# PROJECT PART 1



## CD STORE

## README DOCUMENT

### Prepared For:

Part1 CSI5380 Project submission

### Submitted to:

Professor Dr. Hussein Al Osman

In partial fulfillment of the requirement for

CSI5380 - Systems and Architecture for Electronic Commerce

### Submitted by:

Web Builders

|  |  |  |  |
| --- | --- | --- | --- |
| Sl.No | Team Member | Student ID | Contact ID |
| 1 | Aishvarya Arul Nambi | 7758189 | [aarul094@uottawa.ca](mailto:aarul094@uottawa.ca) |
| 2 | IndhraPriyaShanmugam | 7527045 | [ishan050@uottawa.ca](mailto:ishan050@uottawa.ca) |
| 3 | Priyanka Patel | 7481432 | [ppatel038@uottawa.ca](mailto:ppatel038@uottawa.ca) |
| 4 | RachanaChandrashekar | 7487187 | [rchan092@uottawa.ca](mailto:rchan092@uottawa.ca) |
| 5 | ShruthiMadhurika Naomi | 7564806 | [snaomi011@uottawa.ca](mailto:snaomi011@uottawa.ca) |

Table of Contents

[PROJECT PART 1 1](#_Toc401831975)

[CD STORE 1](#_Toc401831976)

[README DOCUMENT 1](#_Toc401831977)

[1. Instructions for Installation/Configuration: 4](#_Toc401831982)

[1.1 Installing Java (J2SE) 4](#_Toc401831983)

[1.2 Installing Eclipse 5](#_Toc401831984)

[1.3 Installing Tomcat 6](#_Toc401831985)

[1.4 Configuring Tomcat on Eclipse 8](#_Toc401831986)

[1.5 Configuring the browser 9](#_Toc401831987)

[1.6 Testing tomcat setup 10](#_Toc401831988)

[1.7 Installing MySQL 11](#_Toc401831989)

[1.8 Setting up SSL configuration 12](#_Toc401831990)

[1.9 Connection Pool 14](#_Toc401831991)

[1.10 Web Service Configuration 15](#_Toc401831992)

[1.10.1 Creating Web Services: 15](#_Toc401831993)

[2. Organization of Source Code into Packages 1](#_Toc401831995)6

[3. Individual contribution towards the project 17](#_Toc401831996)

[4. Third Party Dependencies 1](#_Toc401831997)7

[5. Testing 18](#_Toc401831998)

[6. References 1](#_Toc401831999)8

# List of Figures:

[Figure 1: Successful installation of java 5](#_Toc401789753)

[Figure 2: Successful installation of Eclipse IDE 5](#_Toc401789754)

[Figure 3: Starting the Tomcat server 6](#_Toc401789755)

[Figure 4: Tomcat running successfully 7](#_Toc401789756)

[Figure 5: Selecting the Tomcat server for configuring the server in Eclipse 8](#_Toc401789757)

[Figure 6: Web Browser configured in Eclipse 9](#_Toc401789758)

[Figure 7: Result in the browser 10](#_Toc401789759)

[Figure 8: Successful installation of MySQL 10](#_Toc401789760)

[Figure 9: Certificate creation using keytool 11](#_Toc401789761)

[Figure 10: Configuring Tomcat's SSL Connector 12](#_Toc401789762)

[Figure 11: Tomcat successfully configured to run on https 12](#_Toc401789763)

[Figure 12: running index.jsp on "localhost:8443"/https 13](#_Toc401789764)

[Figure 13: context.xml file with the resource tag 13](#_Toc401789765)

[Figure 14: Creating orderprocess package 14](#_Toc401789766)

[Figure 15: Creating productcatalog package 14](#_Toc401789767)

[Figure 16: Creating WebService 14](#_Toc401789768)

[Figure 17: Webservice methods 15](#_Toc401789769)

[Figure 18: Source code organized as packages 16](#_Toc401789777)

[ SSLConfiguration 18](#_Toc401789778)

[ Third Party Dependencies 18](#_Toc401789779)

# 

# 1. Instructions for Installation/Configuration:

## 1.1 Installing Java(J2SE)

**Purpose:**

J2SE is a premier platform for developing and deploying application. It is used for compiling Java programs and Servlets.

**SetUp:**

**Step 1:** Download J2SE from <http://www.oracle.com/technetwork/java/index.html>

**Step 2:** Install it to the default directory.

**Step 3:** To test the successful installation, open the command prompt (Start -> type 'cmd' in the search tab and press 'enter').

