

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
2	IndhraPriyaShanmugam	7527045	ishan050@uottawa.ca
3	Priyanka Patel	7481432	ppatel038@uottawa.ca
4	RachanaChandrashekar	7487187	rchan092@uottawa.ca
5	ShruthiMadhurika Naomi	7564806	snaomi011@uottawa.ca

Table of Contents

PROJECT PART 1	1
CD STORE	1
README DOCUMENT.....	1
1. Instructions for Installation/Configuration:	4
1.1 Installing Java (J2SE).....	4
1.2 Installing Eclipse	5
1.3 Installing Tomcat.....	6
1.4 Configuring Tomcat on Eclipse.....	8
1.5 Configuring the browser	9
1.6 Testing tomcat setup	10
1.7 Installing MySQL.....	11
1.8 Setting up SSL configuration	12
1.9 Connection Pool	14
1.10 Web Service Configuration	14
1.10.1 Creating Web Services:	15
2. Organization of Source Code into Packages	166
3. Individual contribution towards the project.....	17
4. Third Party Dependencies	177
5. Testing.....	18
6. References	188

List of Figures:

Figure 1: Successful installation of java	5
Figure 2: Successful installation of Eclipse IDE	6
Figure 3: Starting the Tomcat server	7
Figure 4: Tomcat running successfully	7
Figure 5: Selecting the Tomcat server for configuring the server in Eclipse	9
Figure 6: Web Browser configured in Eclipse	10
Figure 7: Result in the browser	11
Figure 8: Successful installation of MySQL	11
Figure 9: Certificate creation using keytool	12
Figure 10: Configuring Tomcat's SSL Connector	13
Figure 11: Tomcat successfully configured to run on https.....	13
Figure 12: running index.jsp on "localhost:8443"/https.....	14
Figure 13: context.xml file with the resource tag.....	14
Figure 14: Creating orderprocess package	15
Figure 15: Creating productcatalog package	15
Figure 16: Creating WebService.....	16
Figure 17: Webservice methods	16
Figure 18: Source code organized as packages.....	16
• SSLConfiguration	18
• Third Party Dependencies.....	18

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.

```

C:\Program Files\Java\jdk1.7.0_51\bin>javac
Usage: javac <options> <source files>
where possible options include:
  -g               Generate all debugging info
  -g:none          Generate no debugging info
  -g:{lines,vars,source} Generate only some debugging info
  -nowarn          Generate no warnings
  -verbose         Output messages about what the compiler is doing
  -deprecation     Output source locations where deprecated APIs are used
  -classpath <path> Specify where to find user class files and annotations
  -cp <path>        Specify where to find user class files and annotations
  -sourcepath <path> Specify where to find input source files
  -bootclasspath <path> Override location of bootstrap class files
  -extdirs <dirs>    Override location of installed extensions
  -endorseddirs <dirs> Override location of endorsed standards path
  -proc:{none,only} Control whether annotation processing and/or compilation is done.
  -processor <class1>[,<class2>,<class3>...] Names of the annotation processors to run; bypasses default discovery process
  -processorpath <path> Specify where to find annotation processors
  -d <directory>       Specify where to place generated class files
  -s <directory>       Specify where to place generated source files
  -implicit:{none,class} Specify whether or not to generate class files for implicitly referenced files
  -encoding <encoding> Specify character encoding used by source files
  -source <release>    Provide source compatibility with specified release
  -target <release>    Generate class files for specific VM version
  -version            Version information
  -help              Print a synopsis of standard options
  -Akey[=value]      Options to pass to annotation processors
  -X                 Print a synopsis of nonstandard options
  -J<flag>            Pass <flag> directly to the runtime system
  -Werror            Terminate compilation if warnings occur
  @<filename>         Read options and filenames from file

C:\Program Files\Java\jdk1.7.0_51\bin>

```

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.

