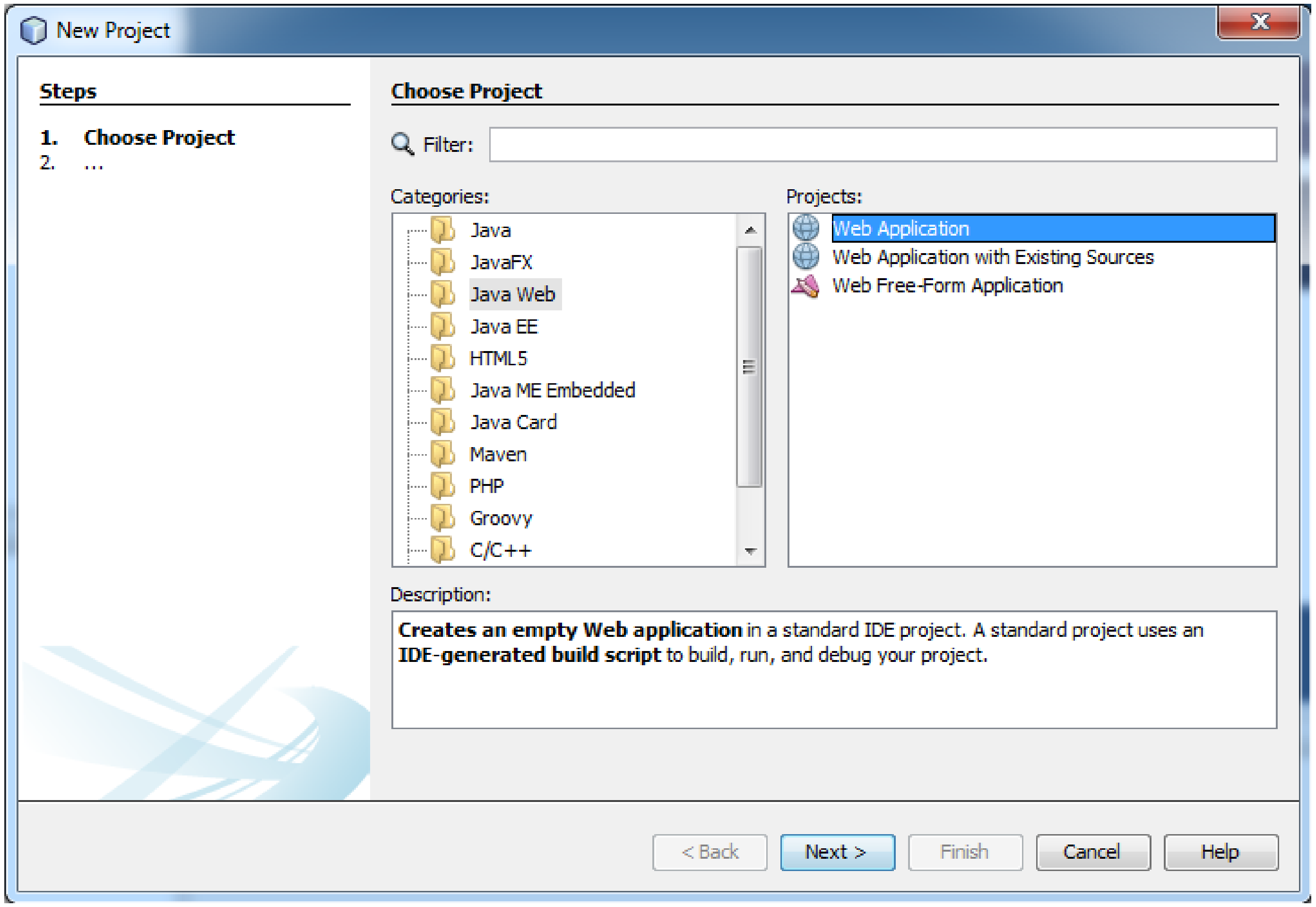
RESTful Web Service Implementation in NetBeans 8.0.2

Prof. (Dr.) Vipul K. Dabhi
Associate Professor,
Dept. of Information Technology,
D. D. University

Major Steps

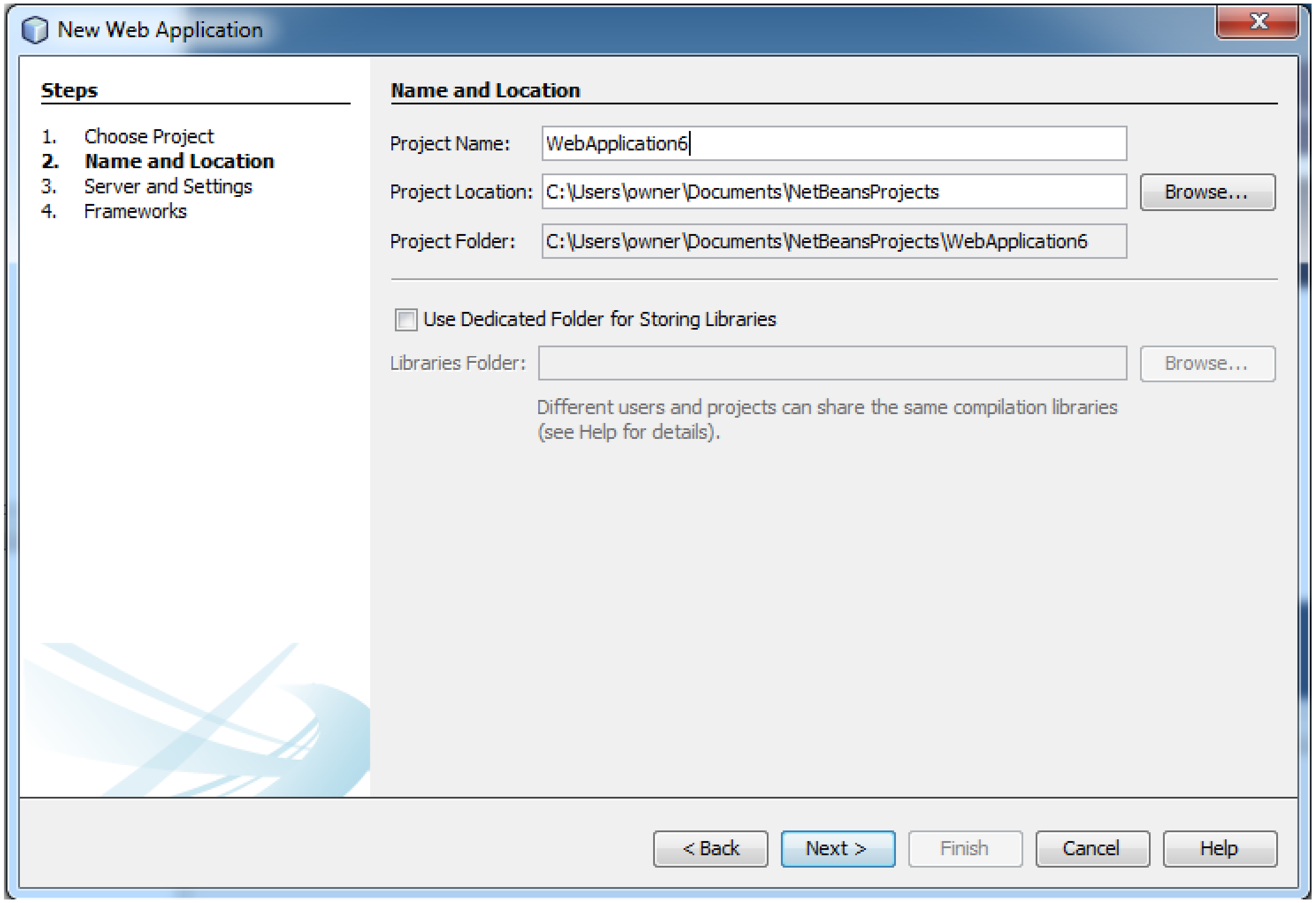
- Create NetBeans Project – Web Application
- Add Jersey Jar File in Project
- Create a Package
  - Create three Java classes under the package
- Add web.xml file in Project
- Launch POSTMAN in Google Chrome
  - Insert URL in address bar
  - Select HTTP Method (GET) to Test RESTful Web Service
  - Select HTTP Method (POST, PUT) and pass data to Test RESTful Web Service.

Create NetBeans Project – Web Application



Creates an empty Web application in a standard IDE project. A standard project uses an **IDE-generated build script** to build, run, and debug your project.

Create NetBeans Project – Web Application


New Web Application

Steps

1. Choose Project
2. **Name and Location**
3. Server and Settings
4. Frameworks

Name and Location

Project Name: WebApplication6

Project Location: C:\Users\owner\Documents\NetBeansProjects

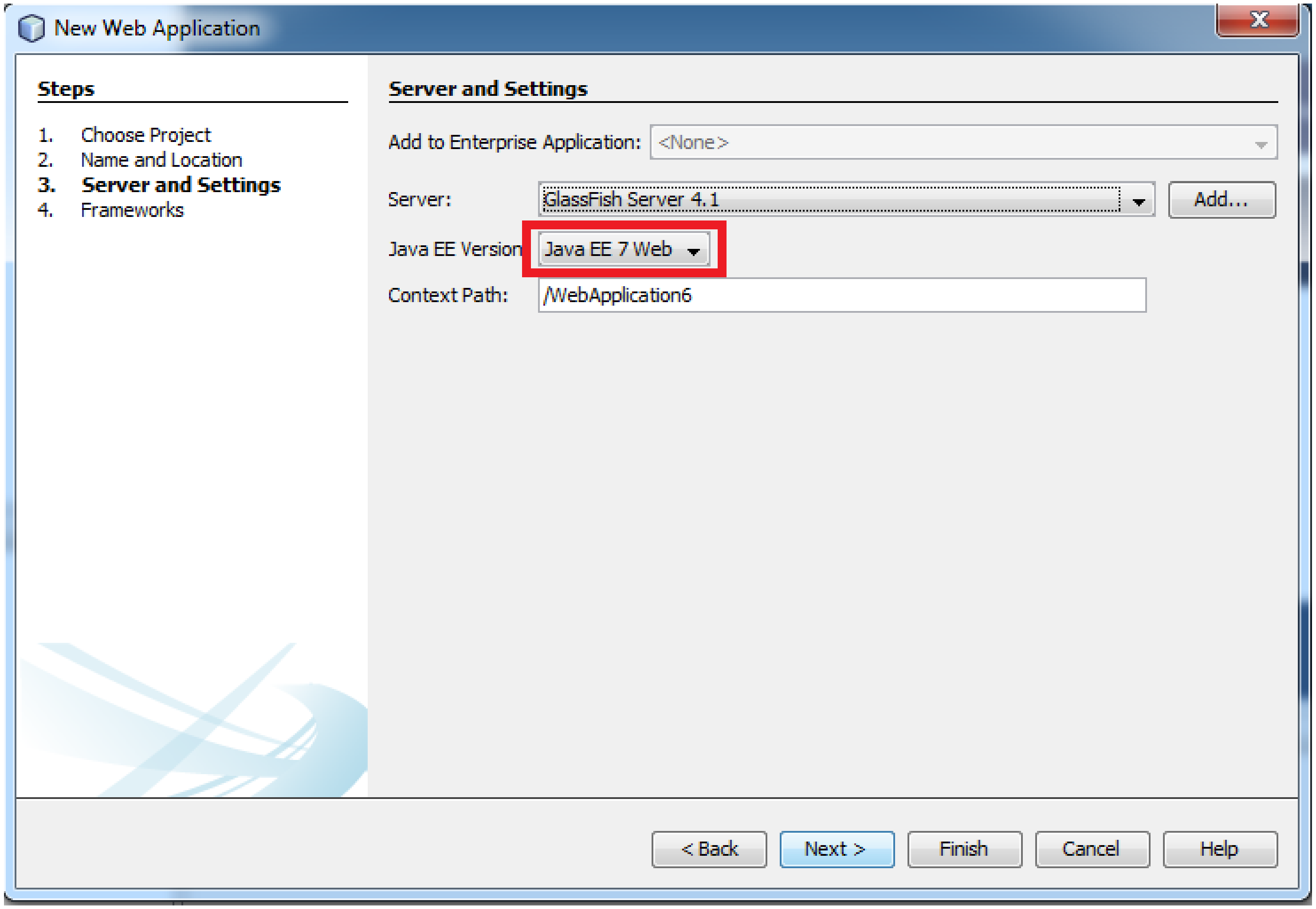
Project Folder: C:\Users\owner\Documents\NetBeansProjects\WebApplication6

☐ Use Dedicated Folder for Storing Libraries

Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).

