****

**College of Engineering**

**Computer Engineering Department**

**CMPE 275 – Enterprise Software Components**

**Class Project**

**Stock Mentor**

**Team**

**Fan Ieong (006450171)**

**Harini Kasturi (007650071)**

**Stuti Vora (005067440)**

**Sudershan Malpani (008067878)**

**Vidushi Desai (008025420)**

**May 8, 2012**

Table of Contents

[Introduction 2](#_Toc324412486)

[Goal of the project 2](#_Toc324412487)

[Purpose of the System 2](#_Toc324412488)

[System Design 3](#_Toc324412494)

[Components Explained 4](#_Toc324412499)

[Graphical User Interface 4](#_Toc324412500)

[Application Controller 5](#_Toc324412504)

[Indexer 5](#_Toc324412506)

[Web Crawler 6](#_Toc324412508)

[Data processor 7](#_Toc324412510)

[Data classifier(Data Mining) 7](#_Toc324412515)

[Data provider 9](#_Toc324412531)

[Information generator 9](#_Toc324412533)

[Prediction Algorithm (Machine Learning) 10](#_Toc324412535)

[Technologies Explored 12](#_Toc324412543)

[Vaadin 12](#_Toc324412544)

[Yarfraw 14](#_Toc324412548)

[Cassandra 15](#_Toc324412550)

[JSoup 16](#_Toc324412554)

[Snapshots 17](#_Toc324412555)

[Installation and Execution Manual 20](#_Toc324412556)

[References 31](#_Toc324412588)

# Introduction

* **Goal of the project**

The goal of the project is to develop a system to understand the Component Based architecture and working with container managed components. By decoupling the functionalities in component based system the main aim is to achieve system granularity with better performance. By applying these concepts to real world problems, understanding the needs for large datasets. To gain technical insights by working with different new technologies such as Vaadin, Yarfraw, Gson/Json and Cassandra.

We can also say that goal of the project is to come up with correlations between types of news (classification) and stock movements. This trend analysis will establish base line of impacts of news on stock price. The second part is to come up with a predictive model based on a certain event in the future. Such analyses are **common techniques in computer industry** in network traffic analysis, performance analysis, and etc. You will gain insights on how the base line is established and how the predictive model would be generated.

* **Purpose of the System**

The objective of this project is to analyze the trend of the stock over time, as in what affected the movement of the stock correlated between types of news (classifications) and stock movements. This trend analysis will establish a base line for the impacts of news and events on stock price. The second part is to come up with a predictive model based on a certain event in the future. To gain insights on how the base line is established and how the predictive model would be generated.

The key aspects that will be demonstrated in this project:

i. Textual search and pattern matching

ii. Data cleaning using common statistical techniques

iii. Data summary and reporting to assist the user in making an informed decision

# System Design

**Figure 1. Logical Design**

Figure 1. Shows the logical design of the Stock Trend Analyzer. The client will enter a particular query. The system will then crawls and index several news and events and provide baseline and predictive analysis of that company’s stock.



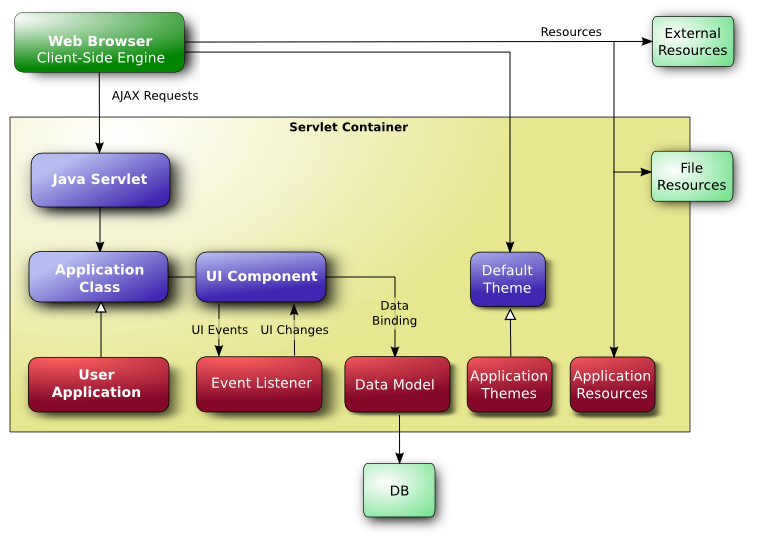
**Figure 2. Physical Design**

Figure 2. Shows the physical design of Stock Trend Analyzer. The detailed design is explained below.

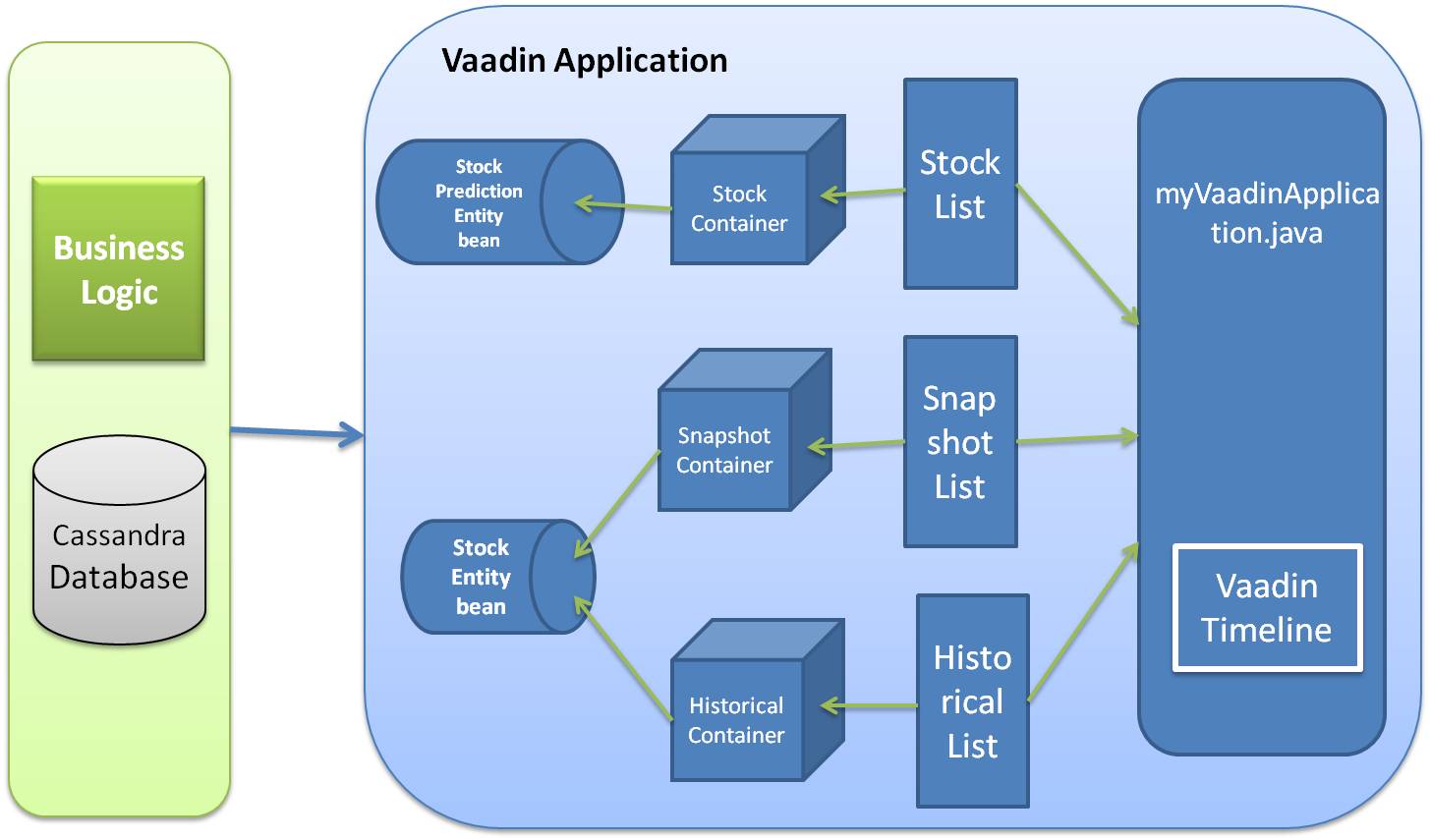
# Components Explained

1. **Graphical User Interface**

The user enters a stock symbol to query the baseline model (historical stock data) and trend analysis. The User Interface also shows real time stock feeds and real time market news. The input is then sent to and handled by Application Controller component. Vaadin, a Java GUI framework, is used for the User Interface.



The entry-point of the system is the application class, which needs to create and manage all necessary user interface components, including windows. User interaction is handled with event listeners, simplified by binding user interface components directly to data. Visual appearance is defined in themes as CSS files. Icons, other images, and downloadable files are handled as *resources*, which can be external or served by the application server or the application itself.



The Vaadin application contains a set of java files which are different in nature and together makes a dynamic, scalable and reusable front end. There is a main java file which extends Vaadin application. It uses all other components to render the data on the browser. In our design, we have two entity beans. One is StockPrediction entity bean, which is used by StockContainer and Stock Predictions are displayed using StockList. The other bean is Stock Prediction bean, which is used by snapshot container and historical container to show snapshot and historical data of the particular ticker value requested by user in the mainVaadinApplication.

* **Application Controller**

The application controller takes the query and passes it to the indexer.

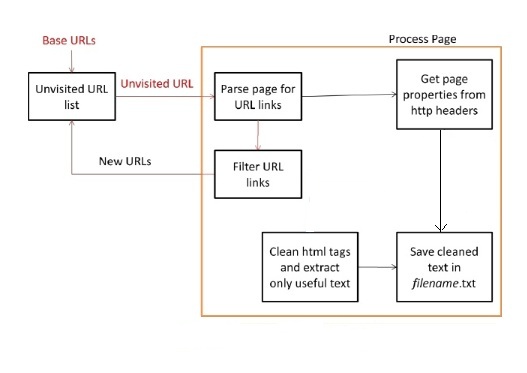
* **Indexer**

The indexer collects the information about the values of the specified fields in the documents of a collection. This component uses Cassandra’s key value database and query optimizer to quickly sort through and order the documents in the collection. These indexes are implemented as B-Tree indexes. It passes the key to the Data Processor component and gets information from Information generator component.

* **Web Crawler**

A Web crawler is a processor/parser of web pages that look for hypertext links and scrape contents of the file contents stored in that particular location. In this case our interest will be the Yahoo! Finance web page. The web crawler is responsible for providing data to the following components:

1. Market News: web crawler parses a specified web page, e.g. Yahoo! Finance (<http://finance.yahoo.com/>) and (recursively) following any links and is on the web page, to obtain news articles.
2. Real time feeds: web crawler processes the Yahoo! Finance RSS news feed (http://finance.yahoo.com/news/category-stocks/rss) to obtain real time new feeds.
3. Stock quotes: web crawler processes the Yahoo! Finance stock live feed to obtain real time stock data.



