International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 4 (2015) pp. 9773-9787

http://www.ripublication.com

Study On Implementing Web Services Using JAVA Spring With Eclipse And Apache Tomcat

Venkadesh.M M.tech[1], Dr.A.Chandra Sekar M.E., Ph.d MISTE[2]

[1]ResearchScholar, Bharath University, Chennai 73, India.
venkadeshkumaresan@yahoo.co.in
[2]Professor-CSC Department ,St.Joseph's College Of Engineering,
Jeppiaar Nagar, Chennai – 600119.
drchandrucse@gmail.com

Abstract

Web Services is used mainly for the purpose of communicating users of different platform within the network connection. Web Services mainly uses SOA architecture. In this paper we are going to implement web services using Java Spring. Spring is an open-source framework most commonly used development framework for Java. This study is based on how to implement web service and using it in spring applications. I have used Apache Tomcat, Eclipse and spring framework to implement web services.

Keywords: Web Services, spring MVC, spring AOP, spring.

1. Introduction

Let us explain the implementation of Web Services in this paper. Web services can be accessed by users of different softwares. Web services [6] can be implemented using various softwares. In this paper we are using RESTful web services [6, 5] using Spring with Java Eclipse. [1] Spring is most commonly used with Java. It is very easy to integrate with other web frameworks. The web component of spring's framework is Spring MVC [2].

This paper deals with the implementation study of Web services using Java Spring with Eclipse and Apache Tomcat. In this paper we have discussed about spring in Section 2. In Section 3 and 4 we are discussing on the Spring MVC [2] and Spring AOP [3] which in short about the spring architecture. In section 5 we deal with Web Services [6]. In the following section we describe the implementation process followed by conclusion and future enhancements. The paper ends with the references section at last.

2. Spring

Spring is the most popular framework used in JAVA applications. It has much number of modules. It is an open-source application program for the Java platform. With the usage of spring Java applications are made easier and simple. [1] The several modules of Spring framework are:

- Spring Core Container
- Aspect-Oriented Programming
- Application Context
- O/R Mapping
- Model- View- Controller
- Web Connect and Utility
- JDBC and DAO

In this paper we are going to deal with spring core container, spring MVC[2] and spring AOP[3].I have dealt this study to create a web domain named "TRAVELSONLINE" where we get the user's id and password and validate using web services. We used the software tool Eclipse to develop and deployed in Tomcat.

3. Spring MVC

Spring MVC [2] is used to develop web domain. Similar to Struts, spring is also a request-based framework. It provides model-view-controller architecture [1]. In our study we are using the following as the mvc components

M- UserDO.java
V- success.jsp
C- WelcomeController.java

Since Spring MVC [2] is highly configurable, Spring also can integrate easily with other popular Web Frameworks like Struts, WebWork, Java Server Faces and Tapestry.

[2] Spring MVC framework mainly works on the basis of DispatcherServlet framework model [2,3,7] (figure 1 [7]).

- The DispatcherServlet framework [2,3] receives the HTTP Request.
- It then maps the request with the Handler Mapping and gets the appropriate control.
- The Controller uses the GET or POST method and takes the appropriate service needed for the request.
- View Resolver gives the specified service for the request send.
- The view option is the process of showing the model data in the browser.

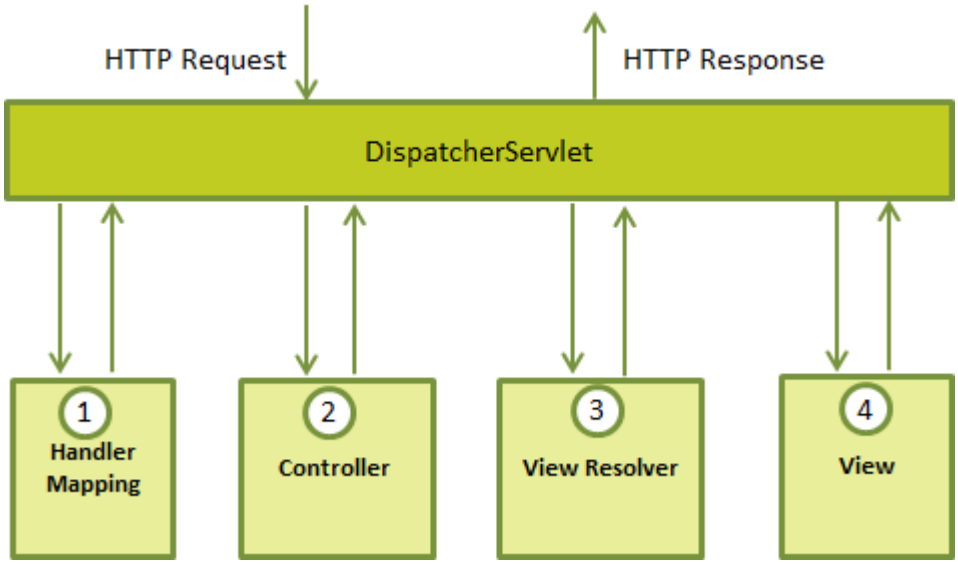

Figure 1. Spring MVC request flow diagram

4. Spring AOP

Spring AOP [3] uses modularity as in OOPS. But the key unit of modularity is Aspect whereas in OOPS it is class. Spring follows its own AOP framework. Aspect Oriented Programming breaks the program logic into small parts. These small parts are called Concerns. Common examples of Aspects are logging, auditing, security, caching, declarative transactions etc. An application may contain many numbers of Aspects as required. Advice gives details about the perfect action to take place before and after the method is executed. The place where the action is taking place is called the Join Point. The set of one or more Join points where the advice is executed is called the Point cut. Aspects has five kinds of Advice namely before, after, around, after-returning, after-throwing [3].