Type 'javac' in the command prompt and press 'enter'. The user should see the usage options on doing this. This ensures successful installation.



#### Figure 1: Successful installation of java

## 1.2 Installing Eclipse

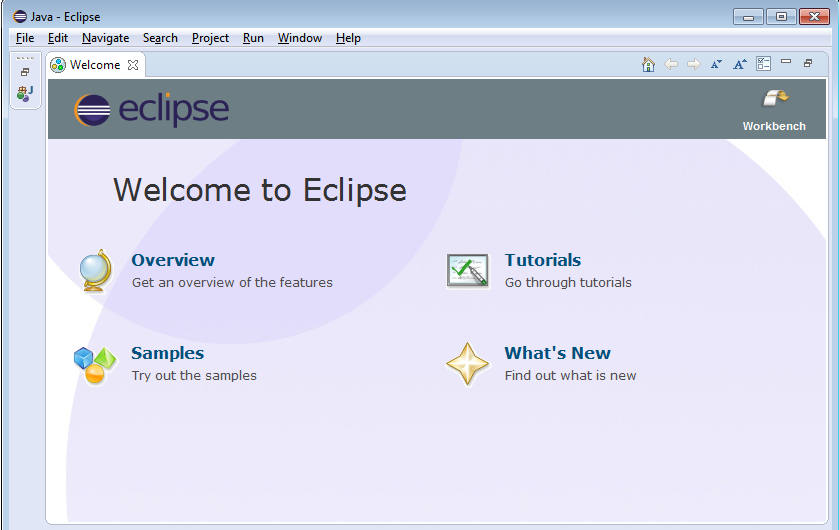
**Purpose:**

Eclipse is a fully-featured integrated development environment (IDE) that lets us develop java applications quickly and easily.

**SetUp:**

**Step 1:** Download Eclipse from <https://www.eclipse.org/downloads/>

**Step 2:** Install the setup. You should see a "Start Page" if the Eclipse is successfully installed, on opening the IDE.



#### Figure 2: Successful installation of Eclipse IDE

## 1.3 Installing Tomcat

**Purpose:**

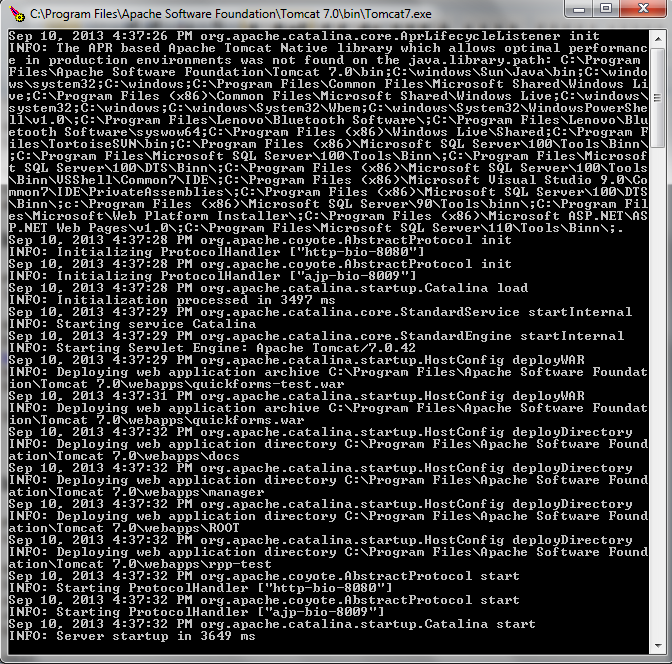
Apache Tomcat is a software used for implementation of the JavaServer Pages technologies. It serves as a server that lets hosting of Java Servlets and other files.