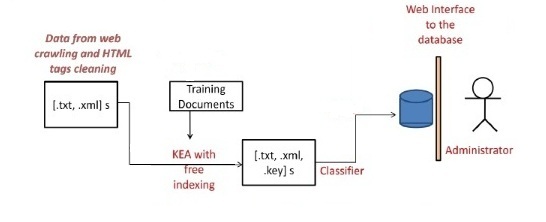
In our system, the web crawler is written with the Yarfraw and Jsoup API. We have used Yarfraw for retrieving URL’s of market news RSS feeds. We also used JSoup, a JSON-based tool, to crawl and retrieve textual data of those articles. All of the articles are stored in a Cassandra Database for further processing.

* **Data processor**

Data processor is responsible for parsing stock quotes feed and new article feeds. It is also responsible of discovering the ‘trend’ of each stock, based on real time and historical data.

* Stock quotes: processed and stored into RDB.
* New articles from RSS Feeds: processed and stored into a RDB table. Using the stock symbol as an index, the article table and the stock quotes table are correlated.
* **Data classifier (Document Classifier)**

We used KEA (Keyword Extraction Algorithm), a Naïve-Bayes algorithm based Document Classifier to extract keywords from new articles. KEA supported vocabulary-based keywords extraction as well as index-free keywords extraction. In our system, we used the index-free extraction tool. The diagram below describes the keywords extraction mechanism:



It is responsible of categorizing stock related news articles into two categories**. So for this the Data is separated into training set, testing set and real time set.** Historical data and financial news from a set range of time (one month worth of data and news in this project) are used to train the system:

1. Good news, something business-positive, better than expected earnings, a new contract, the expectation of new business, the acquiring of key personnel, etc.
2. Bad news, something financially detrimental to the company or its industry, unexpected poor earnings, loss of key clients, loss of key personnel, announcement of bankruptcy, unusual insider selling, etc.

|  |  |
| --- | --- |
| **Keywords** | **Word Counts** |
| universal entertainment | 57 |
| counterclaim | 58 |
| yen | 59 |
| jgbs | 60 |
| thomson | 61 |
| edt | 62 |
| omnicell | 63 |
| nasdaq | 64 |
| adjusted ebitda | 65 |
| ebitda | 66 |
| prior year | 67 |
| prior year period | 68 |
| sim city | 69 |

The word counts were weighted by the stock price change for the article date.

*foreach* (article *in* training set)

*foreach* (word *in* article)

word relevance = Sum over paragraphs

*end foreach*

Total relevance = Sum over all words in article (word relevance)

Multiple each word's relevance by (daily % change for stock / Total relevance)

Add each word relevance to total score for word

*end foreach*

*foreach* (word *in* data)

normalize - divide word's score by number of articles word has appeared in, to get an average score per article

*end foreach*

The output is a map of (words, scores). The returned score is the predicted daily percentage change for the stock in question. We chose this way of weighting the scores so that if a predictor was trained on exactly one word bag and then asked to predict based on that word bag, it would return the actual change for that day exactly.

|  |  |  |
| --- | --- | --- |
| **Keywords** | **Percent Change** | **Word Counts** |
| universal entertainment | -6.1 | 57 |
| counterclaim | -7.1 | 58 |
| yen | 2.5 | 59 |
| jgbs | -2.4 | 60 |
| thomson | -4.3 | 61 |
| edt | 4.3 | 62 |
| omnicell | 2 | 63 |
| nasdaq | 2.3 | 64 |
| adjusted ebitda | 2.5 | 65 |
| ebitda | 3.4 | 66 |
| prior year | -4 | 67 |
| prior year period | -3 | 68 |
| sim city | -9.2 | 69 |

* **Data provider**

The Data provider is responsible for parsing data, typically stock quotes into and from various formats.

* **Information generator**

When information about a particular stock is requested, the information generator applies the data classifier to align new articles related to that stock and uses the prediction algorithm to apply the ‘weight’ of each article to the stock, to provide a prediction of the upcoming trend.

* **Prediction algorithm (Machine Learning)**

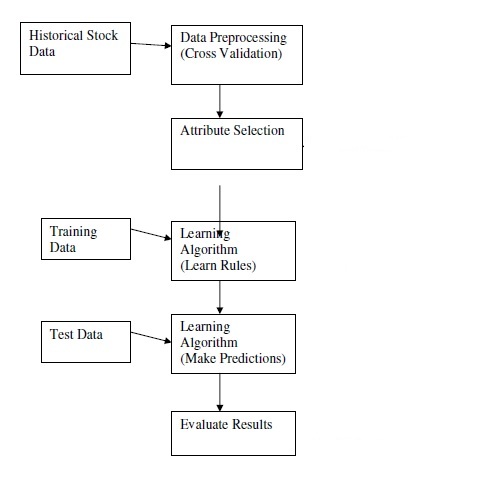
It is an algorithm used to assign a ‘weight’ to each news category. When making a prediction on a test article, the algorithm is:

score = 0

*foreach* (word *in* article)

score += (word score) \* Sum over paragraphs (word count for paragraphs)

*endforeach*



**Training phase**: Used historical stock data as well as news archive for the month of March 2012 to test and tune the prediction model. Come up with ‘prediction accuracy’ based on historical data.

**Testing phase**: Another set of historical data and financial news for the month of April 2012 is then used to created prediction data based on our algorithm, to analyze the accuracy of the system. The real time set is what the user will use. Getting live RSS news feed and stock quotes from yahoo finance, the system will make a prediction of whether the stock will rise or drop, by percentage.

**Operational phase**: Prediction model is used to align live stock data and news feed, and come up with upcoming trend.

|  |  |  |
| --- | --- | --- |
| Date | Symbol | Prediction |
| 2012-03-01 | IBM | -2.3 |
| 2012-03-02 | IBM | 0.5 |
| 2012-03-05 | IBM | -1.3 |
| 2012-03-06 | IBM | -1.4 |
| 2012-03-07 | IBM | 0.9 |
| 2012-03-08 | IBM | 2.5 |
| 2012-03-09 | IBM | 1.6 |

# Technologies Explored

* **Vaadin**
* **Introduction to Vaadin**

**Vaadin is a web application framework for Rich Internet Applications (RIA**). In contrast to Javascript libraries and browser-plugin based solutions, it features robust server-side architecture. This means that the largest part of the application logic runs securely on the server. [Google Web Toolkit (GWT)](http://code.google.com/webtoolkit/) is used on the browser side to ensure a rich and fluent user experience.

**Vaadin is a large collection of UI components.** You compose the application user interface from components such as [Buttons](http://demo.vaadin.com/sampler/#Buttons), [Tables](http://demo.vaadin.com/sampler/#GridsAndTrees), [Trees](http://demo.vaadin.com/sampler/#GridsAndTrees) and [Layouts](http://demo.vaadin.com/sampler#ComponentContainers). The components use events, listeners and data binding to communicate with each other and with your business logic.

**Vaadin is a robust architecture for rapid application development**. The component-based architecture together with statically typed Java language and data binding features help you build applications that are easily modularized and refactored as needed. The [IDE and tooling support](http://vaadin.com/tooling) including [visual designing tool](http://vaadin.com/eclipse/#visual-designer) help you to build web user interface extremely fast.

* **Architecture of Vaadin**

The core piece of the Vaadin Framework is the Java library that is designed tmake creationand maintenance of high quality web-based user interfaces easy. The key idea in the serverdriven programming model of Vaadin is that it lets you forget the web and program user interfaces much like you would program any Java desktop application with conventional toolkits such as AWT, Swing, or SWT. But easier. While traditional web programming is a fun way to spend your time learning new web technologies, you probably want to be productive and concentrate on the application logic. With the serverdriven programming model, Vaadin takes care of managing the user interface in the browser and AJAX communications between the browser and the server. With the Vaadin approach, you do not need to learn and debug browser technologies, such as HTML or JavaScript.



Figure: General Architecture of Vaadin

Vaadin consists of the server-side framework and a client-side engine that runs in the browser as a JavaScript program, rendering the user interface and delivering user interaction to the server. The application runs as a Java Servlet session in a Java application server.

Because HTML, JavaScript, and other browser technologies are essentially invisible to the application logic, you can think of the web browser as only a thin client platform. A thin client displays the user interface and communicates user events to the server at a low level. The control logic of the user interface runs on a Java-based web server, together with your business logic. By contrast, normal client-server architecture with a dedicated client application would include a lot of application specific communications between the client and the server. Essentially removing the user interface tier from the application architecture makes our approach a very effective one. As the Client-Side Engine is executed as JavaScript in the browser, no browser plugins are needed for using applications made with Vaadin. This gives it a sharp edge over frameworks based on Flash, Java Applets, or other plugins. Vaadin relies on the support of GWT for a wide range of browsers, so that the developer doesn't need to worry about browser support.

Behind the server-driven development model, Vaadin makes the best use of AJAX (Asynchronous JavaScript and XML) techniques that make it possible to create Rich Internet Applications (RIA) that are as responsive and interactive as desktop applications.

Hidden well under the hood, Vaadin uses GWT, the Google Web Toolkit, for rendering the user interface in the browser. GWT programs are written in Java, but compiled into JavaScript, thus freeing the developer from learning JavaScript and other browser technologies. GWT is ideal for implementing advanced user interface components (or widgets in GWT terminology) and interaction logic in the browser, while Vaadin handles the actual application logic in the server. Vaadin is designed to be extensible, and you can indeed use any 3rd-party GWT components easily, in addition to the component repertoire offered in Vaadin.

* **Vaadin Goals**

Simply put, Vaadin's ambition is to be the best possible tool when it comes to creating web user interfaces for business applications.

**Right tool for the right purpose**

Because our goals are high, the focus must be clear. This toolkit is designed for creating web applications. It is not designed for creating websites or advertisements demos. For such purposes, you might find (for instance) JSP/JSF or Flash more suitable.

**Simplicity and maintainability**

We have chosen to emphasize robustness, simplicity, and maintainability. This involves following the well-established best practices in user interface frameworks and ensuring that our implementation represents an ideal solution for its purpose without clutter or bloat.

**XML is not designed for programming**

The Web is inherently document-centered and very much bound to the declarative presentation of user interfaces. The Vaadin framework frees the programmer from these limitations. It is far more natural to create user interfaces by programming them than by defining them in declarative templates, which are not flexible enough for complex and dynamic user interaction.

**Tools should not limit your work**

There should not be any limits on what you can do with the framework: if for some reason the user interface components do not support what you need to achieve, it must be easy to add new ones to your application. When you need to create new components, the role of the framework is critical: it makes it easy to create re-usable components that are easy to maintain.

* **Yarfraw**

We use YARFRAW API library to read RSS feeds from finance websites like Yahoo! finance. We have downloaded and explored the features of YARFRAW. Since it supports various data formats like atom format and RSS/RDF, it is very convenient to use since we won't have to worry about the conversions while parsing the data.

* **Cassandra**

The best features of relational databases, key-value stores and document databases. In it the database is known as database only, however the databases and tables are known as key spaces and column families. We can query dynamically in this database. Cassandra likes to fit its indexes in memory.

Its features are:

* 1. It is fast and smart
  2. Has good writing speed than Mongo DB
  3. Scalable
     1. Replication
     2. Auto-Sharding: Cassandra scales horizontally via an auto-sharding (partitioning) architecture.

Cassandra sharding provides:

* Automatic balancing for changes in load and data distribution
* Easy addition of new machines without down time
* Scaling to one thousand nodes
* No single points of failure
* Automatic failover

3. It is a document oriented database, using document based queries (uses JSON and queries are expressed in JSON/javascript).