5. Web Services

[6] Web services are usually identified by Uniform Resource Locator (URL). Web services are a connection of two or more electronic devices in a network to transfer messages between them. Web services connection is made through World Wide Web. A remote computer using any platform can communicate the server with different platform using web service. Web service makes the user access platform independent. Web service has the capacity to convert any existing applications into web applications. Web services are XML based system that exchange information using the internet. There are two types of web services namely

- Big Web Services
- RESTful Web Services

Here we are using RESTful web services (figure 2 [5]). RESTful web services mean REpresentational State Transfer web services [5]. RESTful web services will be the best in integrating with HTTP than the SOAP based services. RESTful web services are completely stateless. It is very easy for the developers to build RESTful web services with Java and JVM (Java Virtual Machine). RESTful web services are a light weight component.

Figure 2 Rest Web Service flow diagram

6. Implementation of spring using Eclipse

The following are the steps to implement spring using Eclipse.

SOFTWARE USED:

- Java Spring 3.2
- Apache Logging
- J2ee1.5

SERVER:

- Apache Tomcat 1.7

TOOLS:

- Eclipse Luna

Step 1:

Open the Eclipse tool. In the Eclipse tool right click and select New option and in that select Other option and then Web option.

Right click- New – Other – Web (figure 3 [8,9,10])

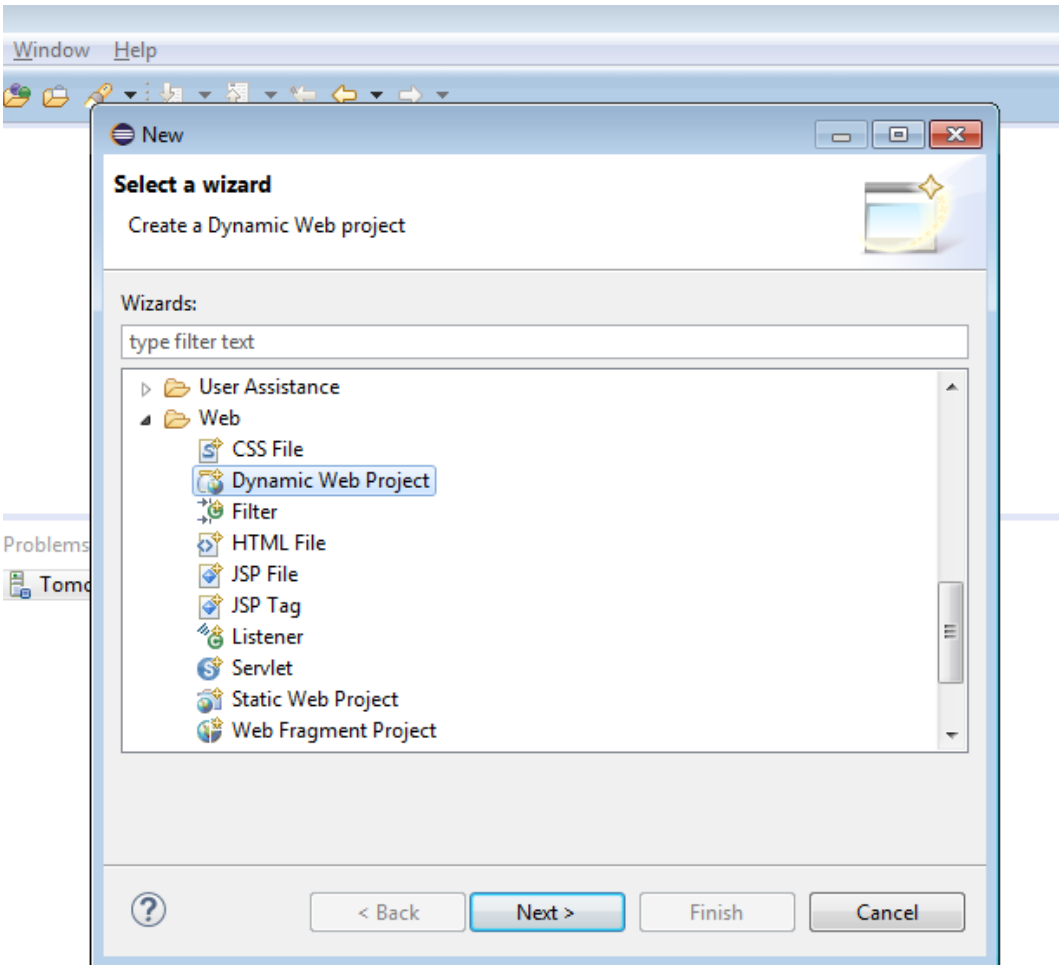

Figure 3. Create dynamic project in Eclipse.

Step 2:

In the Project name tab we have to give TravelsOnline. Select Apache Tomcat in Target Runtime (figure 4[8,9,10]).

