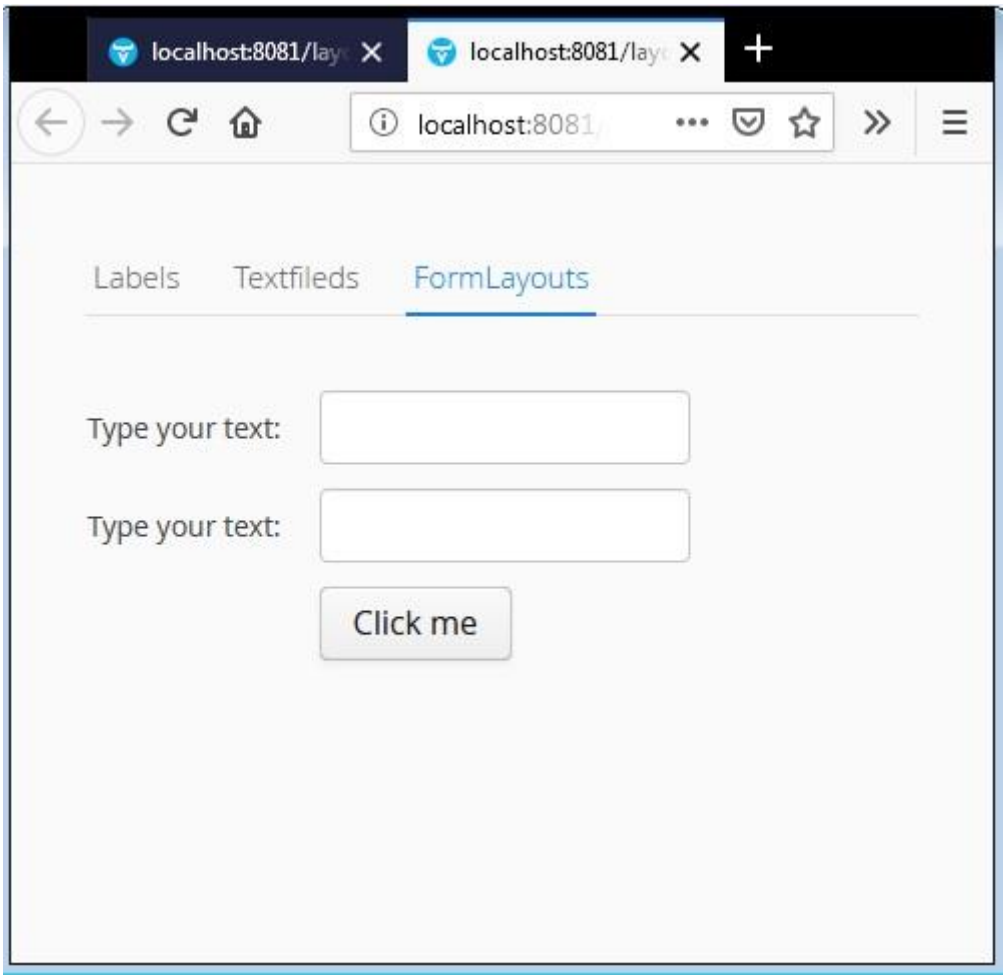
6) Tab Sheets

Τα TabSheets επιτρέπουν στον χρήστη να εναλλάσσονται μεταξύ views χρησιμοποιώντας tabs (βλ. Εικόνα 31).

Για να δημιουργήσουμε ένα κενό tab sheet απλά δημιουργούμε ένα αντικείμενο: TabSheet tab=new TabSheet();

Για να προσθέσουμε ένα component χρησιμοποιούμε την εντολή add tab(component, Το tab που θέλουμε να προστεθεί). Πιο αναλυτικά:

```
protected void init(VaadinRequest vaadinRequest) {
  VerticalLayout layout=new VerticalLayout();

 TabSheet tab=new TabSheet();

 TextField text1=new TextField("Type your text:");
 TextField text2=new TextField("Type your text:");
 Button button=new Button("Click me");

 tab.addTab(new Label("Label 1"), "Labels");
 tab.addTab(new TextField("Type your text:"), "Textfileds");
 tab.addTab(new FormLayout(text1, text2, button), "FormLayouts");
 layout.addComponent(tab);
 setContent(layout);

                    }
```



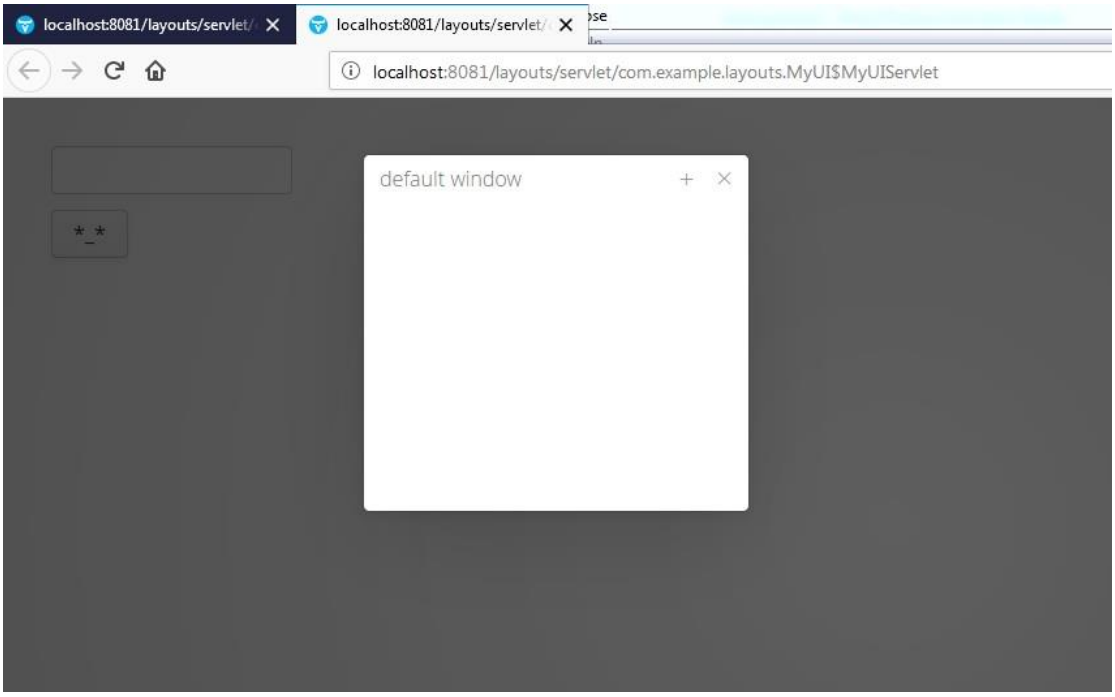
(Εικόνα 31)

7) Windows

Η διαφορά με τα άλλα components είναι ότι το αντικείμενο του window θα πρέπει να υλοποιηθεί μέσα στην κλάση MyUI και έξω από την μέθοδο init.

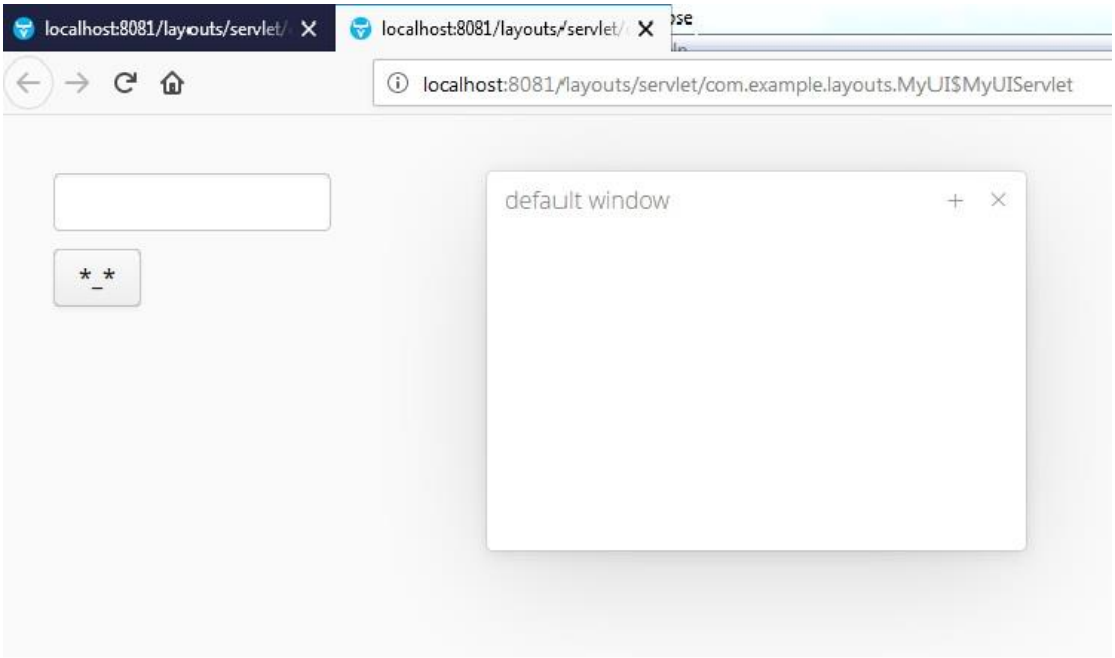
Υπάρχουν δύο ειδών παράθυρα:

- Τα Non modal windows στα οποία μπορώ να τα μετακινήσω (τεχνική drag and drop) και να ασχοληθώ με τα components της σελίδας (βλ. Εικόνα 33).
- Τα modal windows στα οποία μπορώ να ασχοληθώ μόνο μ' αυτά και όχι με τα components της σελίδας. Θα πρέπει να εκτελέσουμε τις εργασίες του παραθύρου ή να κλείσουμε το παράθυρο, μόνο τότε θα μπορούμε να ασχοληθούμε με την σελίδα (βλ. Εικόνα 32). Ένα απλό παράθυρο μπορεί να γίνει modal με τη χρήση της εντολής *set Modal(true).*



modal Window

(Εικόνα 32)



non modal Window

(Εικόνα 33)

```
public class MyUI extends UI {

        Window w1=new Window("default window");

    @Override
    protected void init(VaadinRequest vaadinRequest) {
    VerticalLayout layout=new VerticalLayout();
    TextField text=new TextField();
    Button button=new Button("*_*");
    addWindow(w1);
    w1.setWidth("500px");
    w1.setHeight("500px");
    //w1.setModal(true);
    layout.addComponent(text);
    layout.addComponent(button);
  setContent(layout);

    }
```

Υπάρχουν πολλές εντολές που χρησιμοποιούνται στα Windows. Μερικές απ' αυτές είναι:

- setWidth
- setHeight
- setResizable
- setDraggable
- setPositionX
- setPositionY
- setModal

Κεφάλαιο 6ο
Πίνακες

Οι πίνακες στο Vaadin είναι ίσως το πιο χρήσιμο component γιατί μπορεί να χειρίζεται δεδομένα από μία βάση δεδομένων.