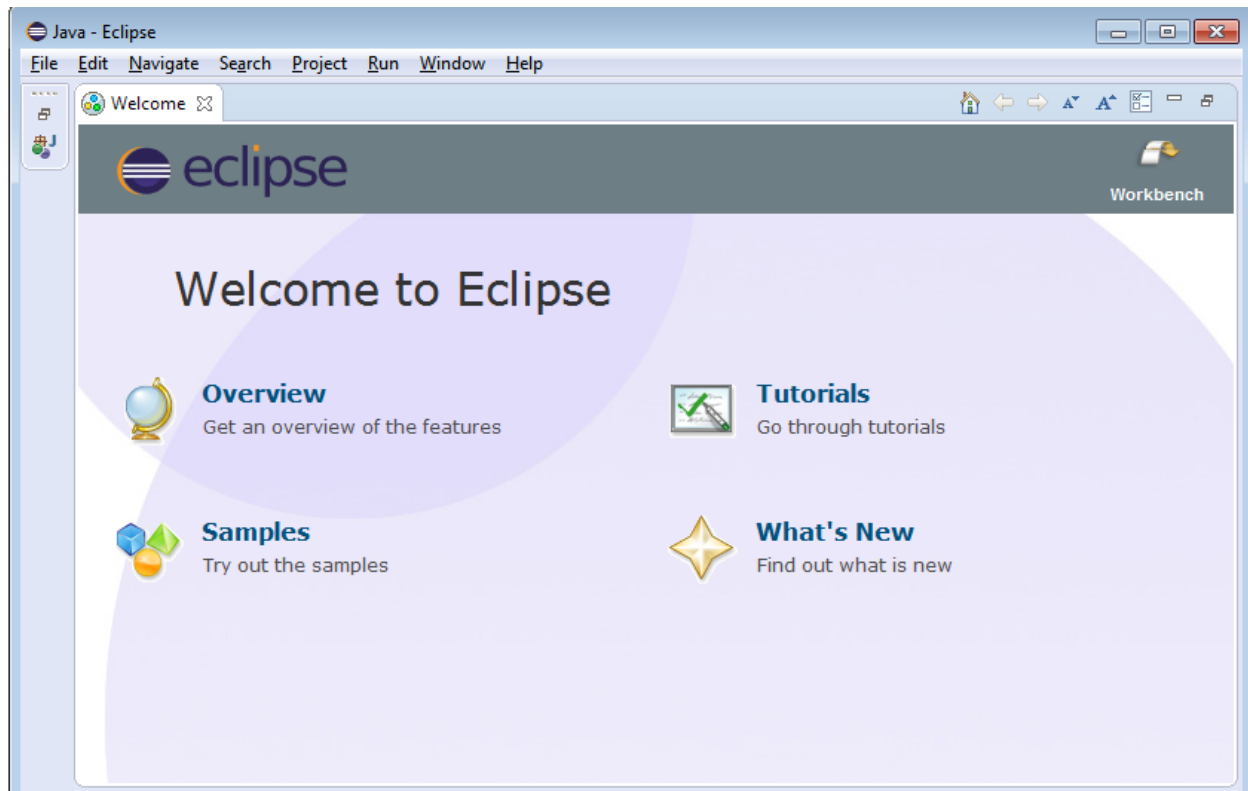


Figure 2: Successful installation of Eclipse IDE

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.

```

C:\Program Files\Apache Software Foundation\Tomcat 7.0\bin\Tomcat7.exe
Sep 10, 2013 4:37:26 PM org.apache.catalina.core.AprLifecycleListener init
INFO: The APR based Apache Tomcat Native library which allows optimal performanc
e in production environments was not found on the java.library.path: C:\Program
Files\Apache Software Foundation\Tomcat 7.0\bin;C:\windows\Sun\Java\bin;C:\windo
ws\System32;C:\windows;C:\Program Files\Common Files\Microsoft Shared\Windows Li
ve;C:\Program Files (x86)\Common Files\Microsoft Shared\Windows Live;C:\windows\
System32;C:\windows;C:\windows\System32\Wbem;C:\windows\System32\WindowsPowerShe
ll\cmd.exe;C:\Program Files\Lenovo\Bluetooth Software\bin;C:\Program Files\Lenovo\Blu
etooth Software\bin;C:\Program Files (x86)\Windows Live\Windows Live;C:\Program Fi
les\TortoiseSUN\bin;C:\Program Files (x86)\Microsoft SQL Server\100\Tools\Binn\
;C:\Program Files\Microsoft SQL Server\100\Tools\Binn\;C:\Program Files\Microsoft
SQL Server\100\DTSPowerShell\bin;C:\Program Files (x86)\Microsoft SQL Server\100\Tools
\Binn\SShell\bin;C:\Program Files (x86)\Microsoft SQL Server\100\Tools\Binn\;C:\Co
mmon7\IDE\PrivateAssemblies;C:\Program Files (x86)\Microsoft SQL Server\100\DTSP
\bin;C:\Program Files (x86)\Microsoft SQL Server\90\Tools\Binn\;C:\Program Fil
es\Microsoft Web Platform Installer\bin;C:\Program Files (x86)\Microsoft ASP.NET AS
P.NET Web Pages\v1.0\bin;C:\Program Files\Microsoft SQL Server\110\Tools\Binn\;.
Sep 10, 2013 4:37:28 PM org.apache.coyote.AbstractProtocol init
INFO: Initializing ProtocolHandler ["http-bio-8080"]
Sep 10, 2013 4:37:28 PM org.apache.coyote.AbstractProtocol init
INFO: Initializing ProtocolHandler ["ajp-bio-8009"]
Sep 10, 2013 4:37:28 PM org.apache.catalina.startup.Catalina load