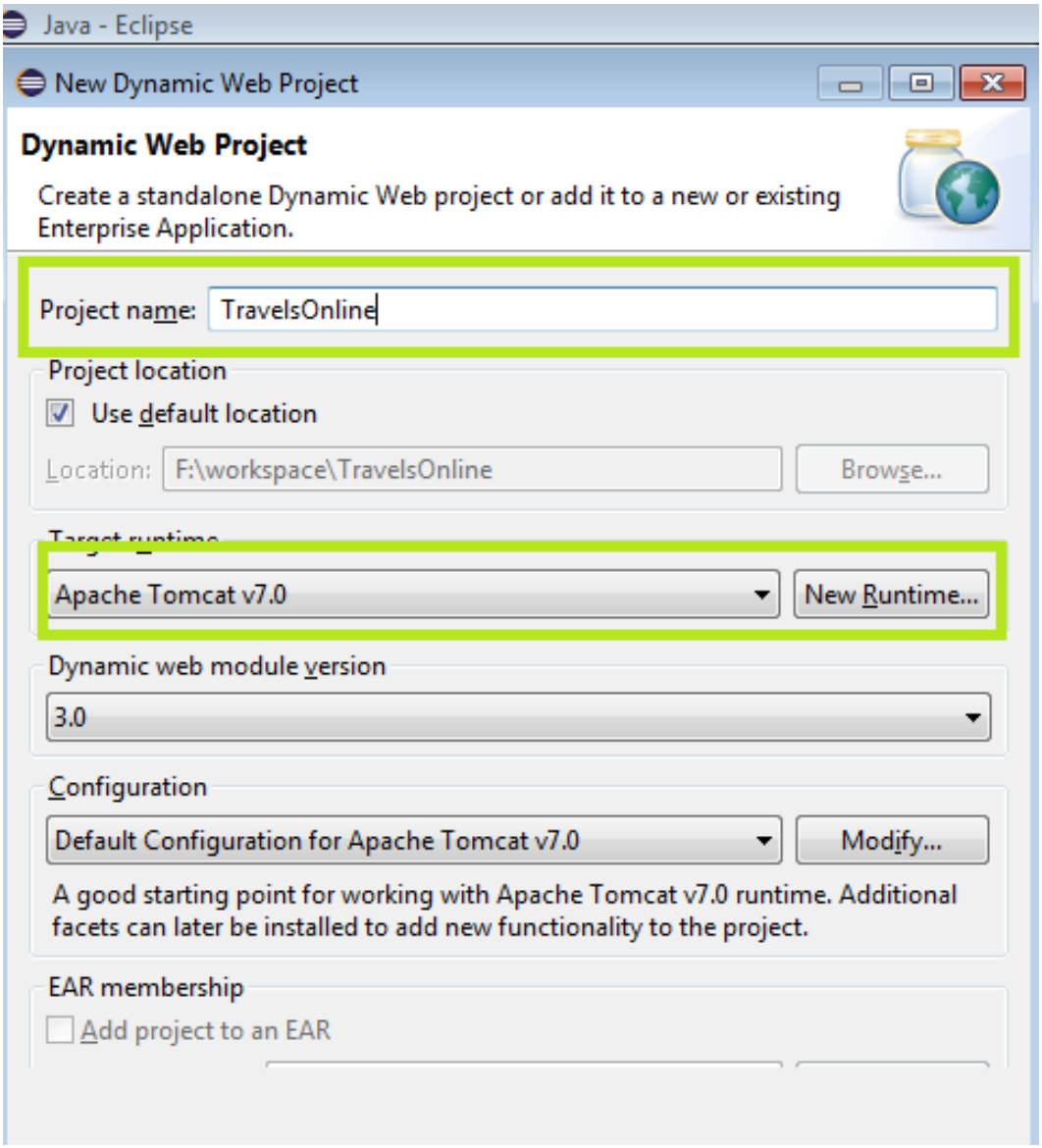

Figure 4. Add domain name to dynamic web project.

Step 3:

We have to copy the required jar files from spring directory and place it to the lib folder and then copy the Apache commons logging and also place it to the lib folder (figure 5[8,10]).