4. Schema Free.

5. Is Performant

* 1. Full index support
  2. No Transactions (has atomic operations)
  3. Memory mapped files (delayed writes or eventual consistency)

6. Its normalized,

7. Is embedded i.e. uses foreign keys as in RDBMS

**Performance:**

For write operations MongoDB uses one core and for read it uses all CPU cores available to it. So one could notice higher speed of MongoDB from single to dual to quad core CPU’s. For performance reasons we can also do connection pooling in MongoDB. But ultimately the writing speed is much higher in Cassandra.

We used Hector API in Java for calling or querying in Cassandra DB. We have done indexing in several columns of column families so that the searching will become indexed.

* **JSoup**

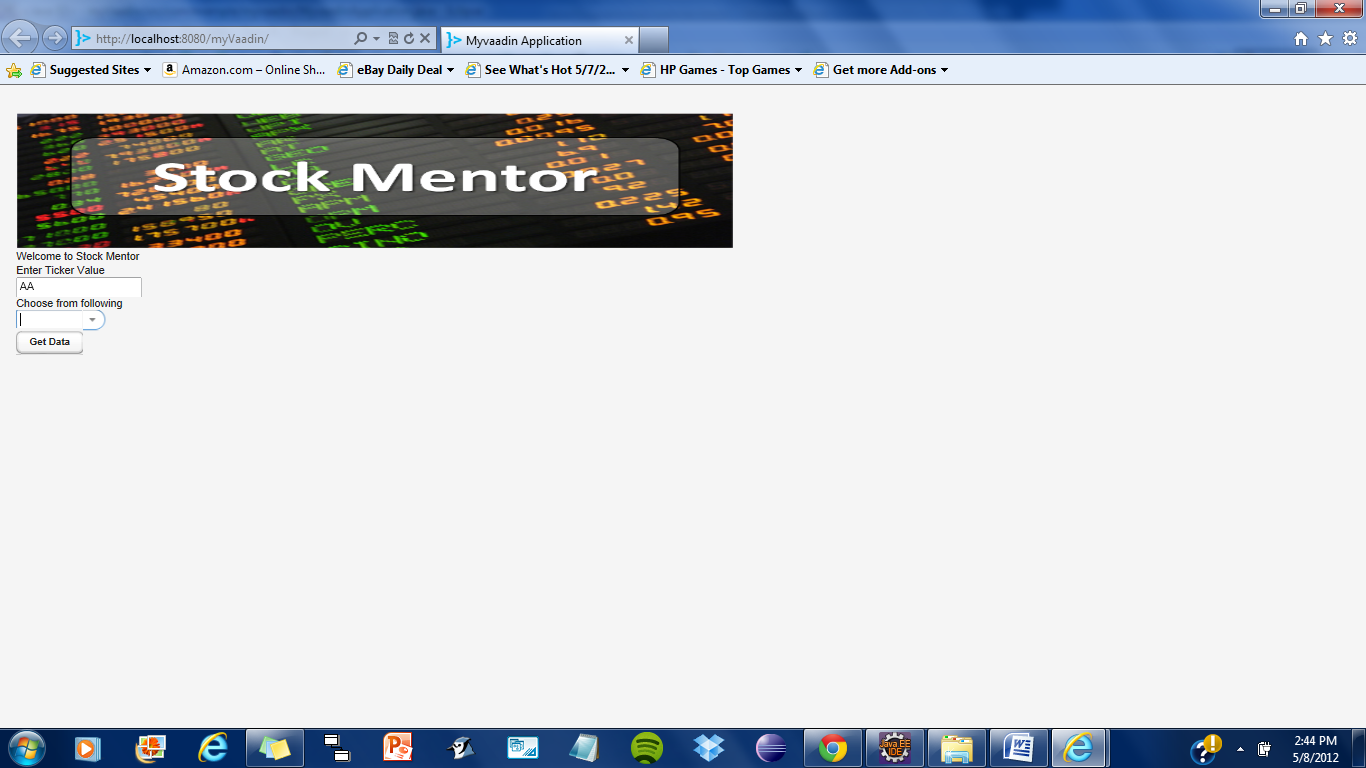
JSoup a Java library for working with real-world HTML. It provides a very convenient API for extracting and manipulating data, using the best of DOM, CSS, and jquery-like methods.

JSoup implements the WHATWG HTML5 specification, and parses HTML to the same DOM as modern browsers do.

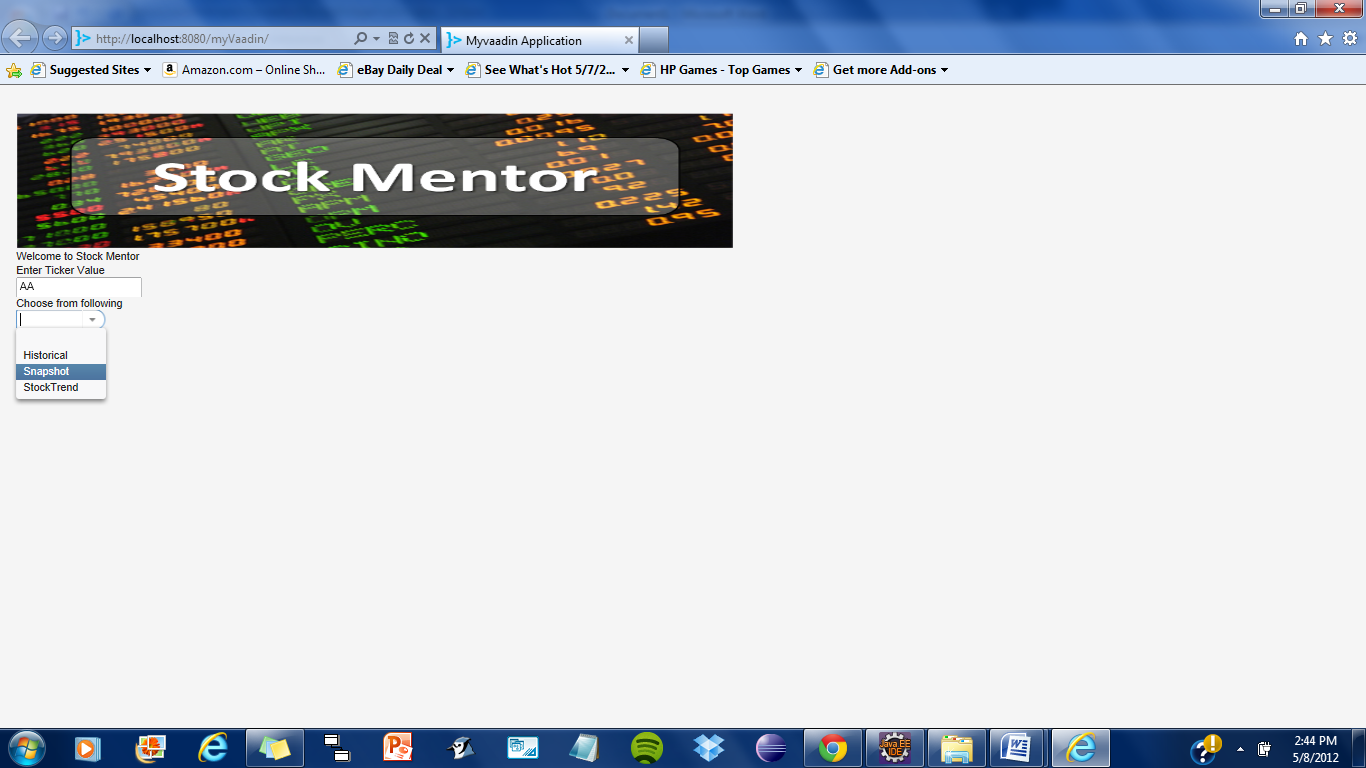
* Scrape and parse HTML from a URL, file, or string
* Find and extract data, using DOM traversal or CSS selectors
* Manipulate the HTML elements, attributes, and text
* Clean user-submitted content against a safe white-list, to prevent XSS attacks
* Output tidy HTML

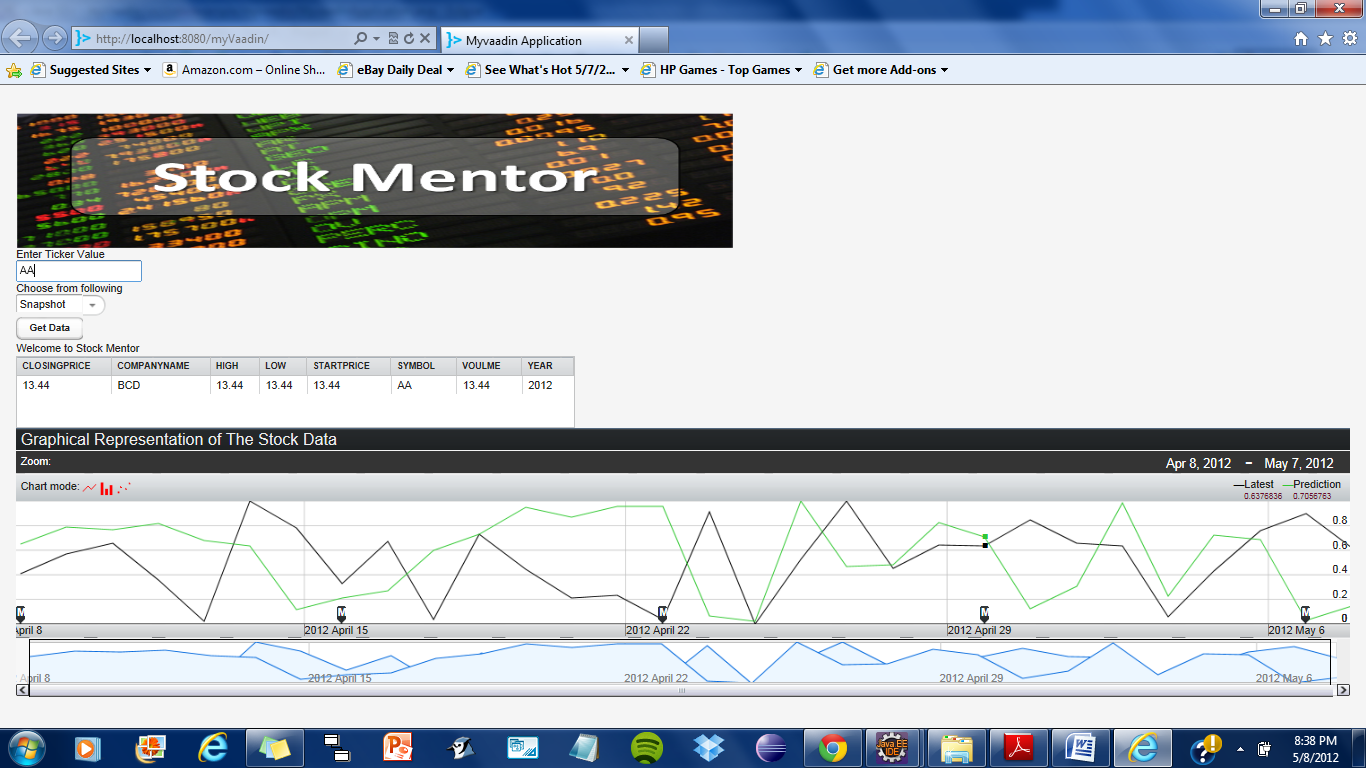
jSoup is designed to deal with all varieties of HTML found in the wild; from pristine and validating, to invalid tag-soup; jsoup will create a sensible parse tree.