Create NetBeans Project – Web Application



New Web Application

Steps

1. Choose Project
2. Name and Location
3. **Server and Settings**
4. Frameworks

Server and Settings

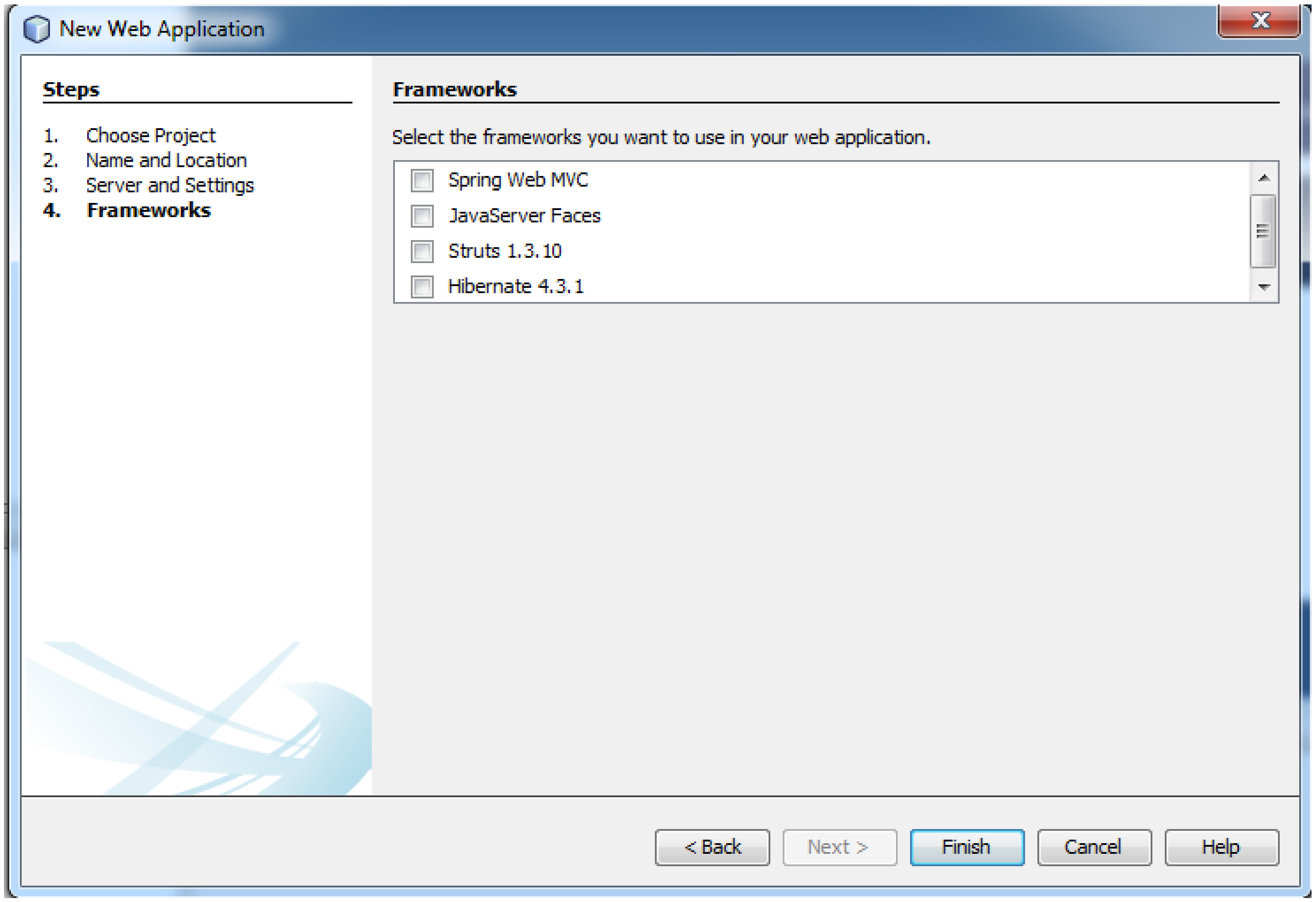
Add to Enterprise Application: <None>

Server: GlassFish Server 4.1

Java EE Version: Java EE 7 Web

Context Path: /WebApplication6

Create NetBeans Project – Web Application



New Web Application

Steps

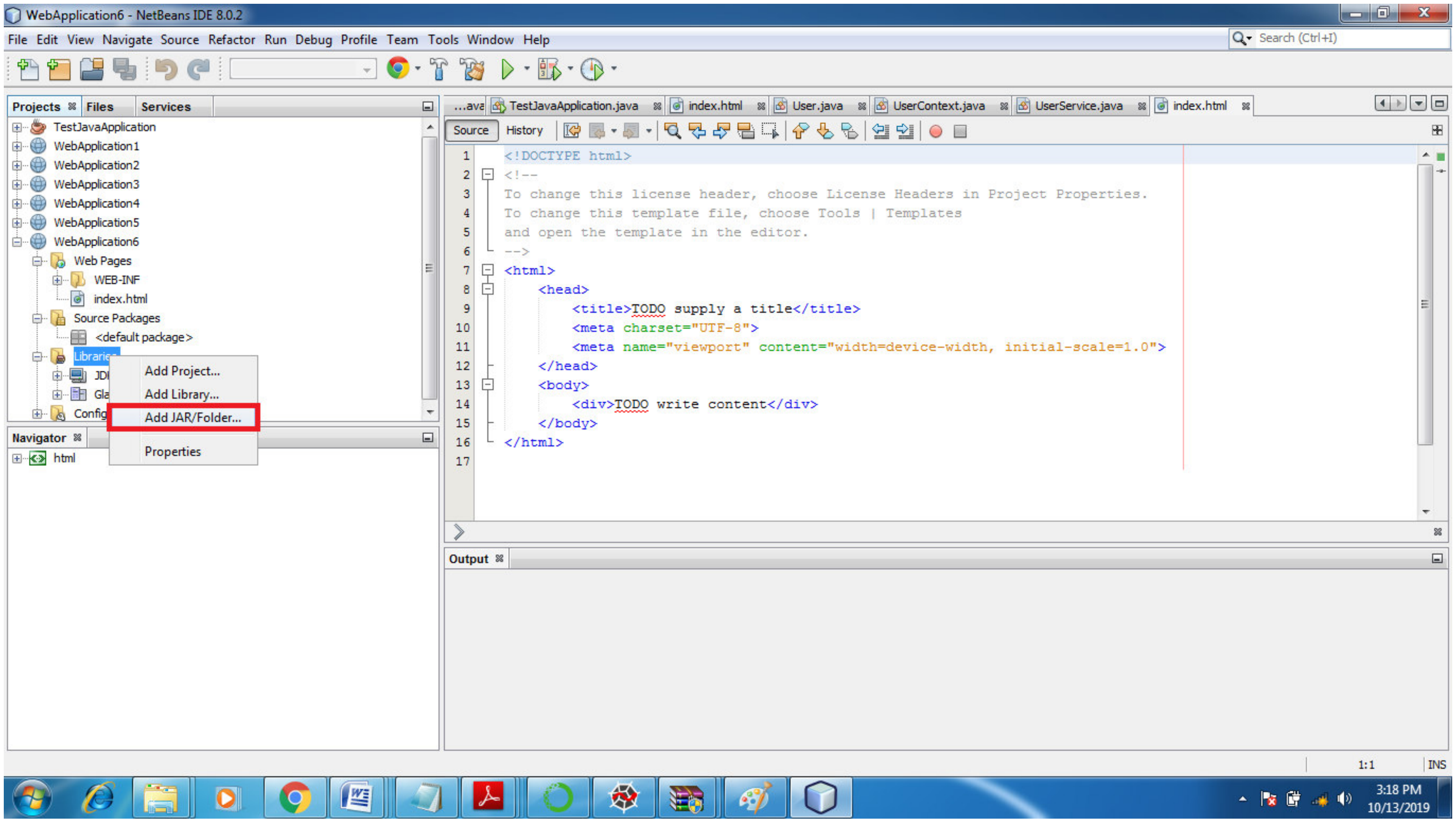
1. Choose Project
2. Name and Location
3. Server and Settings
4. **Frameworks**

Frameworks

Select the frameworks you want to use in your web application.

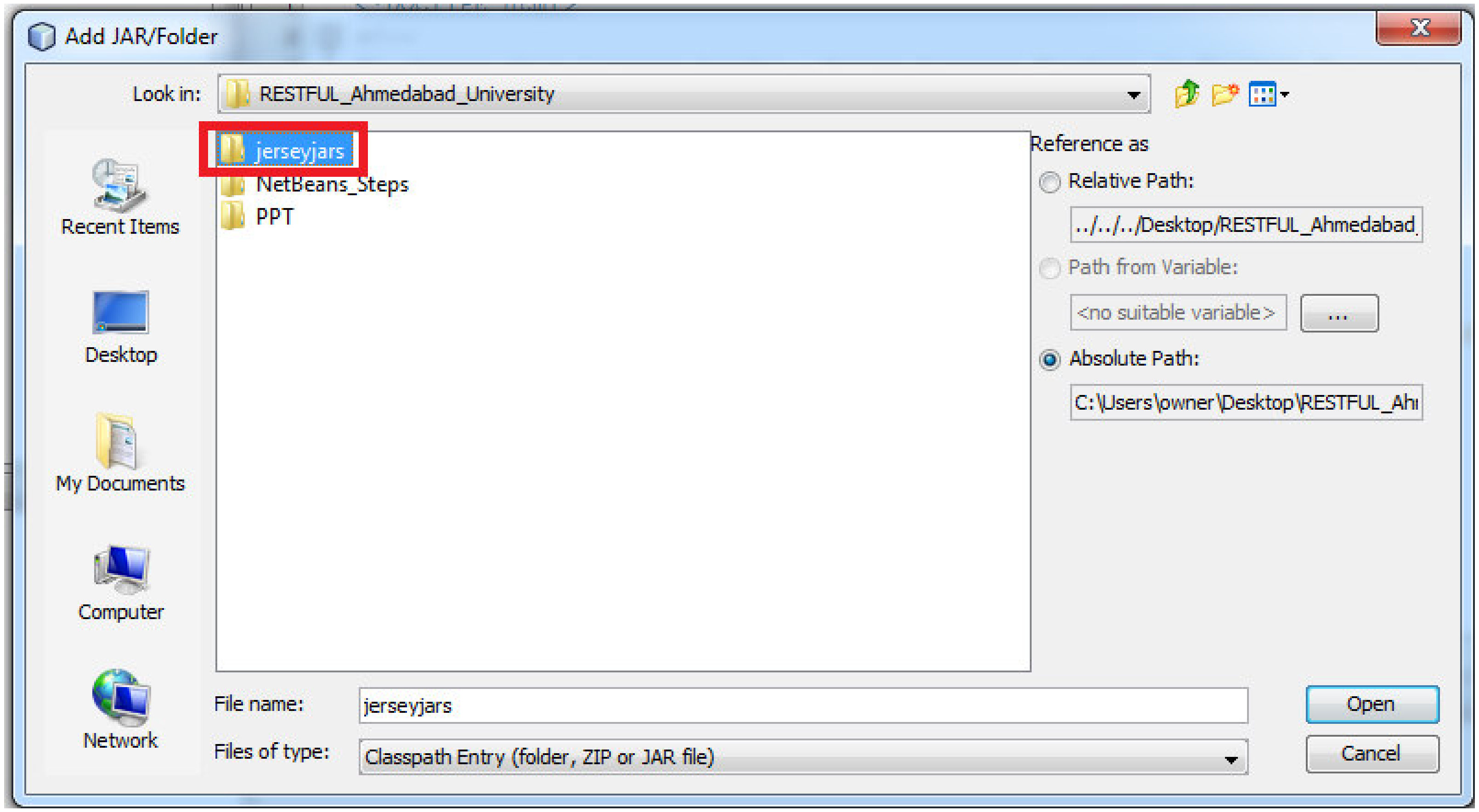
- [ ] Spring Web MVC
- [ ] JavaServer Faces
- [ ] Struts 1.3.10
- [ ] Hibernate 4.3.1

Add Jersey Jar File to Project



```html
<!DOCTYPE html>
<!--
To change this license header, choose License Headers in Project Properties.
To change this template file, choose Tools | Templates
and open the template in the editor.
-->
<html>
    <head>
        <title>TODO supply a title</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <div>TODO write content</div>
    </body>
</html>
```