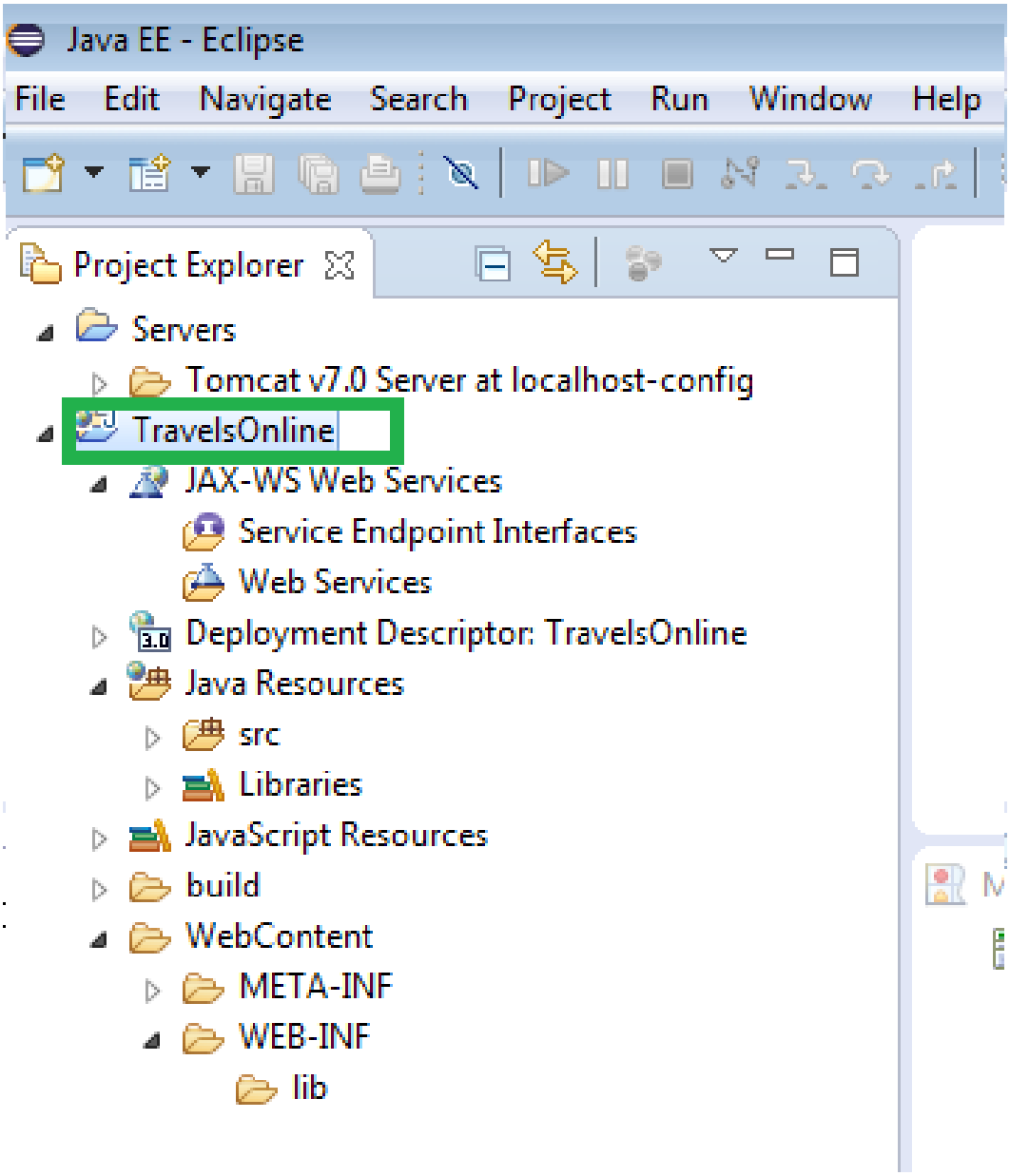

Figure 5. Dynamic web project structure in Eclipse.

Step 4:

We have to create web.xml file under WEB-INF directory (figure 6[8,10]). <welcome-file> tag is used to map the request url to this page. We should configure DispatcherServlet from the Spring application class file. If any request ends with .jsp then tomcat container calls the corresponding WelcomeController.

web.xml

```
<servlet>
<servlet-name>welcome</servlet-name>
<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
<load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
<servlet-name>welcome</servlet-name>
<url-pattern>*.jsp</url-pattern>
</servlet-mapping>

<welcome-file-list>
<welcome-file>
index.jsp
</welcome-file>
</welcome-file-list>

</web-app>
```

Figure 6. WEB.xml file structure

Step 5:

We should create <servlet-name>-servlet.xml file in WEB_INF directory, here we created welcome-servlet.xml. We have to create following files under WebContent directory.

- Index.jsp
- Success.jsp

We have to create WelcomeController file under SRC folder. Deploy this application in Tomcat server (figure 7[8,10]).

web.xml | welcome-servlet.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd">