# Snapshots

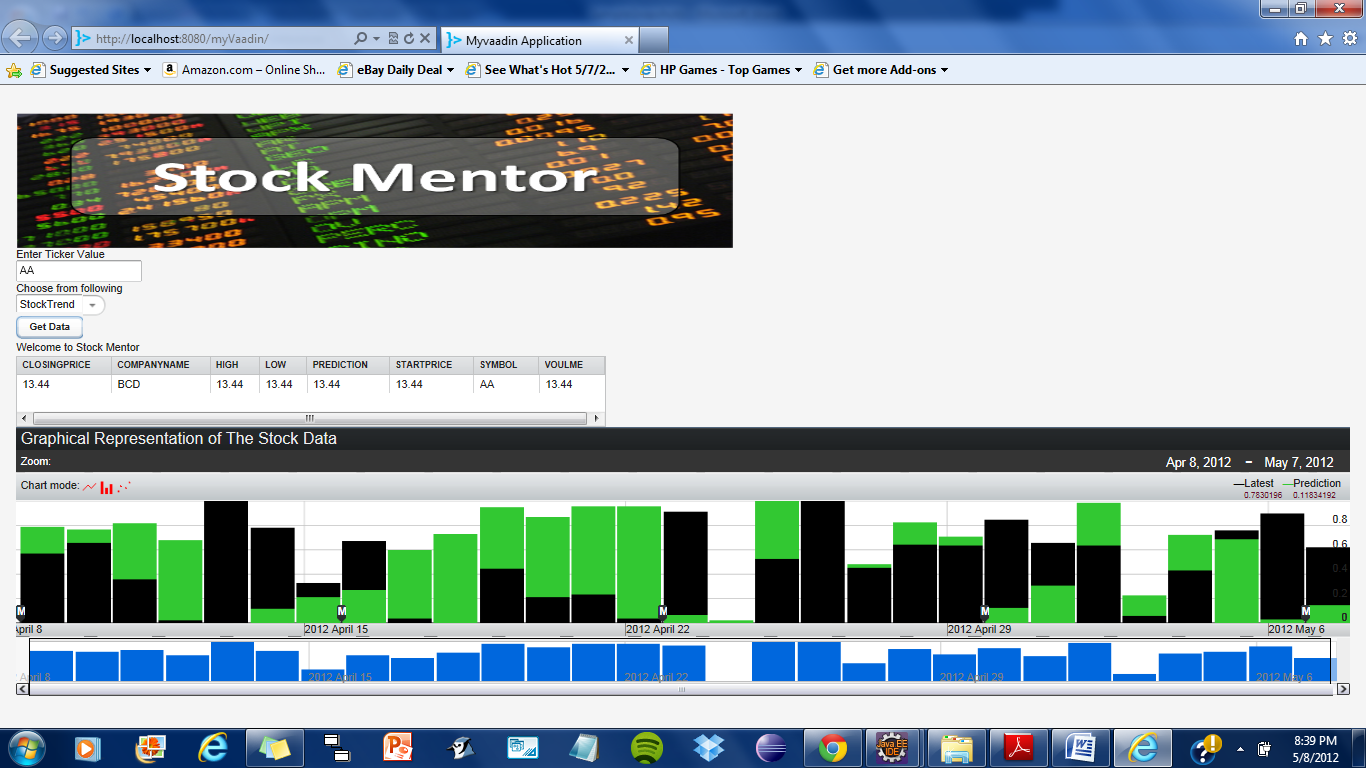


The main page: Shows the list of type of StockData request and the field to enter ticker value

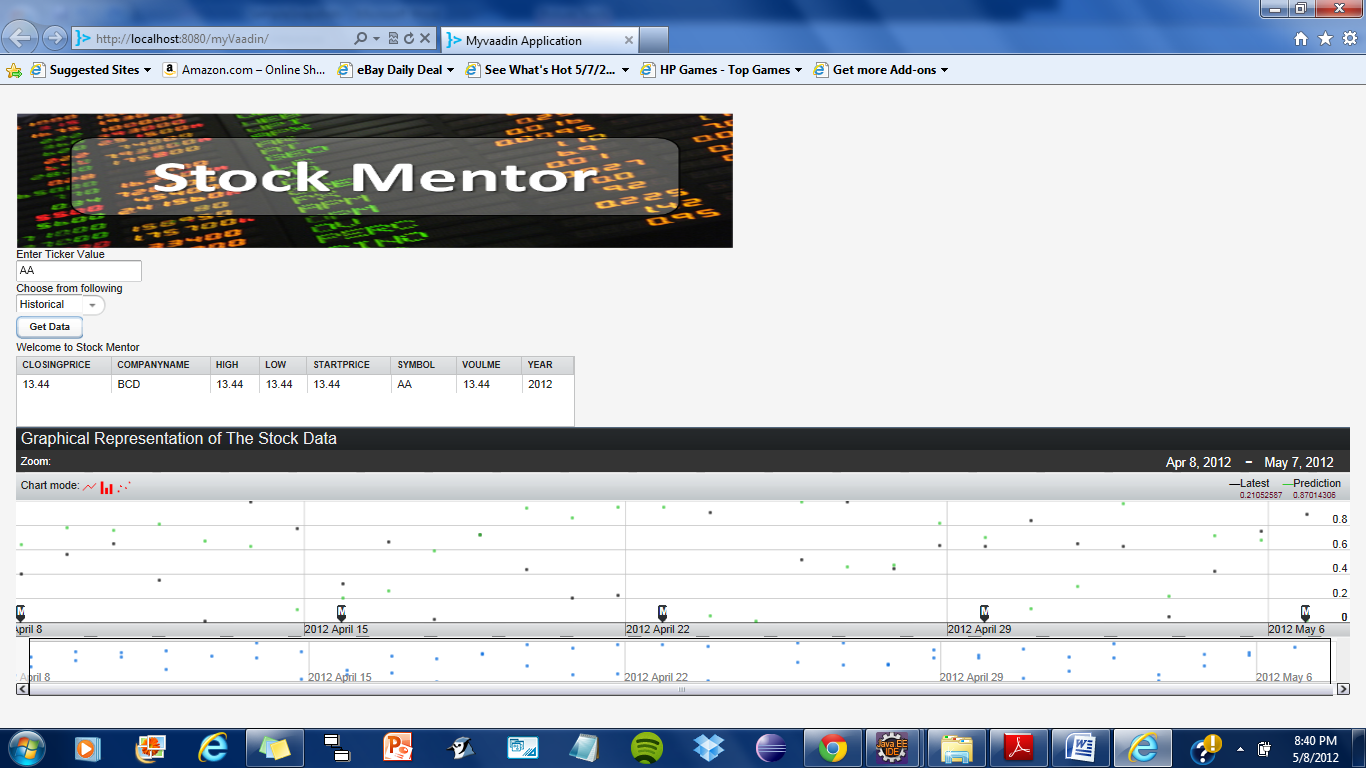




The Snapshot data page for a particular ticker value. This shows a table for snapshot data along with the chart for prediction.



The StockTrend data page for a particular ticker value. This shows a table for StockTrend data along with the chart for prediction.



The Historicaldata page for a particular ticker value. This shows a table for historical data along with the chart for prediction.

# Installation and Execution Manual

There are two basic options for installing:

1. If you use Eclipse, you can install the Vaadin Plugin for Eclipse, as described below in “Vaadin Plugin for Eclipse”

2. Otherwise, download and install the JAR package, as described below in “Installing the JAR Package”.

* **Vaadin Plugin for Eclipse**

If you are using the Eclipse IDE, using the Vaadin plugin should help greatly.

The plugin includes:

• An integration plugin with *wizards* for creating new Vaadin-based projects, themes, and client-side widgets and widget sets.

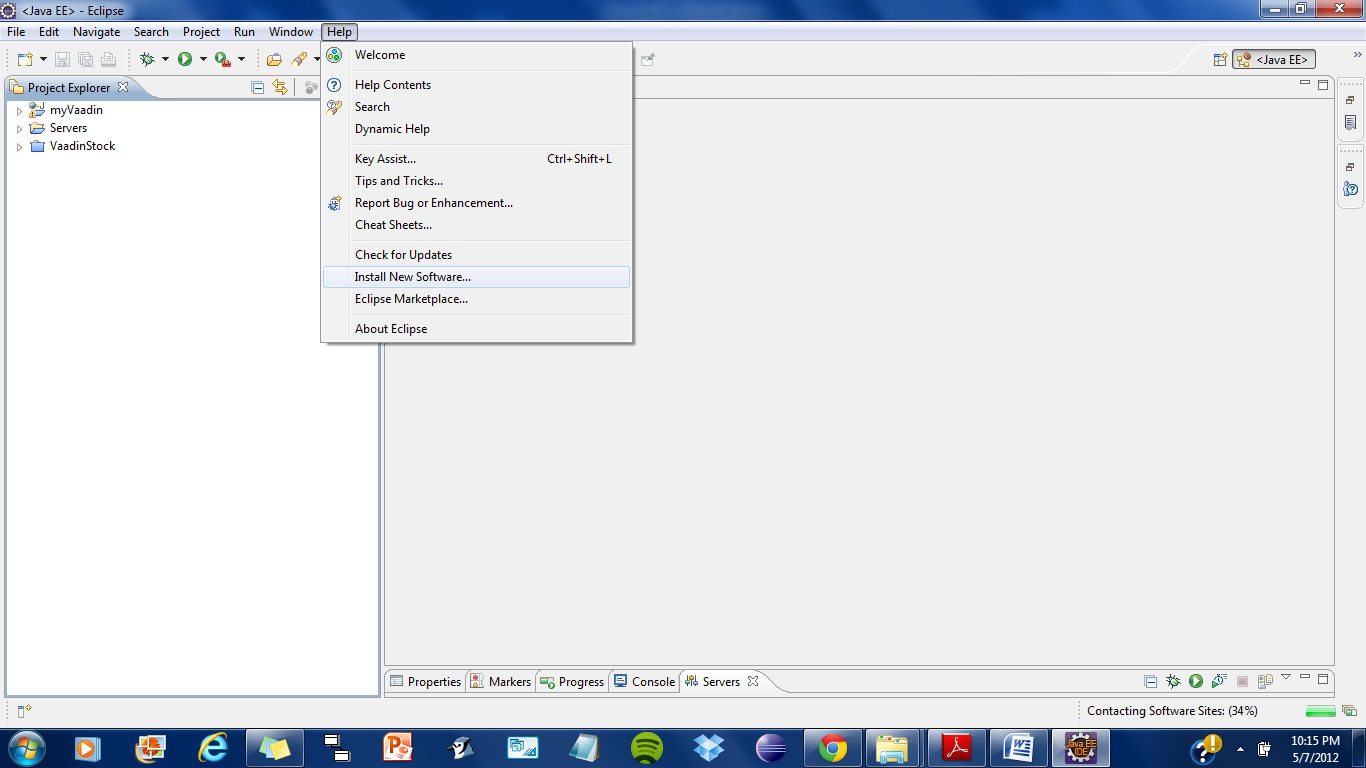
• A *visual editor* for editing custom composite user interface components in a WYSIWYG fashion. With full round-trip support from source code to visual model and back, the editor integrates seamlessly with your development process.