Add Jersey Jar File to Project


Add JAR/Folder

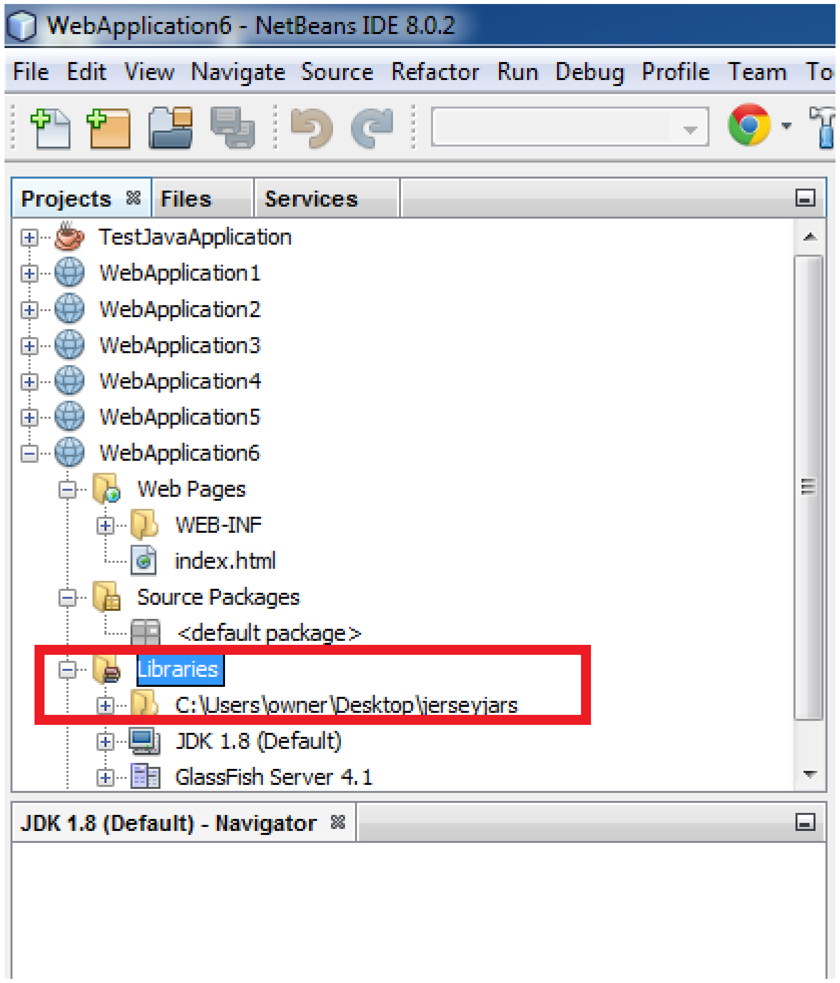
Look in: RESTFUL_Ahmedabad_University

jerseyjars
NetBeans_Steps
PPT

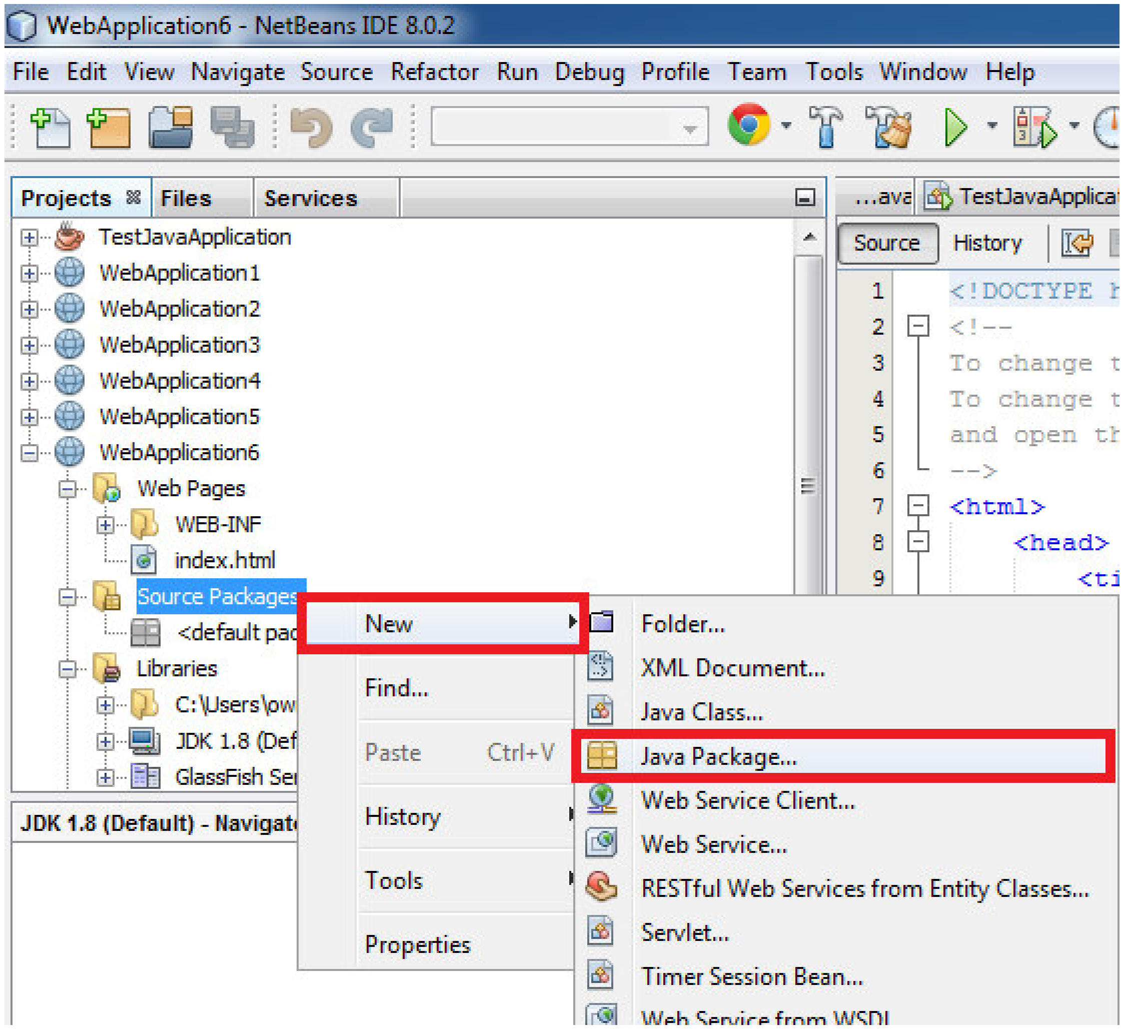
Reference as
Relative Path: ../../../Desktop/RESTFUL_Ahmedabad
Path from Variable: <no suitable variable>
Absolute Path: C:\Users\owner\Desktop\RESTFUL_Ahı

File name: jerseyjars
Files of type: Classpath Entry (folder, ZIP or JAR file)

Add Jersey Jar File to Project

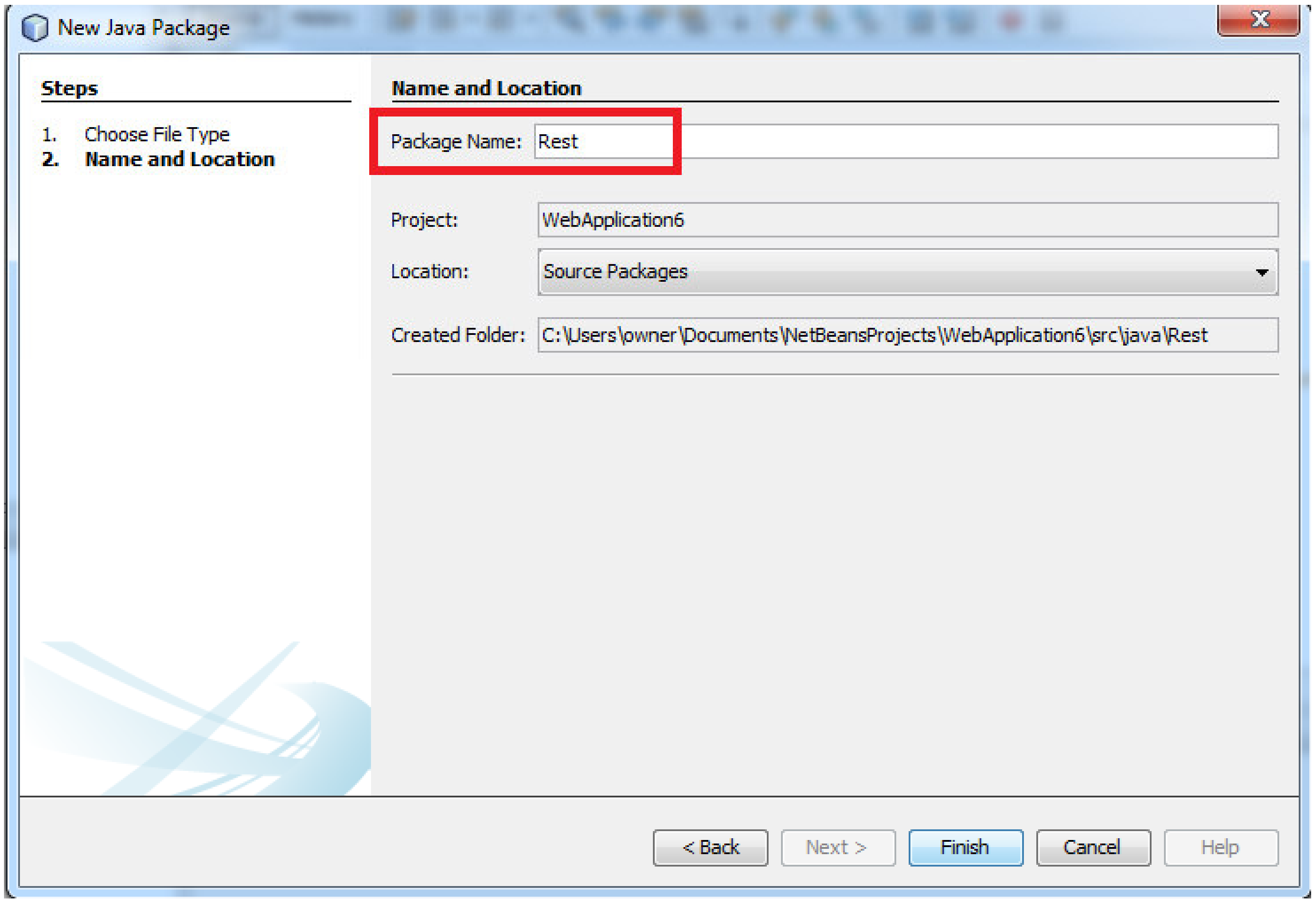


Create Java Package



```
<!DOCTYPE h
<!--
To change t
To change t
and open th
-->
<html>
    <head>
        <ti
```

Create Java Package



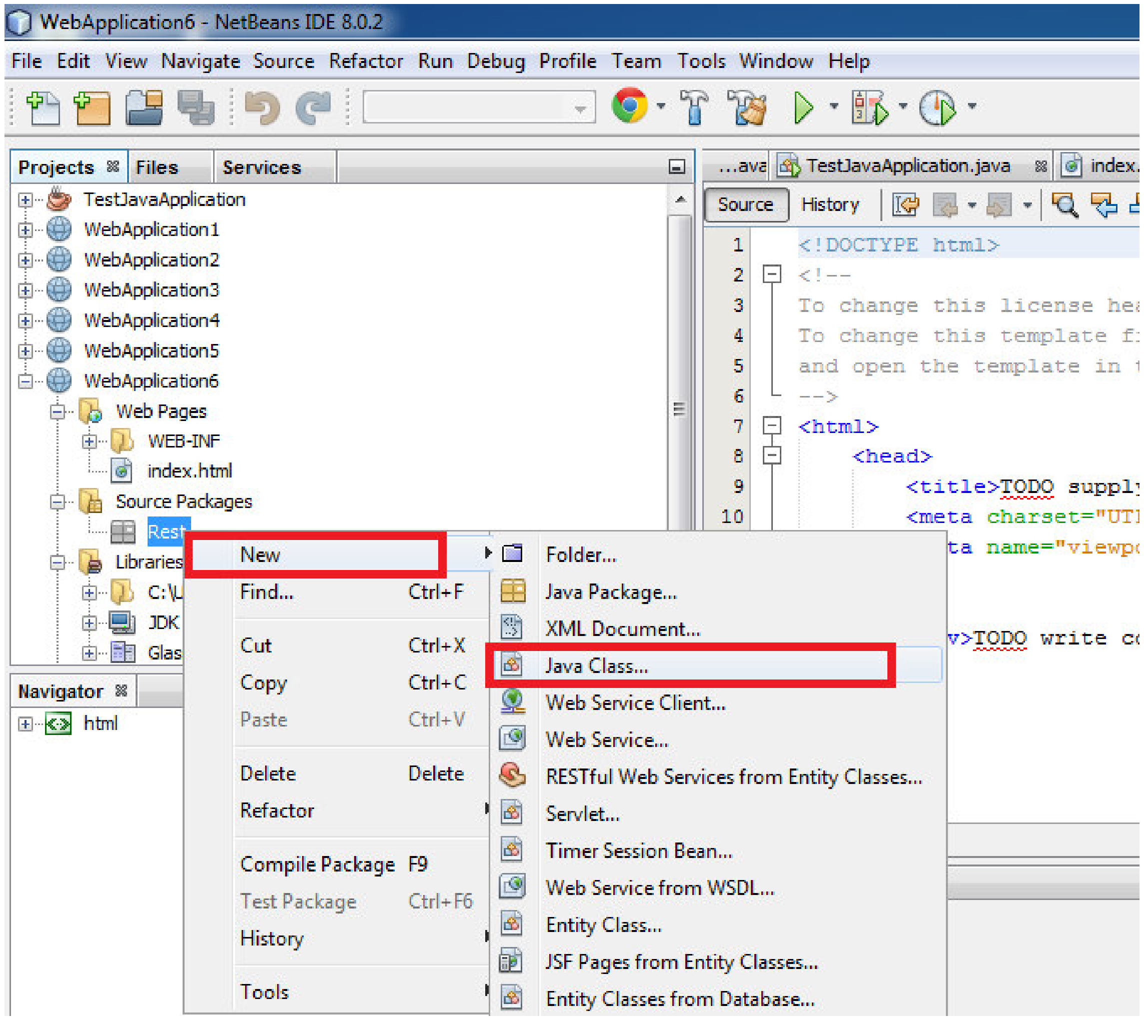
Package Name: Rest

Project: WebApplication6

Location: Source Packages

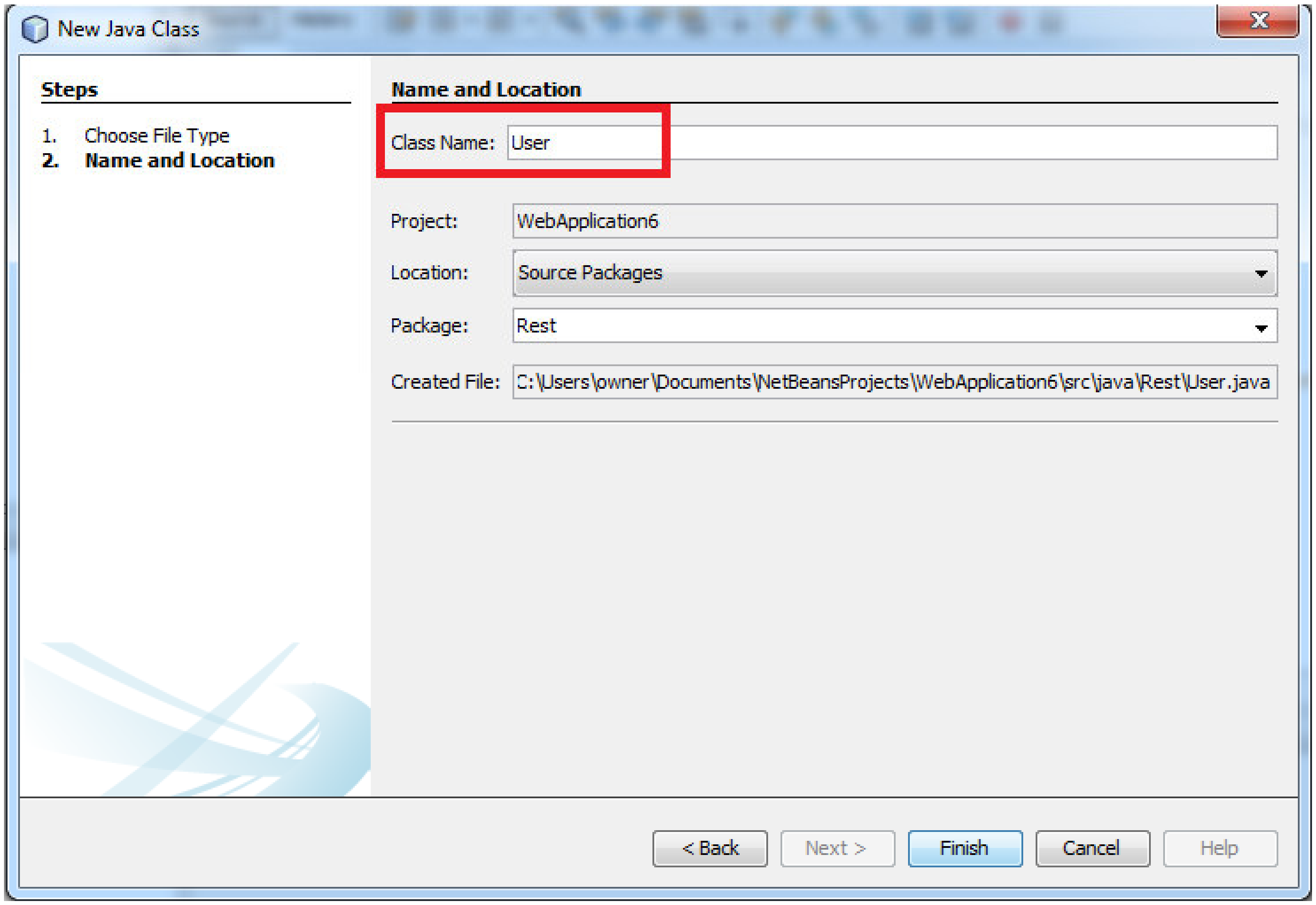
Created Folder: C:\Users\owner\Documents\NetBeansProjects\WebApplication6\src\java\Rest