INFO: Initialization processed in 3497 ms
Sep 10, 2013 4:37:29 PM org.apache.catalina.core.StandardService startInternal
INFO: Starting service Catalina
Sep 10, 2013 4:37:29 PM org.apache.catalina.core.StandardEngine startInternal
INFO: Starting Servlet Engine: Apache Tomcat/7.0.42
Sep 10, 2013 4:37:29 PM org.apache.catalina.startup.HostConfig deployWAR
INFO: Deploying web application archive C:\Program Files\Apache Software Foundat
ion\Tomcat 7.0\webapps\quickforms-test.war
Sep 10, 2013 4:37:31 PM org.apache.catalina.startup.HostConfig deployWAR
INFO: Deploying web application archive C:\Program Files\Apache Software Foundat
ion\Tomcat 7.0\webapps\quickforms.war
Sep 10, 2013 4:37:32 PM org.apache.catalina.startup.HostConfig deployDirectory
INFO: Deploying web application directory C:\Program Files\Apache Software Found
ation\Tomcat 7.0\webapps\docs
Sep 10, 2013 4:37:32 PM org.apache.catalina.startup.HostConfig deployDirectory
INFO: Deploying web application directory C:\Program Files\Apache Software Found
ation\Tomcat 7.0\webapps\manager
Sep 10, 2013 4:37:32 PM org.apache.catalina.startup.HostConfig deployDirectory
INFO: Deploying web application directory C:\Program Files\Apache Software Found
ation\Tomcat 7.0\webapps\ROOT
Sep 10, 2013 4:37:32 PM org.apache.catalina.startup.HostConfig deployDirectory
INFO: Deploying web application directory C:\Program Files\Apache Software Found
ation\Tomcat 7.0\webapps\rpp-test
Sep 10, 2013 4:37:32 PM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["http-bio-8080"]
Sep 10, 2013 4:37:32 PM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["ajp-bio-8009"]
Sep 10, 2013 4:37:32 PM org.apache.catalina.startup.Catalina start
INFO: Server startup in 3649 ms

```

Figure 3: Starting the Tomcat server

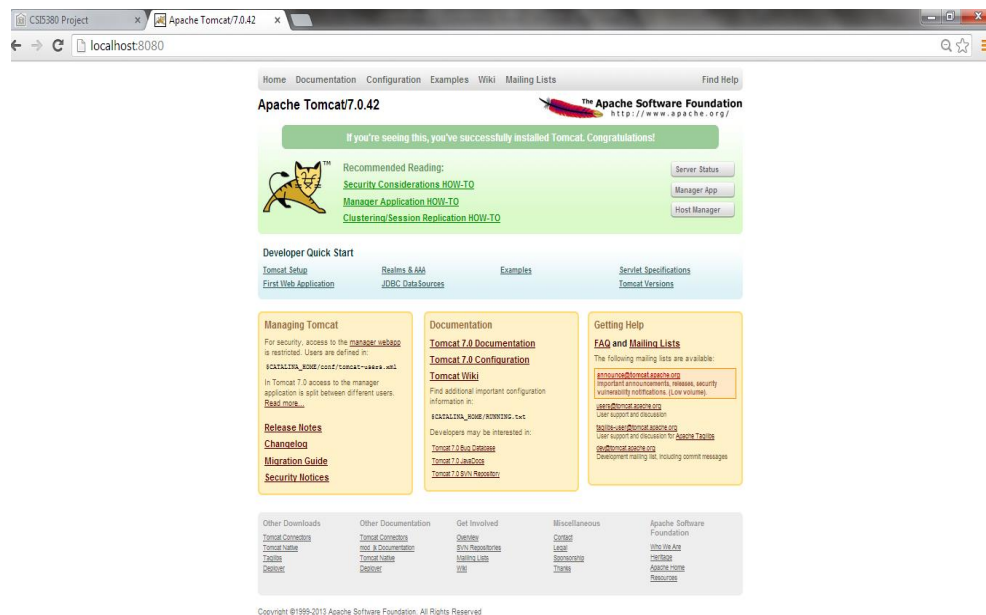


Figure 4: Tomcat running successfully

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.

