Jersey 2.x web service using @FormParam annotation

By SJ | October 11, 2014 | No Comments |

In this article, we will learn and implement @FormParam annotation in JAX-RS Restful web service. @FormParam binds the value/s of form parameter sent by the client in the POST request into formal arguments of Java method

Jersey is the most popular amongst Restful web service development. Latest Jersey 2.x version has been developed by Oracle/Glassfish team in accordance with JAX-RS 2.0 specification. Earlier Jersey 1.x version was developed and supported by Oracle/Sun team

Latest Jersey release version is 2.12 see here and look documentation and API for details. We will implement Jersey examples in the following articles based on latest 2.x version

@FormParam v/s @QueryParam

Use @FormParam for POST request when using form and prefer @QueryParam for GET requests

Annotation Used

- @FormParam (javax.ws.rs.FormParam)
- @Path (javax.ws.rs.Path)
- @GET (javax.ws.rs.GET)

Technology Used

- Java 1.7
- Eclipse Luna IDE
- Jersey-2.12
- Apache Maven 3.2.1
- Apache Tomcat 8.0.54
- Glassfish-4.1

Mavenize or download required jars

Add ***Jersey-2.12*** dependencies to pom.xml

```xml
<properties>
    <jersey.version>2.12</jersey.version>
    <jersey.scope>compile</jersey.scope>
    <compileSource>1.7</compileSource>
    <maven.compiler.target>1.7</maven.compiler.target>
    <maven.compiler.source>1.7</maven.compiler.source>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
</properties>

<dependencies>
    <!-- Jersey core Servlet 2.x implementation -->
    <dependency>
        <groupId>org.glassfish.jersey.containers</groupId>
        <artifactId>jersey-container-servlet-core</artifactId>
        <version>${jersey.version}</version>
        <scope>${jersey.scope}</scope>
    </dependency>
</dependencies>
```

Folks who aren't familiar with Maven concepts or don't require maven for their project, can download the below jars individually from the central repository or maven repository and include them in the classpath

- jersey-container-servlet-core
- javax.ws.rs-api-2.0.1
- jersey-server-2.12
- jersey-common-2.12
- jersey-client-2.12
- jersey-guava-2.12
- javax-annotation-api-1.2
- javax.inject-2.3.0-b10
- hk2-api-2.3.0-b10
- hk2-utils-2.3.0-b10
- aop-alliance-repackaged-2.3.0-b10
- hk2-locator
- javassist-3.18.1-GA
- validation-api-1.1.0.Final
- osgi-resource-locator-1.0.1

Directory Structure

Before moving on, let us understand the directory/package structure once you create project in Eclipse IDE

Maven has to follow certain directory structure

- src/test/java –> test related files, mostly JUnit test cases
- src/main/java –> create java source files under this folder
- src/main/resources –> all configuration files placed here
- src/test/resources –> all test related configuration files placed here
- Maven Dependencies or Referenced Libraries –> includes jars in the classpath
- WEB-INF under webapp –> stores web.xml & other configuration files related to web application

Project Structure (Package Explorer view in Eclipse)

Jars Libraries Used in the Project (Maven Dependencies)

Web application

For any web application, entry point is ***web.xml*** which describes how the incoming http requests are served / processed. Further, it describes about the global-context and local-context param (i.e.; <context-param> & <init-param>) for loading files particular to project requirements & contains respective listener

With this introduction, we will understand how we configured ***web.xml*** for Jersey JAX-RS Restful web service

web.xml (the entry point –> under WEB-INF)

This ***web.xml*** file describes,

POPULAR ARTICLES

Oracle OSB 12c: Service Callout and Routing Table example
Spring JDBC: An example on JdbcTemplate using Annotation
Jersey 2.x web service for uploading/downloading Image file + Java client
Apache CXF: JAX-RS Restful web service using both (JSON + XML) example
Jersey 2.x web service for uploading/downloading Zip file + Java client

