

## RESTful Web Services - First Application

Advertisements

⏪ Previous Page

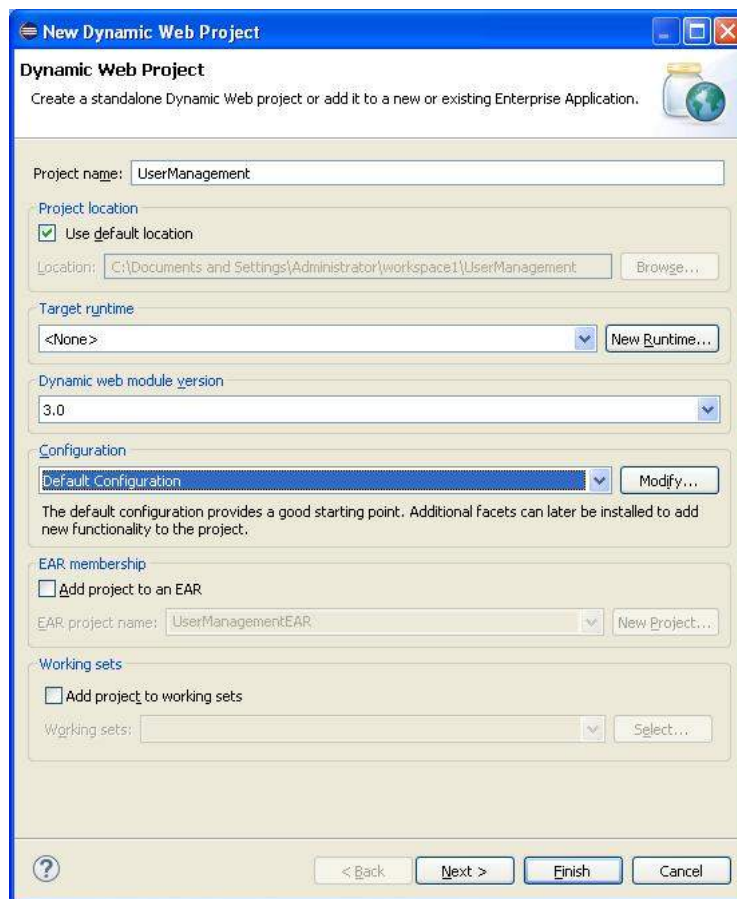
Next Page ⏩

Let us start writing the actual RESTful web services with Jersey Framework. Before you start writing your first example using the Jersey Framework, you have to make sure that you have setup your Jersey environment properly as explained in the RESTful Web Services - Environment Setup chapter. Here, I am also assuming that you have a little working knowledge of Eclipse IDE.

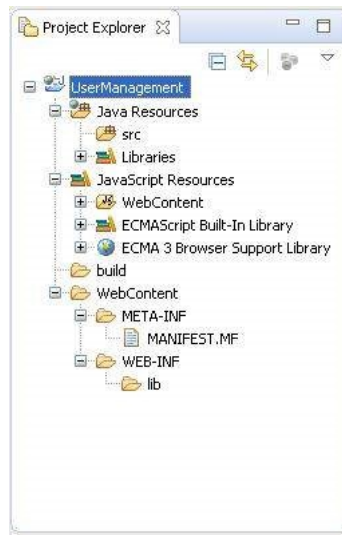
So, let us proceed to write a simple Jersey Application which will expose a web service method to display the list of users.

### Creating a Java Project

The first step is to create a Dynamic Web Project using Eclipse IDE. Follow the option **File** → **New** → **Project** and finally select the **Dynamic Web Project** wizard from the wizard list. Now name your project as **UserManagement** using the wizard window as shown in the following screenshot –



Once your project is created successfully, you will have the following content in your **Project Explorer** –



## Adding the Required Libraries

As a second step let us add Jersey Framework and its dependencies (libraries) in our project. Copy all jars from following directories of download jersey zip folder in WEB-INF/lib directory of the project.

```
\jaxrs-ri-2.17\jaxrs-ri\api
\xaxrs-ri-2.17\jaxrs-ri\ext
\xaxrs-ri-2.17\jaxrs-ri\lib
```

Now, right click on your project name **UserManagement** and then follow the option available in context menu – **Build Path** → **Configure Build Path** to display the Java Build Path window.

Now use **Add JARs** button available under **Libraries** tab to add the JARs present in WEBINF/lib directory.

## Creating the Source Files

Now let us create the actual source files under the **UserManagement** project. First we need to create a package called **com.tutorialspoint**. To do this, right click on src in package explorer section and follow the option – **New** → **Package**.

Next we will create **UserService.java**, **User.java**, **UserDao.java** files under the com.tutorialspoint package.