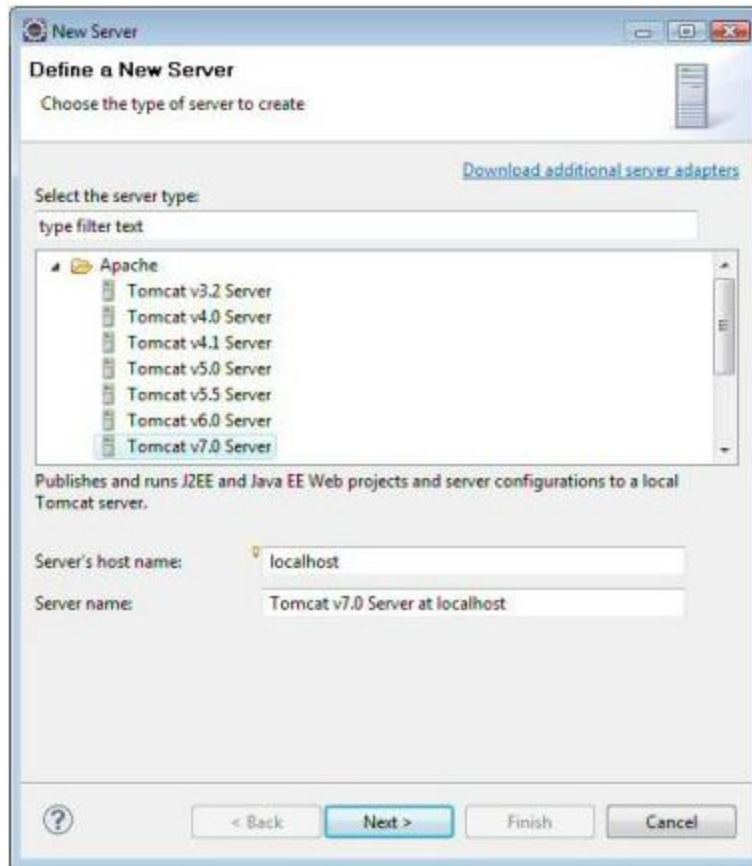


Figure 5: Selecting the Tomcat server for configuring the server in Eclipse

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.