Add Java Class: User.java



```
<!DOCTYPE html>
<!--
To change this license hea
To change this template f
and open the template in t
-->
<html>
    <head>
        <title>TODO suppl
        <meta charset="UT
```

Add Java Class: User.java



Name and Location

Class Name: User

Project: WebApplication6

Location: Source Packages

Package: Rest

Created File: C:\Users\owner\Documents\NetBeansProjects\WebApplication6\src\java\Rest\User.java

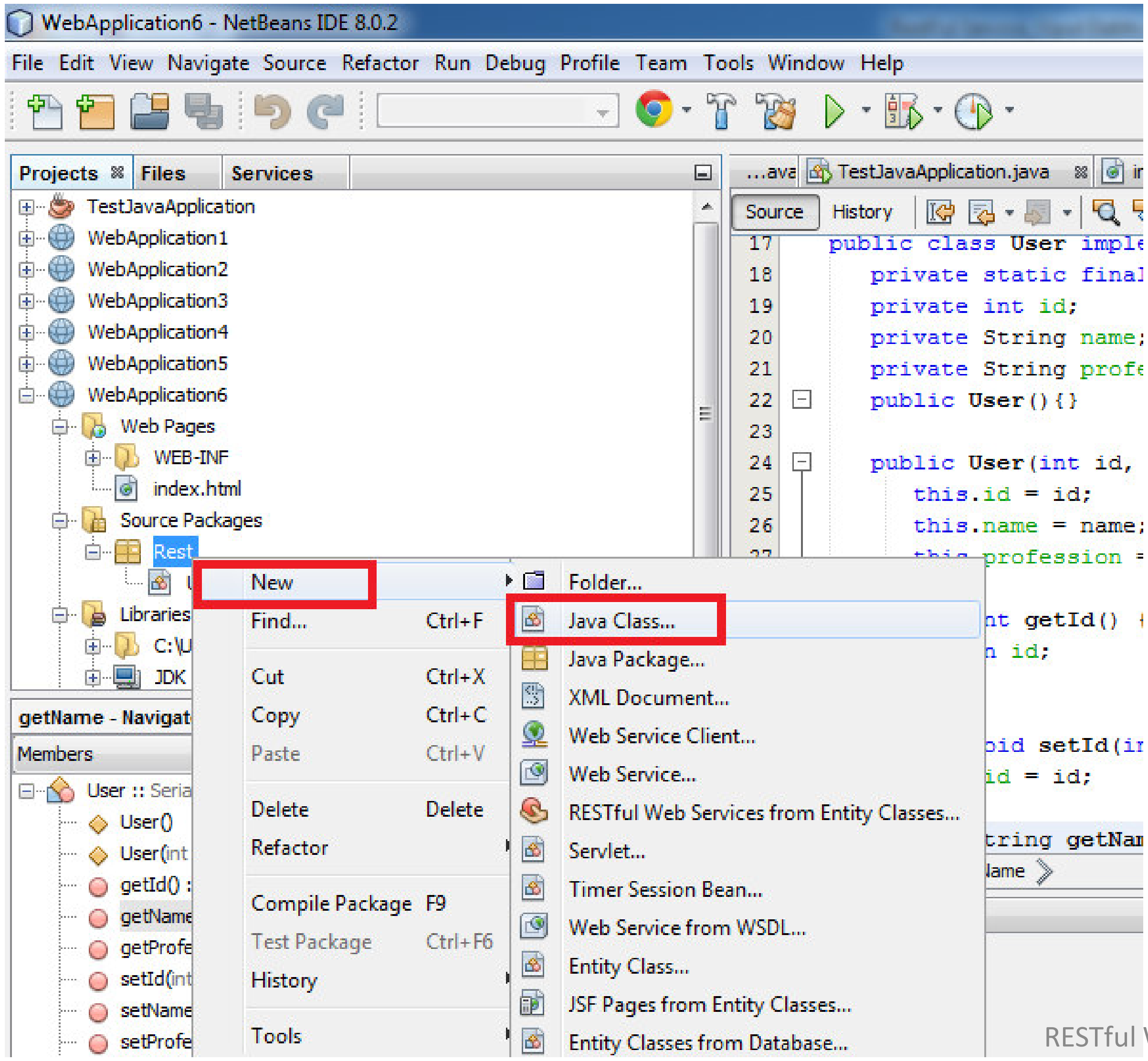
Add Java Class: User.java

```
package Rest;
 /*
(1) This is a Model Class
(2) Annotate Model class with @XmlRootElement. This is required to tell the Jersey to use JAXB for conversion of GenericType
        (List) to XML
(3) Make sure that the model class has no argument constructor. This is required for conversion of model to XML/JSON.
*/
 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;
   }
```

Add Java Class: User.java

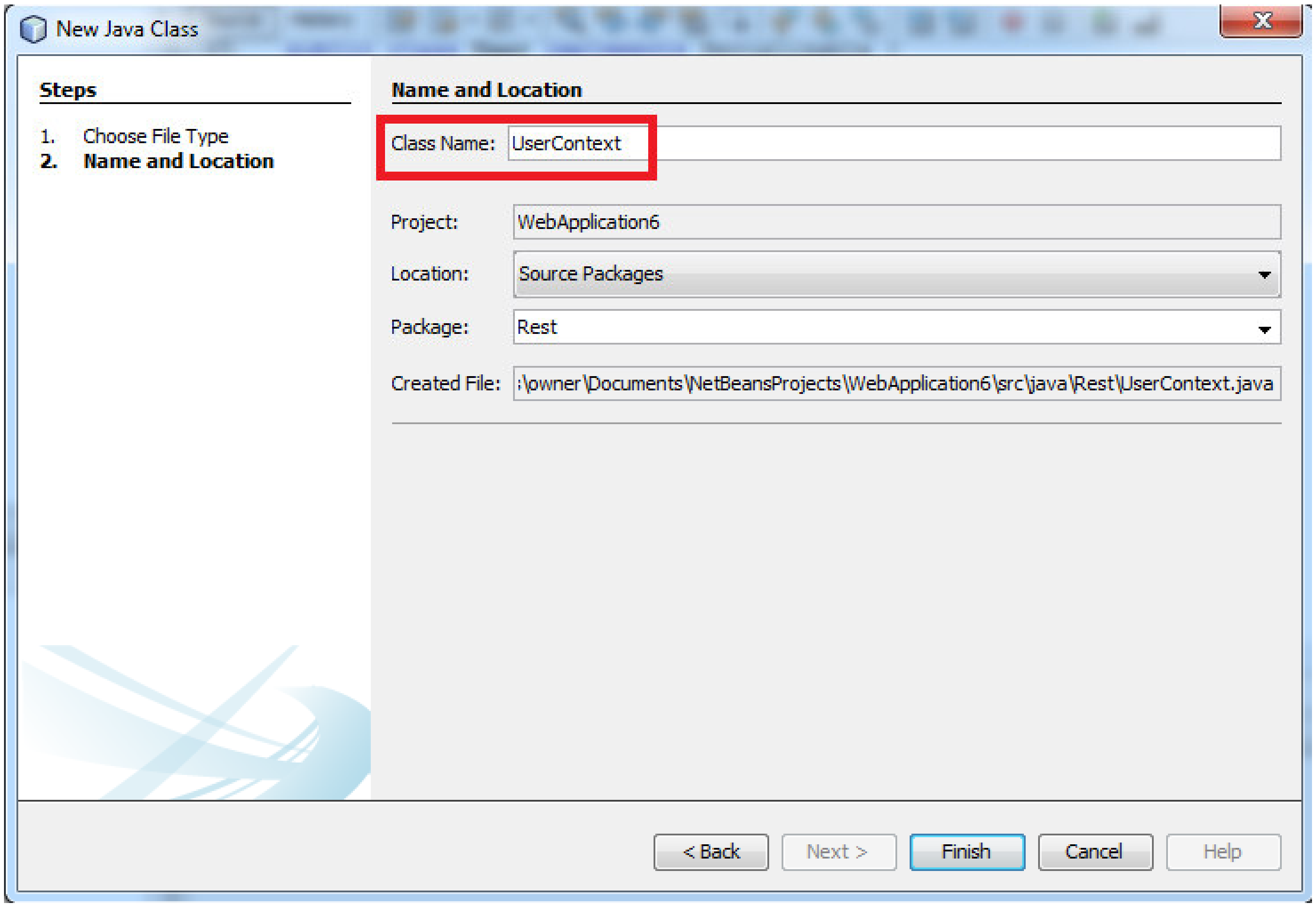
```
 public int getId() {
      return id;
   }

   public void setId (int id) {
      this.id = id;
   }
   public String getName() {
      return name;
   }

   public void setName (String name) {
      this.name = name;
   }
   public String getProfession() {
      return profession;
   }

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

Add Java Class: UserContext.java



```
public class User imple
    private static final
    private int id;
    private String name;
    private String profe
    public User(){}

    public User(int id,
        this.id = id;
        this.name = name;
        this.profession =
```

Add Java Class: UserContext.java


New Java Class

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Class Name: UserContext

Project: WebApplication6

Location: Source Packages

Package: Rest

Created File: :\owner\Documents\NetBeansProjects\WebApplication6\src\java\Rest\UserContext.java

Add Java Class: UserContext.java

```
package Rest;
/* (1) This is a Service class which connects with the Database/File and gets list of users. */
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 UserContext {
   File file = new File("Users23.dat");
  public List<User> getAllUsers(){
    List<User> userList = null;
    try {
       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) {  }
    return userList;
  }
```

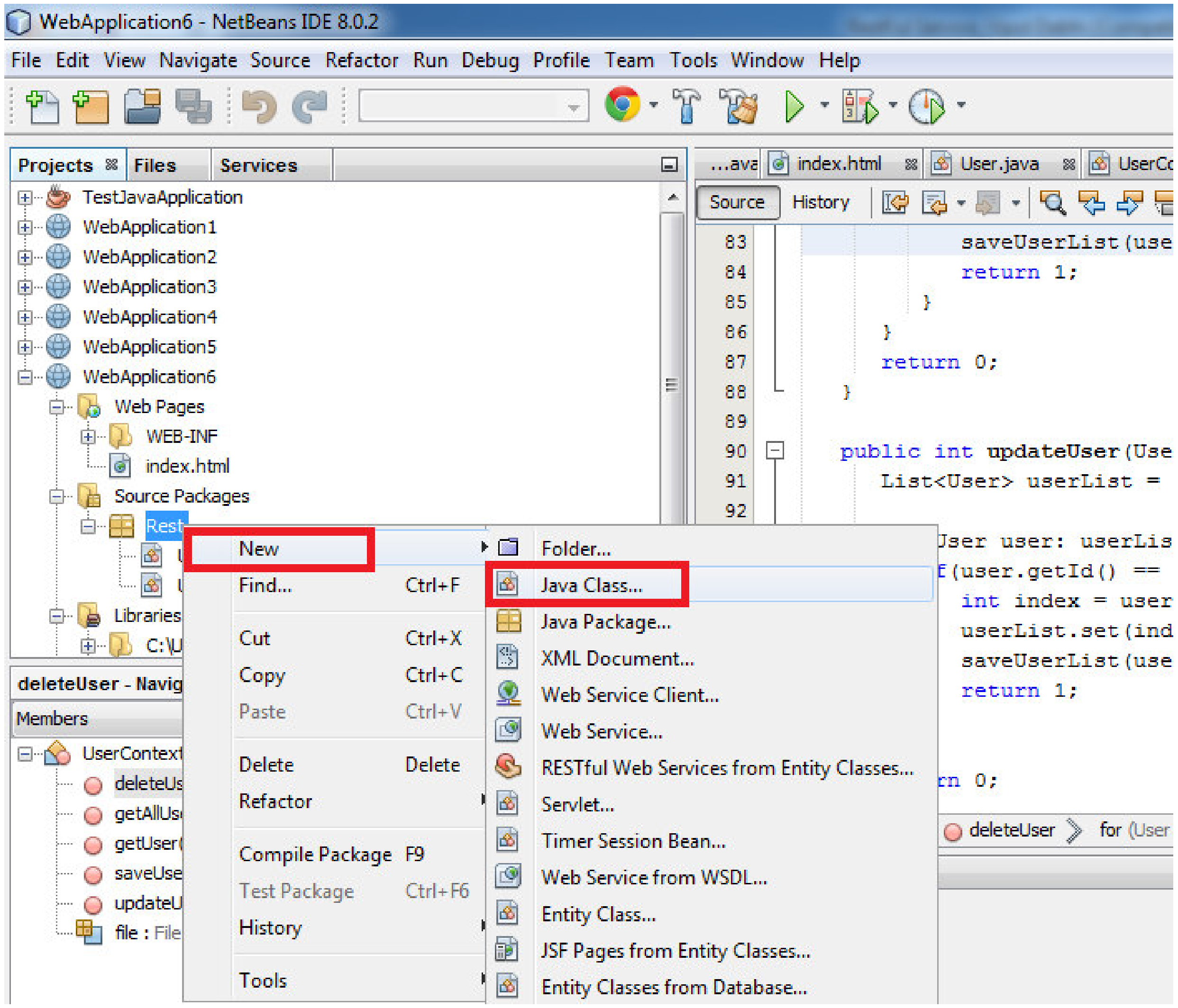
Add Java Class: UserContext.java

