

Spring Application with REST Integration

Source Code Link




















<http://www.mediafire.com/file/xbnhm252vs48kcs/Spring-Rest-Integration-App.zip>

Email: ashok.javatraining@gmail.com

Facebook Group: [Ashok IT School \(https://www.facebook.com/groups/ashokITSchool\)](https://www.facebook.com/groups/ashokITSchool)

Steps to develop Spring app with Rest Integration

Step 1: Create Maven Web Project

- ▼  SpringRestIntegrationApp
 - ▼  src/main/java
 - ▼  com.app.config
 - >  AppConfiguraiton.java
 - >  Applnitializer.java
 - ▼  com.app.controller
 - >  UserRestController.java
 - ▼  com.app.domain
 - >  User.java
 - ▼  com.app.service
 - >  UserService.java
 - >  UserServiceImpl.java
 -  src/main/resources
 -  src/test/java
 - >  JRE System Library [jre1.8.0_161]
 - >  Maven Dependencies
 - >  src
 - >  target
 -  pom.xml

Step 2: Configure maven dependencies in project pom.xml file

```
<properties>
  <springframework.version>4.3.0.RELEASE</springframework.version>
  <jackson.library>2.7.5</jackson.library>
</properties>
<dependencies>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-core</artifactId>
    <version>${springframework.version}</version>
  </dependency>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-web</artifactId>
    <version>${springframework.version}</version>
  </dependency>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>${springframework.version}</version>
  </dependency>
  <dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>3.1.0</version>
  </dependency>
  <dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>${jackson.library}</version>
  </dependency>
</dependencies>
```

Step 3: Create Domain class for Storing and Retrieving the data (User.java)

```
User.java
package com.app.domain;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement(name = "user")
@XmlAccessorType(XmlAccessType.FIELD)
public class User {

    private Integer userid;
    private String username;
    private String email;
    private String gender;

    //setters & getters

    //toString( )

}
```

Step 4: Create UserService.java class to perform Business operations

```
package com.app.service;

import java.util.HashMap;
import java.util.Map;
import org.springframework.stereotype.Service;
import com.app.domain.User;

@Service(value = "service")
public class UserServiceImpl implements UserService {

    private static Map<Integer, User> usersData = new HashMap<Integer, User>();

    public boolean add(User user) {

        if (usersData.containsKey(user.getUserid())) {

            return false;

        } else {

            usersData.put(user.getUserid(), user);

            return true;

        }

    }

    public User get(String uid) {

        System.out.println(usersData);

        if (usersData.containsKey(Integer.parseInt(uid))) {

            return usersData.get(Integer.parseInt(uid));

        }

        return null;

    }

    public boolean update(String uid, User user) {

        if (usersData.containsKey(Integer.parseInt(uid))) {

            usersData.put(Integer.parseInt(uid), user);

            return true;

        }

        return false;

    }

}
```

```
        public boolean delete(String uid) {  
            if (usersData.containsKey(Integer.parseInt(uid))) {  
                usersData.remove(usersData.get(Integer.parseInt(uid)));  
                return true;  
            }  
            return false;  
        }  
    }  
}
```

Step 5: Create RestController (UserRestController.java)

```
package com.app.controller;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.RequestBody;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RequestMethod;  
import org.springframework.web.bind.annotation.RequestParam;  
import org.springframework.web.bind.annotation.ResponseBody;  
import org.springframework.web.bind.annotation.RestController;  
  
import com.app.domain.User;  
import com.app.service.UserService;  
  
@RestController  
public class UserRestController {  
  
    @Autowired(required = true)  
    private UserService service;  
  
    @RequestMapping(value = "/add", method = RequestMethod.POST, consumes = {  
        "application/xml", "application/json" })  
    public @ResponseBody String addUser(@RequestBody User user) {
```

```
        boolean isAdded = service.add(user);  
        if (isAdded) {  
            return "User Added successfully";  
        } else {  
            return "Failed to Add the User..!";  
        }  
    }  
}
```

```
@RequestMapping(value = "/get", produces = { "application/xml", "application/json"  
,method=RequestMethod.GET)
```

```
@ResponseBody
```

```
public User getUserById(@RequestParam(name = "uid") String uid) {  
    System.out.println("Getting User with User Id : "+uid);  
    User user = service.get(uid);  
    return user;  
}
```

```
@RequestMapping(value = "/update", method = RequestMethod.PUT, consumes = {  
"application/xml", "application/json" })
```

```
public @ResponseBody String update(@RequestParam("uid") String uid, @RequestBody  
User user) {
```

```
    boolean isAdded = service.update(uid, user);  
    if (isAdded) {  
        return "User updated successfully";  
    } else {  
        return "Failed to update the User..!";  
    }  
}
```

```
}
```

```
@RequestMapping(value = "/delete", method = RequestMethod.DELETE)
```

```
public @ResponseBody String delete(@RequestParam("uid") String uid) {
```


```
    boolean isAdded = service.delete(uid);  
    if (isAdded) {
```

```
        return "User Deleted successfully";
    } else {
        return "Failed to Delete the User..!";
    }
}

public void setService(UserService service) {
    this.service = service;
}
}
```