• A version of *Book of Vaadin* that you can browse in the Eclipse Help system.

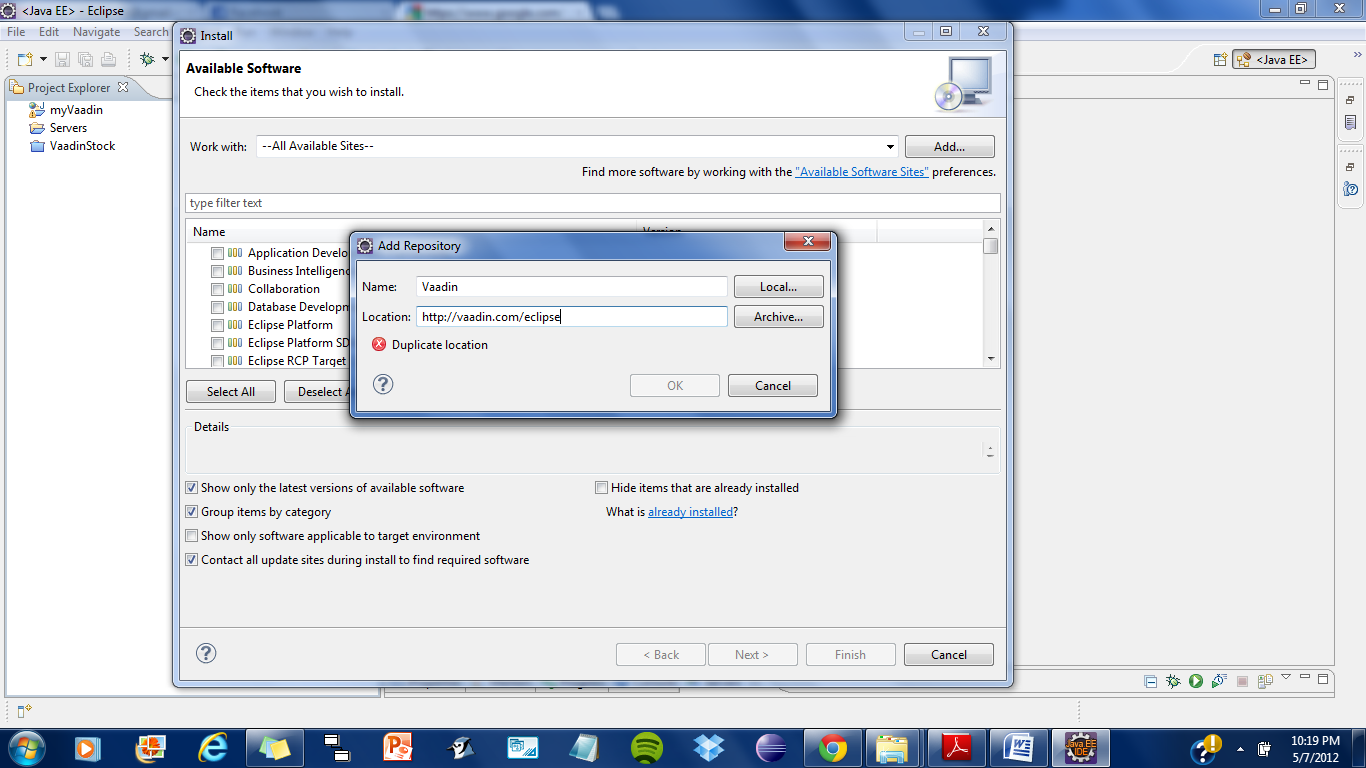
You can install the plugin as follows:

1. Start Eclipse.

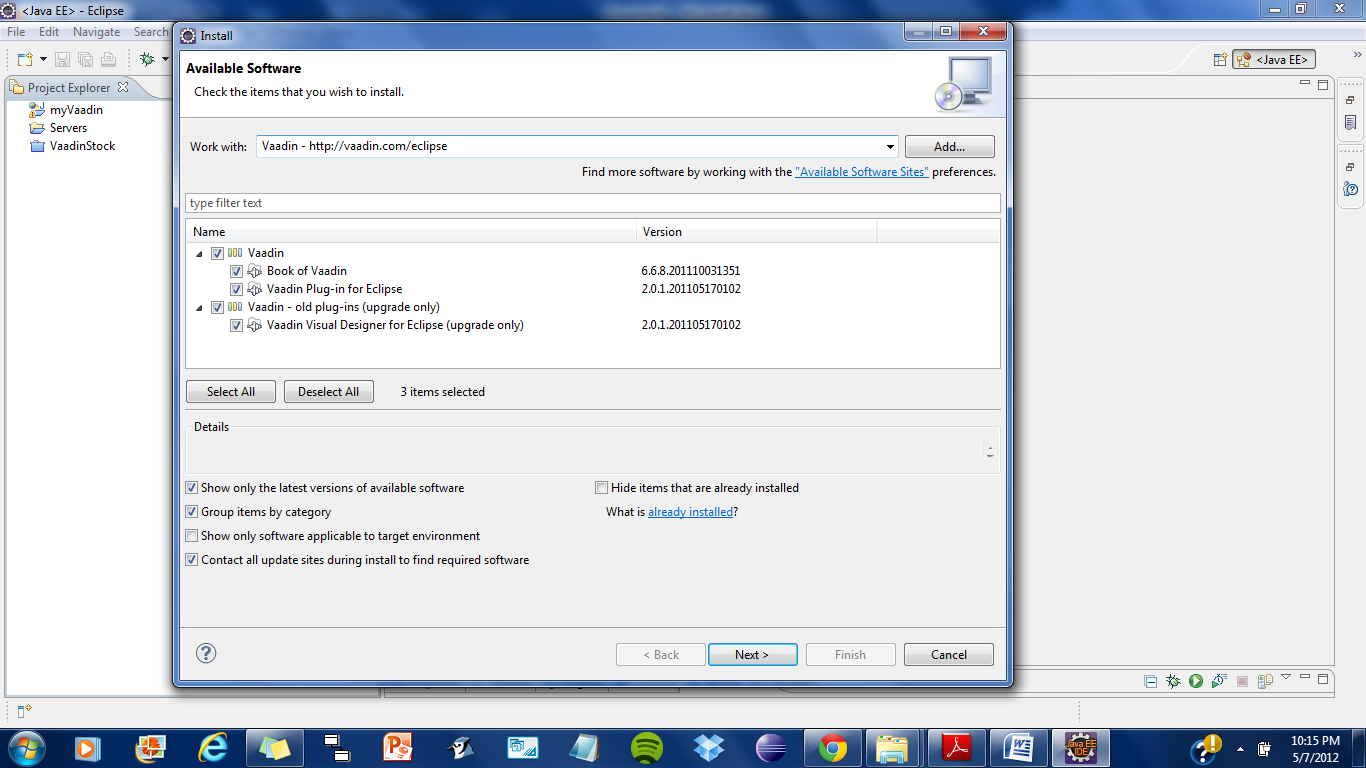
2. Select **Help 🡪 Install new software**



3. Add the Vaadin plugin site by clicking **Add ...**.



Enter the URL of the Vaadin Update Site: http://vaadin.com/eclipse and click **OK**. The



Vaadin site should now appear in the **Software Updates** window. Select all the Vaadin plugins in the tree. Finally, click **Install**.

* **Installing the JAR Package**

You can install the Vaadin JAR package in a few simple steps:

1. Download the newest Vaadin JAR package from the download page at <http://vaadin.com/download/>.

2. Put the JAR in the WEB-APP/lib web library folder in the project.

The location of the WEB-APP/lib folder depends on the project organization.

• In Eclipse projects: WebContent/WEB-INF/lib.

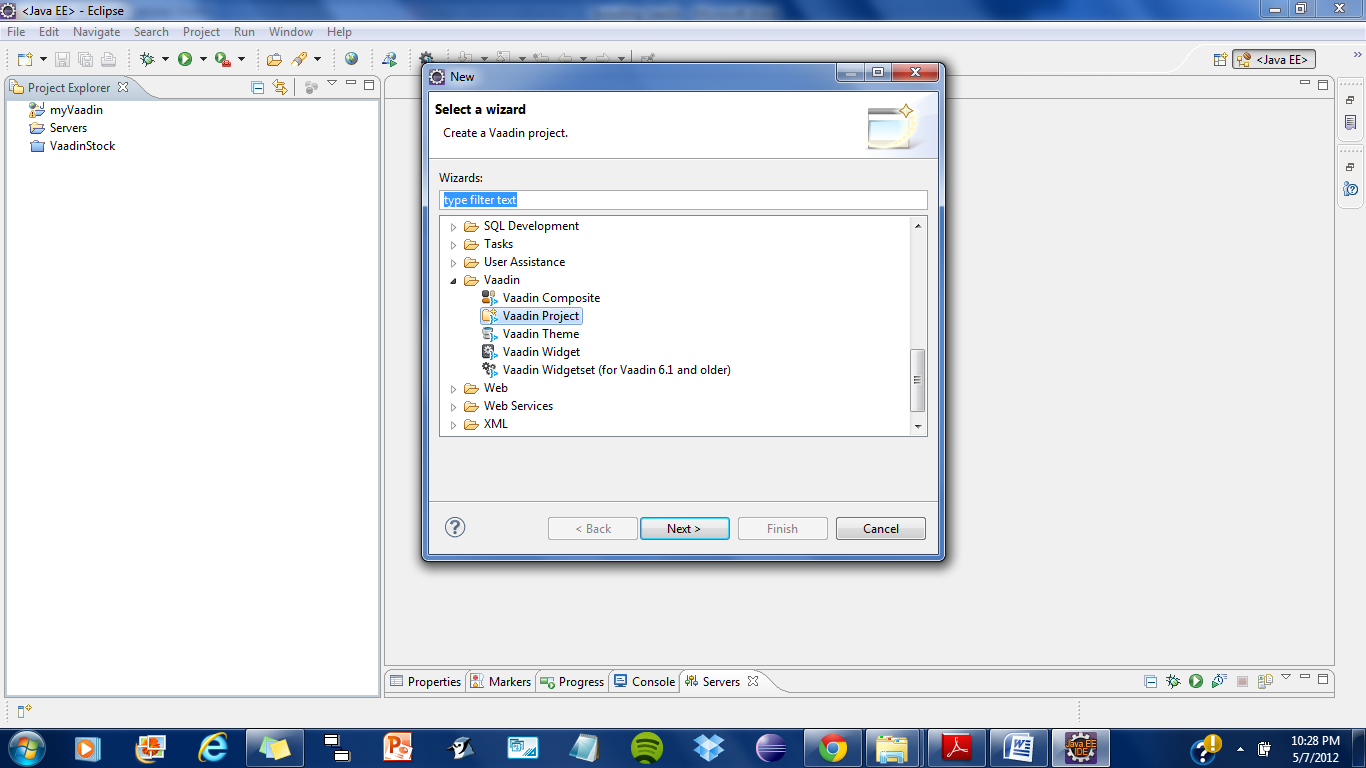
• In Maven projects: src/main/webapp/WEB-INF/lib.

