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 following two important points to note about the main program, UserService.java:

1. First step is to specify a path for the web service using @Path annotation to UserService.
2. Second step is to specify a path for the particular web service method using @Path annotation to method of UserService.

Step 4 - Create 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>

Step 5 - 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 WAR file using eclipse, follow the option **File -> export -> Web > War File** and finally select project UserManagement and destination folder. To deploy war file in Tomcat, place the UserManagement.war in **Tomcat Installation Directory** > webapps directory and start the Tomcat.

Step 6 - Running the Program

We are using [Postman](http://www.getpostman.com/), 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 below result.