<!-- the application context definition for the DispatcherServlet -->

<bean name="/welcome.jsp" class="com.web.WelcomeController"/>

</beans>
```

Figure 7. Welcome-servlet.xml file structure.

Step 6:
Right click Tomcat v7.0 server and then click Add and Remove (figure 8[11]). Step 6 and 7 is used to deploy the TravelsOnline application Apache Tomcat server.

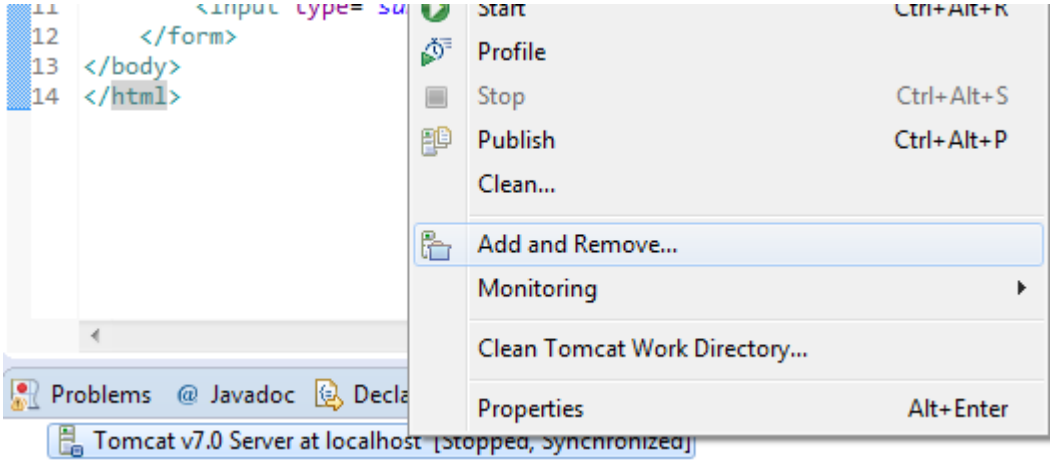

Figure 8. Add dynamic web project in Eclipse

Step 7:
Select TravelsOnline ear file and then Click Add button. At last click the finish button. Now our ear file is deployed in Tomcat(figure 9[11]).

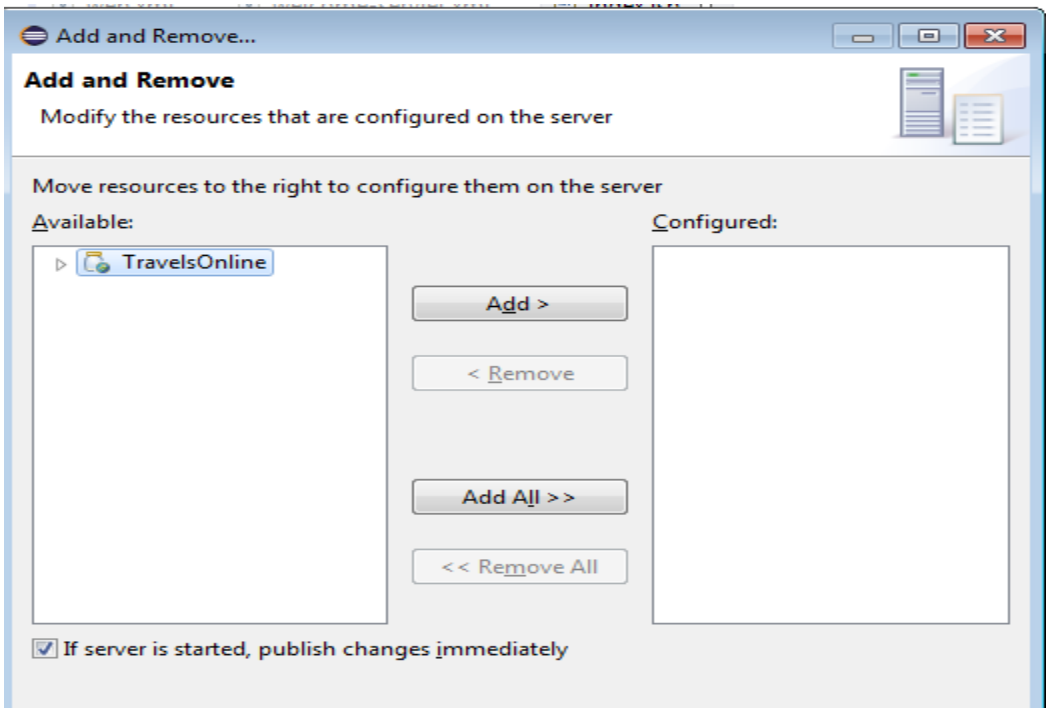

Figure 9. Deploy TravelOnline project in Tomcat.

Step 8:

Start the Tomcat server(figure 10[13]).

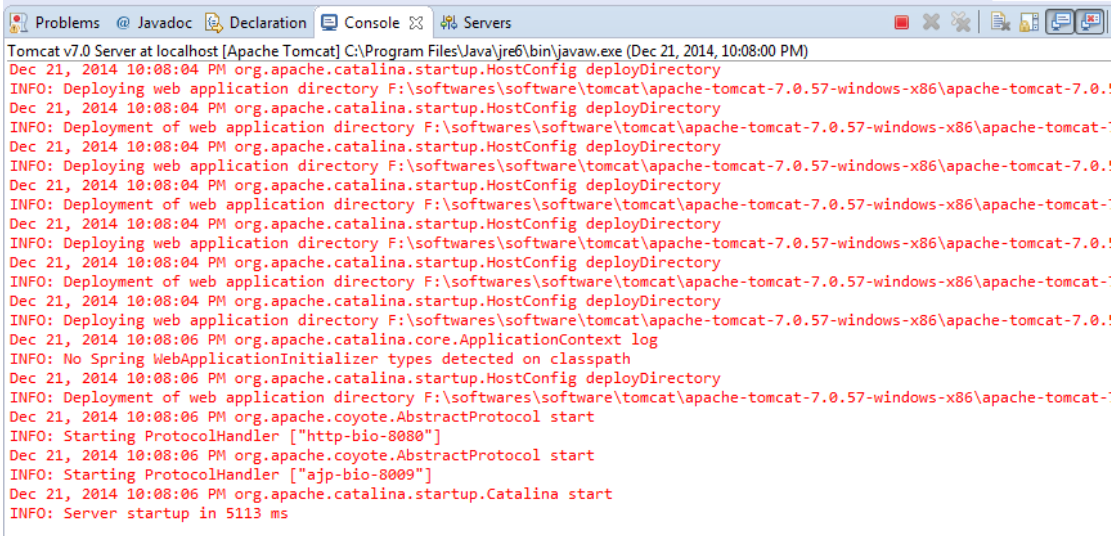

Figure 10. Tomcat server is started in local host.

Step 9:

Check whether Tomcat is running if we use the url http://localhost:8080/ (figure 11[12])

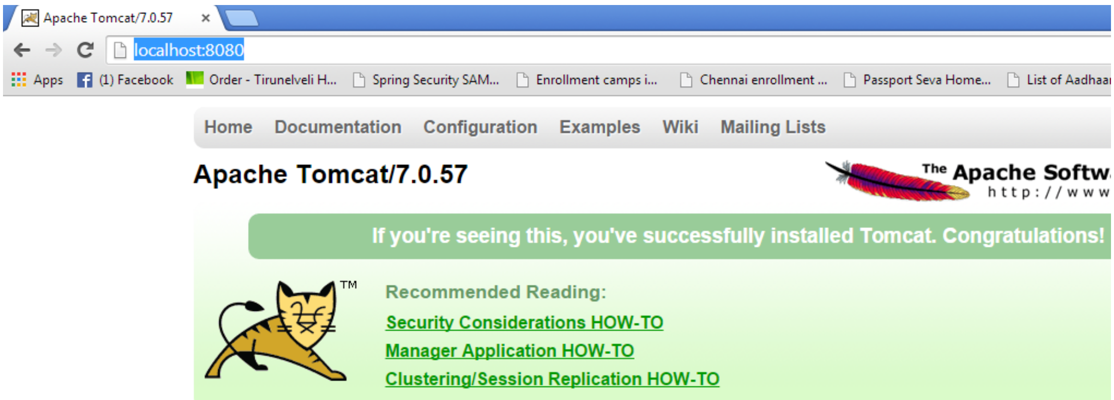

Figure 11. Tomcat server is running in localhost.

Step 10:

This will be the output view in the final step(figure 12[8,9,10]).

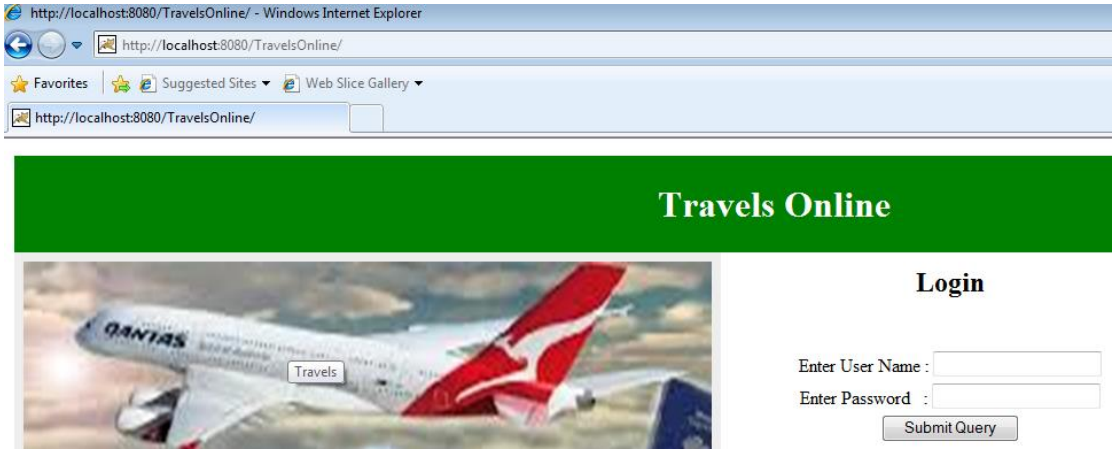