* **Creating a Vaadin Project**

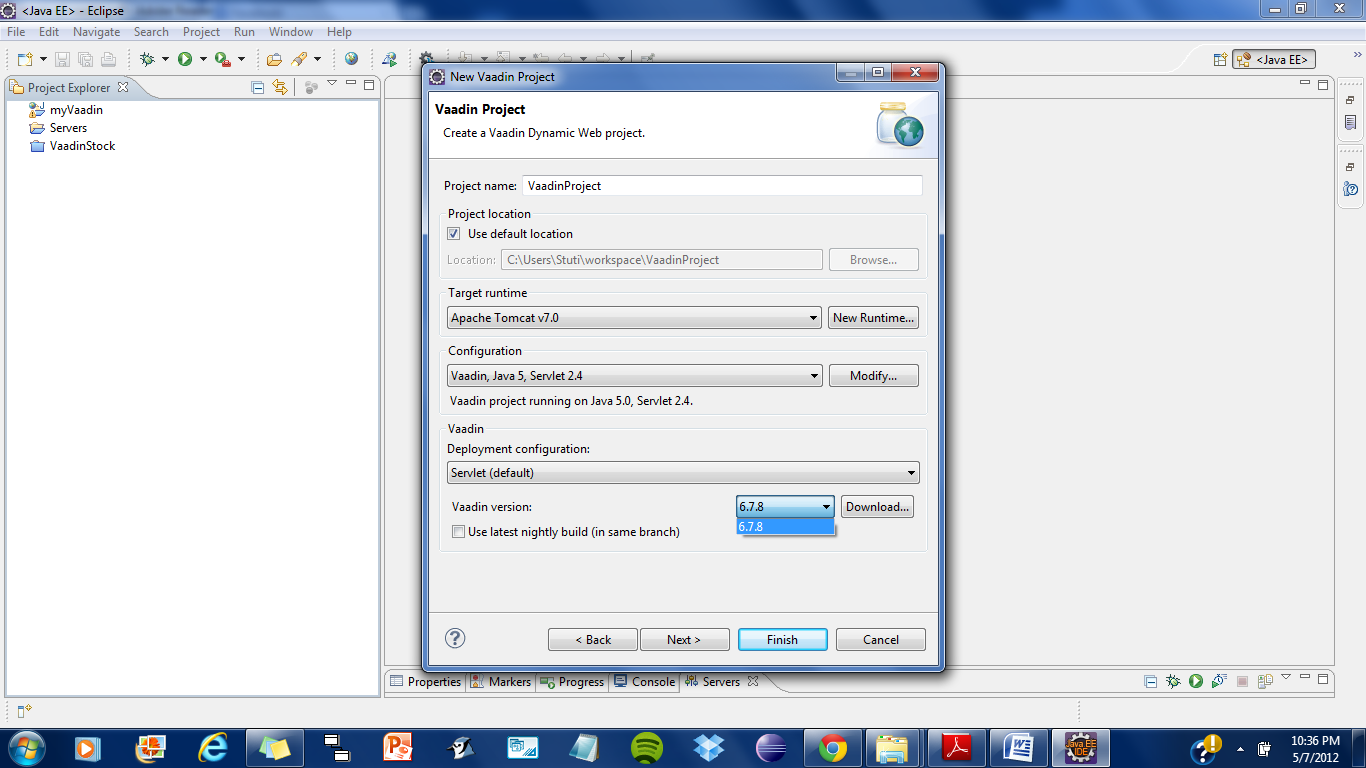
First,launch Eclipse and follow the following steps:

Start creating a new project by selecting from the menu File 🡪 New 🡪 Other

In the New Project window that opens, select Vaadin Project and click Next.



In the Vaadin Project step, you need to set the basic web project settings. You need to give at least the project name and the runtime; the default values should be good for the other settings.



**Project name:**  Give the project a name. The name should be a valid identifier usable cross-platform as a filename and inside a URL, so using only lower-case alphanumeric, underscore, and minus sign is recommended.

**Use default:** Defines the directory under which the project is created.

You should normally leave it as it is. You may need to set the directory, for example, if you are creating an Eclipse project on top of a version-controlled source tree.

**Target runtime:** Defines the application server to use for deploying the application. The server that you have installed, for example Apache Tomcat, should be selected automatically. If not, click New to configure a new server under Eclipse.

**Configuration:** Select the configuration to use; you should normally use the default configuration for the application server.

If you need to modify the project facets, click Modify.

**Deployment configuration:** This setting defines the environment to which the application will be deployed, to generate the appropriate project directory layout and configuration files.

The choices are:

• Servlet (default)

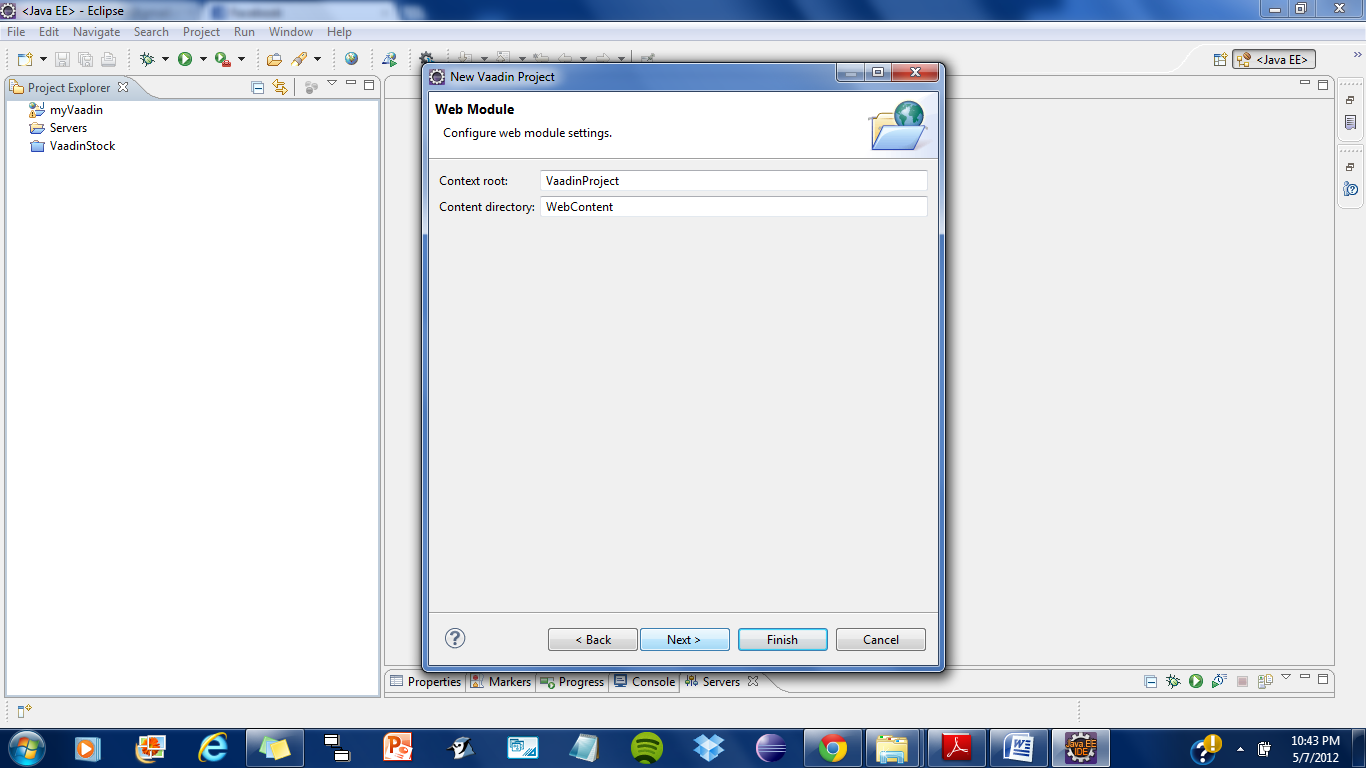
• Google App Engine Servlet

• Generic Portlet

• Old Portlet

**Vaadin version:** Select a version and click Ok to download it. It will appear as a choice in the drop-down list. If you want to change the project to use another version of Vaadin, for example to upgrade to a newer one, you can go to project settings and download and select the other version.

You can click Finish here to use the defaults for the rest of the settings, or click Next.



The settings in the Web Module step define the basic servlet-related settings and the structure of the web application project. All the settings are pre-filled, and you should normally accept them as they are.

**Context Root:** The context root (of the application) identifies the application in the URL used for accessing it. For example, if the server runs in the apps context and the application has vaadinproject context, the URL would be http://example.com/app/url. The wizard will suggest vaadinproject for the context name.

**Content Directory:** The directory containing all the content to be included in the servlet and served by the web server. The directory is relative to the root directory of the project. Java Source Directory The default source directory containing the application sources. The src directory is suggested; another convention common in web applications is to use WebContent/WEB-INF/src, in which case the sources are included in the servlet (but not served in HTTP requests).

You can just accept the defaults and click Next.

The Vaadin project step page has various Vaadin-specific application settings. If you are trying Vaadin out for the first time, you should not need to change anything. You can set most of the settings afterwards, except the creation of the portlet configuration.

**Create project template:** Make the wizard create an application class stub.

**Application Name:** The name of the application appears in the browser window title.

**Base package name:** The name of the Java package under which the application class is to be placed.

**Application class name:** Name of the Vaadin application class.

**Create portlet:** configuration when this option is selected, the wizard will create the files needed for running the application in a portal.

Finally, click Finish to create the project.

Eclipse may ask to switch to J2EE perspective. A Dynamic Web Project uses an external