Το μόνο που χρειάζεται είναι να δημιουργήσουμε ένα αντικείμενο τύπου Table και να εισάγουμε δεδομένα. Ο παρακάτω κώδικας δείχνει έναν πολύ απλό παράδειγμα.

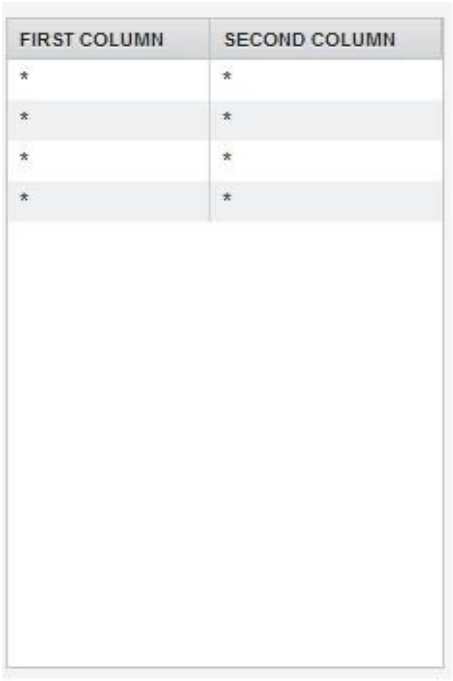
```
protected void init(VaadinRequest request) {
    final VerticalLayout layout = new VerticalLayout();
    layout.setMargin(true);
    setContent(layout);

    Table table=new Table();
    table.addContainerProperty("First Column", String.class, "*");
    table.addContainerProperty("Second Column", String.class, "*");
    table.addItem();
    table.addItem();
    table.addItem();
    table.addItem();
    layout.addComponent(table);
}
```

Εάν θέλουμε να δημιουργήσουμε μία στήλη τότε χρησιμοποιούμε την εντολή addContainerProperty της οποίας η πρώτη παράμετρος αντιπροσωπεύει το ID, η δεύτερη παράμετρος τον τύπο των δεδομένων που θα αποθηκεύονται στον πίνακα (στο συγκεκριμένο παράδειγμα είναι τύπου String) και η τρίτη παράμετρος είναι η default τιμή σε περίπτωση που το κελί δεν έχει τιμή.

Αφού λοιπόν δημιουργήσαμε τις στείλεις του πίνακα, θα πρέπει να ορίσουμε επίσης τον αριθμό των γραμμών. Αυτό γίνεται με την εντολή addItem.

Στο παραπάνω παράδειγμα προσθέτουμε γραμμές χωρίς δεδομένα στα κελιά, άρα οι default τιμές θα είναι το «αστεράκι».

Η παρακάτω Εικόνα δείχνει τα αποτελέσματα του παραπάνω κώδικα (βλ. Εικόνα 34):



(Εικόνα 34)

1) <u>Επικεφαλίδες πινάκων (Headers)</u>

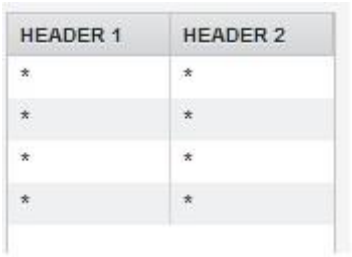
Κατά τη δημιουργία του πίνακα ορίσαμε ταυτόχρονα και την ονομασία των κάθε στηλών δηλ. τις επικεφαλίδες. Τι γίνεται όμως εάν θέλουμε να μετονομάσουμε μία στήλη; Το vaadin έχει προνοήσει και διαθέτει μία σειρά εντολών. Συγκεκριμένα με την εντολή setColumnHeader μπορούμε να μετονομάσουμε μία στήλη. Η ανάπτυξη της εντολής φαίνεται στον ακόλουθο κώδικα:

table.setColumnHeader ("First Column", "Header Changed");

(Εικόνα 35)

Επίσης υπάρχει η δυμνατότητα να αλλάξουμε ταυτόχρονα όλες τις στήλες ενός πίνακα με την εντολή setColumnHeaders. Πιο Συγκεκριμένα:

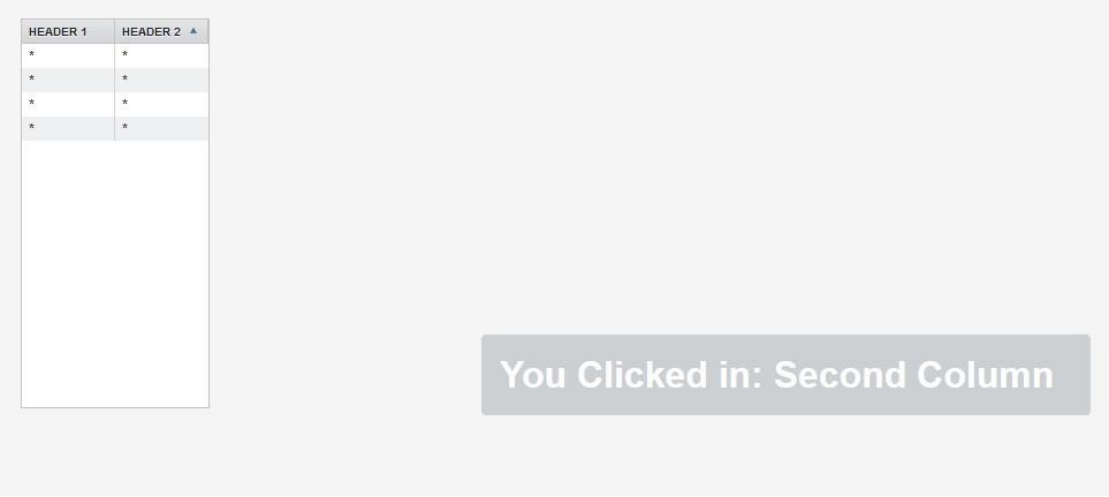
table.setColumnHeaders (new String[]{"Header 1", "Header 2"});



(Εικόνα 36)

Οι στήλες έχουν το δικό τους Click Listener και με αυτό τον τρόπο εάν κάνουμε click σε μία στήλη τότε θα εκτελεστεί ένα τμήμα κώδικα που έχουμε ορίσει εμείς. Για παράδειγμα στον παρακάτω κώδικα όταν κάνουμε click με το ποντίκι μας σε μία στήλη τότε θα μας εμφανίσει ένα μήνυμα (βλ. εικ 37).

```
table.addHeaderClickListener(new Table.HeaderClickListener() {
    @Override
    public void headerClick(Table.HeaderClickEvent event) {

        Notification.show("You Clicked in: "+event.getPropertyId().toString());

    }
});
```



(Εικόνα 37)

Αξίζει να σημειωθεί ότι το ID της κάθε στήλης παραμένει το ίδιο δηλ. όπως τα θέσαμε κατά τη δημιουργία του πίνακα (First Column και Second Column αντίστοιχα). Ακόμα και εάν αλλάξουμε το header της κάθε στήλης το ID παραμένει ίδιο. Αυτό φαίνεται και από την παραπάνω Εικόνα 37 στην οποία κάναμε click στην κεφαλίδα της δεύτερης στήλης. Παρόλο που η δεύτερη στήλη ονομάζεται “Header 2”, το μήνυμα που εμφανίστηκε είναι «Second Column» (βλ. Εικόνα 37).

2) Footers

Με τον όρο «Footer» εννοούμε μία επιπλέον γραμμή που εμφανίζεται στο τέλος του πίνακα, κυρίως για να δείξουμε το άθροισμα των γραμμών ή ότι άλλο σχετίζεται με τις στήλες. Το «Footer» δεν εμφανίζεται παρά μόνο εάν εκτελέσουμε την εντολή setFooterVisible(true).

Εάν θέλουμε να ορίσουμε μία τιμή στο “Footer” θα πρέπει να πληκτρολογήσουμε την εντολή setColumnFooter(“column ID”, “String Value”).

Ο ακόλουθος κώδικας αντιπροσωπεύει ένα παράδειγμα που δείχνει πως εμφανίζεται το “Footer”:

```
protected void init(VaadinRequest request) {
    final VerticalLayout layout = new VerticalLayout();
    layout.setMargin(true);
    setContent(layout);

    Table table=new Table();

    table.addContainerProperty("First Column", Integer.class, 0);
    table.addContainerProperty("Second Column", Integer.class, 0);

    int column1=0, column2=0;

    for(int i=1;i<5;i++){
       table.addItem(new Object[]{i, i*2}, i);
       column1+=i;
       column2+=i*2;

    }
```

```
        layout.addComponent(table);

        table.setColumnHeaders(new String[]{"Header 1", "Header 2"});


        table.setFooterVisible(true);
        table.setColumnFooter("First Column", "Sum of HEADER 1="+column1);
        table.setColumnFooter("Second Column", "Sum of HEADER 2="+column2);

    }
```

Σε περίπτωση που εκτελεστεί ο κώδικας, στον Browser θα εμφανιστούν τα εξής αποτελέσματα (βλ. Εικόνα 38):

HEADER 1	HEADER 2
1	2
2	4
3	6
4	8
Sum of HEADER 1=10	Sum of HEADER 2=20

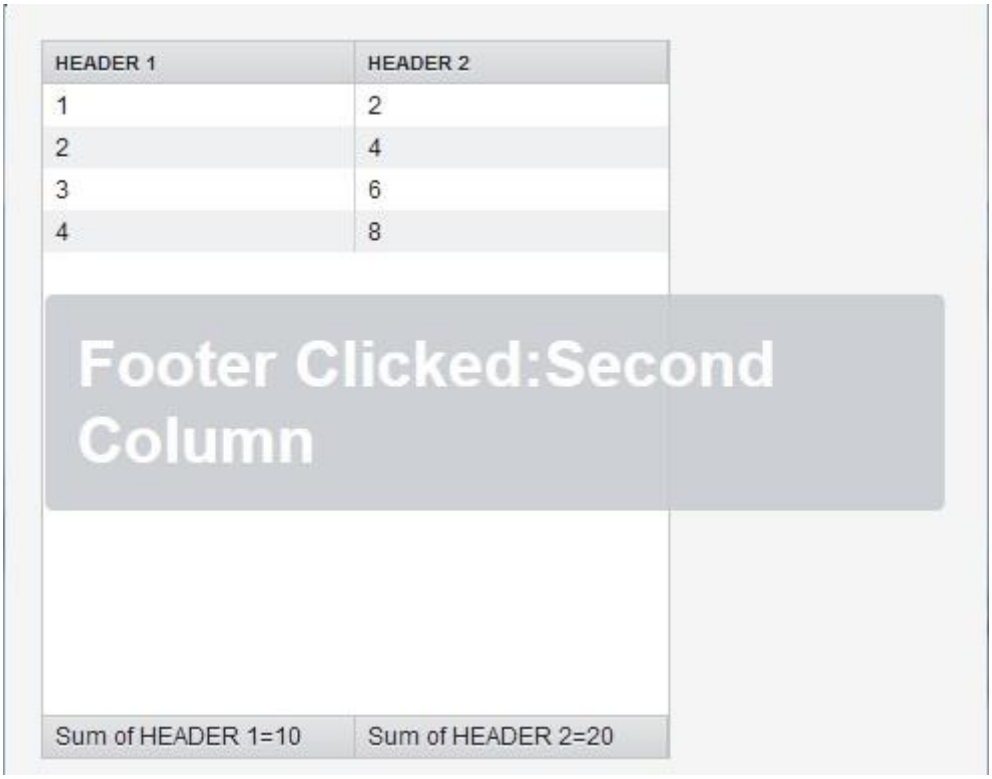
(Εικόνα 38)

Όπως στο “Header” έτσι και στο “Footer” μπορεί να προστεθεί ο ανάλογος Listener. Έτσι, εάν κάνουμε click σε ένα σημείο του “Footer” τότε εκτελείτε ο κώδικας που του έχουμε ορίσει εμείς.