**SetUp:**

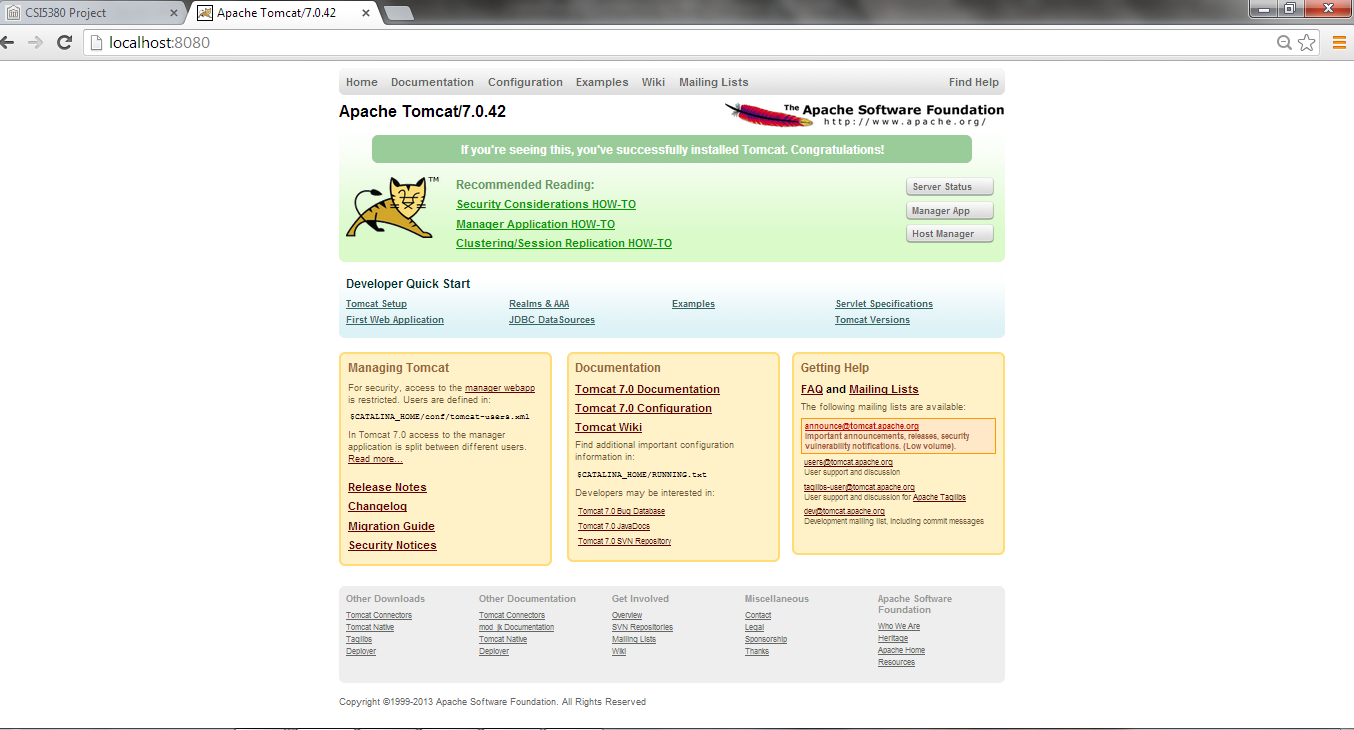
**Step 1:** Download Tomcat 7(or above) binary from <http://tomcat.apache.org/>

**Step 2:** Include the Servlet library in the class path

**Step 3:** Navigate to Tomcat's bin (" C:\Program Files (x86)\Apache Software Foundation\Tomcat 7.0\bin”)folder and start the tomcat service.



#### Figure 3: Starting the Tomcat server



#### Figure 4: Tomcat running successfully

## 1.4 Configuring Tomcat on Eclipse

**Step1:** From Eclipse, navigate to "Servers" tab at bottom.

**Step2:** Right click on the "Servers" tab, New, Server, Apache, Tomcat v7.0 and navigate to the folder where you unzipped Tomcat

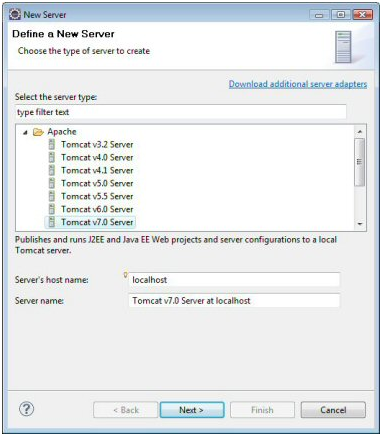
**Step3:** You should now see “Tomcat v7.0 Server at localhost” listed under the Servers tab at the bottom.