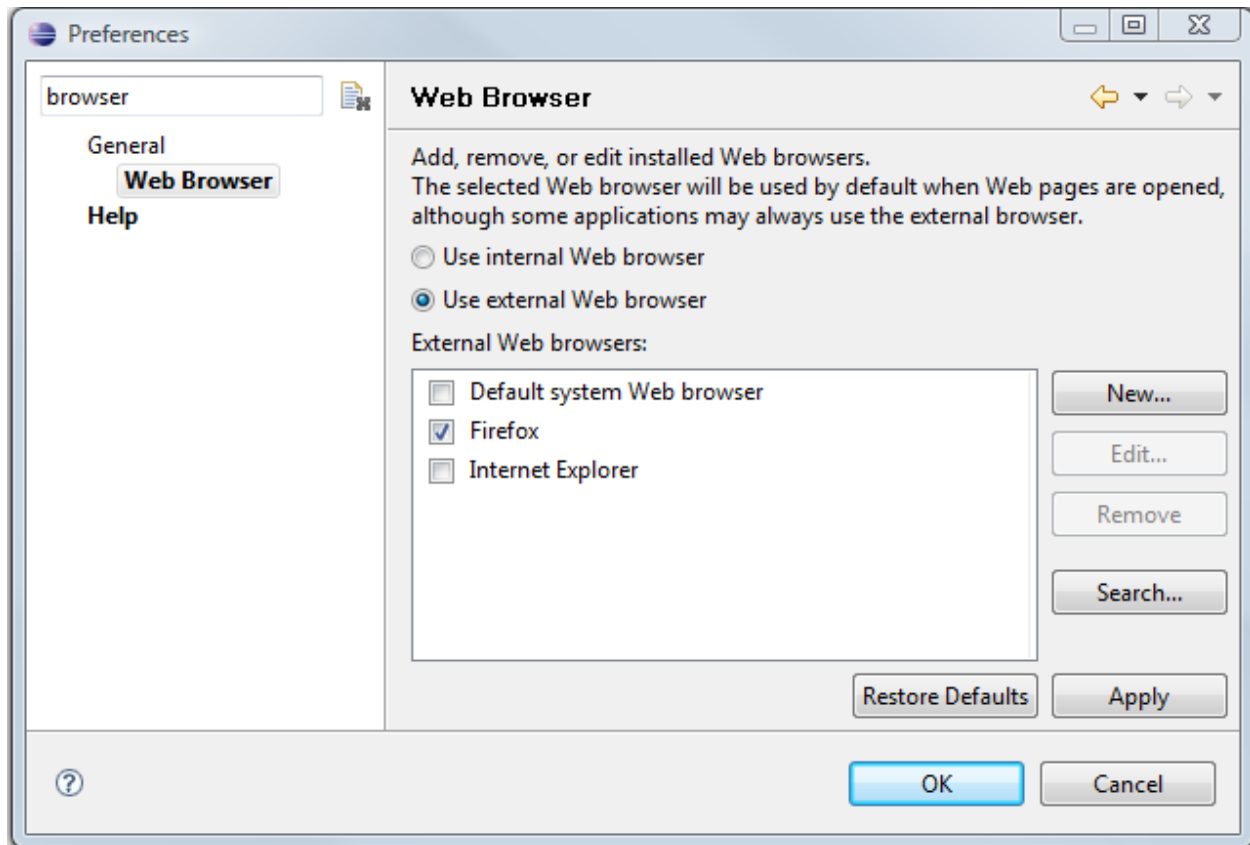


Figure 6: Web Browser configured in Eclipse

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.

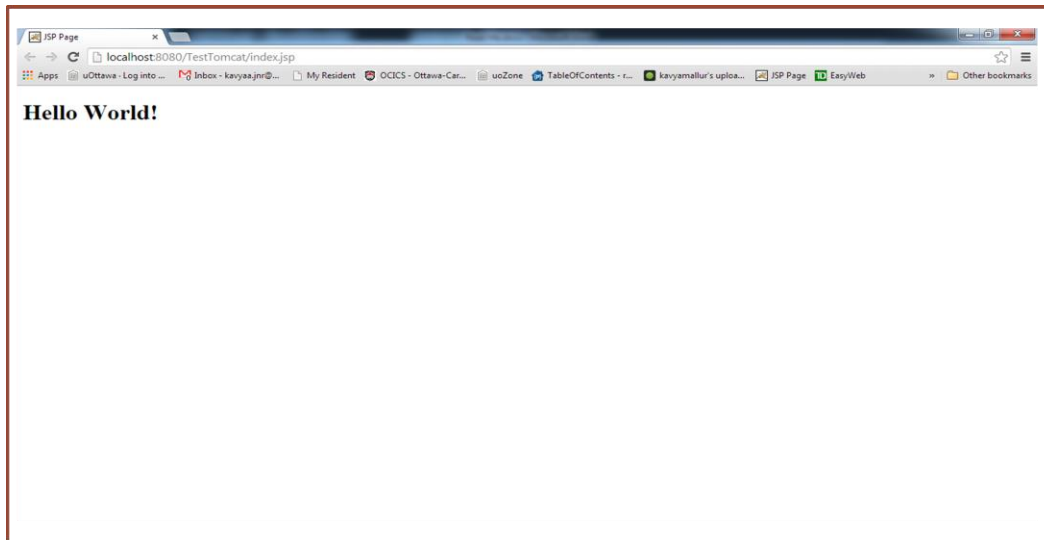


Figure 7: Result in the browser

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/>

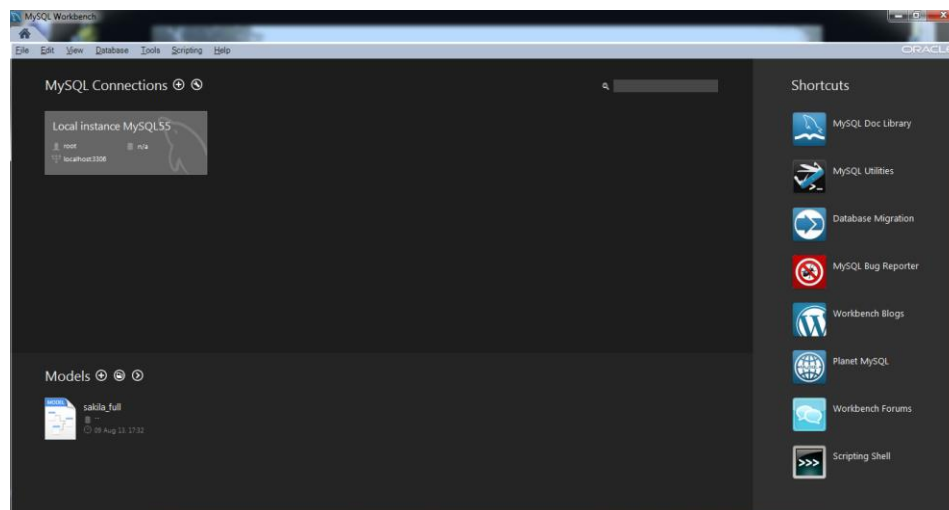


Figure 8: Successful installation of MySQL

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.