Figure 12. TravelsOnline domain main page.

7. Implementation of web services using spring with Java Eclipse

In implementation of web services we have to follow similar steps from step 1 to step 4 from implementation of spring using eclipse. After that the steps to be followed are as follows:

Step 1:

We created a new project LoginWebServices. Create Web.xml and <servlet-name>-servlet.xmlfigure 13[8].

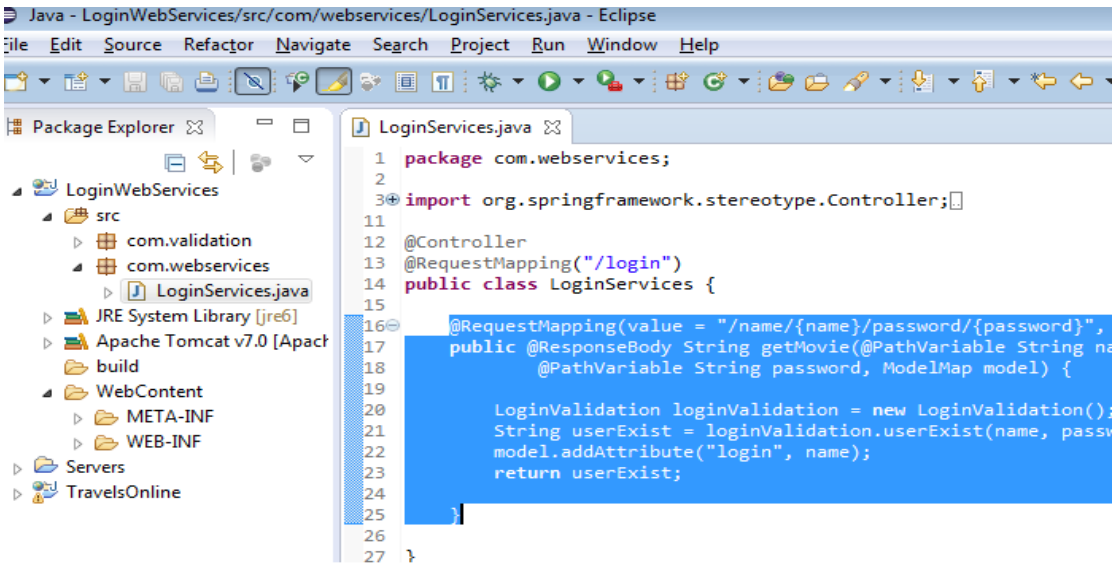

Figure 13. Spring controller structure.

[4] We created a java class named LoginServices.java. To make this class a web services we are using annotations @Controller [2, 3]. Now this class acts as a web service. The annotation @RequestMapping [2, 3] is used to access the web services. Here we are using the service login. We can access the web services using the following url http://localhost:8080/LoginWebServices/login. TravelsOnline page gets the username and password and uses this web services to authenticate the user credentials. Based on the web services response TravelsOnline page allows the user to enter the page.