**Step 4:** Click on Servers tab at bottom. Right click on Tomcat v7.0, choose “Start”.

**Step 5:**Open <http://localhost:8080> in a browser. This will result in a 404 error message.

**Step 6:** Copy the ROOT Web App into Eclipse to rectify the error message.

Eclipse forgets to copy the default apps (ROOT, examples, etc.) when it creates a Tomcat folder inside the Eclipse workspace. Go to C:\apache-tomcat-7.0.34\webapps, R-click on the ROOT folder and copy it. Then go to your Eclipse workspace, go to the .metadata folder, and search for "wtpwebapps". You should find something like your-eclipse-workspace\.metadata\.plugins\org.eclipse.wst.server.core\tmp0\wtpwebapps (or .../tmp1/wtpwebapps if you already had another server registered in Eclipse). Go to the wtpwebapps folder, R-click, and paste ROOT (say "yes" if asked if you want to merge/replace folders/files). Then reload http://localhost/ to see the Tomcat welcome page.

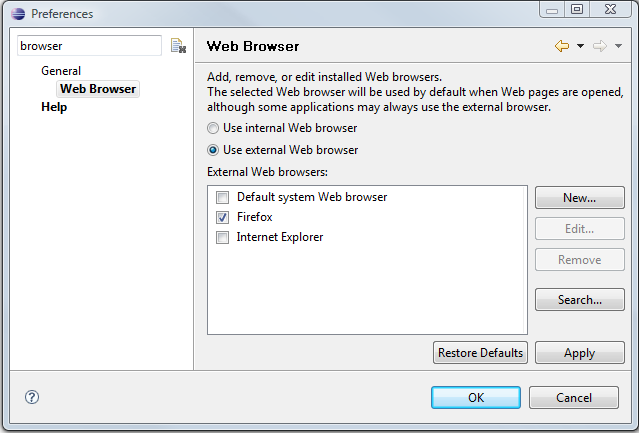


#### Figure 5: Selecting the Tomcat server for configuring the server in Eclipse

## 1.5 Configuring the browser

**Step 1:** Open the browser preference page via Window > Preferences > General > Web Browser.

**Step 2:** Select the “User external Web browser” radio button and choose from the available external web browsers.



#### Figure 6: Web Browser configured in Eclipse

## 1.6 Testing tomcat setup

**Step 1:** Start the tomcat as explained previously.

**Step 2:** Create a new project. Click on "File" and select "New Project" from the list.

**Step 3:** Select "Web" and Dynamic Web Project. For “Target Runtime” choose “Apache Tomcat v7.0”. Give it a name “Test”. Click on "Next". Accept all other defaults. Click on "Finish" to create a project successfully.

**Step 4:** You will now observe "index.jsp" under "WebContent" folder being created. Right click on this "index.jsp" file and click on "Run”

**Step 5:** You will observe that the index.jsp file is now run on the configured browser successfully displaying "Hello World!" message to the user.