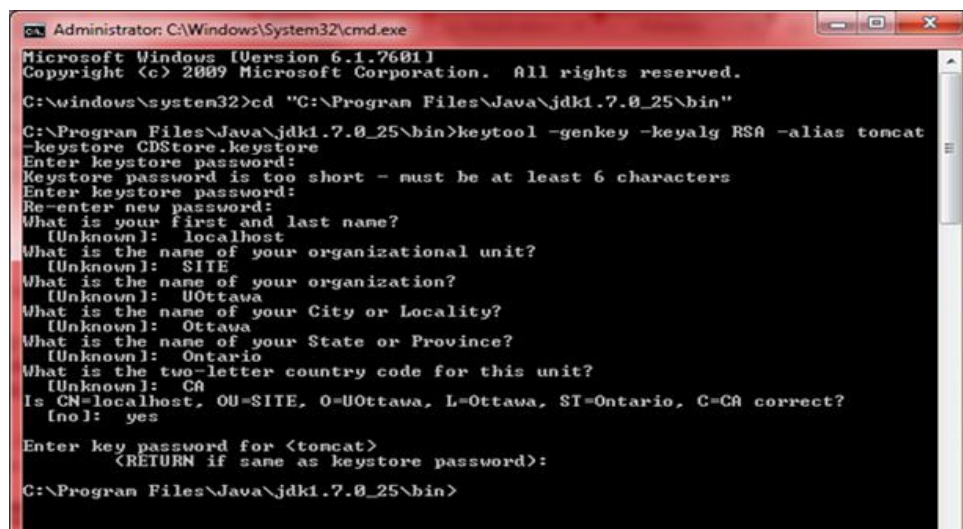
SetUp:

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.

```
$JAVA_HOME/bin/keytool-genkey-keyalgRSA -alias tomcat -keystore<Keystore file name>
```

Note:

RSA algorithm is preferred secure algorithm.
The command will create a new keystore file



```
Administrator: C:\Windows\System32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\windows\system32>cd "C:\Program Files\Java\jdk1.7.0_25\bin"

C:\Program Files\Java\jdk1.7.0_25\bin>keytool -genkey -keyalg RSA -alias tomcat
-keystore CDStore.keystore
Enter keystore password:
Keystore password is too short - must be at least 6 characters
Enter keystore password:
Re-enter new password:
What is your first and last name?
[Unknown]: localhost
What is the name of your organizational unit?
[Unknown]: SITE
What is the name of your organization?
[Unknown]: UOttawa
What is the name of your City or Locality?
[Unknown]: Ottawa
What is the name of your State or Province?
[Unknown]: Ontario
What is the two-letter country code for this unit?
[Unknown]: CA
Is CN=localhost, OU=SITE, O=UOttawa, L=Ottawa, ST=Ontario, C=CA correct?
[no]: yes
Enter key password for <tomcat>
<RETURN if same as keystore password>:

C:\Program Files\Java\jdk1.7.0_25\bin>
```

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.