Step 6: Create AppConfig and AppInitiazer classes

AppConfiguration.java



```
AppConfiguration.java
package com.app.config;

import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;

@Configuration
@EnableWebMvc
@ComponentScan(basePackages = "com.app.*")
public class AppConfiguration {
}
```

AppInitializer.java

```
AppInitializer.java X
package com.app.config;

import org.springframework.web.servlet.support.AbstractAnnotationConfigDispatcherServletInitializer;

public class AppInitializer extends AbstractAnnotationConfigDispatcherServletInitializer {

    @Override
    protected Class<?>[] getRootConfigClasses() {
        return new Class[] { AppConfig.class };
    }

    @Override
    protected Class<?>[] getServletConfigClasses() {
        return null;
    }

    @Override
    protected String[] getServletMappings() {
        return new String[] { "/rest/*" };
    }
}
```

Step 7: Deploy the Application into server (I have used Apache Tomcat 8.0)

Step 8: Test the Application using POSTMAN plugin in Google Chrome

Testing Add User

URI: http://localhost:6060/SpringRestIntegrationApp/rest/add

Method Type: POST

Consumes: {application/xml, application/json}

Produces: text/plain

Request Body Data: In XML Format

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

```
<user>
```

```
<userid>101</userid>
```

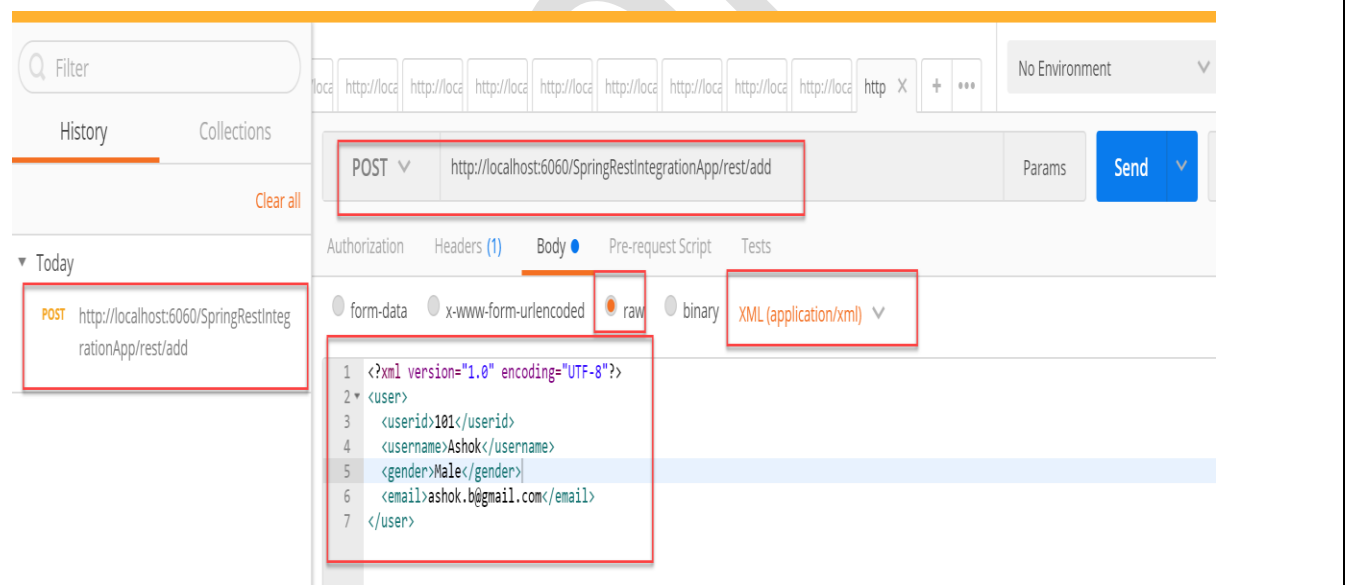
```
<username>Raju</username>
```

```
<gender>Male</gender>
```

```
<email>Raj@gmail.com</email>
```

```
</user>
```