RECENT ARTICLES

- Interview Questions and Answers on Java JDBC
- Java JDBC: Metadata of ResultSet using ResultSetMetaData Interface
- Java JDBC: Metadata of database using DatabaseMetaData Interface
- Java JDBC: Order By clause using JDBC Statement Interface
- Java JDBC: Like clause condition using JDBC Statement Interface

- Like any JEE web framework register "*org.glassfish.jersey.servlet.ServletContainer*" with servlet container
- http requests with URL pattern ***"/rest/*"*** will be sent to the registered servlet called "jersey-servlet" i.e.; *(org.*glassfish.jersey.servlet.ServletContainer*)*
- Set <init-param> with <param-name> as "*jersey.config.server.provider.packages*" and <param-value> describing the qualified package name of the JAX-RS annotated Resource/Provider classes. In this example, "*com.jersey.series.formparam*"
- <welcome-file-list> files under this tag is the start-up page

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.

    <display-name>Jersey-FormParam</display-name>

    <!-- Jersey Servlet -->
    <servlet>
        <servlet-name>jersey-servlet</servlet-name>
        <servlet-class>org.glassfish.jersey.servlet.ServletContainer</
        <!-- Register resources and providers -->
        <init-param>
            <param-name>jersey.config.server.provider.packages</param-
            <param-value>com.jersey.series.formparam</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>

    <servlet-mapping>
        <servlet-name>jersey-servlet</servlet-name>
        <url-pattern>/rest/*</url-pattern>
    </servlet-mapping>

    <!-- welcome file -->
    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
    </welcome-file-list>

</web-app>
```

Let's see coding in action

URL Pattern

Http url for any common web application is http://<server>:<port>/<root-context>/<from_here_application_specific_path>

In our example, we are going to deploy the war into Tomcat 8.0 server so our server and port are *localhost* and *8080* respectively. Context root is the project name i.e.; Jersey-FormParam. Initial path for this application is http://localhost:8080/Jersey-FormParam

We have configured ***"/rest/*"*** as url-pattern for the "jersey-servlet" servlet in web.xml and at class-level path configured is ***"/bookservice"*** using @Path annotation. Next respective path for each method annotated with @Path (method-level)

Book Service interface

Defines just one simple abstract method to add new book information

NOTE: It's always a good programming practice to do ***code-to-interface*** and have its implementation separately

IBookService.java

```
package com.jersey.series.formparam;

import javax.ws.rs.core.Response;

public interface IBookService {

    public Response addBookInfo(String bookName, String author, String
}
```

Book Service implementation

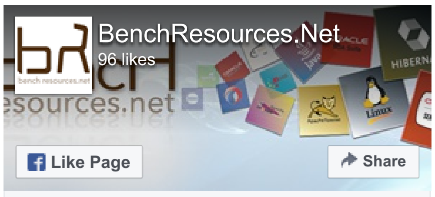
Implements above interface returning Response object

TAGS

Apache CXF constructor constructor overloading database default constructor Eclipse Hibernate insert interview question Java java8 JAX-RS JAX-WS JAXB JDBC JDBC Connection steps JDeveloper 12c Jersey JSON keyword Maven method overloading Metro mysql oops Oracle OSB overloading parametrized constructor REST RestEasy select SOA SOAP Spring Spring AOP Spring Core Spring JDBC Spring MVC Spring Security table update Weblogic Web Services XML

BenchResources Net

G+ Suivre



Be the first of your friends to like this

A simple method which takes **three-arguments** using @FormParam annotation (*multiple parameters*)
URL: http://localhost:8080/Jersey-FormParam/rest/bookservice/bookinfo

POST request from the form will invokes this method when the relative action (path) is set to ***"rest/bookservice/bookinfo"***

Note: Jersey doesn't inherit JAX-RS annotations. So we are annotating Resource/Provider classes and then defining qualified package name in ***web.xml***

BookServiceImpl.java

```
package com.jersey.series.formparam;

import javax.ws.rs.FormParam;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.core.Response;