Στον παρακάτω κώδικα, όταν κάνουμε click σε κάποιο σημείο του “Footer” τότε εμφανίζεται ένα μήνυμα και μας ενημερώνει για το ID της στήλης στην οποία αντιστοιχεί.

```
table.addFooterClickListener(new Table.FooterClickListener(){
    @Override
    public void footerClick(Table.FooterClickEvent event) {
        Notification.show("Footer Clicked:"+event.getPropertyId());
    }

});
```

Τα αποτελέσματα απεικονίζονται στην ακόλουθη Εικόνα 39:



(Εικόνα 39)

3) <u>Ανάγνωση & Εγγραφή δεδομένων σε πίνακα (Reading and Writing data to tables)</u>

Αρχικά δημιουργούμε έναν πίνακα 5x5 (5 στηλών και 5 γραμμών) που θα έχουν ως στοιχεία τον χαρακτήρα «*». Ο τρόπος δημιουργίας του πίνακα έχει περιγραφεί λεπτομερέστερα σε προηγούμενες ενότητες, χρησιμοποιώντας αναλυτικά παραδείγματα.

Σε αυτή την ενότητα θα επικεντρωθούμε στην ανάγνωση και την εγγραφή δεδομένων από ένα κελί ενός πίνακα.

Η ανάγνωση και αντιγραφή γίνεται με τις εντολές «property.getValue()» και «property.setValue» οι οποίες παίρνουν μία τιμή από το κελί που του ορίσαμε ή θέτουν μία τιμή στο επιθυμητό κελί. Άρα το πιο σπουδαίο είναι να περιγράψουμε με κάποιο τρόπο ένα κελί το οποίο θα ανήκει σε συγκεκριμένη στήλη και γραμμή του πίνακα. Αυτό γίνεται δημιουργώντας μία «property» η οποία θα δείχνει σε συγκεκριμένο κελί το οποίο προσδιορίζεται από ID της στήλης και το ID της γραμμής.

Οι εντολές θα συμπεριλαμβάνονται σε ένα ItemClickListener οι οποίες θα εκτελούνται κάθε φορά που κάνουμε click σε ένα κελί του πίνακα.

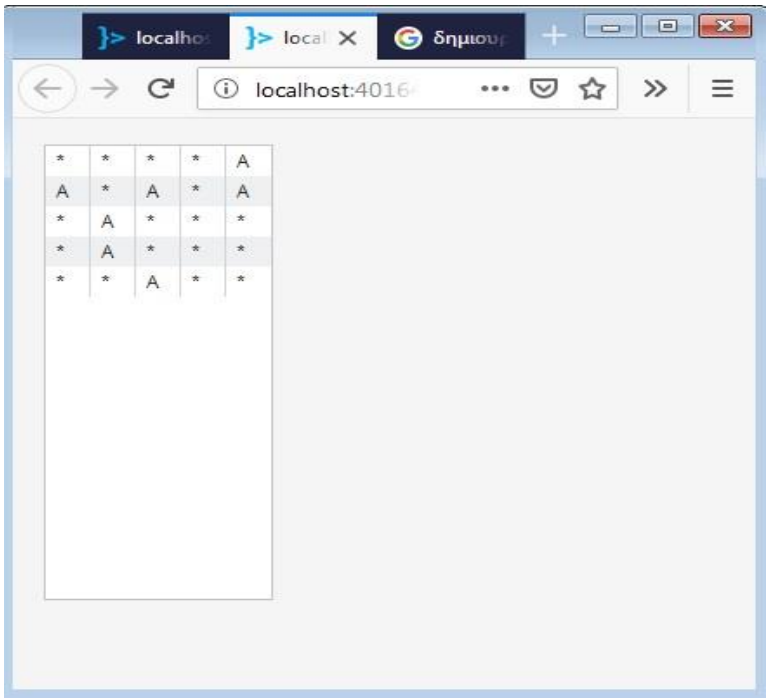
Ακολουθεί αναλυτικά ο κώδικας που χρησιμοποιείται ως παράδειγμα για την επεξήγηση των εντολών εγγραφής και ανάγνωσης δεδομένων σε έναν πίνακα:

```
public class MyVaadinUI extends UI implements ItemClickListener
{
int tableColumns=5;
int tableRows=5;
   @Override
   protected void init(VaadinRequest request) {
      VerticalLayout mainLayout=new VerticalLayout();
      mainLayout.setMargin(true);
      mainLayout.setSpacing(true);
      Table table=new Table();
      table.setColumnHeaderMode(Table.ColumnHeaderMode.HIDDEN);

      for(int column=0; column<tableColumns; column++){
         table.addContainerProperty(column, String.class, "*");
      }

      for(int row=0;row<tableRows;row++){
         table.addItem(row);
      }
      table.addItemClickListener((ItemClickEvent.ItemClickListener) this);
```

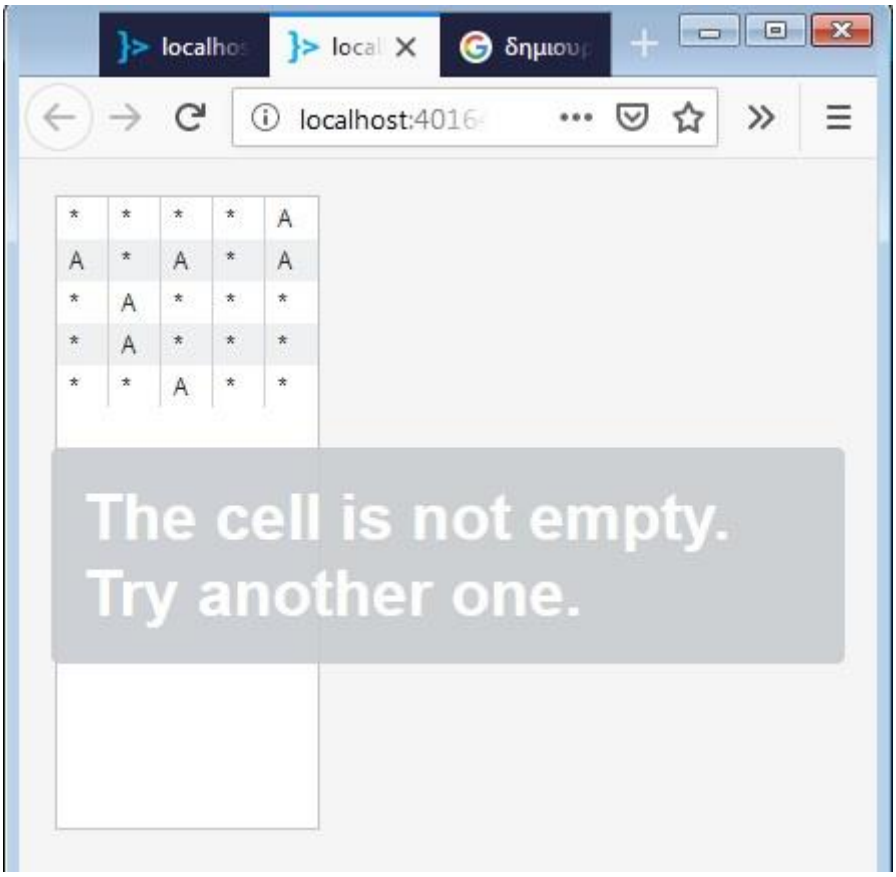
```
            mainLayout.addComponent(table);
            setContent(mainLayout);
        }
        @Override
        public void itemClick(ItemClickEvent event) {
            Property
    property=event.getItem().getItemProperty(event.getPropertyId());
            if(property.getValue().equals("A")){
                Notification.show("The cell is not empty. Try another one.");
            }
            else property.setValue("A");
        }
    }
```

Επεξήγηση κώδικα: Δημιουργείται ένας πίνακας 5x5 στοιχείων και κάθε κελί έχει ως default τον χαρακτήρα αστεράκι («*»). Όταν κάνουμε click σε ένα κελί τότε διαβάζουμε την τιμή που έχει δηλ. το αστεράκι («*») και στη συνέχεια αντικαθίσταται από το γράμμα «A». Εάν παρόλα αυτά, κατά λάθος, κάνουμε click σε ένα κελί που περιέχει τον χαρακτήρα «A», τότε μας βγάζει το μήνυμα «The cell is not empty. Try another one.».