POSTMAN Screenshot



Testing Fetch User

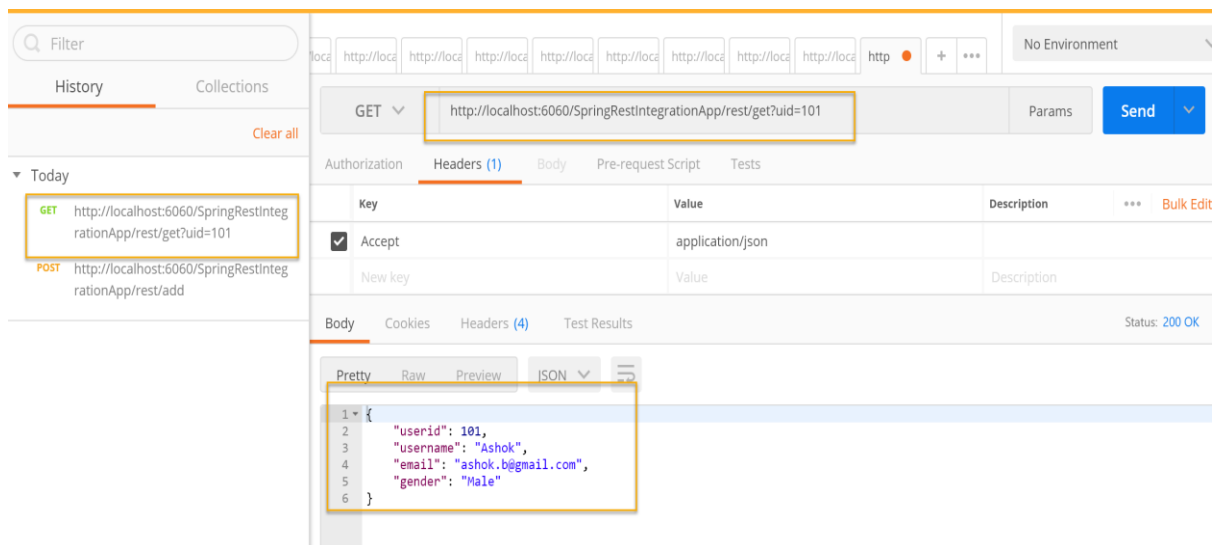
URI: http://localhost:6060/SpringRestIntegrationApp/rest/get?uid=101

Method Type: GET

Input: Request Parameter (? uid=101)