### User.java

```
package com.tutorialspoint;

import java.io.Serializable;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;
@XmlRootElement(name = "user")

public class User implements Serializable {
    private static final long serialVersionUID = 1L;
    private int id;
    private String name;
    private String profession;
    public User(){}

    public User(int id, String name, String profession){
        this.id = id;
        this.name = name;
        this.profession = profession;
    }
    public int getId() {
        return id;
    }
    @XmlElement
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    @XmlElement
    public void setName(String name) {
        this.name = name;
    }
    public String getProfession() {
        return profession;
    }
}
```

```

    }
    @XmlElement
    public void setProfession(String profession) {
        this.profession = profession;
    }
}

```

### UserDao.java

```

package com.tutorialspoint;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.ArrayList;
import java.util.List;

public class UserDao {
    public List<User> getAllUsers(){

        List<User> userList = null;
        try {
            File file = new File("Users.dat");
            if (!file.exists()) {
                User user = new User(1, "Mahesh", "Teacher");
                userList = new ArrayList<User>();
                userList.add(user);
                saveUserList(userList);
            }
            else{
                FileInputStream fis = new FileInputStream(file);
                ObjectInputStream ois = new ObjectInputStream(fis);
                userList = (List<User>) ois.readObject();
                ois.close();
            }
        } catch (IOException e) {
            e.printStackTrace();
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
        return userList;
    }
    private void saveUserList(List<User> userList){
        try {
            File file = new File("Users.dat");
            FileOutputStream fos;
            fos = new FileOutputStream(file);
            ObjectOutputStream oos = new ObjectOutputStream(fos);
            oos.writeObject(userList);
            oos.close();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

```

### UserService.java

```

package com.tutorialspoint;

import java.util.List;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;
@Path("/UserService")

public class UserService {
    UserDao userDao = new UserDao();
    @GET
    @Path("/users")
    @Produces(MediaType.APPLICATION_XML)
    public List<User> getUsers(){
        return userDao.getAllUsers();
    }
}

```

There are two important points to be noted about the main program,

### UserService.java

The first step is to specify a path for the web service using @Path annotation to the UserService.

The second step is to specify a path for the particular web service method using @Path annotation to method of UserService.

## Creating the Web.xml configuration File

You need to create a Web.xml Configuration file which is an XML file and is used to specify Jersey framework servlet for our application.

### web.xml

```
<?xml version = "1.0" encoding = "UTF-8"?>
<web-app xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
  xmlns = "http://java.sun.com/xml/ns/javaee"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
  http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
  id = "WebApp_ID" version = "3.0">
  <display-name>User Management</display-name>
  <servlet>
    <servlet-name>Jersey RESTful Application</servlet-name>
    <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>
    <init-param>
      <param-name>jersey.config.server.provider.packages</param-name>
      <param-value>com.tutorialspoint</param-value>
    </init-param>
  </servlet>
  <servlet-mapping>
    <servlet-name>Jersey RESTful Application</servlet-name>
    <url-pattern>/rest/*</url-pattern>
  </servlet-mapping>
</web-app>
```

### Deploying the Program

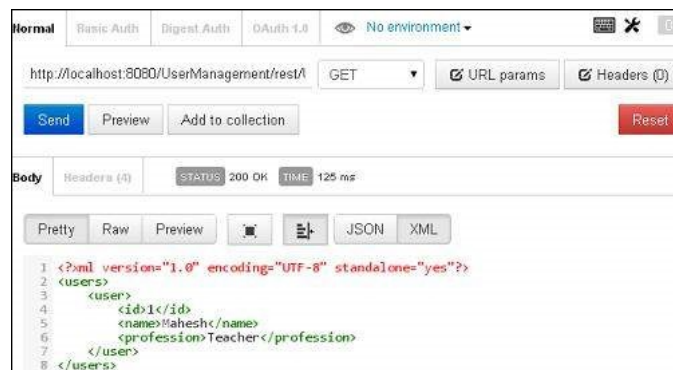
Once you are done with creating source and web configuration files, you are ready for this step which is compiling and running your program. To do this, using Eclipse, export your application as a war file and deploy the same in tomcat.

To create a WAR file using eclipse, follow the option **File** → **export** → **Web** → **War File** and finally select project UserManagement and destination folder. To deploy a war file in Tomcat, place the UserManagement.war in the **Tomcat Installation Directory** → **webapps directory** and start the Tomcat.

### Running the Program

We are using Postman , a Chrome extension, to test our webservices.

Make a request to UserManagement to get list of all the users. Put `http://localhost:8080/UserManagement/rest/UserService/users` in POSTMAN with GET request and see the following result.



Congratulations, you have created your first RESTful Application successfully.

⏪ Previous Page

Next Page ⏩

Advertisements



Write for us   FAQ's   Helping   Contact

© Copyright 2017. All Rights Reserved.

Enter email for newsletter

go