```
public void saveUserList(List<User> userList1){
      try {
         FileOutputStream fos;
         fos = new FileOutputStream(file);
         ObjectOutputStream oos = new ObjectOutputStream(fos);
         oos.writeObject(userList1);
         oos.close();
      } catch (FileNotFoundException e) {
         e.printStackTrace();
      } catch (IOException e) {
         e.printStackTrace();
      }
   }

   public User getUser(int id){
      List<User> users = getAllUsers();
       for(User user: users){
         if(user.getId() == id){
             System.out.println("getID is"+user.getId()+"AND ID is"+id);
             return user;
         }
      }
      return null;
   }
```

Add Java Class: UserContext.java

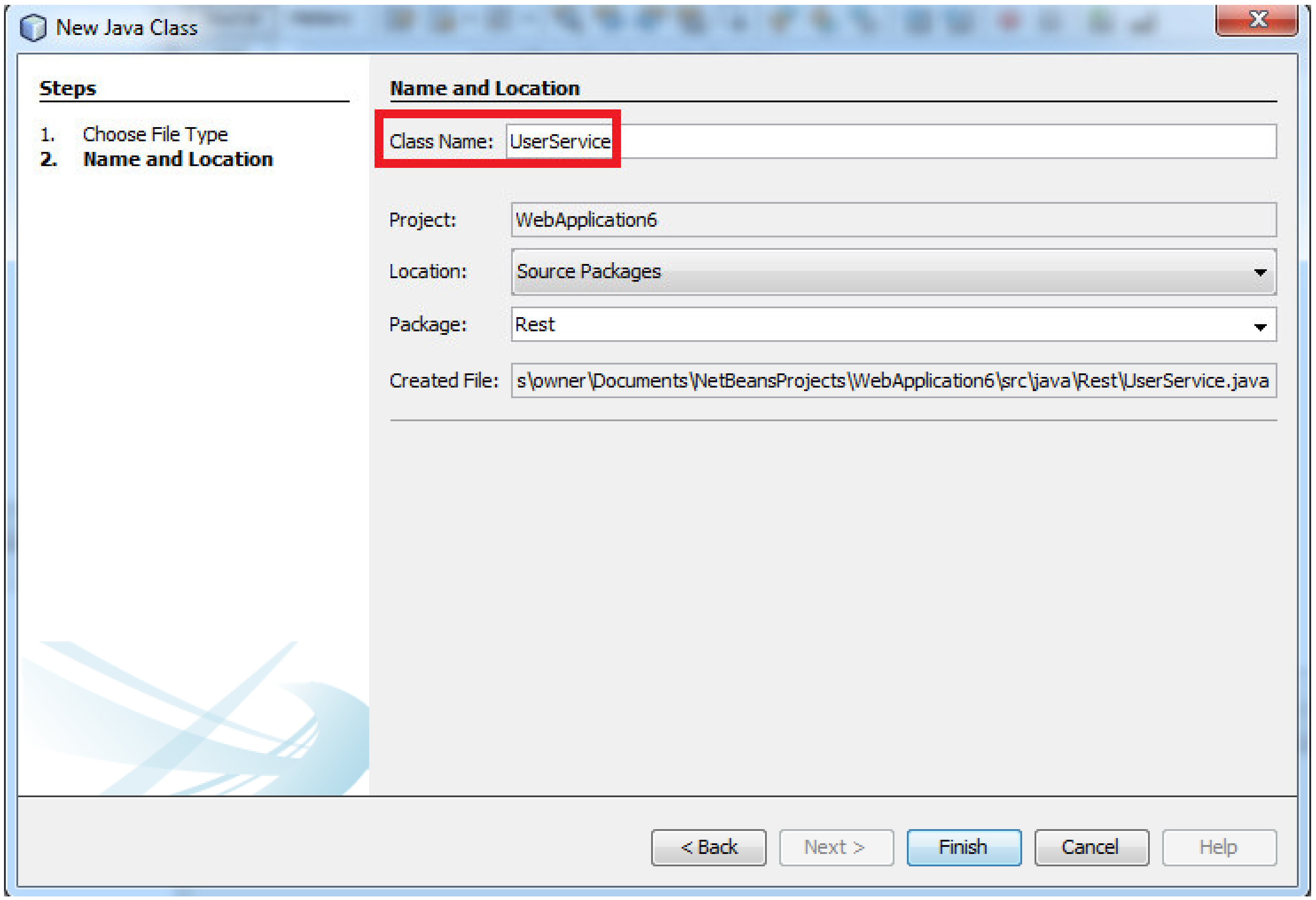
```
public int deleteUser(int id){
      System.out.println("Inside Delete User Method of User Context");
      List<User> userList = getAllUsers();
      for(User user: userList){
        if(user.getId() == id){
           System.out.println("Inside Delete User getID is"+user.getId()+"AND ID is"+id);
           int index = userList.indexOf(user);
           userList.remove(index);
           saveUserList(userList);
           return 1;
        }
     }
     return 0;
  }
public int updateUser(User pUser){
     List<User> userList = getAllUsers();
      for(User user: userList){
        if(user.getId() == pUser.getId()){
           int index = userList.indexOf(user);
           userList.set(index, pUser);
           saveUserList(userList);
           return 1;
        }
     }
     return 0;
  }
}
```

Add Java Class: UserService.java



```
            saveUserList(use
            return 1;
        }
    }
    return 0;
}

public int updateUser(Use
    List<User> userList =
```

Add Java Class: UserService.java



Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Class Name: UserService

Project: WebApplication6

Location: Source Packages

Package: Rest

Created File: s\owner\Documents\NetBeansProjects\WebApplication6\src\java\Rest\UserService.java

Add Java Class: UserService.java

```
/*
(1) This is a Resource Class - Resource Handler
 (2) Jersey servlet looks for resource class to handle request submitted to Resource URL - http://localhost:8080/WebApplication2/test/UserService
 (3) Add methods (GET,POST etc..) to this class that returns response
@Path("/UserService") annotation tells Jersey to execute different (GET, POST) methods of this class when the URL is called -
        http://localhost:8080/WebApplication2/test/UserService
 (4) To tell the Jersey that there is a Resource class available in this package, configure init-param of Jersey Servlet in web.xml.
<param-name>jersey.config.server.provider.packages</param-name>
<param-value>Rest</param-value> : The name of package where you want Jersey to lookup for Resource class. Just lookup the classes available under this package
when somebody makes request to http://localhost:8080/WebApplication2/test/UserService. There may be one class which can handle this request.
 There may be multiple classes under the same package. Jersey looks this class as possible contender to handle request.
We want to convey Jersey that this class which handles /UserService URL. Therefore we use Path annotation @Path("/UserService").
 (5) We have mapped the URL: http://localhost:8080/WebApplication2/test/UserService. But there could be different HTTP requests (GET, POST etc..)
which can be sent to this URL. Moreover, there can be multiple methods available in the class. Jersey provides a way to map HTTP Method to a Java Method of class.
Annotate method with the right HTTP method annotation (@GET, @POST etc..)
 (6)How Jersey sends the response of the method. What is the format of the response it has to send?
Annotate method with @Produces specifying response format. MediaType is an enumeration that contains lots of the
standard content type (TEXT_PLAIN, APPLICATION_JSON, APPLICATION_XML).
 (7) The annotation @Path("/UserService") is applicable to whole class. We want method level annotations for URL
        http://localhost:8080/WebApplication2/test/UserService/users/1
Jersey allows Path annotations for methods also.
Subsequent Path: @Path("/users/{userid}") adds on to @Path("/UserService").
{userid} is a variable piece of URL and not hardcoded.  The way to tell Jersey that a portion of URL is variable, is to use {} brackets.
 Jersey has a feature to extract the value of variable portion of URL. You have to use annotation @PathParam.
public User getUser(@PathParam("userid") int userid)
 (8) When you make POST request from POSTMAN always remember to set the Content-Type value to application/json in the Header of request.
Provide Raw data.
Accept the Model (User) type as argument to bind to the request body. Jersey converts the request body to Model instance.
Use @Consumes to specify the expected request body format.
*/
```

Add Java Class: UserService.java

```
package Rest;

import java.util.Iterator;
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;
import javax.ws.rs.*;

 @Path("/UserService")
public class UserService {
   UserContext userDao = new UserContext();
   private static final String SUCCESS_RESULT="<result>success</result>";
   private static final String FAILURE_RESULT="<result>failure</result>";
   @GET
   @Path("/users")
   @Produces(MediaType.APPLICATION_JSON) // Because we want to return list of users in JSON format
   public List<User> getUsers(){
      return userDao.getAllUsers();
   }

 @GET
   @Path("/users/{userid}")
   @Produces(MediaType.APPLICATION_JSON)
   public User getUser(@PathParam("userid") int userid){
      System.out.println("The ID received in GET is"+userid);
      return userDao.getUser(userid);
   }
```

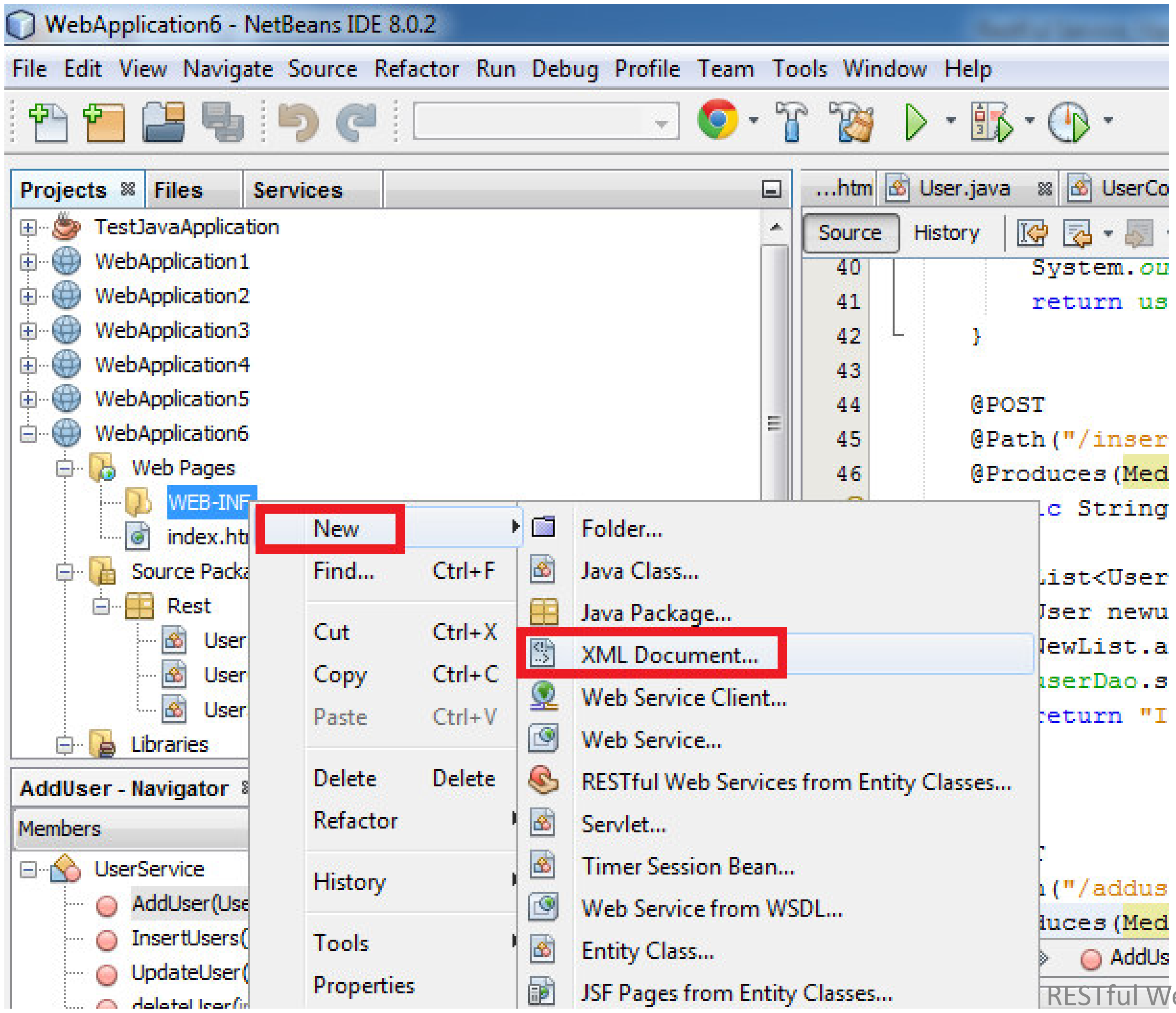
Add Java Class: UserService.java

```
@POST
 @Path("/insertuser")
 @Produces(MediaType.TEXT_PLAIN)
 public String InsertUsers()
 {
    List<User> NewList = userDao.getAllUsers();
    User newuser = new User(3,"ABC","XYZ");
    NewList.add(newuser);
    userDao.saveUserList(NewList);
    return "Inserted";
 }

@POST
 @Path("/adduser")
 @Produces(MediaType.APPLICATION_JSON)
 @Consumes(MediaType.APPLICATION_JSON)
 public User AddUser(User user)
 {
    System.out.println("Inside Add User Method");
    List<User> NewList = userDao.getAllUsers();
    NewList.add(user);
    userDao.saveUserList(NewList);
    return user;
 }
```

Add Java Class: UserService.java

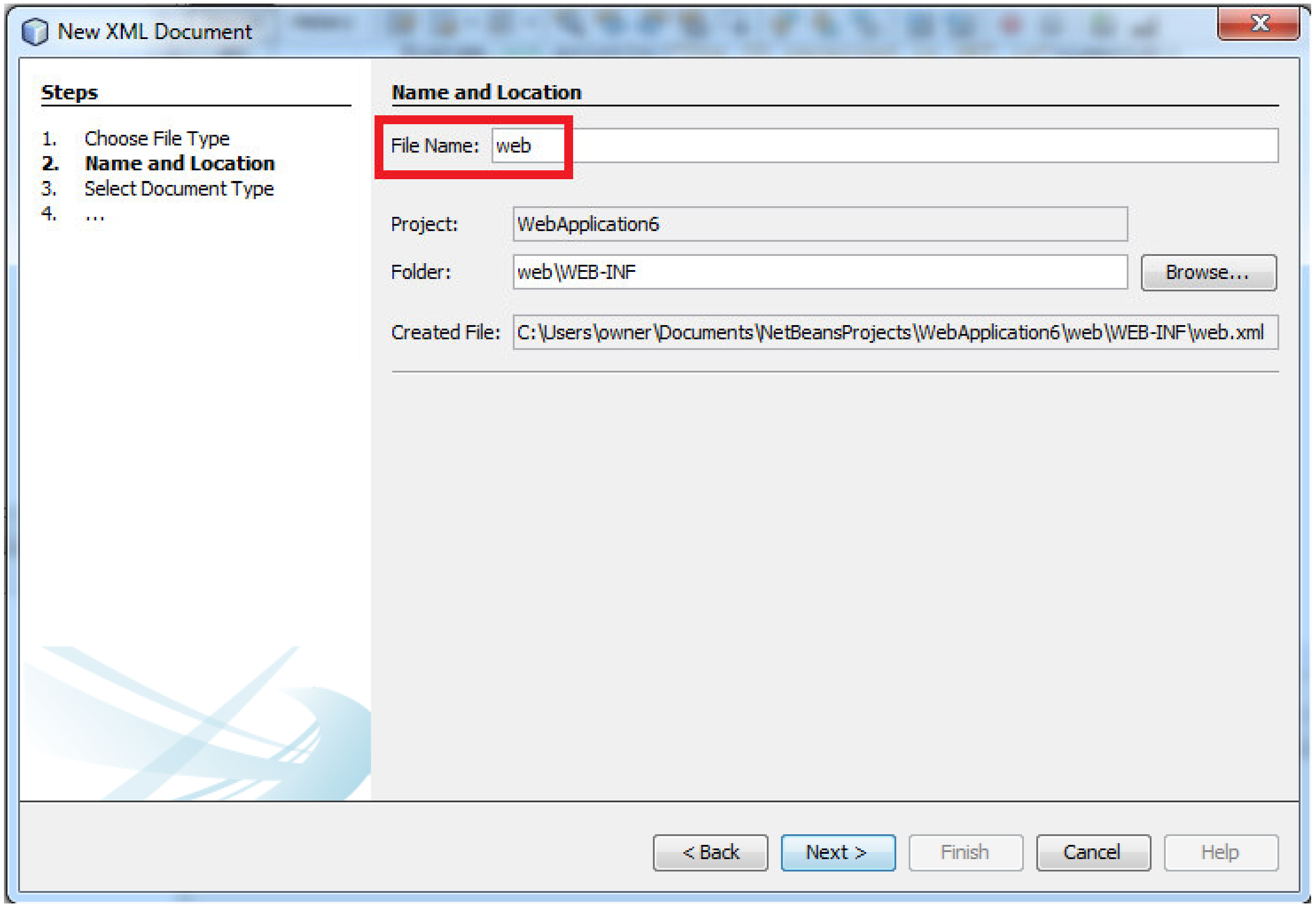
```
@PUT
 @Path("/updateuser")
 @Produces(MediaType.APPLICATION_JSON)
 @Consumes(MediaType.APPLICATION_JSON)
 public User UpdateUser(User user) {
 System.out.println("The ID received in GET is"+user.getId());
 int result = userDao.updateUser(user);
 if(result==1)
      System.out.println("Success in Update");
 else
      System.out.println("Failure in Update");
 return user;
 }