#### Figure 7: Result in the browser

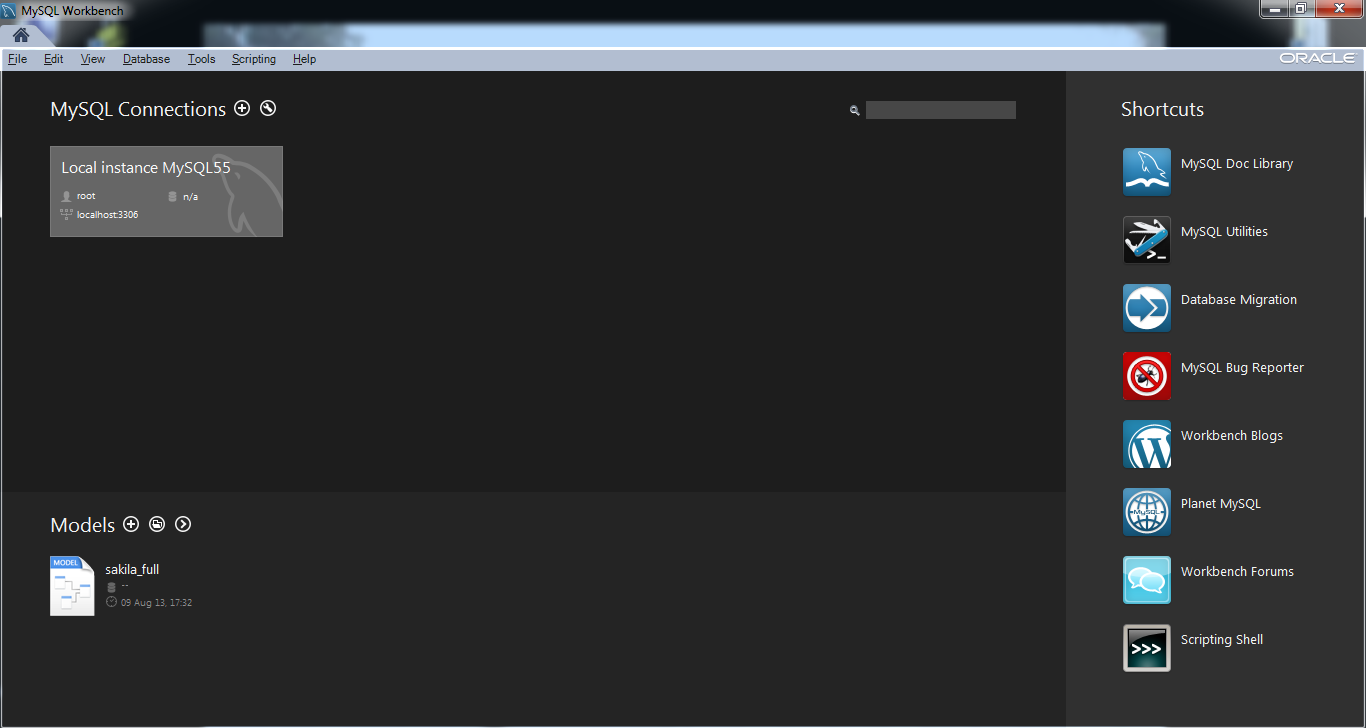
## 1.7 Installing MySQL

**Purpose:**

It is the widely used open-source DBMS in web applications. We have used MySQL along with the MySQL workbench a front-end tool to create and manage SQL databases

**SetUp:**

**Step 1:** Download and install MySQL (mysql-installer-community-5.5.33.2) from <http://dev.mysql.com/downloads/>



#### Figure 8: Successful installation of MySQL

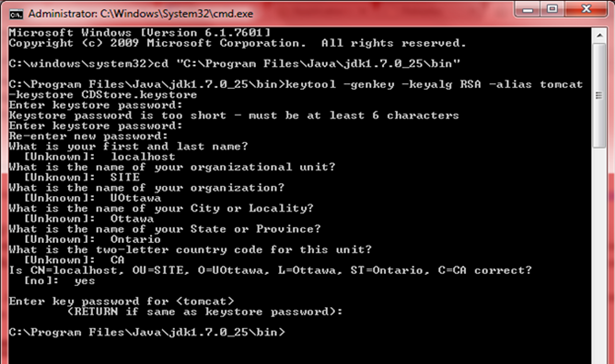
## 1.8 Setting up SSL configuration

**Purpose:**

SSL protocol provides security for communications between the client and server by implementing encrypted data and certificate-based authentication.

**SetUp:**

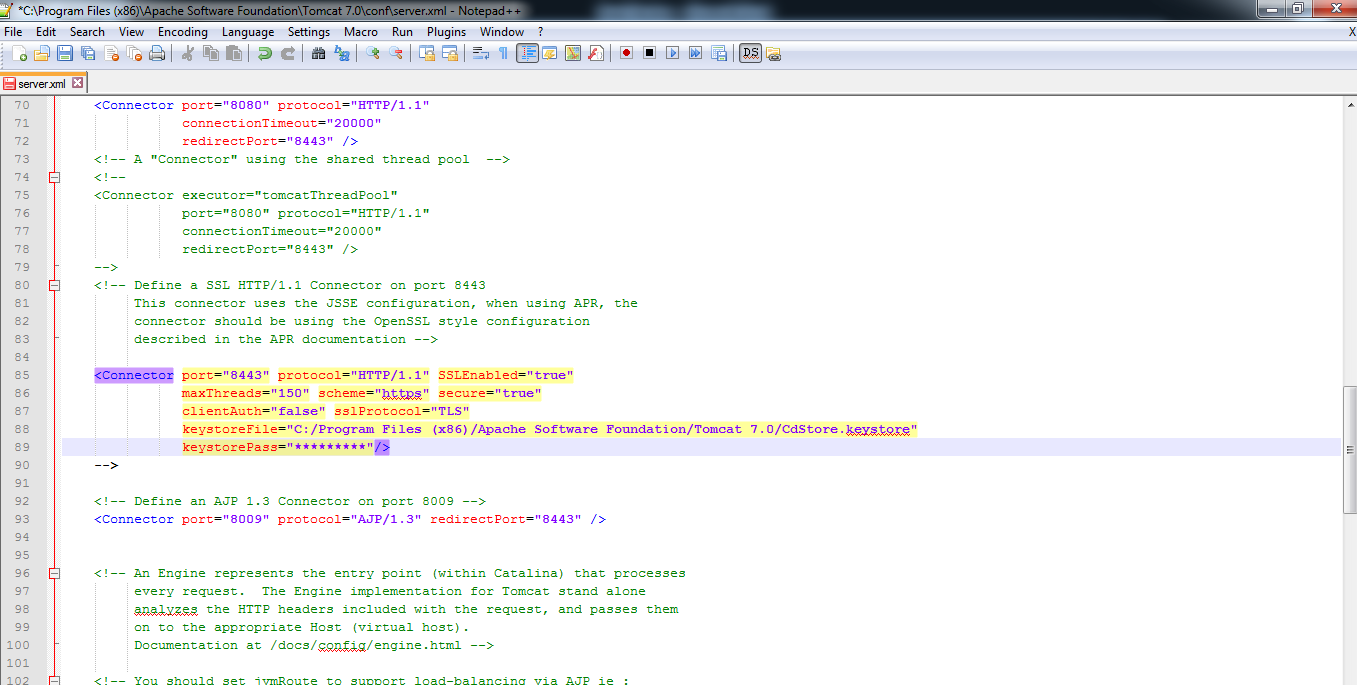
**Step 1:** To have your webpages running securely on https, it is necessary to configure SSL.First step is to create keystore. "Keytool" included in the JDK is used to create keystore.