Produces: {application/xml, application/json}

POSTMAN Screenshot



Testing Update User

URL: `http://localhost:6060/SpringRestIntegrationApp/rest/update?uid=101`

Method Type: PUT

Input in URL: User Id (Request Parameter)? `uid=101`

Consumes: `{application/xml, application/json}`

Produces: `text/plain`

Request Body Data: in XML format

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

`<user>`

`<userid>101</userid>`

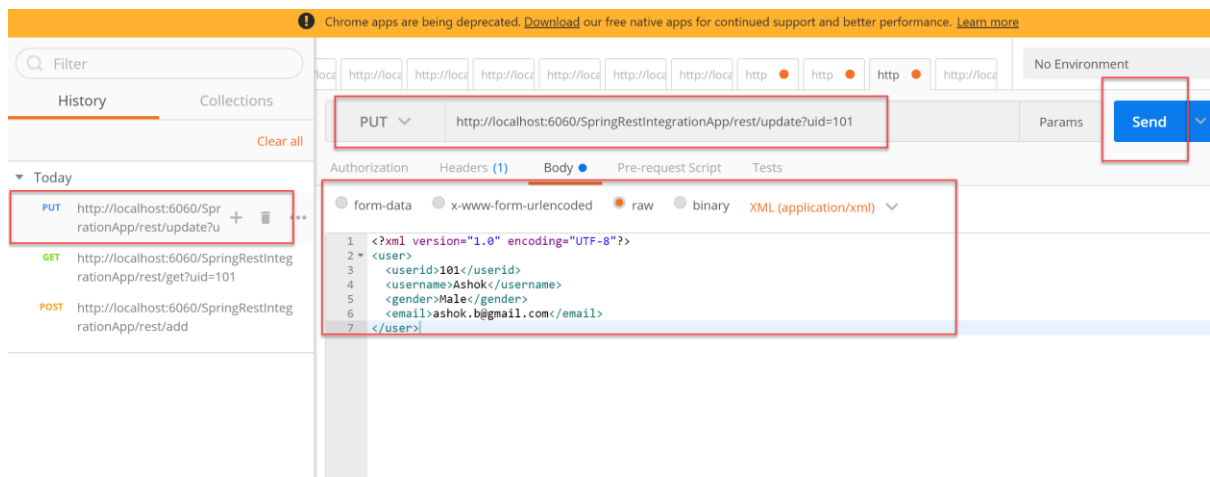
`<username>Ashok</username>`

`<gender>Male</gender>`

`<email>ashok.b@gmail.com</email>`

`</user>`

POSTMAN Screenshot



Testing Delete User

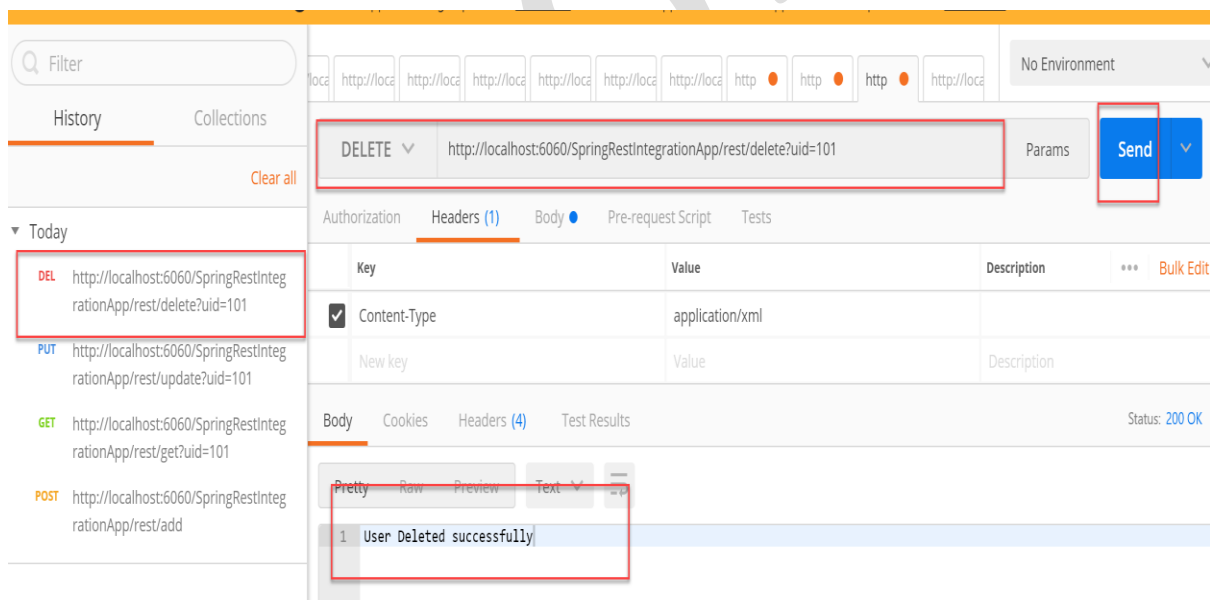
URL: http://localhost:6060/SpringRestIntegrationApp/rest/delete?uid=101

Method Type: DELETE

Input: Request Parameter (? uid=101)

Produces: text/plain

POSTMAN Screenshot



-----Thankyou-----