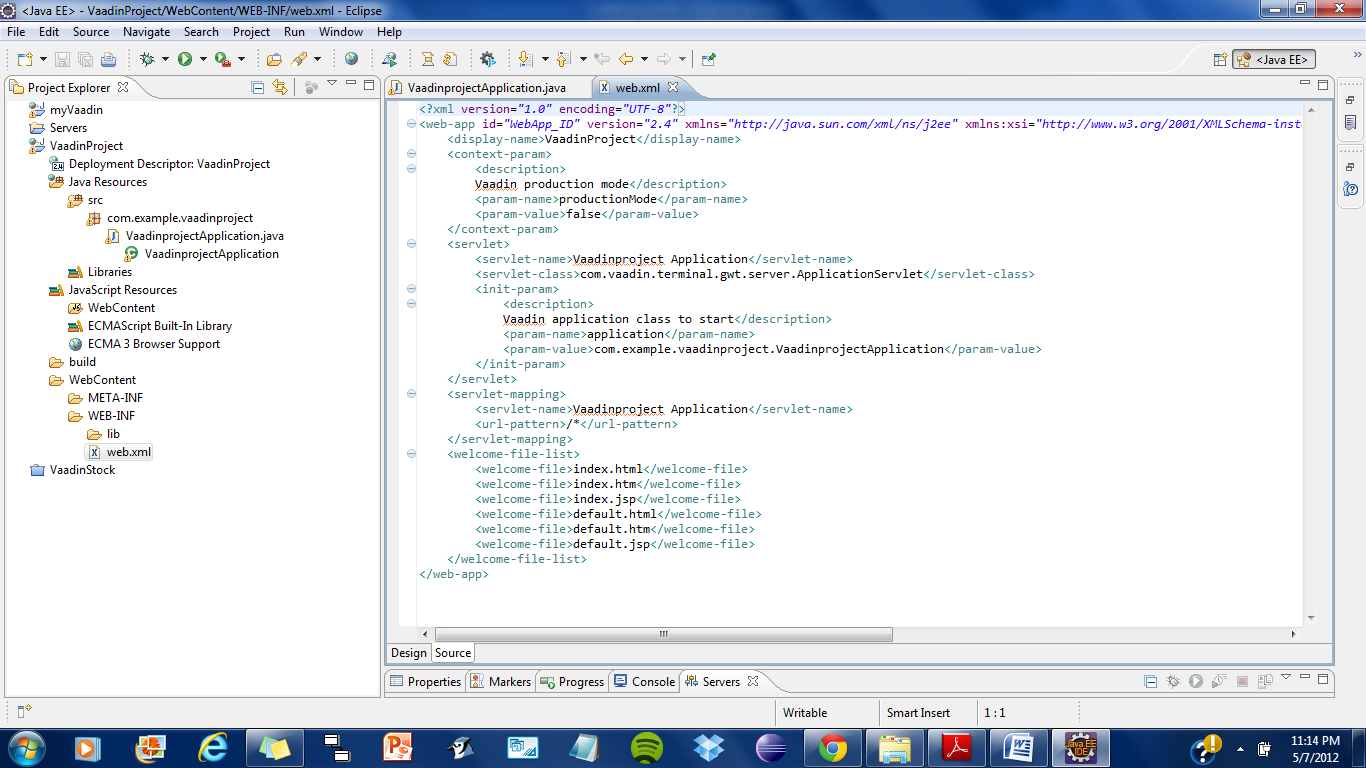
web server and the J2EE perspective provides tools to control the server and manage application deployment. Click Yes.

Exploring the Project

After the New Project wizard exists, it has done all the work for us: Vaadin libraries are installed in the WebContent/WEB-INF/lib directory, an application class skeleton has been written to src directory, and WebContent/WEB-INF/web.xml already contains a deployment descriptor. The application class created by the plugin contains the following code:



The web.xml is the congfiguration file which is as follows:



* **Setting Up and Starting the Web Server**

Eclipse IDE for Java EE Developers has the Web Standard Tools package installed, which supports control of various web servers and automatic deployment of web content to the server when changes are made to a project.

Make sure that Tomcat was installed with user permissions. Configuration of the web server in Eclipse will fail if the user does not have write permissions to the configuration and deployment directories under the Tomcat installation directory.

Follow the following steps.

1. Switch to the Servers tab in the lower panel in Eclipse. List of servers should be empty after Eclipse is installed. Right-click on the empty area in the panel and select New Server.

2. Select Apache Tomcat v7.0 Server and set Server's host name as localhost, which should be the default. If you have only one Tomcat installed, Server runtime has only one choice. Click Next.

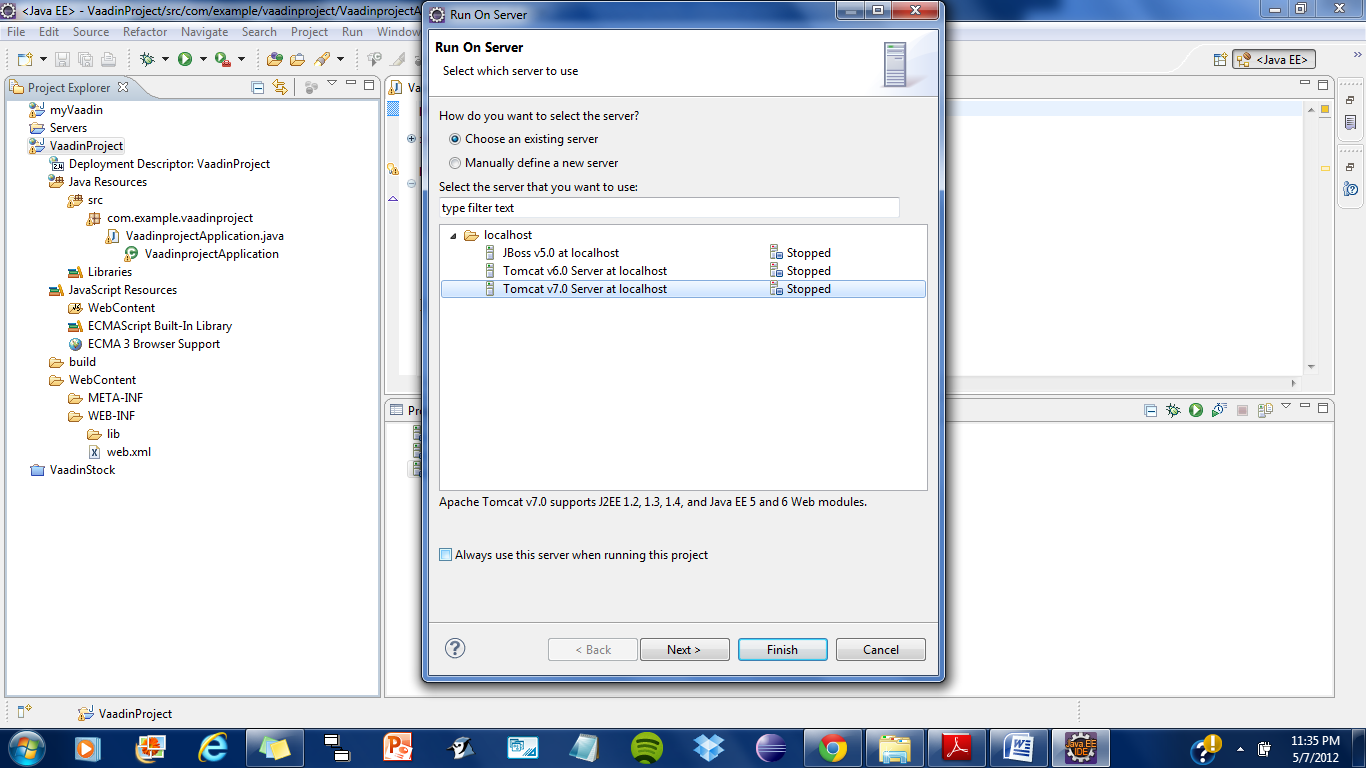
3. Add your project to the server by selecting it on the left and clicking Add to add it to the configured projects on the right. Click Finish.

The server and the project are now installed in Eclipse and are shown in the Servers tab. To start the server, right-click on the server and select Debug. To start the server in non-debug mode, select Start.



5. The server starts and the WebContent directory of the project is published to the server on http://localhost:8080/vaadinproject/.

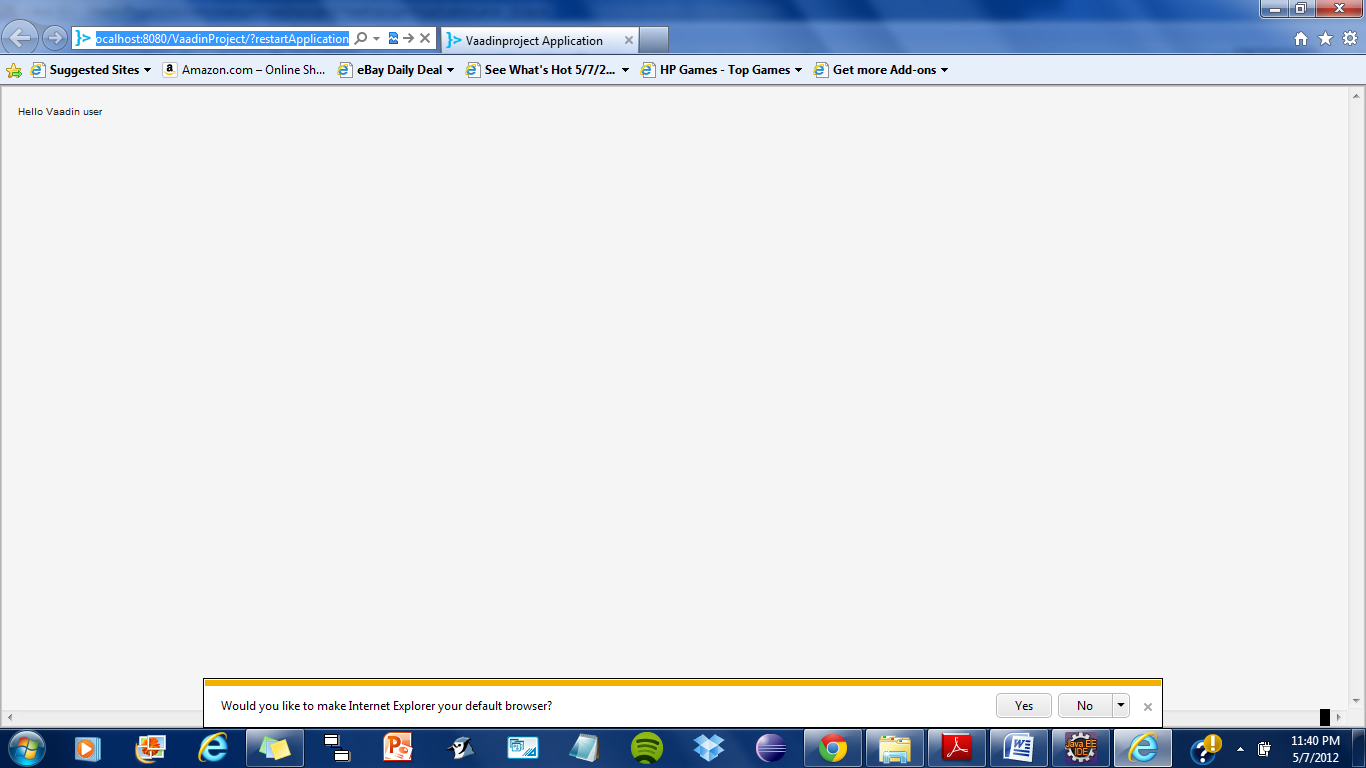
* Running

Starting application is as easy as selecting vaadinproject from the Project Explorer and then Run on Server. Then Select Tomcat 7.0 at localhost. Eclipse then opens the application in built-in web browser. To debug, you can insert break points in the Java code by double-clicking on the left margin bar of the source code window.

* Restarting Application Session

When you open the URL for the application, it creates a new user session. The session is preserved even if you reload the page. Moreover, as Eclipse likes to do hot deployment to Tomcat, and Tomcat likes to persist sessions on server shutdown, you may experience a problem that the application doesn't return to its initial state after modifying code or even restarting the server.

Adding the ?restartApplication” parameter in the URL tells the Vaadin servlet to create a new Application instance when reloading the page.