#### Figure 9: Certificate creation using keytool

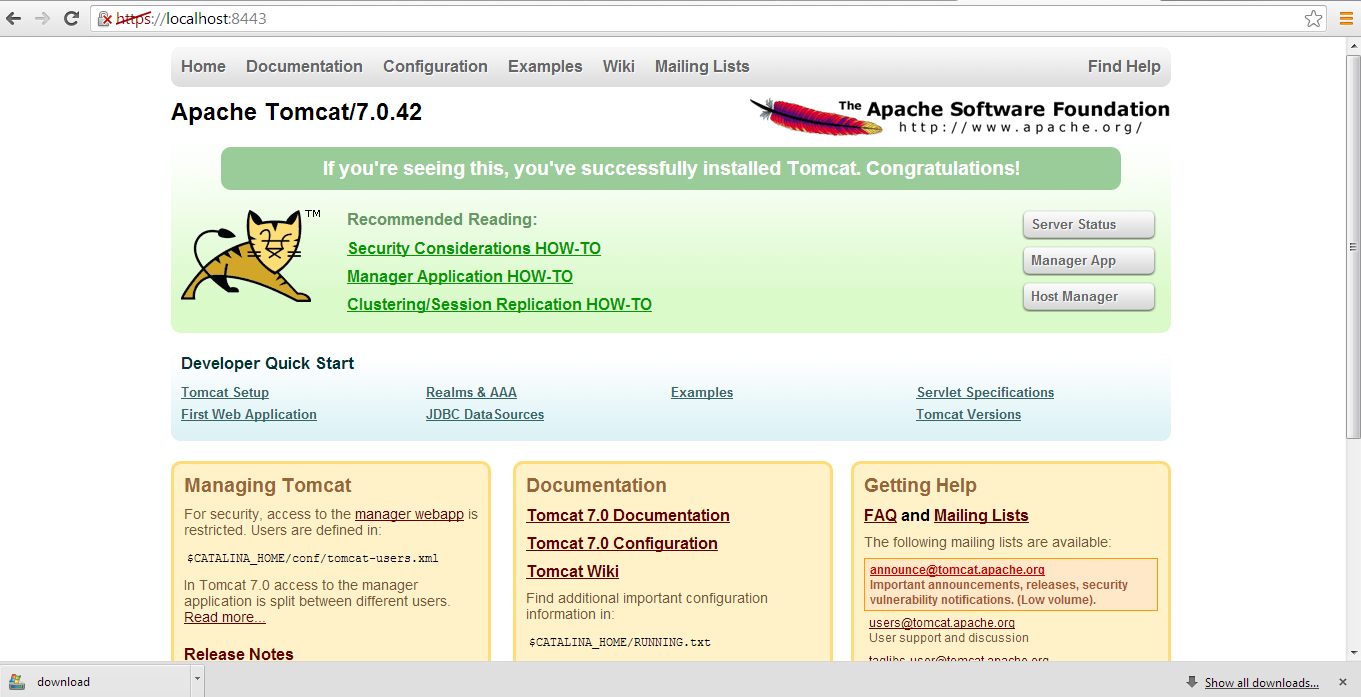
**Step 2:** Copy the keystore file generated to tomcat folder (copy CDStore.keystore generated to ":\Program Files\Apache Software Foundation\Tomcat 7.0")

**Step 3:**Configure Tomcat's SSL connector next. This can be done by changing the "Server.xml" file content in the conf folder of tomcat.