Στις ακόλουθες εικόνες φαίνονται τα αποτελέσματα (βλ. εικ 40, Εικόνα 41):



(Εικόνα 40)



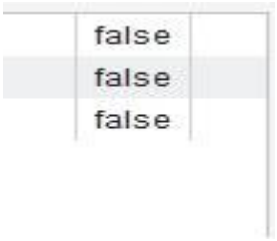
(Εικόνα 41)

4) <u>Επεξεργάσιμος Πίνακας (Editable table)</u>

Ας υποθέσουμε ότι δημιουργούμε ένα απλό πίνακα με τρεις στήλες και τρεις γραμμές. Οι εντολές για την δημιουργία του εν λόγω πίνακα είναι οι ακόλουθες:

```
table.addContainerProperty("String", String.class, "");
table.addContainerProperty("Boolean", Boolean.class, false);
table.addContainerProperty("Date", Date.class, null);
table.addContainerProperty("item", Item.class, item);
table.addItem();
table.addItem();
table.addItem();
```

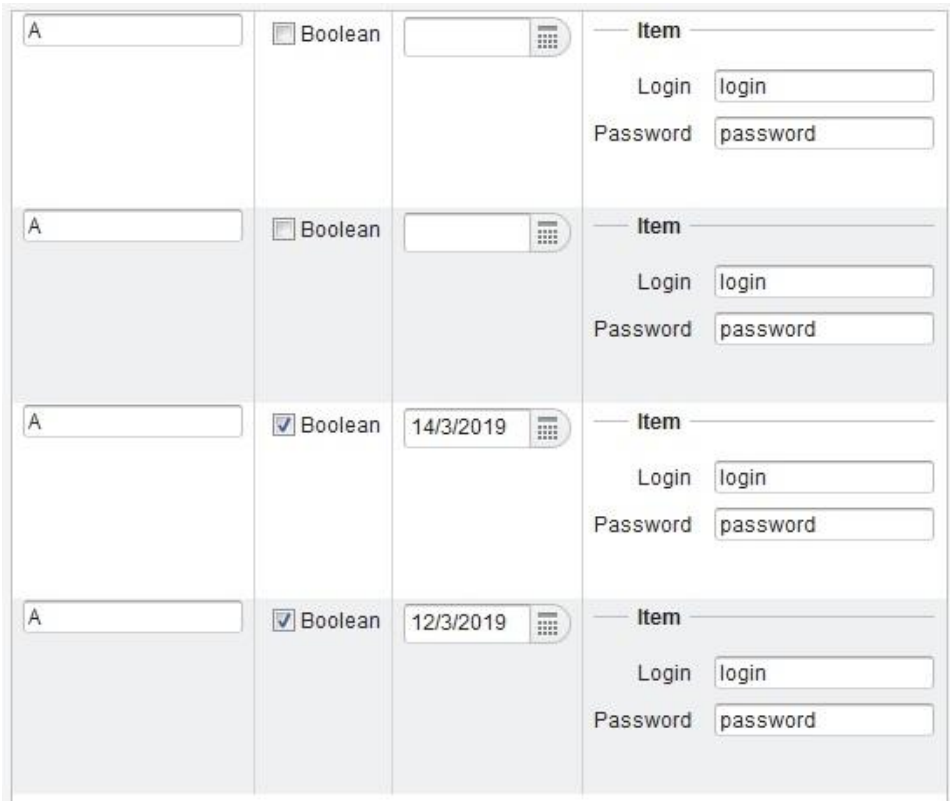
Η απεικόνιση του πίνακα σε έναν φυλλομετρητή (Web Browser) φαίνεται στην ακόλουθη Εικόνα 42:



(Εικόνα 42)

Το Vaadin χρησιμοποιεί την εντολή **set Editable(true)** η οποία «βλέπει» τον τύπο των δεδομένων που έχουμε ορίσει σε κάθε στήλη και ανάλογα εμφανίζει το αντίστοιχο component. Για παράδειγμα στον παραπάνω πίνακα έχουμε ορίσει ότι η πρώτη στήλη θα περιέχει δεδομένα τύπου String, άρα με τη χρήση της εν λόγω εντολής μας εμφανίζει ένα TextBox. Το ίδιο συμβαίνει για παράδειγμα και με την Τρίτη στήλη που έχουμε ορίσει ότι θα περιλαμβάνει δεδομένα τύπου Ημερομηνίας. Με τη χρήση της εντολής setEditable(true) θα εμφανίζεται ένα πεδίο στο οποίο μπορούμε να πληκτρολογήσουμε μία ημερομηνία ή μπορούμε να επιλέξουμε την Ημερομηνία κάνοντας κλικ στο εικονίδιο που βρίσκεται στα δεξιά του πεδίου.

Τα αποτελέσματα φαίνονται στην ακόλουθη Εικόνα 43:



(Εικόνα 43)

Ο κώδικας για τη δημιουργία του παραπάνω πίνακα, χρησιμοποιώντας την εντολή setEditable(true), είναι ο ακόλουθος:

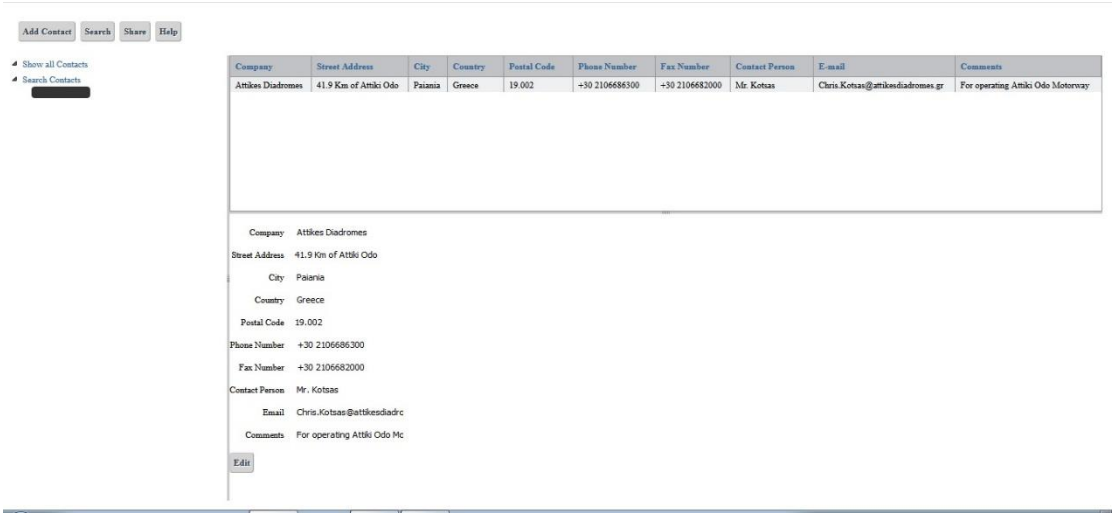
```
protected void init(VaadinRequest request) {

        User user=new User();
        user.setLogin("login");
        user.setPassword("password");
        Table table=new Table();
        BeanItem<User> item=new BeanItem<User>(user);
        table.addContainerProperty("String", String.class, "");
        table.addContainerProperty("Boolean", Boolean.class, false);
        table.addContainerProperty("Date", Date.class, null);
        table.addContainerProperty("item", Item.class, item);
        table.addItem();
        table.addItem();
        table.addItem();
        table.addItem(new Object[] {"", true, new Date(), item}, 4);

        VerticalLayout mainLayout=new VerticalLayout();
        mainLayout.setMargin(true);
        mainLayout.setSpacing(true);

            table.setColumnHeaderMode(Table.ColumnHeaderMode.HIDDEN);

            table.addItemClickListener((ItemClickEvent.ItemClickListener) this);

            table.setEditable(true);
            mainLayout.addComponent(table);
            setContent(mainLayout);
        }

        @Override
        public void itemClick(ItemClickEvent event) {
```

```
        Property
property=event.getItem().getItemProperty(event.getPropertyId());
        if(property.getValue().equals("A")){
            Notification.show("The cell is not empty. Try another one.");
        }
        else property.setValue("A");
    }

}
```

ΚΕΦΑΛΑΙΟ 7ο
Επεξήγηση εφαρμογής

Η εφαρμογή που υλοποιήθηκε είναι μία ολοκληρωμένη εφαρμογή η οποία καταχωρεί το πελατολόγιο μιας εταιρείας. Έτσι για παράδειγμα στην εφαρμογή καταχωρείται το όνομα της εταιρείας με την οποία συνεργαζόμαστε, τη διεύθυνσή της, την τοποθεσία της, τα στοιχεία επικοινωνίας, ο υπεύθυνος της εταιρείας με τον οποίο επικοινωνούμε κάθε φορά που θέλουμε κάτι και ένα σύντομο σχόλιο με το αντικείμενο με το οποίο ασχολείται η εταιρεία.

Στην ακόλουθη Εικόνα φαίνεται ένα παράδειγμα με τα πεδία που προαναφέραμε συμπληρωμένα (βλ. Εικόνα 44):



(Εικόνα 44)

Η συγκεκριμένη εφαρμογή, η οποία υλοποιήθηκε με τη βοήθεια του Vaadin 7, περιλαμβάνει τρία πακέτα που το κάθε πακέτο περιλαμβάνει τις δικές του κλάσεις.

Πιο αναλυτικά το πακέτο «com.example.UserInterface» περιέχει όλες τις τάξεις που έχουν να κάνουν με το User Interface δηλ. με ότι βλέπουμε σε έναν φυλλομετρητή.

Το πακέτο «com.example.data» περιέχει τις κλάσεις που χρησιμοποιούνται για τη δημιουργία ενός Container και για τα default δεδομένα που εμφανίζονται στην εφαρμογή.

Τέλος το πακέτο «com.example.diplomatikiFinal» περιέχει τη κλάση «diploamtikiUI» η οποία συνδέει όλες τις κλάσεις απ' όλα τα πακέτα που χρησιμοποιούνται στο πρόγραμμα. Ο τρόπος σύνδεσης είναι δημιουργώντας διάφορα αντικείμενα στη

συγκεκριμένη κλάση. Επίσης η εν λόγω κλάση περιέχει και τη συνάρτηση init η οποία εκτελείται κάθε φορά που κάνουμε μία αλλαγή.

Στις επόμενες ενότητες θα παρουσιαστεί λεπτομερέστερα το κάθε πακέτο και θα επεξηγηθεί ο κώδικας από κάθε κλάση μαζί με τα αποτελέσματά τους.

a) UserInterface πακέτο

1. *HelpWindow κλάση*

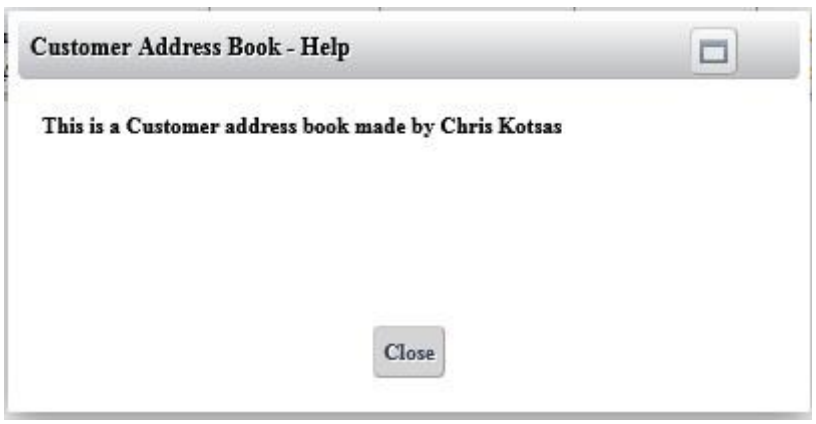
```
package com.example.UserInterface;

import com.vaadin.shared.ui.label.ContentMode;
import com.vaadin.ui.Alignment;
import com.vaadin.ui.Button;
import com.vaadin.ui.Label;
import com.vaadin.ui.VerticalLayout;
import com.vaadin.ui.Window;
import com.vaadin.ui.Button.ClickEvent;

public class HelpWindow extends Window {

    public HelpWindow(){
        center();
        setCaption("Customer Address Book - Help");
        setHeight("200px");
        setWidth("400px");
        VerticalLayout mainLayout=new VerticalLayout();
        mainLayout.setSpacing(true);
        mainLayout.setMargin(true);
        Label label=new Label("<b>This is a Customer address book made by
    Chris Kotsas</b>", ContentMode.HTML);
        Button closeWindow=new Button("Close");
        setClosable(false);
        mainLayout.addComponent(label);
        mainLayout.addComponent(closeWindow);
```

```
            setContent(mainLayout);
            mainLayout.setExpandRatio(label,1);
            mainLayout.setComponentAlignment(closeWindow,
        Alignment.BOTTOM_CENTER);
            mainLayout.setSizeFull();

            closeWindow.addClickListener(new Button.ClickListener() {

                @Override
                public void buttonClick(ClickEvent event) {
                    // TODO Auto-generated method stub
                    close();

                }
            });

    }
}
```

Ο παραπάνω κώδικας δημιουργεί ένα παράθυρο, με συγκεκριμένες διαστάσεις, το οποίο εμφανίζεται στο κέντρο της εφαρμογής. Το παράθυρο αυτό έχει σαν τίτλο το όνομα της εφαρμογής δηλ. «Customer Address Book» και δίνει πληροφορίες για το δημιουργό της εφαρμογής.