 @DELETE
 @Path("/deleteuser/{userid}")
 @Produces(MediaType.TEXT_PLAIN)
 @Consumes(MediaType.APPLICATION_JSON)
 public String deleteUser(@PathParam("userid") int userid)
 {
   System.out.println("The ID received in DELETE is"+userid);
   int result = userDao.deleteUser(userid);
   System.out.println("Value of Result is"+result);
   if(result == 1){
     return "SUCCESS";
   }
   return "FAILURE";
 }
}
```

Add web.xml file to Project



```
System.ou
    return us
}

@POST
@Path("/inser
@Produces(Med
```

Add web.xml file to Project



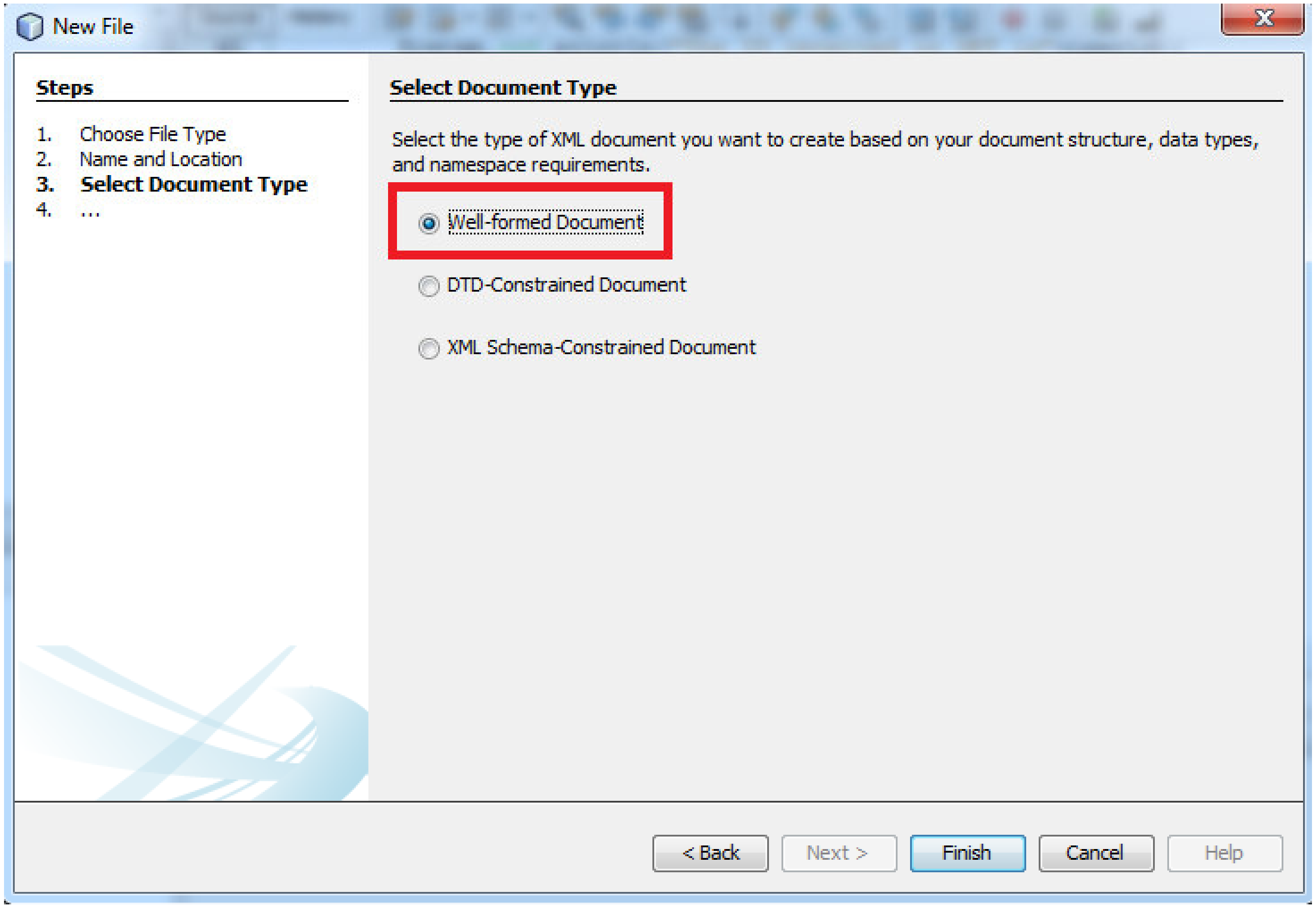
File Name: web

Project: WebApplication6

Folder: web\WEB-INF

Created File: C:\Users\owner\Documents\NetBeansProjects\WebApplication6\web\WEB-INF\web.xml

Add web.xml file to Project



New File

Steps

1. Choose File Type
2. Name and Location
3. **Select Document Type**
4. ...

Select Document Type

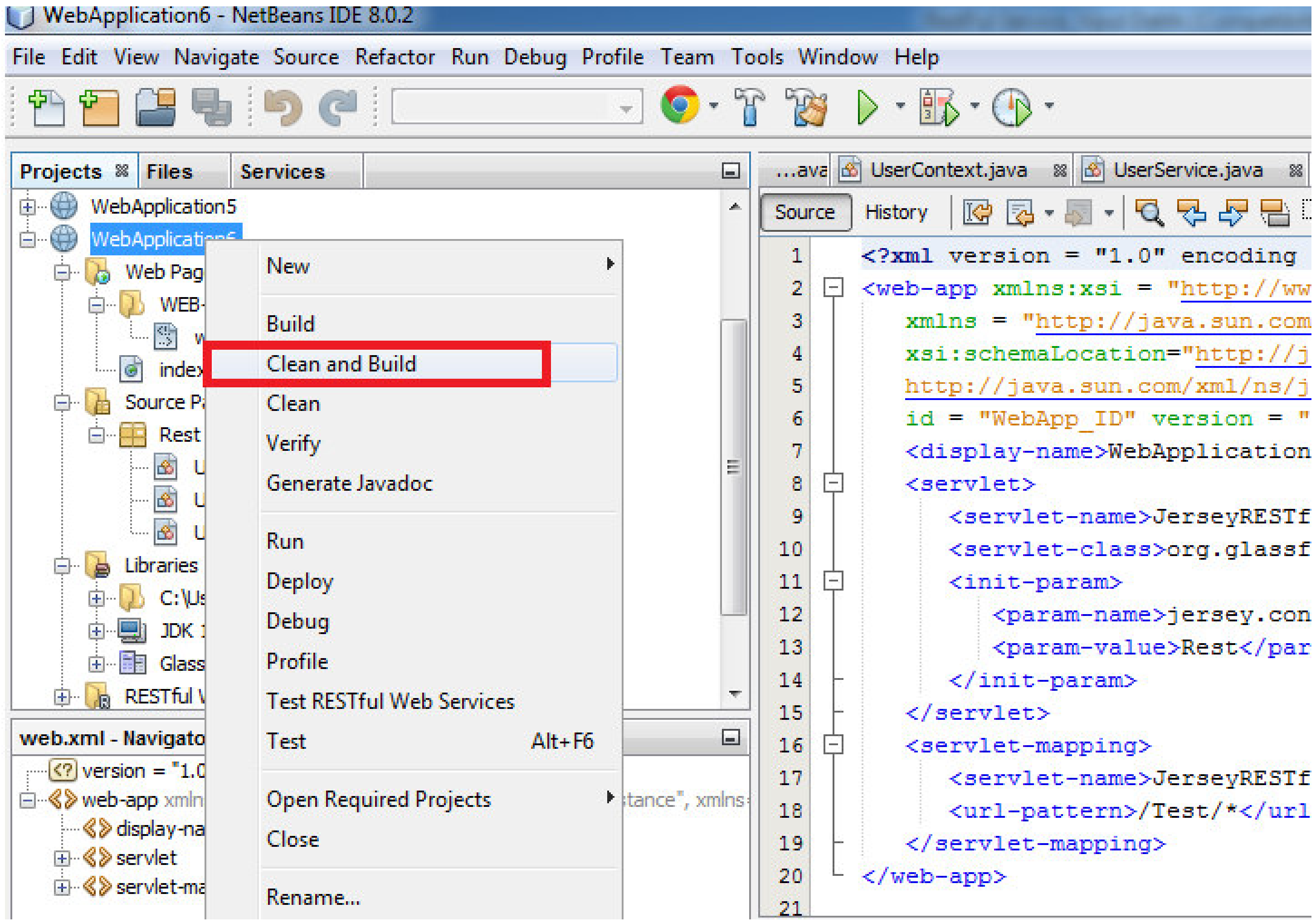
Select the type of XML document you want to create based on your document structure, data types, and namespace requirements.

- [✓ Well-formed Document]
- DTD-Constrained Document
- XML Schema-Constrained Document

Add web.xml file to Project

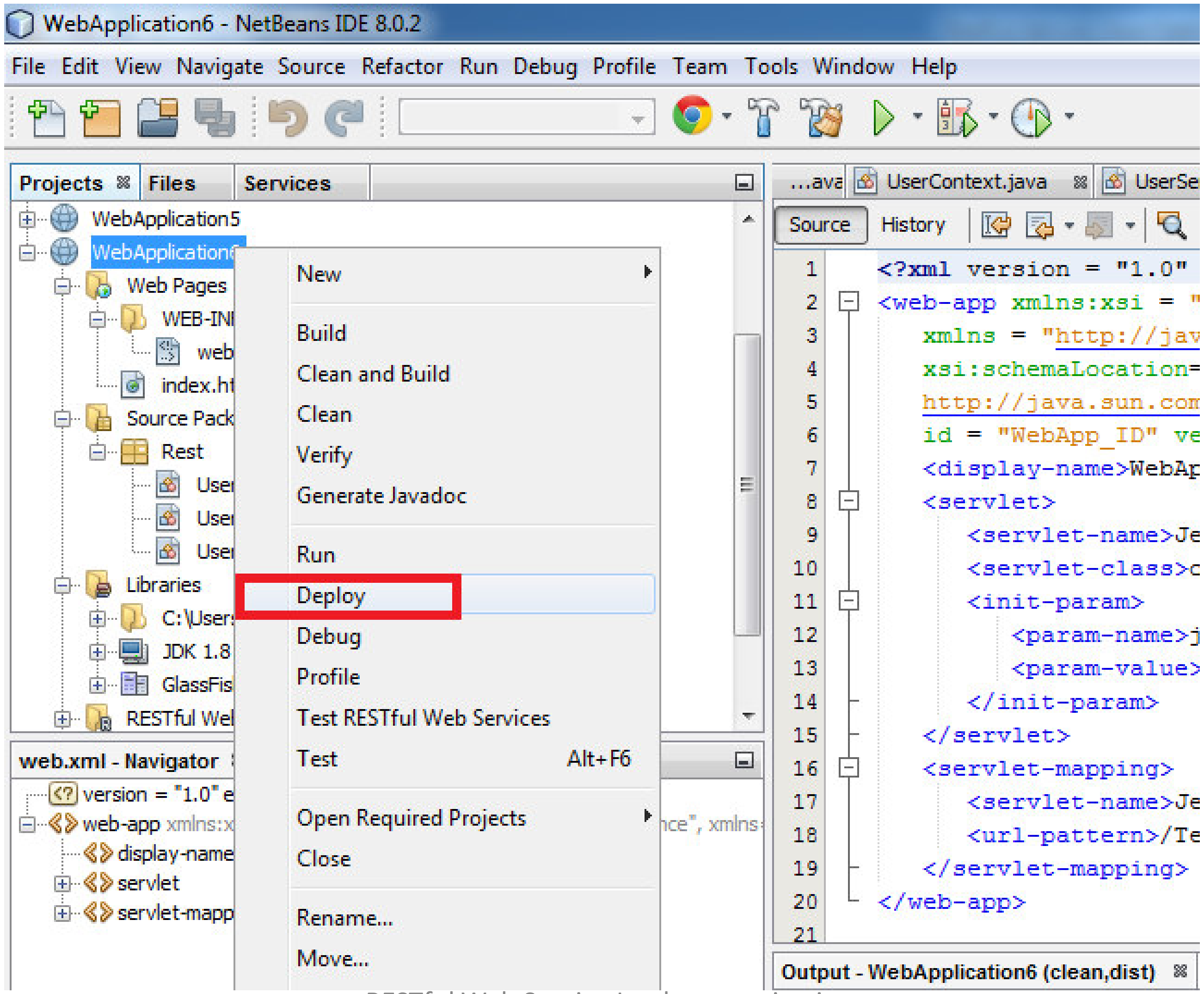
```
<?xml version = "1.0" encoding = "UTF-8"?>
<web-app xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
  xmlns = "http://java.sun.com/xml/ns/javaee"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
  http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
  id = "WebApp_ID" version = "3.0">
  <display-name>WebApplication6</display-name>
  <servlet>
    <servlet-name>JerseyRESTfulApplication</servlet-name>
    <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>
    <init-param>
      <param-name>jersey.config.server.provider.packages</param-name>
      <param-value>Rest</param-value>
    </init-param>
  </servlet>
  <servlet-mapping>
    <servlet-name>JerseyRESTfulApplication</servlet-name>
    <url-pattern>/Test/*</url-pattern>
  </servlet-mapping>
</web-app>
```

Clean and Build Project



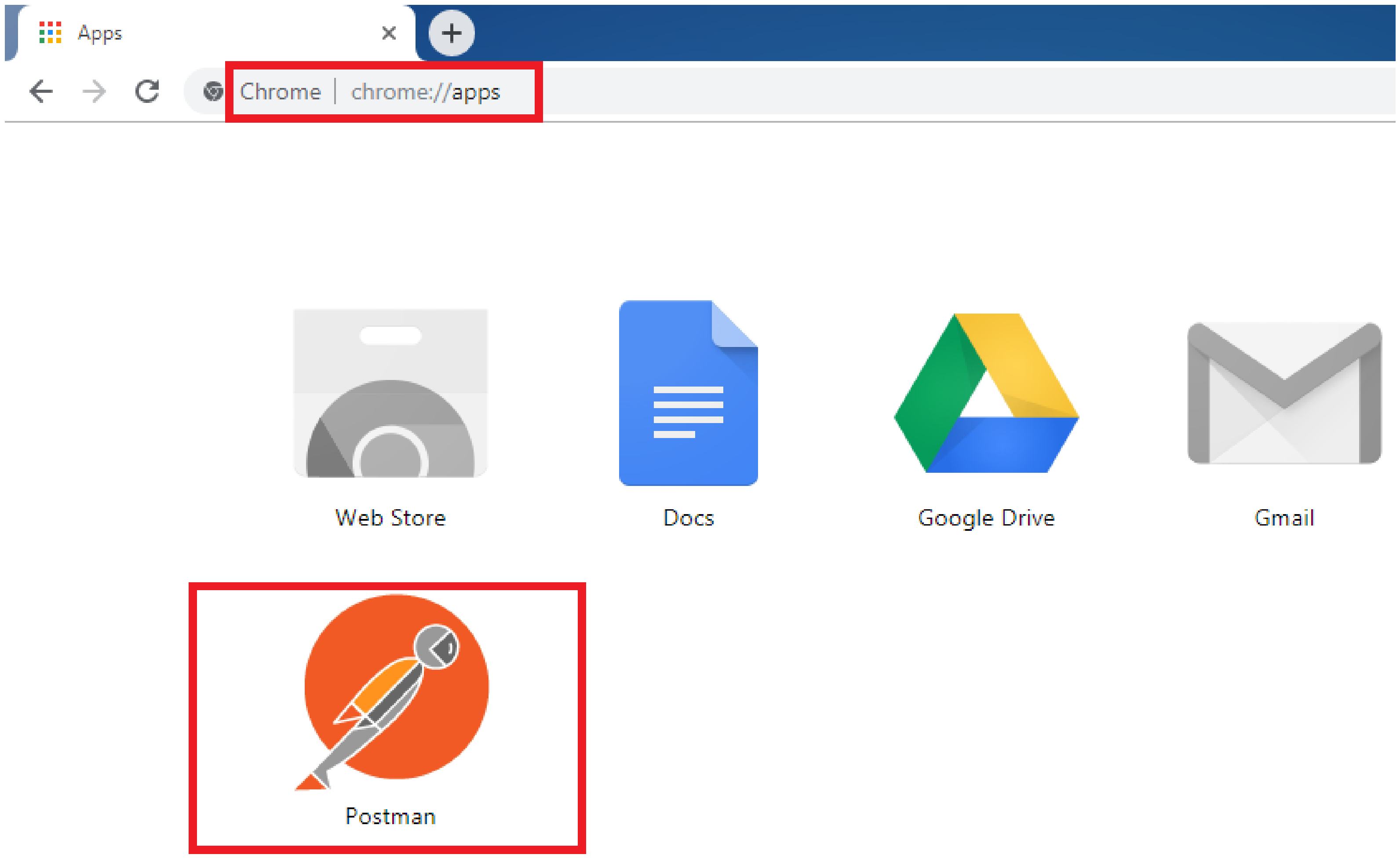
```
<?xml version = "1.0" encoding
<web-app xmlns:xsi = "http://ww
    xmlns = "http://java.sun.com
    xsi:schemaLocation="http://j
    http://java.sun.com/xml/ns/j
    id = "WebApp_ID" version = "
    <display-name>WebApplication
    <servlet>
        <servlet-name>JerseyRESTf
        <servlet-class>org.glassf
        <init-param>
            <param-name>jersey.con
            <param-value>Rest</par
        </init-param>
    </servlet>
    <servlet-mapping>
        <servlet-name>JerseyRESTf
        <url-pattern>/Test/*</url
    </servlet-mapping>
</web-app>
```

Deploy Project to Application Server

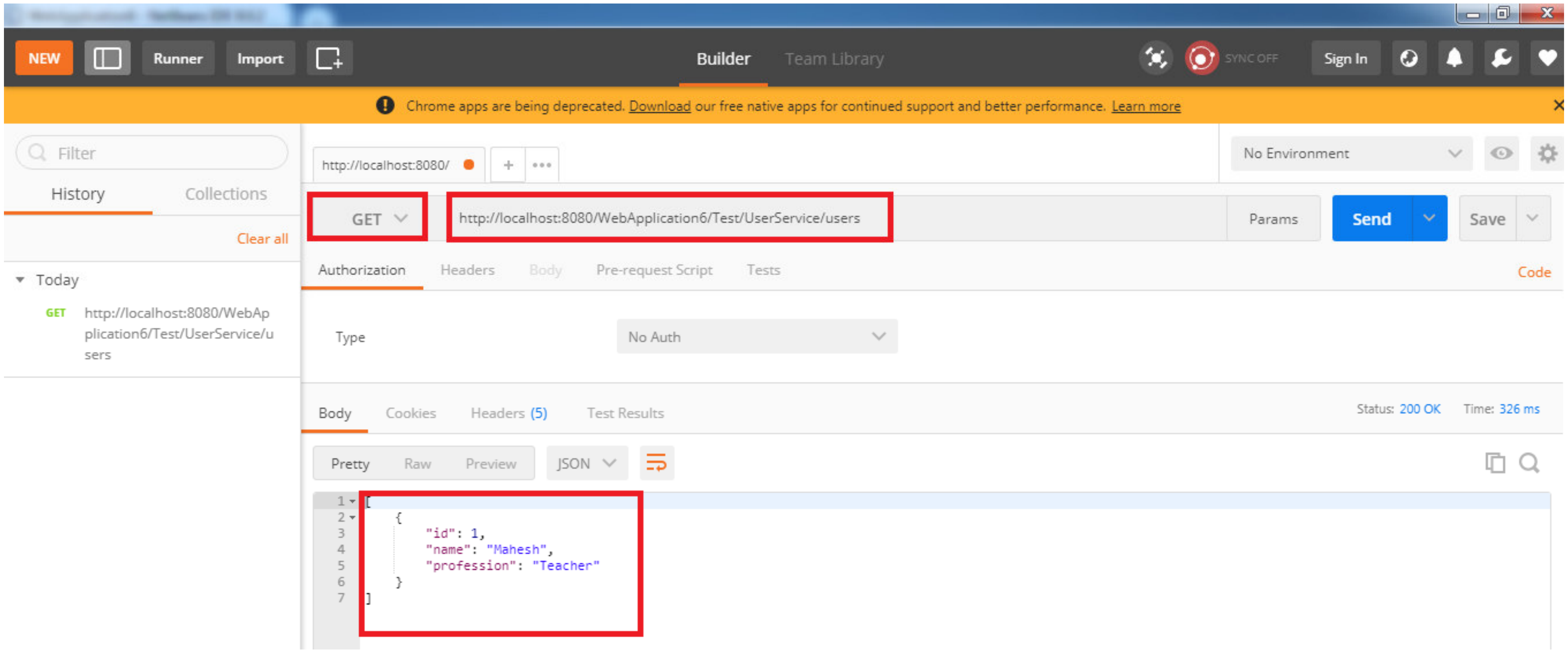