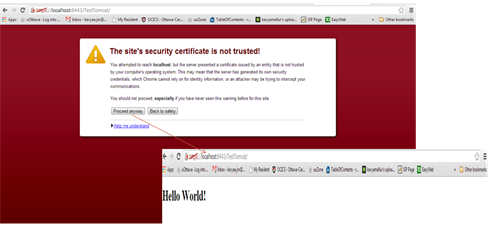
#### Figure 10: Configuring Tomcat's SSL Connector

**Step 4:** Check if the Tomcat is successfully configured for https. Provide "[https://localhost:8443](https://localhost:8443/)" in the Browser URL and verify that you will see tomcat up and running successfully.



#### Figure 11: Tomcat successfully configured to run on https

**Step 5:** Run the "index.jsp" file created earlier, on https port to verify that Tomcat is successfully configured.Since we have not added the certificate yet to the trusted root, we will see a message displayed. Click on "Proceed anyway" to view the webpage successfully.



#### Figure 12: Running index.jsp on "localhost: 8443"/https

## 1.9 Connection Pool

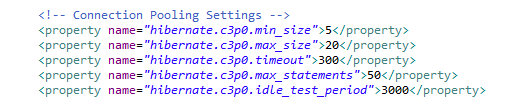
**Purpose:**

Connection pools is a cache of database connections maintained. Connections can be reused when future requests to the DB are required.

Connection pools can be used to enhance the performance of executing commands on a DB.

**SetUp:**

To set up connection pooling, include the following commands in the **hibernate.config.xml**. It defines the minimum and maximum size of the connection pools to enhance scalability.



#### Figure 13: hibernate.config.xml file with the resource tag

## 

## 1.10 Web Service Configuration

**Purpose:**

Web services are application services that allow different applications from different sources to communicate with each other. We have implemented two web services in our project.

* ProductCatalogWS: Used for obtaining all the products and their details.
* OrderProcessingWS: Used to get user details and also to add order details with the user selected items into database successfully.

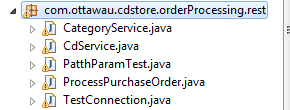
### 1.10.1 Creating Web Services:

**SetUp:**

The web service we have implemented in our project is **Representational state transfer** (**REST**). It is an architectural style consisting of a coordinated set of architectural constraints applied to components, connectors, and data elements, within a distributed system. We used **JERSEY** framework for implementing RESTful web services. JERSEY can be downloaded and installed from the link: <https://jersey.java.net/>. JERSEY should be added to the build path before implementing it.

**Step 1:** Create a package called "com.ottawau.cdstore.orderProcessing.rest" in the backend.

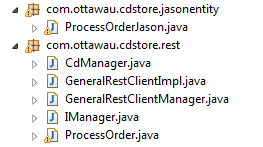
Right click on the project and select "new java package". Provide "orderprocessing.rest" as the package name.



#### Figure 14: Creating orderprocessing.rest package

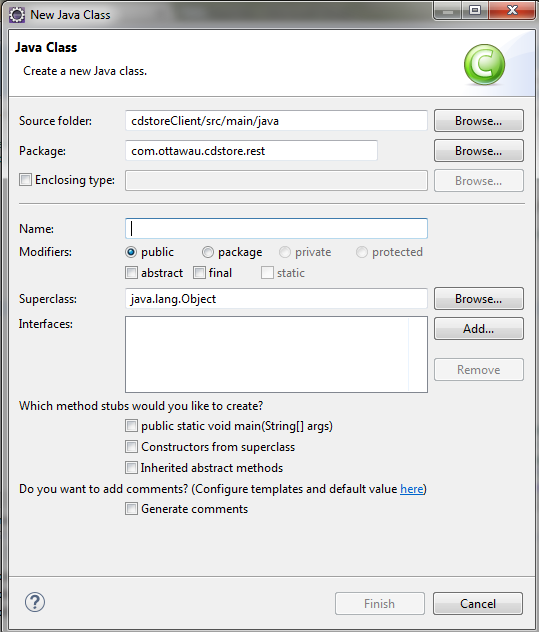
**Step 2:** Similarly, Create a package called "cdstore.rest" at the client end.