Περιέχει επίσης ένα πλήκτρο, που εάν πατηθεί, τότε κλείνει το παράθυρο.

Τα αποτελέσματα φαίνονται στην παρακάτω Εικόνα 45:



(Εικόνα 45)

2. ShareWindow κλάση

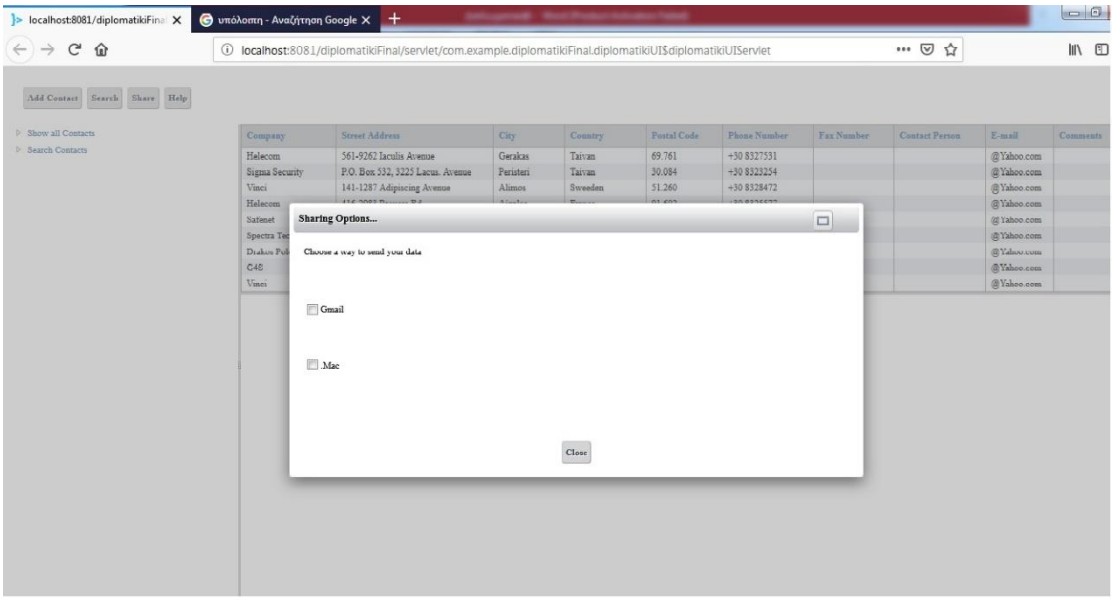
```
package com.example.UserInterface;

import com.vaadin.ui.Alignment;
import com.vaadin.ui.Button;
import com.vaadin.ui.CheckBox;
import com.vaadin.ui.Label;
import com.vaadin.ui.VerticalLayout;
import com.vaadin.ui.Window;
import com.vaadin.ui.Button.ClickEvent;

public class ShareWindow extends Window {

    public ShareWindow() {

        setModal(true);
        setWidth("50%");
        setHeight("50%");
        center();
        setClosable(false);
        setCaption("Sharing Options...");

        VerticalLayout layout=new VerticalLayout();
        layout.setSpacing(true);
        layout.setMargin(true);
        Label label=new Label("Choose a way to send your data");
        CheckBox check1=new CheckBox("Gmail");
        CheckBox check2=new CheckBox(".Mac");
        Button close=new Button("Close");

        layout.addComponent(label);
        layout.addComponent(check1);
        layout.addComponent(check2);
        layout.addComponent(close);
```

```
                layout.setSizeFull();
                layout.setComponentAlignment(close,
Alignment.BOTTOM_CENTER);

                setContent(layout);

                close.addClickListener(new Button.ClickListener() {

                        @Override
                        public void buttonClick(ClickEvent event) {
                                // TODO Auto-generated method stub
                                close();
                        }
                });

        }

}
```

Η κλάση αυτή εμφανίζει ένα ακόμα παράθυρο στο κέντρο της εφαρμογής το οποίο περιέχει δύο τρόπους αποστολής των δεδομένων (βλ Εικόνα 46).

(Εικόνα 46) Share Window

Εκτός από το περιεχόμενο του, η κύρια διαφορά με το προηγούμενο παράθυρο είναι στην εντολή «setModal(true)». Η εντολή αυτή δίνει τη δυνατότητα να ασχοληθούμε μόνο με το παράθυρο. Δηλ. εάν θέλουμε να κάνουμε κλικ σε ένα άλλο σημείο της εφαρμογής τότε δε μπορούμε. Θα πρέπει να κλείσουμε το εν λόγω παράθυρο και στη συνέχεια να περιηγηθούμε στην υπόλοιπη εφαρμογή (βλ. Εικόνα 47).



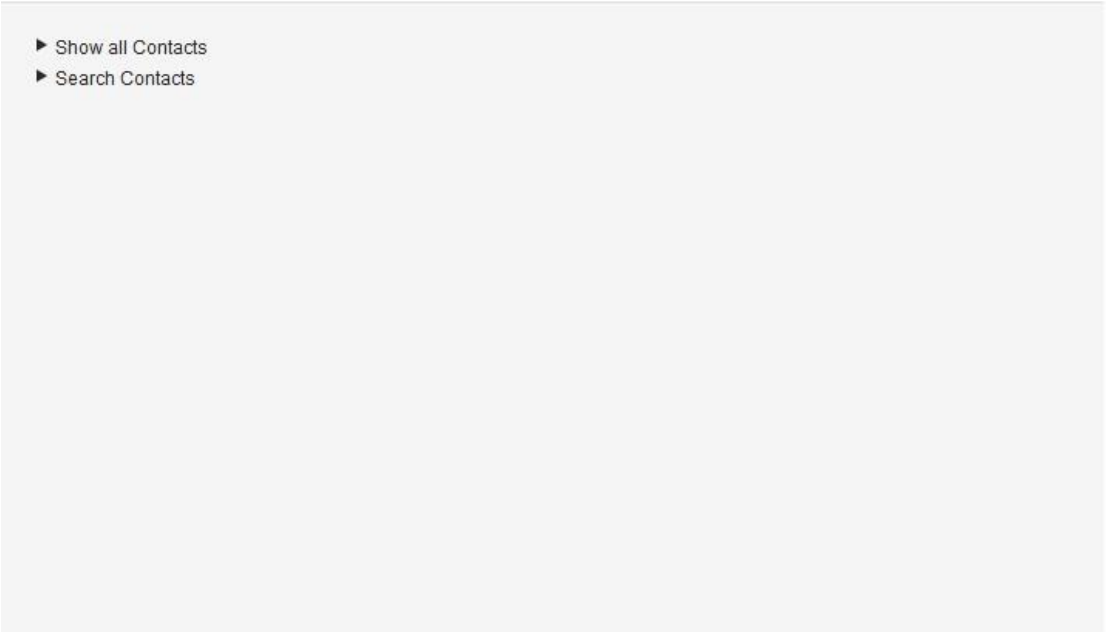
(Εικόνα 47) Share Window με χρήση της εντολής setModal (true)

3. NavigationTree κλάση

```
package com.example.UserInterface;

import com.example.diplomatikiFinal.diplomatikiUI;
import com.vaadin.event.ItemClickEvent.ItemClickListener;
import com.vaadin.ui.Tree;

public class NavigationTree extends Tree {

    public static final String SHOW_ALL="Show all Contacts";
    public static final String SEARCH="Search Contacts";

    public NavigationTree(){
        addItem(SHOW_ALL);
```

```
            addItem(SEARCH);

        }

        public NavigationTree(diplomatikiUI app){
            addItem(SHOW_ALL);
            addItem(SEARCH);
            setSelectable(true);
            setNullSelectionAllowed(false);
            addItemClickListener((ItemClickListener) app);
        }

}
```

Ο κώδικας της Navigation Tree κλάσης είναι πολύ απλός. Περιέχει δύο constructors οι οποίοι σκοπό έχουν να υλοποιήσουν ένα δέντρο. Το δέντρο περιέχει μόνο δύο αλφαριθμητικά, το «Show all Contacts" και το "Search Contacts". Τα αποτελέσματα φαίνονται στην παρακάτω Εικόνα 48.