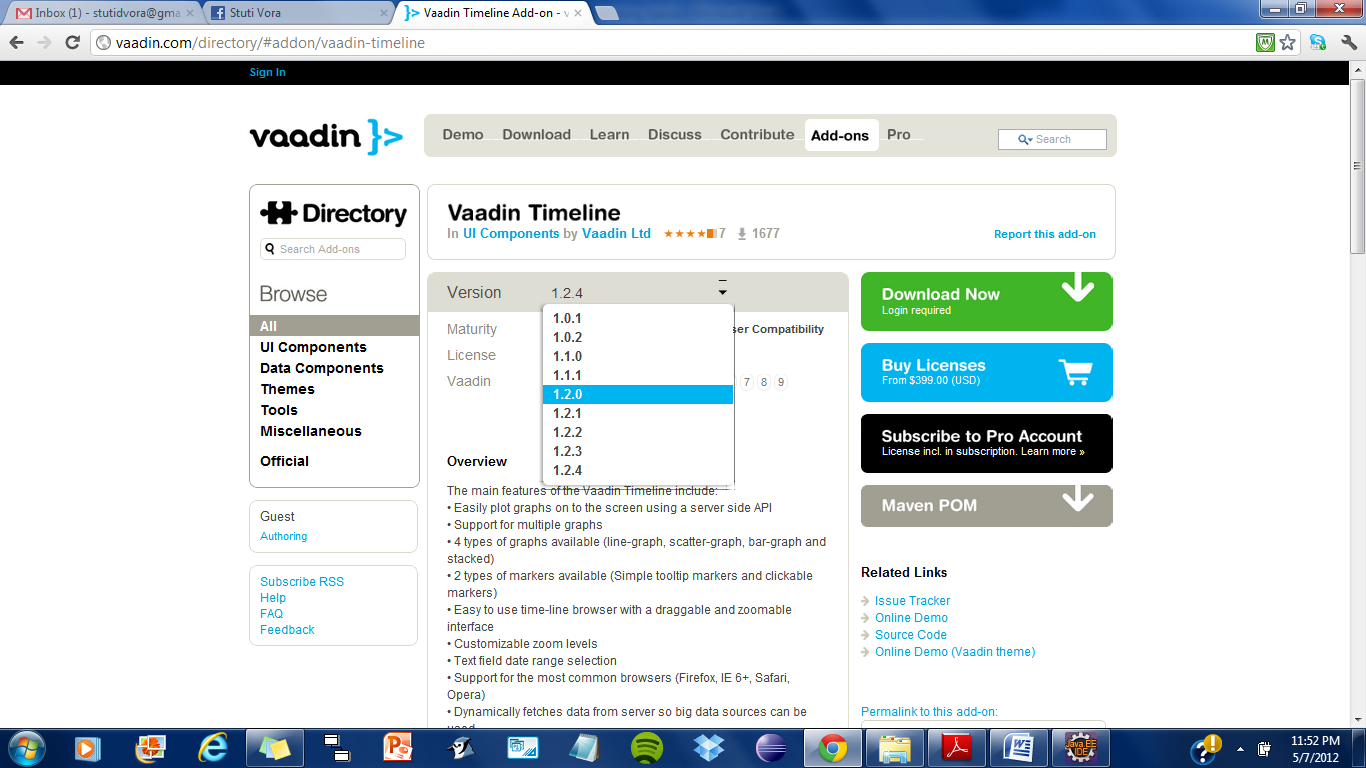
* **Add-ons**

In addition to the components, layouts, themes, and data sources built in into the core Vaadin library, many others are available as add-ons, either from the Vaadin Directory or from independent sources. Both commercial and free components exist under various licenses. Installation of themes, data sources, and components built with server-side component composition is simple, just dropping a JAR package in a project and, usually, compiling the included widget set (the client-side implementation).

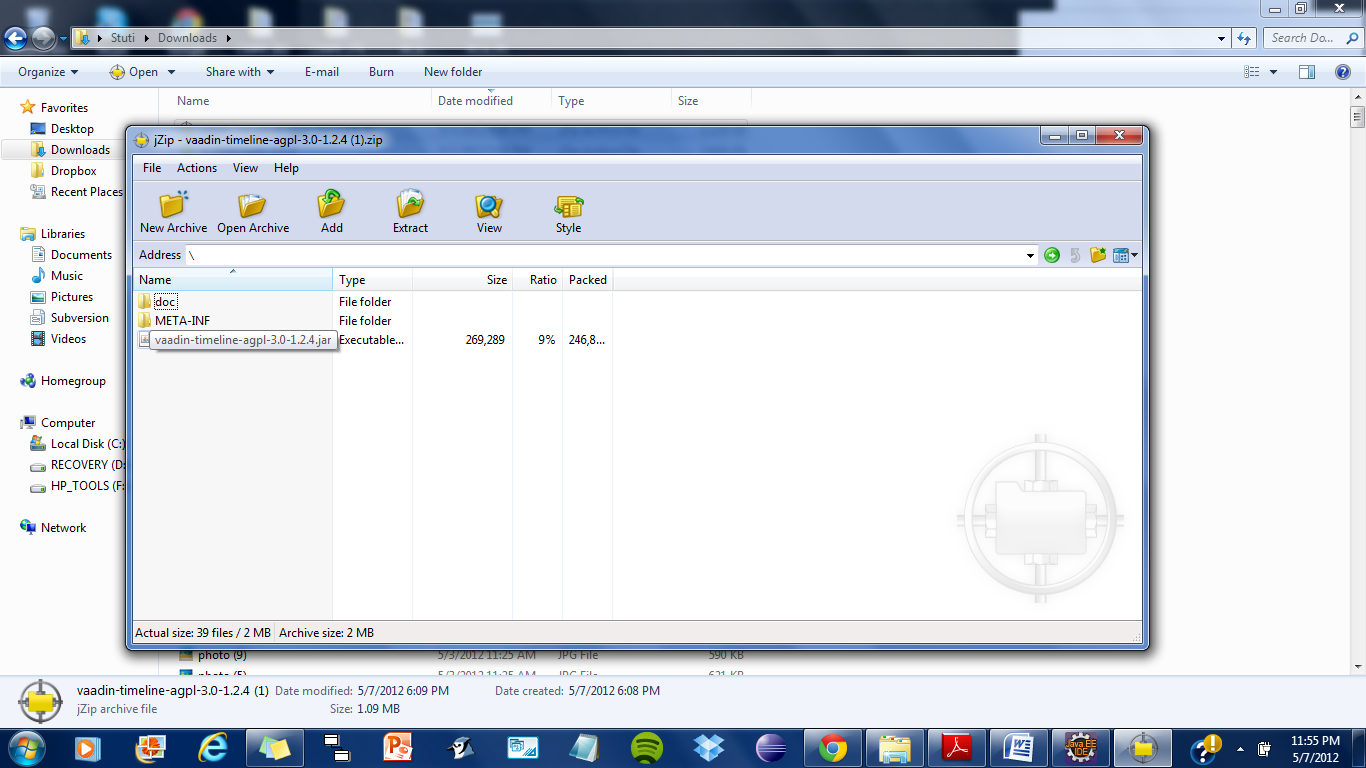
**Downloading Add-ons from Vaadin Directory**

Vaadin Directory at http://vaadin.com/directory/ provides a rich collection of add-ons for Vaadin.You can download Directory add-on packages from the details page of an add-on.

**We have used Timeline add-on in our project.**

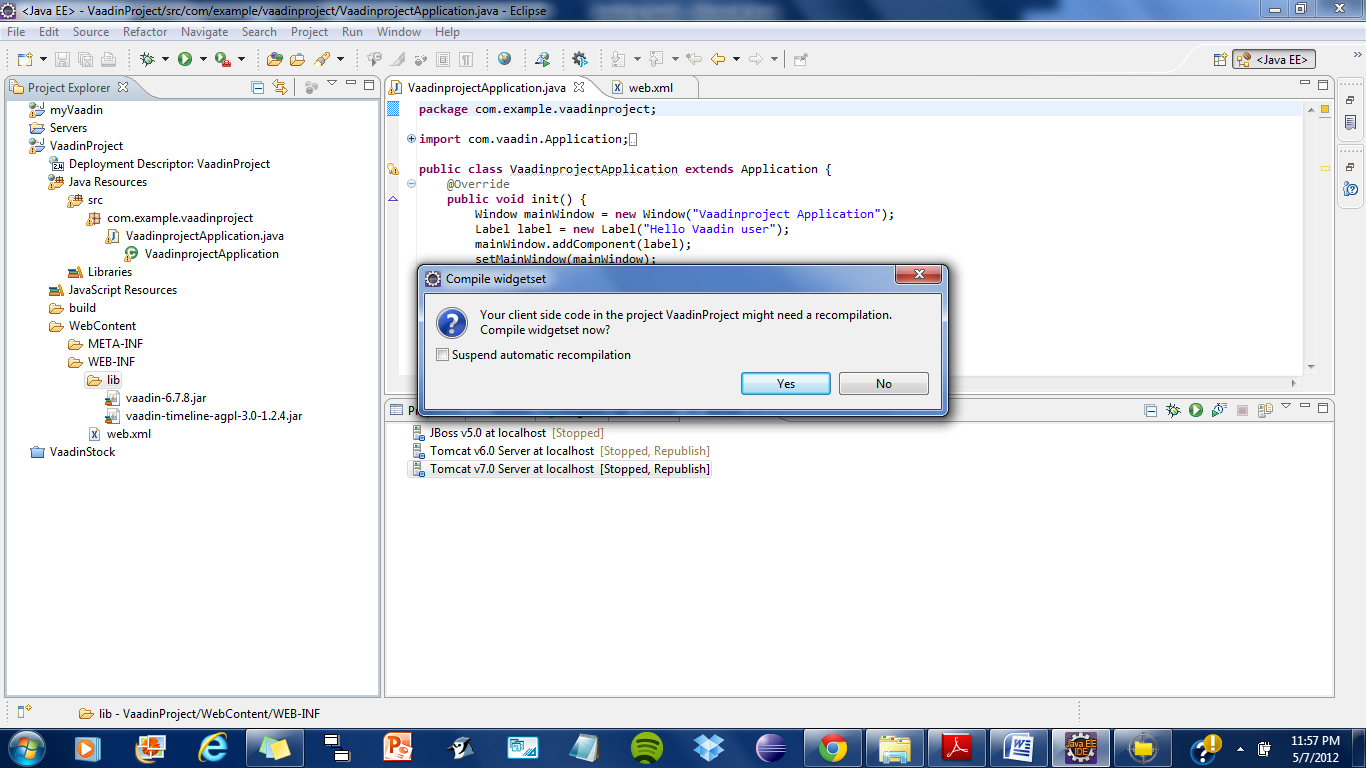


1. Select the version; some add-ons have several versions available. The latest is shown by default, but you can choose another the version to download from the dropdown menu in the header of the details page.
2. Click **Download Now** and save the JAR or Zip file on your computer.



1. If the add-on is packaged in a Zip package, unzip the package and follow any instructions

provided inside the package. Typically, you just need to copy a JAR file to your web project under the WEB-INF/lib directory.



1. You need to compile the client-side implementations of the add-on components, that is, a *widget set*. This is the case for majority of add-ons, except for pure server-side, theme, or data binding add-ons. You must recompile the widget set if you install a new version of the add-on or the Vaadin library. See the subsequent sections for detailed instructions for compiling widget sets. Click yes for compiling widgetset
2. Update the project in web server and possibly restart the server.

# References

1. Book of Vaadin : https://vaadin.com/book
2. Vaadin Timeline Add-on : http://vaadin.com/directory/#addon/vaadin-timeline
3. What is Vaadin : <https://vaadin.com/learn>
4. KEA API: <http://code.google.com/p/kea-algorithm/>
5. JSOUP: <http://jsoup.org/>
6. YARFRAW: http://yarfraw.sourceforge.net/intro.html