Right click on the project and select "new java package". Provide "cdstore.rest" as the package name.



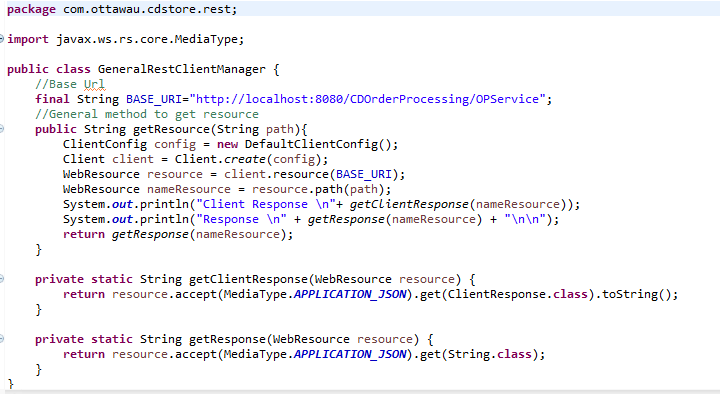
#### Figure 15: Creating cdstore.rest package

**Step 3:** Right click on the package and create new webservice class. Provide webservice names "depending on the functionalities written within it and click on "Finish".



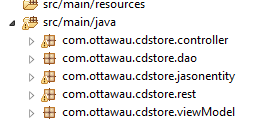
#### Figure 16: Creating Web Service

**Step 4:** Add suitable methods under the webservices.



#### Figure 17: Webservice methods

# 2. Organization of Source Code into Packages



#### Figure 18: Source code organized as packages

# 3. Individual contribution towards the project

|  |  |
| --- | --- |
| **Team Member** | **Member's contribution** |
| Aishvarya Arul Nambi  Indhra Priya Shanmugam | Code Implementation   * Session Controller * DB Agent |
| Priyanka Patel  Rachana Chandrasekar | Code Implementation   * Web services * DB Agent |
| Aishvarya Arul Nambi  Indhra Priya Shanmugam | Code Implementation   * Web Services * Jsps |
| Aishvarya Arul Nambi  Indhra Priya Shanmugam  Priyanka Patel  Rachana Chandrasekar  ShruthiMadhurika Naomi | Architecture Design   * Jsps   Testing  Documentation   * Design Document * Read Me * Test Document |
| Aishvarya Arul Nambi  Indhra Priya Shanmugam  ShruthiMadhurika Naomi | Architecture Design,   * Website Design(Html, CSS, JavaScript and JQuery),   Testing  Documentation   * Read Me * Design Document * Test Document |
| Priyanka Patel  Rachana Chandrasekar  ShruthiMadhurika Naomi | Code Implementation   * Jsps * Website Design(Html, CSS, JavaScript and JQuery) |

# 4. Third Party Dependencies

* **mysql-connector-java-5.1.26**

The MySQL Connector provides access to MySQL Database using ODBC API. mysql-connector-java-5.1.26-bin.jaris placed under Tomcat's lib folder to setup the connection pool and connection between the Java EE and MySQL DataBase in our project.

* **Google gson library**

gson-2.2.4 is the Java library that we have used in our project to convert Java objects into their JSON representation. Gson, along with providing simple toJson() and fromJson() methods to convert Java objects to JSON and vice-versa, also provides extensive support to Java Generics and allows custom representation for objects.

* **JQuery**

JQuery is the fast and feature-rich JavaScript library, used in our project which made event handling, HTML document traversal and manipulation simpler.

# 5. Testing

Kindly refer "Application Testing" document for all the application related testing screenshots.

# 6. References

#### SSL Configuration

<http://tomcat.apache.org/tomcat-6.0-doc/ssl-howto.html>

<http://www.mulesoft.com/tomcat-ssl>

<http://tomcat.apache.org/>

#### Third Party Dependencies

<https://code.google.com/p/google-gson/>

<http://dev.mysql.com/doc/refman/5.6/en/connector-odbc.html>

<http://jquery.com/>