(Εικόνα 48)

4. Person Form κλάση

```
package com.example.UserInterface;

import java.util.Arrays;
import java.util.Iterator;
import java.util.List;

import com.example.data.Person;
import com.example.data.PersonCotnainer;
import com.example.diplomatikiFinal.diplomatikiUI;
import com.vaadin.data.Item;
import com.vaadin.data.util.BeanItem;
import com.vaadin.data.validator.EmailValidator;
import com.vaadin.data.validator.RegexpValidator;
import com.vaadin.ui.Button;
import com.vaadin.ui.Button.ClickEvent;
import com.vaadin.ui.Button.ClickListener;
import com.vaadin.ui.ComboBox;
import com.vaadin.ui.Field;
import com.vaadin.ui.Form;
import com.vaadin.ui.FormLayout;
import com.vaadin.ui.HorizontalLayout;
import com.vaadin.ui.TextField;

@SuppressWarnings("deprecation")
        public class PersonForm extends Form implements ClickListener {


private Button save = new Button("Save", (ClickListener) this);
private Button cancel = new Button("Cancel", (ClickListener) this);
private Button edit = new Button("Edit", (ClickListener) this);
private diplomatikiUI app;
```

```
private final ComboBox cities = new ComboBox("City");

private boolean newContactMode = false;
private Person newPerson = null;

        public PersonForm() {
                addField("First Name", new TextField("First Name"));
                addField("Last Name", new TextField("Last Name"));
                HorizontalLayout footer = new HorizontalLayout();
                footer.setSpacing(true);
                footer.addComponent(save);
                footer.addComponent(cancel);
                setFooter(footer);
                }

        public PersonForm(diplomatikiUI app) {
                        this.app = app;

cities.setNewItemsAllowed(true);

cities.setNullSelectionAllowed(false);
cities.addItem("");

PersonCotnainer ds = app.getDataSource();
for (Iterator<Person> it = ds.getItemIds().iterator();it.hasNext();){
        String city = (it.next()).getCity();
        cities.addItem(city);
}

        setBuffered(true);
                HorizontalLayout footer = new HorizontalLayout();
                footer.setSpacing(true);
                footer.addComponent(save);
                footer.addComponent(cancel);
```

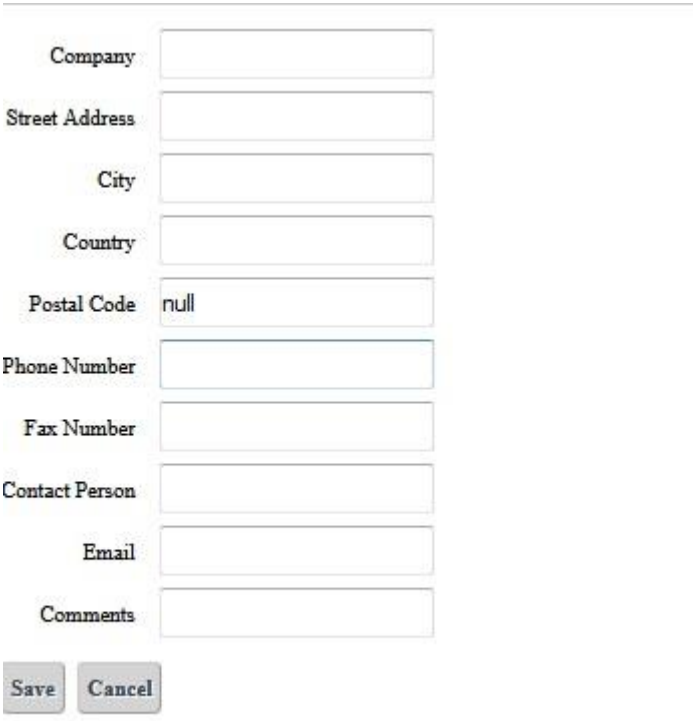
```
            footer.addComponent(edit);
            footer.setVisible(false);
            setFooter(footer);
        }

@Override
public void buttonClick(ClickEvent event) {
        Button source = event.getButton();
        if (source == save) {

            if (!isValid()) {
                return;
            }

            commit();

            if (newContactMode) {
            // We need to add the new person to the container
            Item addedItem = app.getDataSource().addItem(newPerson);

            //
            // We must update the form to use the Item from our datasource
            // as we are now in edit mode
            //
            setItemDataSource(addedItem);
            newContactMode = false;
            }

                setReadOnly(true);
            } else if (source == cancel) {
                        if (newContactMode) {
                            newContactMode = false;
                            setItemDataSource(null);
                        } else {
```

```
                    discard();
                }
                setReadOnly(true);
            } else if (source == edit) {
                setReadOnly(false);
            }
        }

        @Override
        public void setItemDataSource(Item newDataSource) {
            newContactMode = false;
            if (newDataSource != null) {
                List<Object> orderedProperties = Arrays

.asList(PersonCotnainer.NATURAL_COL_ORDER);
            super.setItemDataSource(newDataSource, orderedProperties);
            setReadOnly(true);
            getFooter().setVisible(true);

            } else {
                super.setItemDataSource(null);
                getFooter().setVisible(false);
                }
            }


            @Override
            public void setReadOnly(boolean readOnly) {
                super.setReadOnly(readOnly);
                save.setVisible(!readOnly);
                cancel.setVisible(!readOnly);
                edit.setVisible(readOnly);
            }
```

```
        public void addContact() {
            // Create a temporary item for the form
            newPerson = new Person();
            setItemDataSource(new BeanItem(newPerson));
            newContactMode = true;
            setReadOnly(false);
        }

    }
```

Η Person Form κλάση εμφανίζει όλα τα πεδία τα οποία συμπληρώνουμε. Μετά την εισαγωγή των στοιχείων θα πρέπει να πατήσουμε το πλήκτρο “Save” για την καταχώρηση των δεδομένων ή “Cancel” για να μην αποθηκευτούν.

Τα πεδία της εν λόγω κλάσης φαίνονται στην παρακάτω Εικόνα 49:



(Εικόνα 49)

5. *Person List κλάση*

```
package com.example.UserInterface;

import com.example.data.PersonCotnainer;
import com.example.diplomatikiFinal.diplomatikiUI;
import com.vaadin.data.Property;
import com.vaadin.data.Property.ValueChangeEvent;
import com.vaadin.ui.Table;

public class PersonList extends Table {

        public PersonList(diplomatikiUI app){
                setSizeFull();
                setContainerDataSource(app.getDataSource());
                setVisibleColumns(PersonCotnainer.NATURAL_COL_ORDER);
                setColumnHeaders(PersonCotnainer.COL_HEADERS_ENGLISH);
                setSelectable(true);
                setImmediate(true);
                setNullSelectionAllowed(false);
                addValueChangeListener((Property.ValueChangeListener) app);

        }
}
```

Η συγκεκριμένη κλάση εμφανίζει όλες τις καταχωρήσεις που έχουμε πραγματοποιήσει. Στην ακόλουθη Εικόνα εμφανίζεται ένα μικρό δείγμα (βλ. Εικόνα 50):

Company	Street Address	City	Country	Postal Code	Phone Number	Fax Number	Contact Person	E-mail	Comments
Helecom	561-9262 Iaculis Avenue	Gerakas	Taivan	69.761	+30 8327531			@Yahoo.com	
Sigma Security	P.O. Box 532, 3225 Lacus. Avenue	Peristeri	Taivan	30.084	+30 8323254			@Yahoo.com	
Vinci	141-1287 Adipiscing Avenue	Alimos	Sweeden	51.260	+30 8328472			@Yahoo.com	
Helecom	416-2983 Posuere Rd.	Aigaleo	France	91.602	+30 8325577			@Yahoo.com	
Safenet	Ap #183-928 Scelerisque Road	Ymmitos	United Kingdom	25.730	+30 8325055			@Yahoo.com	
Spectra Tech	P.O. Box 298, 9401 Mauris St.	Koropi	France	78.416	+30 8322297			@Yahoo.com	
Drakos Polemis	6897 Suscipit Rd.	Kifisia	Austria	10.788	+30 8327377			@Yahoo.com	
G4S	Ap #992-5769 Nunc Street	Marousi	United Kingdom	32.738	+30 8321888			@Yahoo.com	
Vinci	2603 Bibendum. Av.	agia Paraskevi	Greece	45.220	+30 8320725			@Yahoo.com	

(Εικόνα 50)

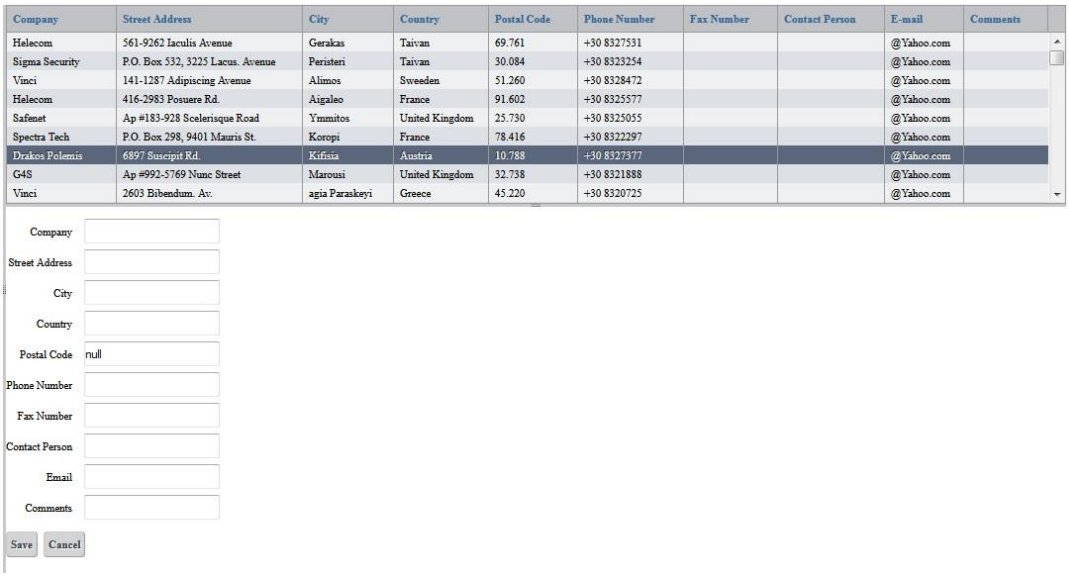
6. List View κλάση

```
package com.example.UserInterface;

import com.vaadin.ui.TextField;
import com.vaadin.ui.VerticalSplitPanel;

public class ListView extends VerticalSplitPanel {

    public ListView(PersonList personList, PersonForm personForm){
        //setSizeFull();

        setFirstComponent(personList);
        setSecondComponent(personForm);
        setSplitPosition(35);
    }
}
```

Η List View κλάση είναι η πιο εύκολη απ' όλες τις κλάσεις που έχουν δημιουργηθεί μέχρι τώρα. Το μόνο που κάνει είναι να δημιουργεί ένα κάθετο Panel δηλ. ένα χώρο στον οποίο τα αντικείμενα που θα προσθέσουμε θα τοποθετούνται το ένα κάτω από το άλλο.

Το Panel χωρίζεται κάθετα στη θέση που ορίζεται από την εντολή «setSplitPosition(35)». Στο επάνω τμήμα απεικονίζεται η κλάση «PersonList» ενώ στο κάτω τμήμα εμφανίζεται η «PersonForm» κλάση.