```
<?xml version = "1.0"
<web-app xmlns:xsi = "
    xmlns = "http://jav
    xsi:schemaLocation=
    http://java.sun.com
    id = "WebApp_ID" ve
    <display-name>WebAp
    <servlet>
        <servlet-name>Je
        <servlet-class>c
        <init-param>
            <param-name>j
            <param-value>
        </init-param>
    </servlet>
    <servlet-mapping>
        <servlet-name>Je
        <url-pattern>/Te
    </servlet-mapping>
</web-app>
```

Launch POSTMAN in Chrome



Test HTTP GET Method



Chrome apps are being deprecated. Download our free native apps for continued support and better performance. Learn more

GET http://localhost:8080/WebApplication6/Test/UserService/users

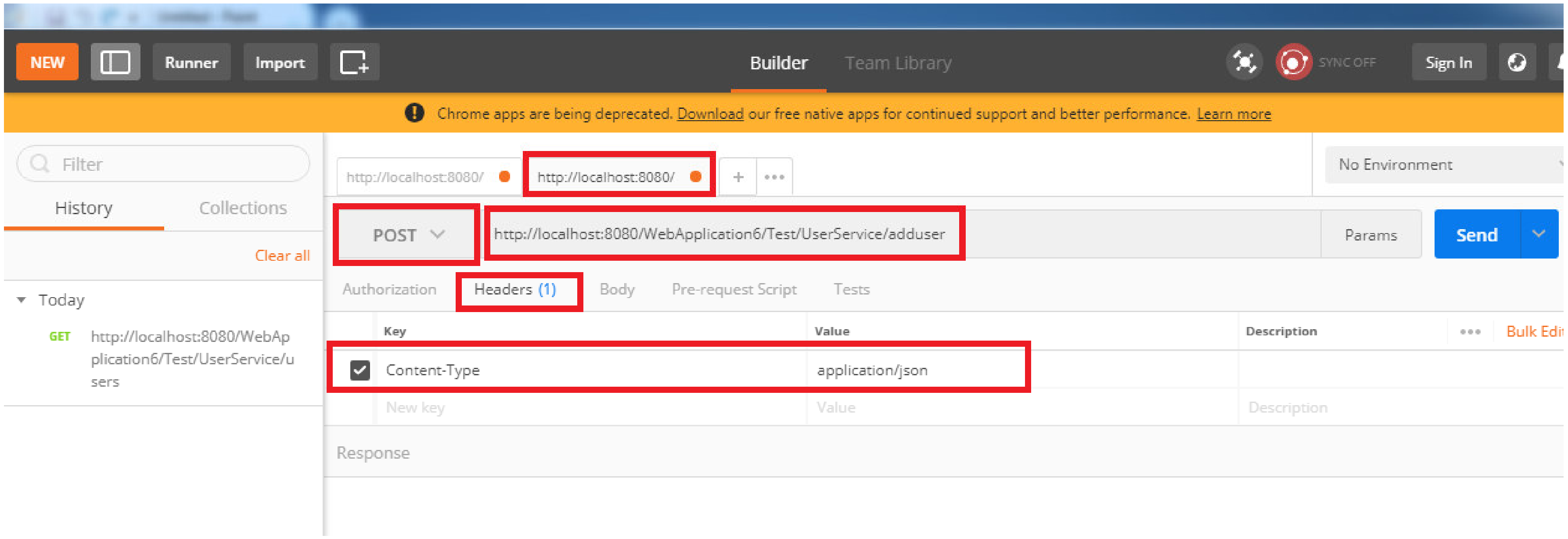
GET http://localhost:8080/WebApplication6/Test/UserService/users

Type: No Auth

Status: 200 OK Time: 326 ms