To access the web services directly use the url http://localhost:8080/LoginWebServices/login/name/venkat/password/password (figure 14 [8]).

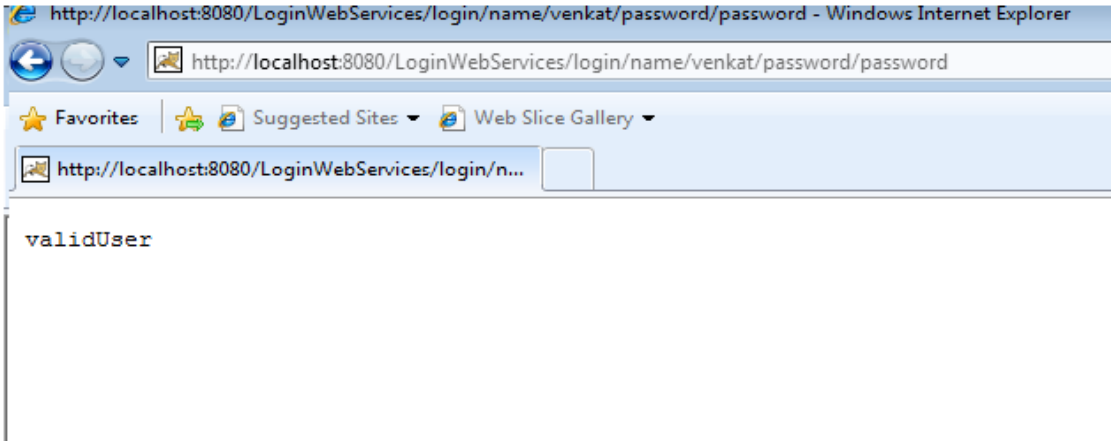

Figure 14. Direct access the REST web services.

Step 2:

This is the final output step. We have user name- venkat and password- password and click the submit button.

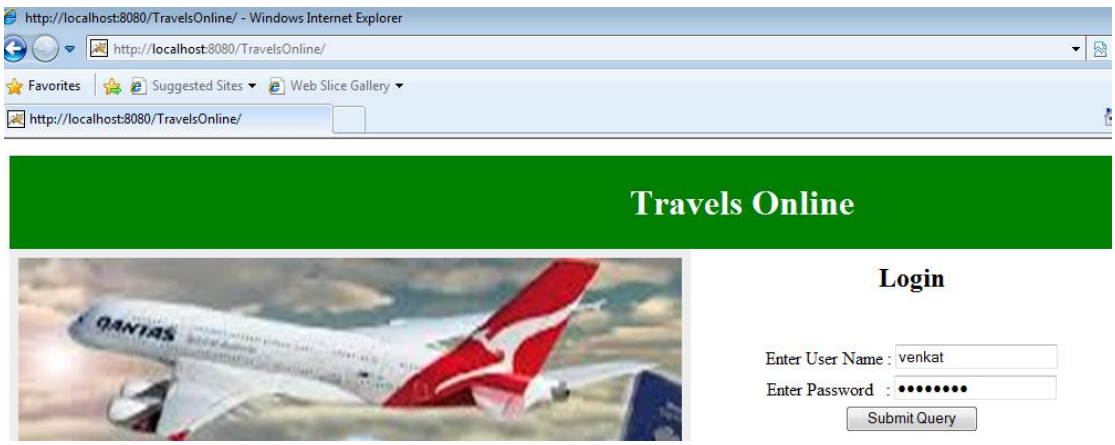

Figure 15. Enter valid user name and password.

After clicking the submit button if the user name is valid we get the following page (figure 15, 16 [8]).

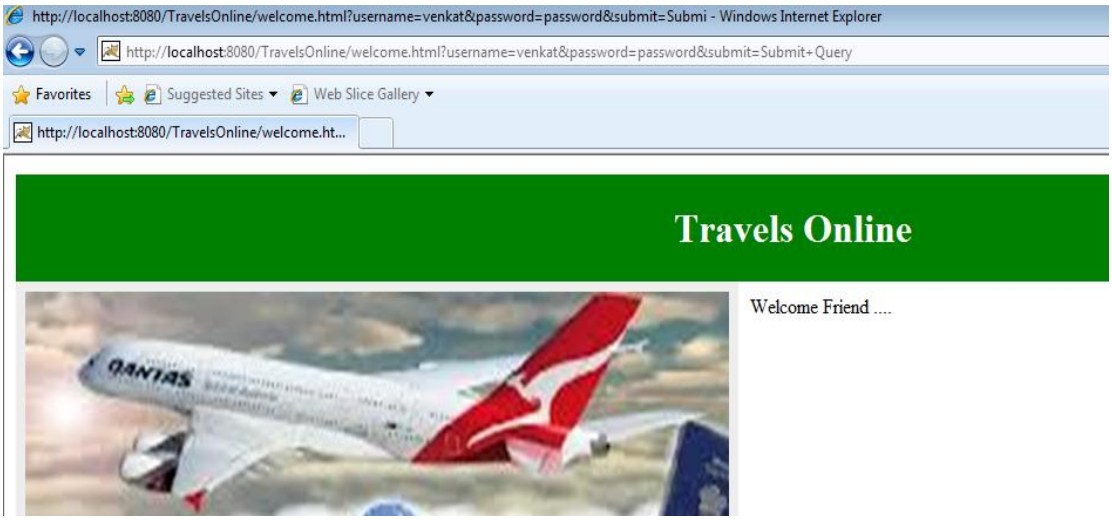

Figure 16. TravelsOnline domain main page.

If we use invalid user name and password the following pages appear (figure 17, 18[8]).
Username- sindhu
Password- venkat

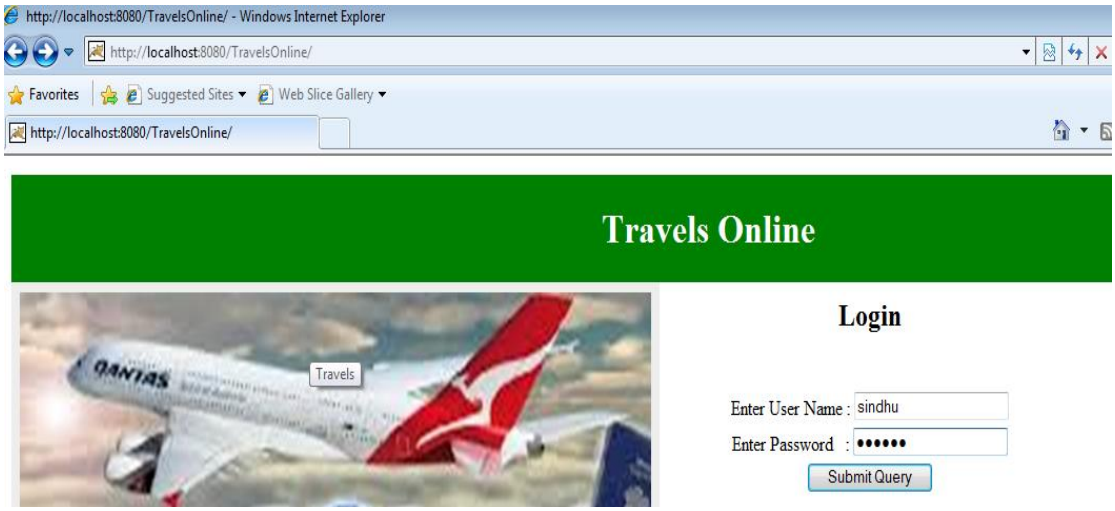

Figure 17. Enter in valid username/password.

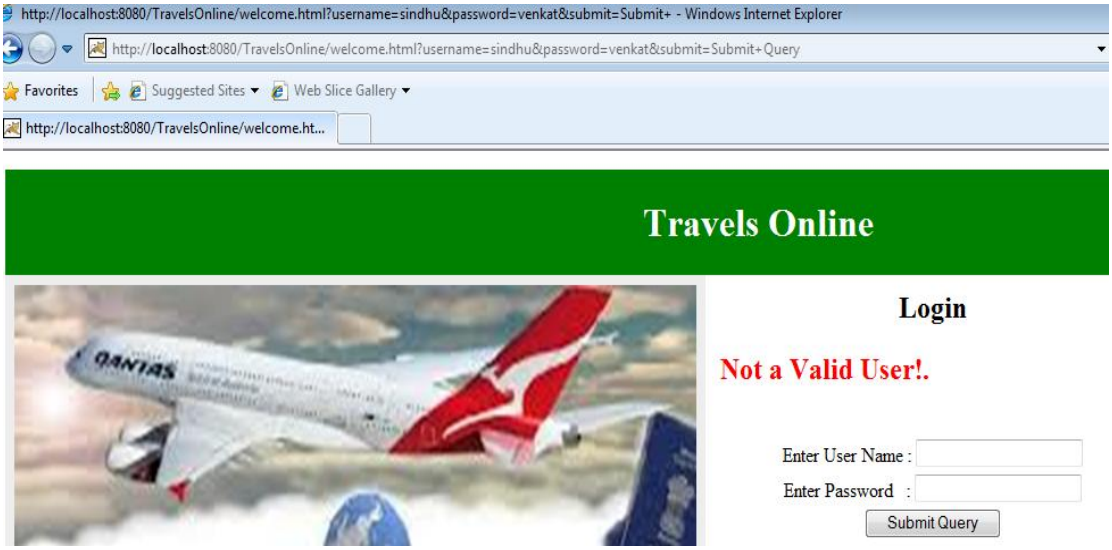

Figure 18. In valid username / password error.

7. Conclusion

This study deals with the Web services and Spring with Java Eclipse. It gives details about the Spring web component Spring MVC. This study gives details about the implementation steps of Spring using Eclipse and web services using Spring with Java Eclipse.