Στην παρακάτω Εικόνα φαίνεται το αποτέλσμα της συγκεκριμένης κλάσης (βλ. Εικόνα 51):



(Εικόνα 51)

7. Search View κλάση

```
package com.example.UserInterface;

import java.awt.event.FocusEvent;
import java.awt.event.FocusListener;
import java.util.ArrayList;

import com.example.data.PersonCotnainer;
import com.example.data.SearchFilter;
import com.example.diplomatikiFinal.diplomatikiUI;
import com.vaadin.data.Property;
import com.vaadin.data.Property.ValueChangeEvent;
import com.vaadin.data.Property.ValueChangeListener;
import com.vaadin.event.FieldEvents.BlurListener;
import com.vaadin.ui.Alignment;
import com.vaadin.ui.Button;
import com.vaadin.ui.CheckBox;
import com.vaadin.ui.ComboBox;
import com.vaadin.ui.FormLayout;
import com.vaadin.ui.NativeSelect;
import com.vaadin.ui.Notification;
import com.vaadin.ui.Panel;
import com.vaadin.ui.TextField;
import com.vaadin.ui.Button.ClickEvent;
import com.vaadin.ui.Button.ClickListener;

public class Search View extends Panel {

        private TextField text=new TextField("Search");
        private ComboBox choice=new ComboBox("Field Search");
        private CheckBox saveSearch=new CheckBox("Save Search");
        private TextField searchName=new TextField("Search Name");
        private Button search=new Button("Search");
```

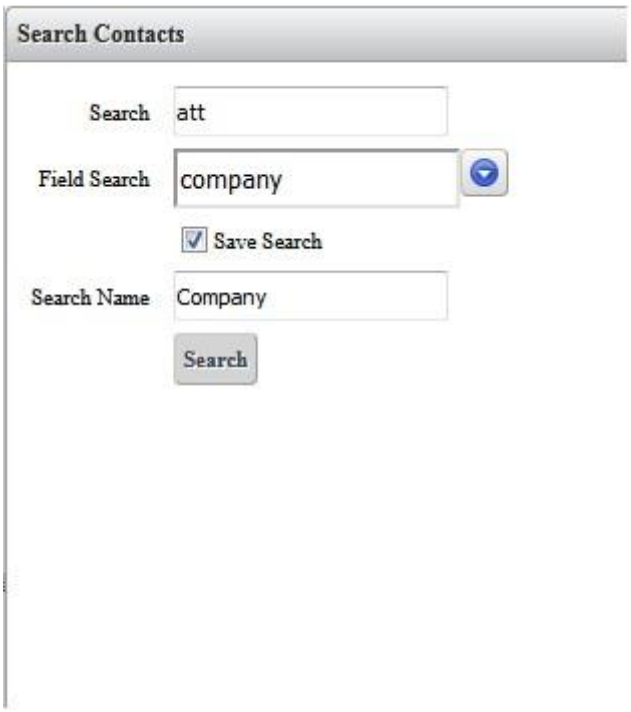
```
private FormLayout layout=new FormLayout();
private diplomatikiUI app;

public SearchView(final diplomatikiUI app){

        this.app=app;

        setCaption("Search Contacts");
        setSizeFull();

        setContent(layout);

        saveSearch.setValue(true);

        choice.addItems(PersonCotnainer.NATURAL_COL_ORDER);

        layout.setSpacing(true);
        layout.setMargin(true);

        choice.setValue("lastName");
        choice.setNullSelectionAllowed(false);

        layout.addComponent(text);
        layout.addComponent(choice);
        layout.addComponent(saveSearch);
        layout.addComponent(searchName);
        layout.addComponent(search);

        saveSearch.setImmediate(true);


saveSearch.addValueChangeListener(new Property.ValueChangeListener() {

    @Override
```

```
        public void valueChange(ValueChangeEvent event) {
                        // TODO Auto-generated method stub
        boolean check=(boolean) event.getProperty().getValue();
            if(check==false){
                searchName.setVisible(false);

            }
            else {
                searchName.setVisible(true);

                }

            }

        });


        search.addClickListener(new ClickListener() {

            @Override
            public void buttonClick(ClickEvent event) {
                performSearch();

                }
        });

        }


public void performSearch(){

String searchTerm=(String) text.getValue();
SearchFilter searchFilter=new SearchFilter(choice.getValue(),searchTerm,
                        (String) searchName.getValue());
```

```
            if(saveSearch.booleanValue()){
                app.saveSearch(searchFilter);
            }
            app.search(searchFilter);

        }

}
```

Η κλάση Search View μας δίνει τη δυνατότητα να αναζητούμε τις επαφές που ικανοποιούν ορισμένα κριτήρια. Για παράδειγμα στο πεδίο «Search» πληκτρολογούμε τη λέξη κλειδί και στο πεδίο «Field Search» επιλέγουμε την στήλη στην οποία θα αναζητηθεί.

Η συγκεκριμένη κλάση μας δίνει τη δυνατότητα να αποθηκεύουμε τα αποτελέσματα μιας αναζήτησης με σκοπό να χρησιμοποιηθούν άμεσα στο μέλλον. Το μόνο που χρειάζεται να κάνουμε είναι να επιλέξουμε το checkbox “Save Search” και να δώσουμε ένα όνομα στο πεδίο “Search Name” (βλ. Εικόνα 52).



(Εικόνα 52)

b) Data πακέτο

1. Person κλάση

```
package com.example.data;

import java.io.Serializable;

public class Person implements Serializable {

        private String company=" ";
        private String streetAddress="";
        private String city=" ";
        private String country=" ";
        private Integer postalCode=null;
        private String phoneNumber=" ";
        private String faxNumber=" ";
        private String contactPerson=" ";
        private String email=" ";
        private String comments=" ";
        public String getCompany() {
                return company;
        }
        public void setCompany(String company) {
                this.company = company;
        }
        public String getStreetAddress() {
                return streetAddress;
        }
        public void setStreetAddress(String streetAddress) {
                this.streetAddress = streetAddress;
        }
        public String getCity() {
                return city;
        }
```

```
public void setCity(String city) {
        this.city = city;
}
public String getCountry() {
        return country;
}
public void setCountry(String country) {
        this.country = country;
}
public Integer getPostalCode() {
        return postalCode;
}
public void setPostalCode(Integer postalCode) {
        this.postalCode = postalCode;
}
public String getPhoneNumber() {
        return phoneNumber;
}
public void setPhoneNumber(String phoneNumber) {
        this.phoneNumber = phoneNumber;
}
public String getFaxNumber() {
        return faxNumber;
}
public void setFaxNumber(String faxNumber) {
        this.faxNumber = faxNumber;
}
public String getContactPerson() {
        return contactPerson;
}
public void setContactPerson(String contactPerson) {
        this.contactPerson = contactPerson;
}
public String getEmail() {
```

```
        return email;
    }
    public void setEmail(String email) {
        this.email = email;
    }
    public String getComments() {
        return comments;
    }
    public void setComments(String comments) {
        this.comments = comments;
    }


}
```

Η κλάση Person περιέχει όλα τα πεδία στα οποία θα καταχωρούμε δεδομένα. Οι μεταβλητές-πεδία είναι αλφαριθμητικά τα οποία έχουν οριστεί ως Private, που σημαίνει ότι μπορούν να χρησιμοποιούνται μόνο από την ίδια κλάση.

Εάν θέλουμε να έχουμε πρόσβαση στα συγκεκριμένα πεδία από άλλη κλάση τότε χρησιμοποιούμε τις συναρτήσεις «gettes» και «settes» που έχουν οριστεί.

2. Person Container κλάση

```
package com.example.data;

import java.io.Serializable;
import java.util.Random;

import com.vaadin.data.util.BeanItemContainer;

public class PersonCotnainer extends BeanItemContainer<Person> implements
Serializable {
```

```
        public static final Object[] NATURAL_COL_ORDER = new Object[] {
        "company", "streetAddress", "city", "country", "postalCode",
"phoneNumber","faxNumber", "contactPerson", "email", "comments" };

        public static final String[] COL_HEADERS_ENGLISH = new String[] {
        "Company", "Street Address", "City", "Country", "Postal Code", "Phone
Number","Fax Number", "Contact Person", "E-mail","Comments" };

        public PersonCotnainer() throws InstantiationException,
IllegalAccessException{
                super(Person.class);
        }

public static PersonCotnainer createTestData() {
        final String[] companyNames = { "Magnetic", "Spectra Tech", "Sigma
Security", "Safenet", "Monitor Electronics", "Ote", "G4S", "Makara", "Kapsch",
"AM2C", "Sunlight", "Drakos Polemis", "Helecom", "Vinci" };

        final String[] cities = { "Athens", "Kifisia", "Peristeri", "Marousi","Koropi",
"Alimos", "Agios Dimitrios", "Marousi", "Ymmitos", "Ano Losia","Xalandri", "agia
Paraskeyi", "Aigaleo", "Gerakas", "Paiania" };

        final String countries [] = { "Greece", "Sweeden", "Austria",
"Germany", "France", "United Kingdom", "Spain", "Belgium","Portugal", "Italy",
"Taivan", "Tokyo", "Turky" };

        final String streetAddresses[] = { "4215 Blandit Av.", "452-8121 Sem
Ave","279-4475 Tellus Road", "4062 Libero. Av.", "7081 Pede. Ave",
"6800 Aliquet St.", "P.O. Box 298, 9401 Mauris St.","161-7279 Augue Ave", "P.O. Box
496, 1390 Sagittis. Rd.","448-8295 Mi Avenue", "6419 Non Av.",
"659-2538 Elementum Street", "2205 Quis St.","252-5213 Tincidunt St.", "P.O. Box
175, 4049 Adipiscing Rd.","3217 Nam Ave", "P.O. Box 859, 7661 Auctor St.","2873
Nonummy Av.", "7342 Mi, Avenue","539-3914 Dignissim. Rd.", "539-3675 Magna
```

```
Avenue","Ap #357-5640 Pharetra Avenue", "416-2983 Posuere Rd.","141-1287
Adipiscing Avenue", "Ap #781-3145 Gravida St.",
"6897 Suscipit Rd.", "8336 Purus Avenue", "2603 Bibendum. Av.","2870 Vestibulum
St.", "Ap #722 Aenean Avenue","446-968 Augue Ave", "1141 Ultricies Street","Ap
#992-5769 Nunc Street", "6690 Porttitor Avenue",
"Ap #105-1700 Risus Street","P.O. Box 532, 3225 Lacus. Avenue", "736 Metus
Street","414-1417 Fringilla Street", "Ap #183-928 Scelerisque Road",
"561-9262 Iaculis Avenue" };

PersonCotnainer c = null;
Random r = new Random(0);
try {
        c = new PersonCotnainer();
        for (int i = 0; i < 100; i++) {
            Person p = new Person();
            p.setCompany(companyNames[r.nextInt(companyNames.length)]);
            p.setCity(cities[r.nextInt(cities.length)]);
            p.setCountry(countries[r.nextInt(countries.length)]);
            p.setEmail(p.getContactPerson().toLowerCase() + "@Yahoo.com");
            p.setPhoneNumber("+30 832" + r.nextInt(10) + r.nextInt(10)
                                + r.nextInt(10) + r.nextInt(10));
            int n = r.nextInt(100000);
            if (n < 10000) {
                n += 10000;
             }
                p.setPostalCode(n);
                p.setStreetAddress(streetAddresses[r.nextInt(streetAddresses.length)]);
                c.addItem(p);
           }
        } catch (InstantiationException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
        } catch (IllegalAccessException e) {
                        // TODO Auto-generated catch block
```

```
                    e.printStackTrace();
            }

        return c;
    }

}
```

Ο σκοπός της κλάσης αυτής είναι να δημιουργεί 100 επαφές. Τα δεδομένα που εμφανίζονται σε κάθε πεδίο επιλέγονται με τυχαίο τρόπο από τους πίνακες company Names, cities, countries, streetAddressess.