```
[
    {
        "id": 1,
        "name": "Mahesh",
        "profession": "Teacher"
    }
]
```

Test HTTP POST Method



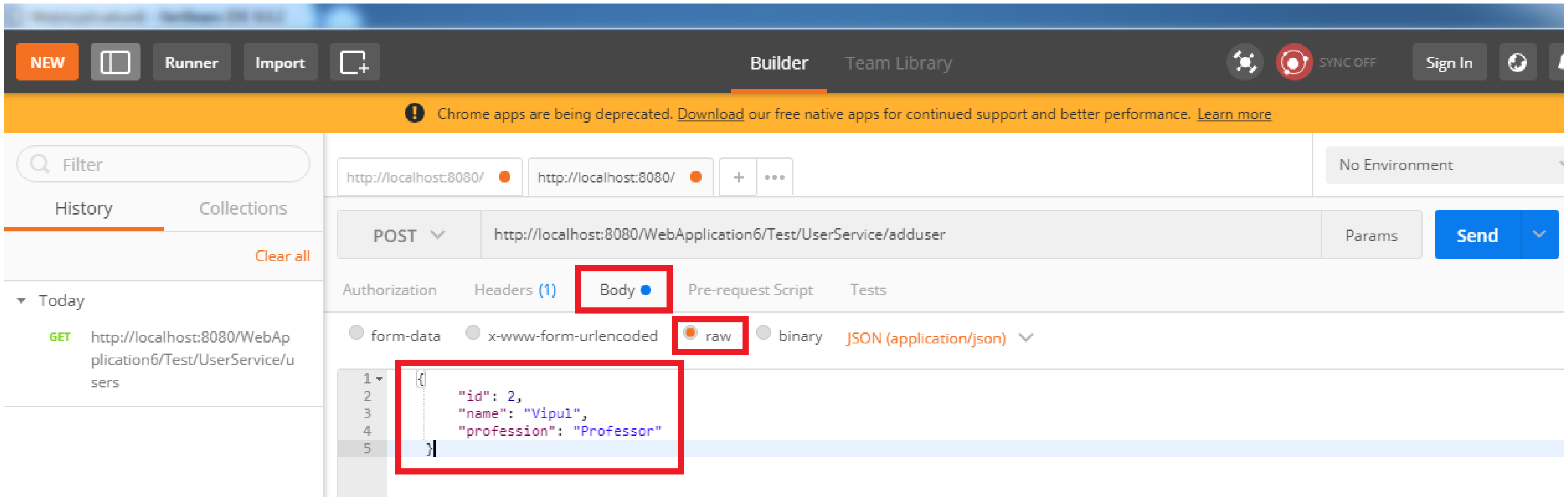
Chrome apps are being deprecated. Download our free native apps for continued support and better performance. Learn more

GET http://localhost:8080/WebApplication6/Test/UserService/users

POST http://localhost:8080/WebApplication6/Test/UserService/adduser

	Key	Value	Description
✓	Content-Type	application/json	
	New key	Value	Description

Test HTTP POST Method



Chrome apps are being deprecated. Download our free native apps for continued support and better performance. Learn more

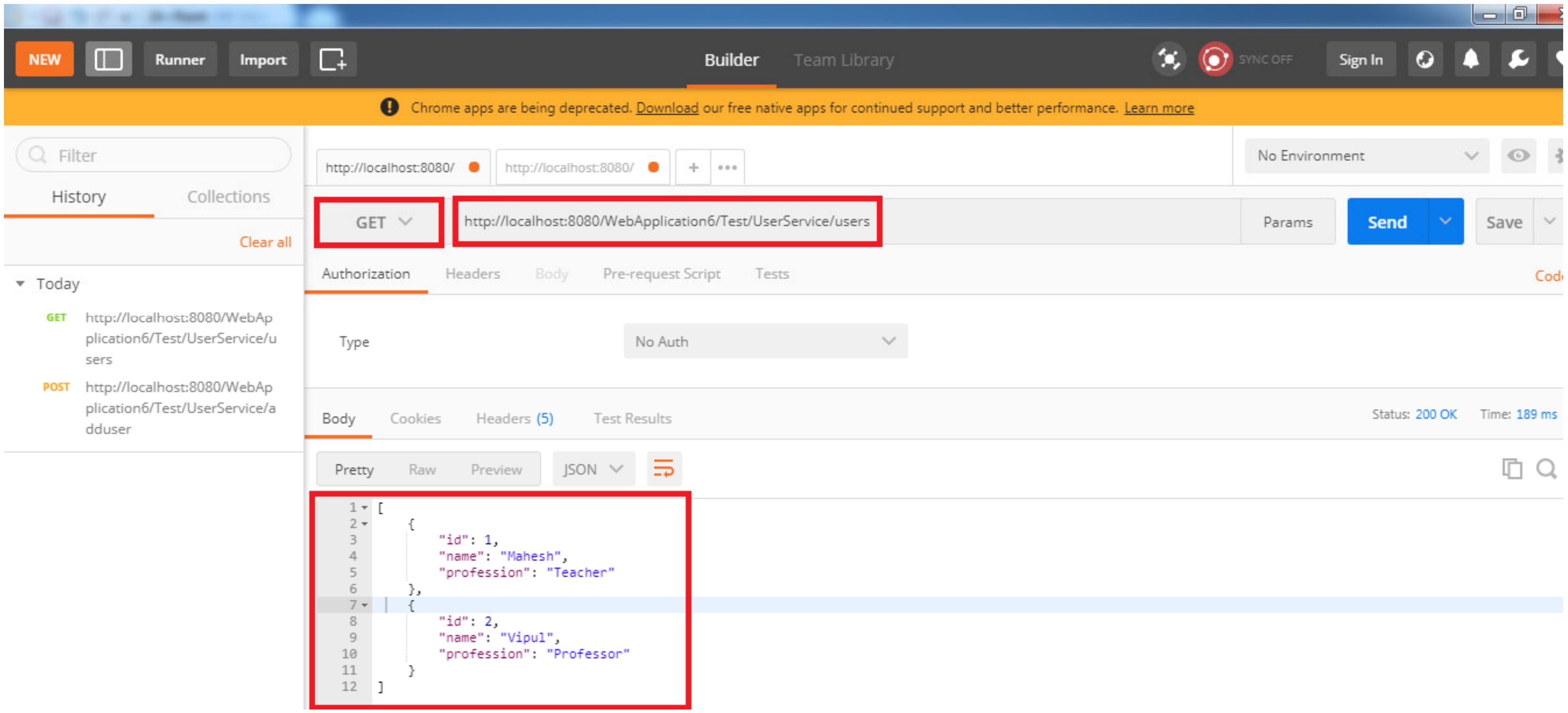
GET http://localhost:8080/WebApplication6/Test/UserService/users

POST http://localhost:8080/WebApplication6/Test/UserService/adduser

form-data x-www-form-urlencoded [✓ raw] binary JSON (application/json)

```
{
    "id": 2,
    "name": "Vipul",
    "profession": "Professor"
}
```

Test HTTP POST Method



Chrome apps are being deprecated. Download our free native apps for continued support and better performance. Learn more

GET http://localhost:8080/WebApplication6/Test/UserService/users

POST http://localhost:8080/WebApplication6/Test/UserService/adduser

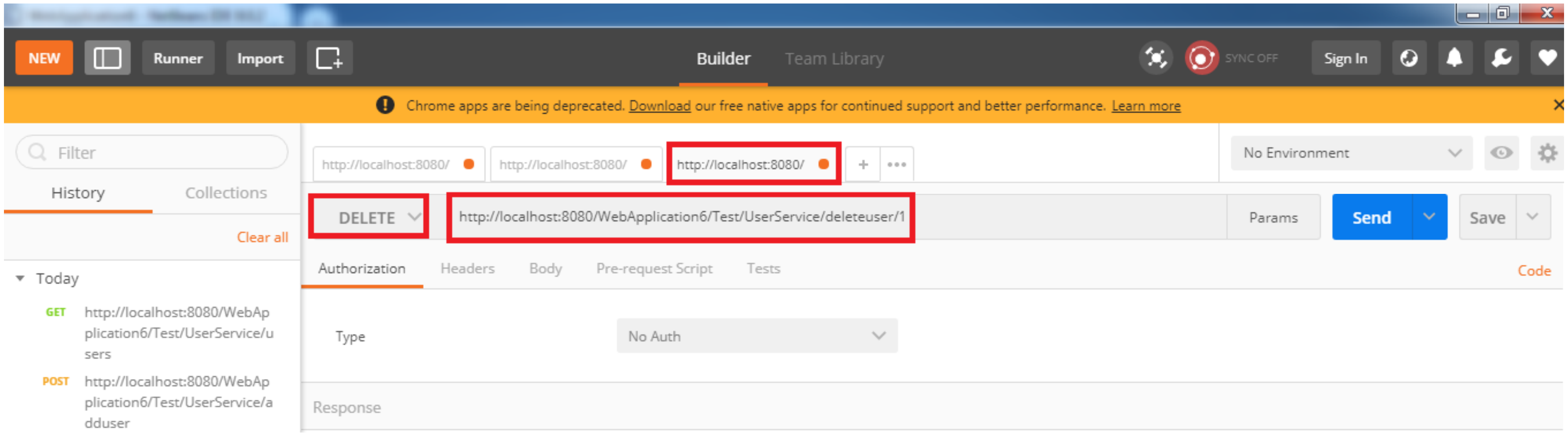
GET http://localhost:8080/WebApplication6/Test/UserService/users

Type: No Auth

Status: 200 OK Time: 189 ms

```
[
    {
        "id": 1,
        "name": "Mahesh",
        "profession": "Teacher"
    },
    {
        "id": 2,
        "name": "Vipul",
        "profession": "Professor"
    }
]
```

Test HTTP DELETE Method



Chrome apps are being deprecated. Download our free native apps for continued support and better performance. Learn more

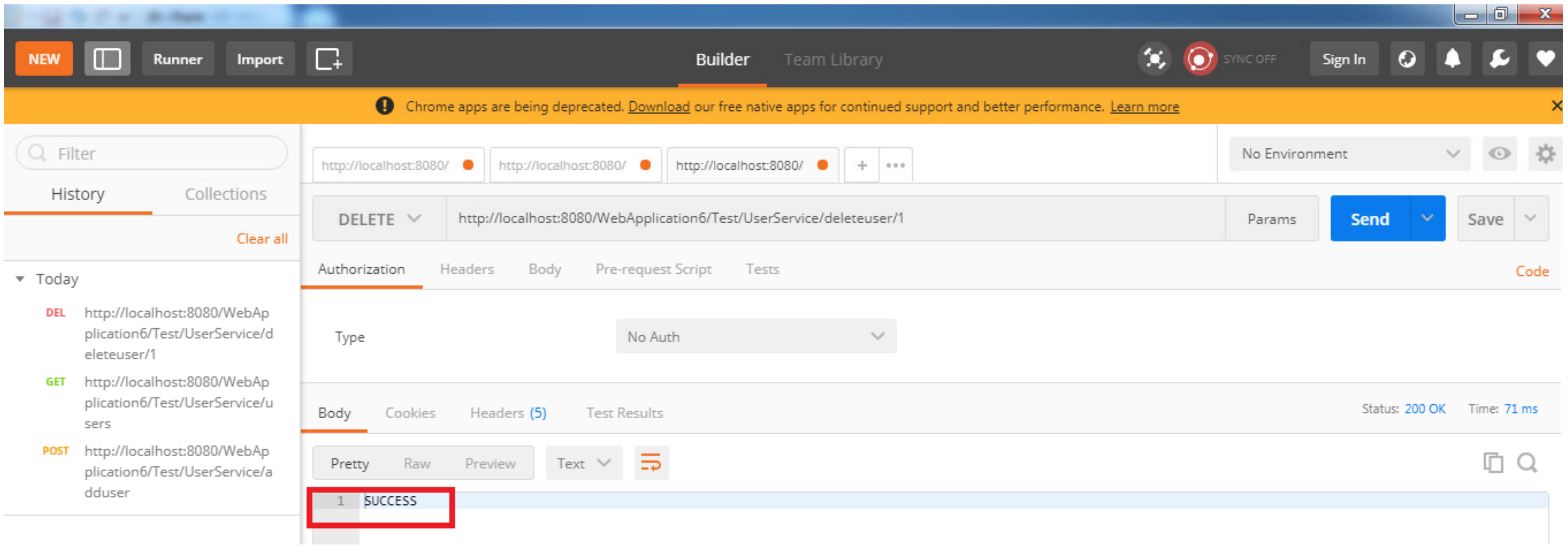
GET http://localhost:8080/WebApplication6/Test/UserService/users

POST http://localhost:8080/WebApplication6/Test/UserService/adduser

DELETE http://localhost:8080/WebApplication6/Test/UserService/deleteuser/1

Type: No Auth

Test HTTP DELETE Method



Chrome apps are being deprecated. Download our free native apps for continued support and better performance. Learn more

DEL http://localhost:8080/WebApplication6/Test/UserService/deleteuser/1

GET http://localhost:8080/WebApplication6/Test/UserService/users

POST http://localhost:8080/WebApplication6/Test/UserService/adduser

DELETE http://localhost:8080/WebApplication6/Test/UserService/deleteuser/1

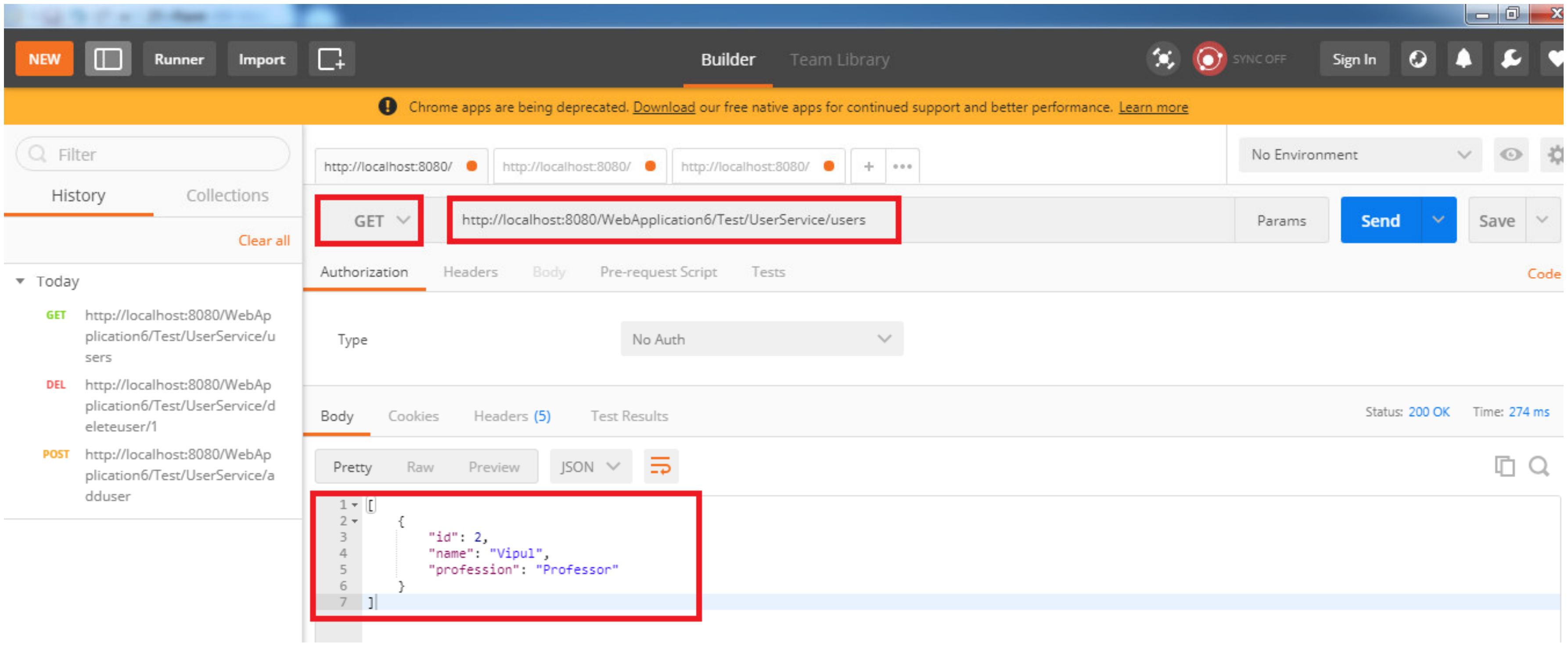
Authorization Headers Body Pre-request Script Tests

Type: No Auth

Body Cookies Headers (5) Test Results — Status: 200 OK Time: 71 ms

```
SUCCESS
```

Test HTTP DELETE Method



History:

GET http://localhost:8080/WebApplication6/Test/UserService/users

DEL http://localhost:8080/WebApplication6/Test/UserService/deleteuser/1

POST http://localhost:8080/WebApplication6/Test/UserService/adduser

GET http://localhost:8080/WebApplication6/Test/UserService/users

Type: No Auth

Status: 200 OK Time: 274 ms

```
[
    {
        "id": 2,
        "name": "Vipul",
        "profession": "Professor"
    }
]
```