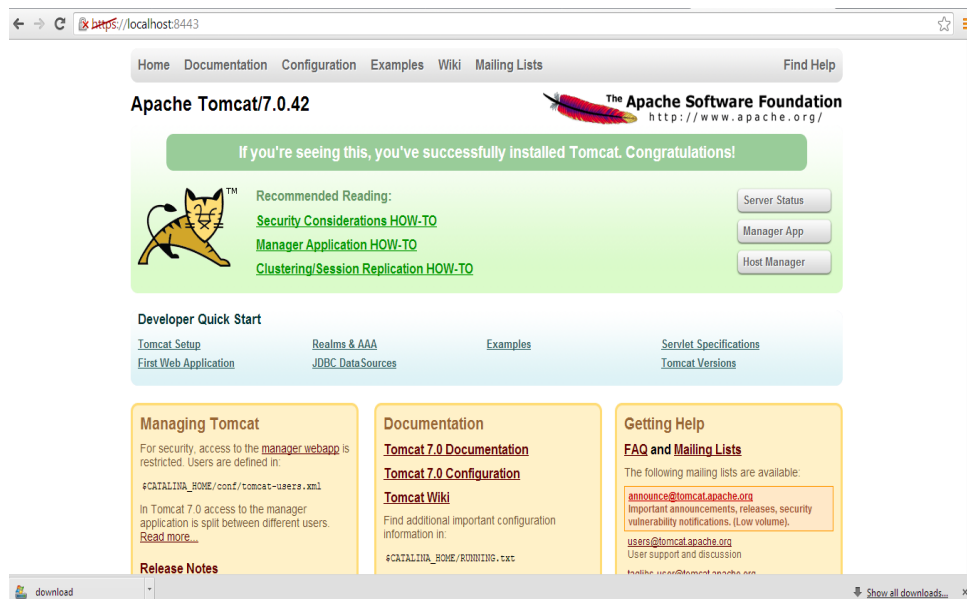
Uncomment the Connector Port=8443 portion

```
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
maxThreads="150" scheme="https" secure="true"
clientAuth="false" sslProtocol="TLS"
keystoreFile="C:/Program Files/Apache Software Foundation/Tomcat
7.0/CD_Store_Keystore.keystore"
keystorePass="*****"/>
```

```
70 <!-- A <Connector> using the shared thread pool -->
71 <Connector port="8080" protocol="HTTP/1.1"
72         connectionTimeout="20000"
73         redirectPort="8443" />
74
75 <!--
76 <Connector executor="tomcatThreadPool"
77         port="8080" protocol="HTTP/1.1"
78         connectionTimeout="20000"
79         redirectPort="8443" />
80
81 <!-- Define a SSL HTTP/1.1 Connector on port 8443
82 This connector uses the JSSE configuration, when using APR, the
83 connector should be using the OpenSSL style configuration
84 described in the APR documentation -->
85 <Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
86         maxThreads="150" scheme="https" secure="true"
87         clientAuth="false" sslProtocol="TLS"
88         keystoreFile="C:/Program Files (x86)/Apache Software Foundation/Tomcat 7.0/CDStore.keystore"
89         keystorePass="*****" />
90
91 <!-- Define an AJP 1.3 Connector on port 8009 -->
92 <Connector port="8009" protocol="AJP/1.3" redirectPort="8443" />
93
94 <!-- An Engine represents the entry point (within Catalina) that processes
95 every request. The Engine implementation for Tomcat stand alone
96 analyzes the HTTP headers included with the request, and passes them
97 on to the appropriate Host (virtual host).
98 Documentation at /docs/config/engine.html -->
99
100 <!-- You should set vmRoute to support load-balancing via AJP ie :
101
102
```

Figure 10: Configuring Tomcat's SSL Connector

Step 4: Check if the Tomcat is successfully configured for https. Provide "<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.

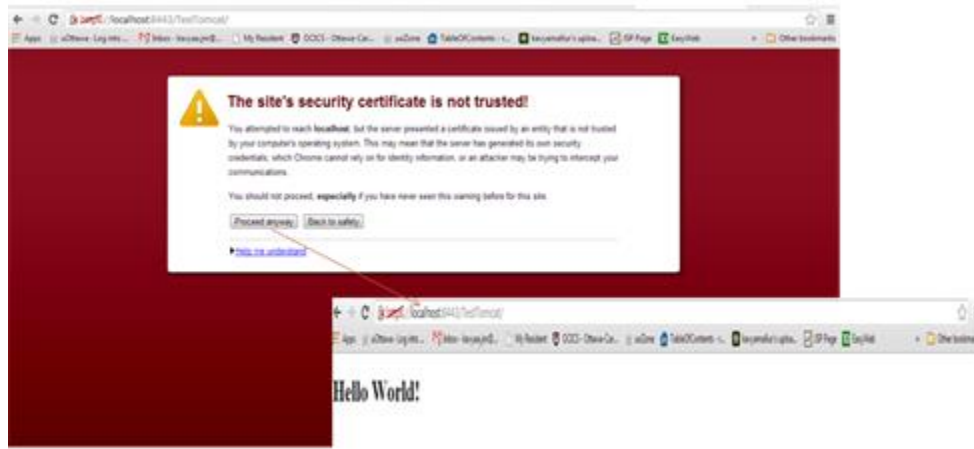


Figure 12: Running index.jsp on "localhost: 8443"/https

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.

```
<!-- Connection Pooling Settings -->
<property name="hibernate.c3p0.min_size">5</property>
<property name="hibernate.c3p0.max_size">20</property>
<property name="hibernate.c3p0.timeout">300</property>
<property name="hibernate.c3p0.max_statements">50</property>
<property name="hibernate.c3p0.idle_test_period">3000</property>
```

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.

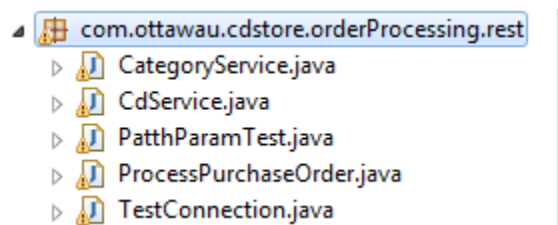


Figure 14: Creating orderprocessing.rest package

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.