3. Search Filter κλάση

```
package com.example.data;

import java.io.Serializable;

public class SearchFilter implements Serializable {

        private final String term;
        private final Object propertyId;
        private String searchName;

public SearchFilter (Object propertyId, String searchTerm, String name){
                this.propertyId=propertyId;
                this.term=searchTerm;
                this.searchName=name;
        }

public String getSearchName() {
                return searchName;
        }
```

```
public void setSearchName(String searchName) {
		this.searchName = searchName;
	}

public String getTerm() {
		return term;
	}


public Object getPropertyId() {
		return propertyId;
	}

public String toString(){
		return getSearchName();
	}

}
```

c) <u>Diplomatiki Final πακετο</u>

1. <u>Diplomatiki UI κλάση</u>

```
package com.example.diplomatikiFinal;

import javax.servlet.annotation.WebServlet;

import com.example.UserInterface.HelpWindow;
import com.example.UserInterface.ListView;
import com.example.UserInterface.NavigationTree;
import com.example.UserInterface.PersonForm;
import com.example.UserInterface.PersonList;
import com.example.UserInterface.SearchView;
import com.example.UserInterface.ShareWindow;
```

```
import com.example.data.PersonCotnainer;
import com.example.data.SearchFilter;

import com.vaadin.annotations.Theme;
import com.vaadin.annotations.VaadinServletConfiguration;
import com.vaadin.data.Item;
import com.vaadin.data.Property;
import com.vaadin.data.Property.ValueChangeEvent;
import com.vaadin.event.ItemClickEvent;
import com.vaadin.event.ItemClickEvent.ItemClickListener;
import com.vaadin.server.VaadinRequest;
import com.vaadin.server.VaadinServlet;
import com.vaadin.ui.Button;
import com.vaadin.ui.Component;
import com.vaadin.ui.HorizontalLayout;
import com.vaadin.ui.HorizontalSplitPanel;
import com.vaadin.ui.Label;
import com.vaadin.ui.Notification;
import com.vaadin.ui.TextField;
import com.vaadin.ui.UI;
import com.vaadin.ui.VerticalLayout;
import com.vaadin.ui.Button.ClickEvent;

/**
 * This UI is the application entry point. A UI may either represent a browser window
 * (or tab) or some part of a html page where a Vaadin application is embedded.
 * <p>
 * The UI is initialized using {@link #init(VaadinRequest)}. This method is intended to
be
 * overridden to add component to the user interface and initialize non-component
functionality.
 */
@Theme("mytheme")
public class diplomatikiUI extends UI implements Button.ClickListener,
```

```
Property.ValueChangeListener, ItemClickListener{

        private Button add_contact=new Button("Add Contact");
        private Button search=new Button("Search");
        private Button share=new Button("Share");
        private Button help=new Button("Help");
        VerticalLayout mainLayout=new VerticalLayout();
        HorizontalSplitPanel medSection=new HorizontalSplitPanel();

        NavigationTree navigationTree=new NavigationTree(this);
        private ListView listView=null;
        private PersonList personList=null;
        private PersonForm personForm=null;
        private HelpWindow window=null;
        private SearchView searchView=null;

        private PersonCotnainer dataSource=PersonCotnainer.createTestData();

    @Override
    protected void init(VaadinRequest vaadinRequest) {

        creatingMainLayout();

            }

    @WebServlet(urlPatterns = "/*", name = "diplomatikiUIServlet", asyncSupported =
true)
    @VaadinServletConfiguration(ui = diplomatikiUI.class, productionMode = false)
    public static class diplomatikiUIServlet extends VaadinServlet {
    }

// ----------  Creating Functions ----------------------------------
```

```
//////// start  function creatingMainLayout ////////////////////////////
    public void creatingMainLayout(){
            mainLayout.setSpacing(true);
            mainLayout.setMargin(true);
            mainLayout.addComponent(createToolbar());
            mainLayout.addComponent(medSection);
            mediumSection();
            mainLayout.setExpandRatio(medSection,1);
            mainLayout.setSizeFull();
            setContent(mainLayout);
            setTheme("liferay");

            setMainComponent(getListView());

    }

    //////// end creatingMainLayout ////////////////////////////////


    //////// start  function createToolbar ////////////////////////////

    public HorizontalLayout createToolbar(){
    HorizontalLayout layoutToolbar=new HorizontalLayout();
    layoutToolbar.setSpacing(true);
    layoutToolbar.setMargin(true);
    layoutToolbar.addComponent(add_contact);
    layoutToolbar.addComponent(search);
    layoutToolbar.addComponent(share);
    layoutToolbar.addComponent(help);

//---------------add_contact click Listener ////////////////////////
```

```
add_contact.addClickListener(new Button.ClickListener() {

        @Override
        public void buttonClick(ClickEvent event) {
            // TODO Auto-generated method stub
            final Button source= event.getButton();
            if(source==add_contact){
                addNewContact();
            }
        }
    });


////////////end add_contact click Listener ////////////////////


//// help click Listener ////////////////////////////////////////////////

  help.addClickListener(new Button.ClickListener() {

        @Override
        public void buttonClick(ClickEvent event) {
            // TODO Auto-generated method stub
            window=new HelpWindow();
            addWindow(window);
            //window.setSizeUndefined();


        }
    });

 /////////// end help click Listener ////////////////////////

////share click Listener ////////////////////////////////////////////////
```

```
share.addClickListener(new Button.ClickListener() {

        @Override
        public void buttonClick(ClickEvent event) {
            // TODO Auto-generated method stub

            ShareWindow share=new ShareWindow();
            addWindow(share);

        }
    });

////////////// end share click Listener //////////////////////////

//----------------------------------------------------------------


/////////////////////////////Search click Listener ///////////////////////


search.addClickListener(new Button.ClickListener() {

        @Override
        public void buttonClick(ClickEvent event) {
            // TODO Auto-generated method stub
            final Button source=event.getButton();
            if(source==search)
                showSearchView();

        }
    });


////////////////////////// Search click Listener ////////////////////////////
```

```
    return layoutToolbar;
    }

    ///////// end createToolbar ////////////////////////////////

//////// start  function mediumSection ////////////////////////////////

    public void mediumSection(){
            medSection.setSizeFull();
            medSection.setSplitPosition(20);
            medSection.setFirstComponent(navigationTree);


    }

    ///////// end mediumSection ////////////////////////////////

    ///////// start  setMainComponent  ////////////////////////////////

    public void setMainComponent(Component c){
            medSection.setSecondComponent(c);


    }

    ///////// end setMainComponent ////////////////////////////////

    ///////// start  getListView  ////////////////////////////////

    public ListView getListView(){
```

```
        if (listView==null) {

                personList=new PersonList(this);
                personForm=new PersonForm(this);
                listView=new ListView(personList, personForm);

        }

        return listView;

  }

  ///////// end getListView  ////////////////////////////////

 ///////// start  getDataSource  ////////////////////////////////

  public PersonCotnainer getDataSource(){
        return dataSource;
  }

 //////////// end getDataSource ////////////////////////

 ///////// start  getSearchView  ////////////////////////////////

    public SearchView getSearchView(){
          if(searchView==null)
                searchView=new SearchView(this);
        return searchView;
    }

  /// //////////// end getSearchView ////////////////////////

 ////////////////////////////////////////////////////////////////////////
```

```
        @Override
        public void buttonClick(ClickEvent event) {
                // TODO Auto-generated method stub
                Button button=event.getButton();
                if(button==search)
                        showSearchView();

        }

   ////////////// start  showSearchView ////////////////////////////////

        public void showSearchView(){
                setMainComponent(getSearchView());
        }


////////////////// end showSearchView ////////////////////////////////

        @Override
        public void valueChange(ValueChangeEvent event) {
                // TODO Auto-generated method stub
                Property property=event.getProperty();
                if(property==personList){
                        Item item=personList.getItem(personList.getValue());
                        if(item!=personForm.getItemDataSource()){
                                personForm.setItemDataSource(item);
                        }
                }
        }

////////////////////////////////////////////////////////////////////

        @Override
```

```
public void itemClick(ItemClickEvent event) {
        // TODO Auto-generated method stub
        if(event.getSource()==navigationTree){
                Object itemId=event.getItemId();
                if(itemId!=null){
                        if(NavigationTree.SHOW_ALL.equals(itemId)){
                                getDataSource().removeAllContainerFilters();
                                showListView();
                        }
                        else if(NavigationTree.SEARCH.equals(itemId)){
                                showSearchView();
                        }
                        else if(itemId instanceof SearchFilter){
                                search((SearchFilter) itemId);
                        }
                }
        }

}

////////////////////////////////////////////////////////////////////////////

public void showListView(){
        setMainComponent(getListView());
}

////////////////////////////////////////////////////////////////////////////

public void addNewContact(){
        showListView();
        personForm.addContact();
}

public void search(SearchFilter searchFilter) {
```

```
            // TODO Auto-generated method stub
            getDataSource().removeAllContainerFilters();
            getDataSource().addContainerFilter(searchFilter.getPropertyId(),
                        searchFilter.getTerm(), true, false);
            showListView();

        }

        public void saveSearch(SearchFilter searchFilter) {
            // TODO Auto-generated method stub
            navigationTree.addItem(searchFilter);
            navigationTree.setParent(searchFilter, NavigationTree.SEARCH);
            navigationTree.setChildrenAllowed(searchFilter, false);
            navigationTree.expandItem(NavigationTree.SEARCH);
            navigationTree.setValue(searchFilter);

        }

}
```

Βιβλιογραφία

- Alejandro Duarte, “Vaadin 7 UI Design By Example”, Packt Publishing, July 2013 - ISBN: 978-1782162261.
- Nicolas Frankel, “Learning Vaadin 7, Second Edition”, Packt Publishing, September 2013 - ISBN: 978-1782169772.
- Jaroslav Holan, Ondrej Kvasnovsky, “Vaadin 7 Cookbook”, Packt Publishing, April 2013 - ISBN: 978-1849518802
- Nicolas Frankel, “Learning Vaadin”, Packt Publishing, October 2011 - ISBN: 978-1849515221.
- “BOOK OF VAADIN - VAADIN 7 EDITION”, 2013 - ISBN: 978-9529319701
- “Book of Vaadin”, Oy IT Mill Ltd, 2010