8. Future Enhancements

In the future environment we will concentrate in using hibernate with relational database system to store the user information and to retrieve and validate the information.

9. References

[1] http://en.wikipedia.org/wiki/Spring_Framework
[2] http://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html
[3] http://www.javatpoint.com/spring-aop-tutorial
[4] http://www.petrikainulainen.net/spring-mvc-test-tutorial/
[5] http://www.chemaxon.com/products/jchem-web-services/
[6] http://en.wikipedia.org/wiki/Web_service
[7] http://www.tutorialspoint.com/spring/spring_web_mvc_framework.htm
[8] http://www.srccodes.com/p/article/3/Tomcat-Hello-World-Servlet-using-Eclipse-IDE
[9] http://www.deepakgaikwad.net/index.php/2009/02/08/spring-mvc-tutorial-

with-eclipse-and-tomcat.html
[10] http://viralpatel.net/blogs/spring-3-mvc-create-hello-world-application-spring-3-mvc/
[11]http://help.eclipse.org/kepler/index.jsp?topic=%2Forg.eclipse.stardust.docs.wst%2Fhtml%2Fwst-integration%2Fdynamic-web-proj.html
[12] http://tomcat.apache.org/tomcat-7.0-doc/setup.html
[13] http://cooljavateacher.com/2013/04/01/setup-apache-tomcat-in-eclipse/