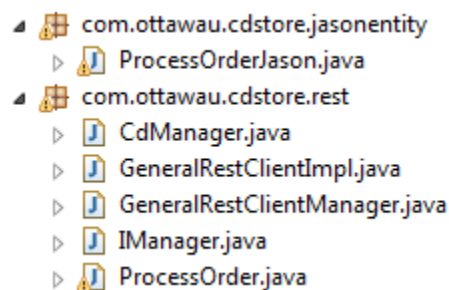


Figure 15: Creating cdstore.rest package

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".

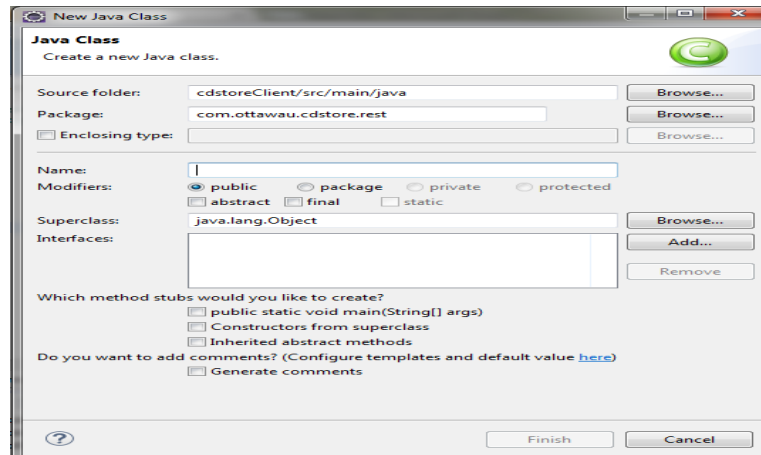


Figure 16: Creating Web Service

Step 4: Add suitable methods under the webservices.

```
package com.ottawau.cdstore.rest;

import javax.ws.rs.core.MediaType;

public class GeneralRestClientManager {
    //Base Url
    final String BASE_URI="http://localhost:8080/CDOrderProcessing/OPService";
    //General method to get resource
    public String getResource(String path){
        ClientConfig config = new DefaultClientConfig();
        Client client = Client.create(config);
        WebResource resource = client.resource(BASE_URI);
        WebResource nameResource = resource.path(path);
        System.out.println("Client Response \n"+ getClientResponse(nameResource));
        System.out.println("Response \n" + getResponse(nameResource) + "\n\n");
        return getResponse(nameResource);
    }

    private static String getClientResponse(WebResource resource) {
        return resource.accept(MediaType.APPLICATION_JSON).get(ClientResponse.class).toString();
    }

    private static String getResponse(WebResource resource) {
        return resource.accept(MediaType.APPLICATION_JSON).get(String.class);
    }
}
```

Figure 17: Webservice methods

2. Organization of Source Code into Packages

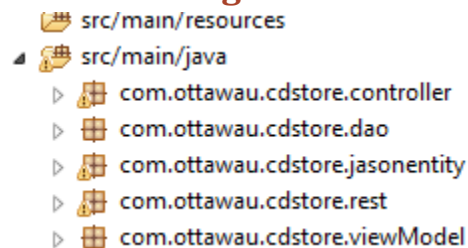


Figure 18: Source code organized as packages

3. Individual contribution towards the project

Team Member	Member's contribution
Aishvarya Arul Nambi Indhra Priya Shanmugam	Code Implementation <ul style="list-style-type: none"> • Session Controller • DB Agent
Priyanka Patel Rachana Chandrasekar	Code Implementation <ul style="list-style-type: none"> • Web services • DB Agent
Aishvarya Arul Nambi Indhra Priya Shanmugam	Code Implementation <ul style="list-style-type: none"> • Web Services • Jsps
Aishvarya Arul Nambi Indhra Priya Shanmugam Priyanka Patel Rachana Chandrasekar ShruthiMadhurika Naomi	Architecture Design <ul style="list-style-type: none"> • Jsps Testing Documentation <ul style="list-style-type: none"> • Design Document • Read Me • Test Document
Aishvarya Arul Nambi Indhra Priya Shanmugam ShruthiMadhurika Naomi	Architecture Design, <ul style="list-style-type: none"> • Website Design(Html, CSS, JavaScript and JQuery), Testing Documentation <ul style="list-style-type: none"> • Read Me • Design Document • Test Document
Priyanka Patel Rachana Chandrasekar ShruthiMadhurika Naomi	Code Implementation <ul style="list-style-type: none"> • 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.jar is 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/>