@Path("bookservice")
public class BookServiceImpl implements IBookService {

    // http://localhost:8080/Jersey-FormParam/rest/bookservice/bookinf
    @POST
    @Path("/bookinfo")
    public Response addBookInfo(
            @FormParam("name") String bookName,
            @FormParam("author") String author,
            @FormParam("category") String category){

        String bookInfo = "Book [Name=" + bookName +  ", Author=" + au
        return Response.status(200).entity(bookInfo).build();
    }
}
```

View Technologies

index.html (\src\main\webapp\index.html)
Start up page which has anchor link to "addNewBook.html" with value Click here à redirects to addNewBook.html page

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Jersey-FormParam</title>
</head>
<body style="align: center">
    <b>Welcome to Jersey 2.x web service learnings</b>
    <br />
    <a href="https://jersey.java.net/">Jersey Home Page</a>
    <br />
    <a href="https://jersey.java.net/documentation/latest/index.html">
        documentation</a>
    <br />
    <a href="https://jersey.java.net/download.html">Latest Jersey
        bundle</a>
    <br />
    <br />
    <br />
    <br />
    <a href="addNewBook.html">Click here </a>to add new book
</body>
</html>
```

addNewCustomer.html (\src\main\webapp\addNewBook.html)
Simple HTML form with POST action (method) which takes three inputs from the end user to *add new book information* to the database

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http:
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1
<title>Jersey - @FormParam</title>
</head>
<body>
    <center>
        <b>Welcome to the JAX-RS Restful web service Learning</b>
        <h4>An Example on @FormParam Annotation</h4>
        <form method="post" action="rest/bookservice/bookinfo">
            <table>
                <tr>
                    <td>Book Name:</td>
                    <td><input type="text" name="name" /></td>
                </tr>
                <tr>
                    <td>Author:</td>
                    <td><input type="text" name="author" /></td>
                </tr>
                <tr>
                    <td>Category:</td>
                    <td><input type="text" name="category" /></td>
                </tr>
                <tr>
                    <td colspan="1"><input name="submit" type="submit"
                        value="Add New Book" /></td>
                    <td colspan="1"><input name="reset" type="reset" v
                </tr>
            </table>
        </form>
    </center>
</body>
</html>
```

Tomcat-8.0.12 Deployment

- Run maven command to build the war: ***mvn clean install*** (use command prompt or integrated maven in eclipse IDE
- Copy(ctrl+c) the war file from the target folder
- Paste(ctrl+v) it into apache tomcat (webapps folder)
- Start the tomcat server

Glassfish-4.1 Deployment

- Run maven command to build the war: ***mvn clean install*** (use command prompt or integrated maven in eclipse IDE
- Once you see "BUILD SUCCESS" after running maven command, keep the war file ready to be deployed
- There are two ways to deploy war file into Glassfish-4.1
  1. Online
  2. Offline
- Click here to understand above deployments process in detail

Test the service !!

Testing

Access the default URL http://localhost:8080/Jersey-FormParam which has a anchor link to addNewBook.html page, apart from some basic links to Jersey pages

Click on the link, you will be redirected to addNewBook.html page which displays fields for entering *Book-Name, Author* & *Category* to capture new book information and store it in the database

Or else directly access the URL for the same page
http://localhost:8080/Jersey-FormParam/addNewBook.html –> displays below page

Enter details of book information like

Book Name	:	Diagnostic Microbiology
Author	:	Patricia Tille
Category	:	Microbiology

And click ***"Add New Book"*** button

Clicking ***"Add New Book"*** button will invokes POST method configured in the form tag with action=*"rest/bookservice/bookinfo"*

On successful execution of the request, a success message is returned to the user as shown below

2. Java client

Uses Client, ClientBuilder, WebTarget and Response classes from core JAX-RS package javax.ws.rs.client for invoking Restful web service

Note: Commented lines will help out us to various request header parameters

TestBookService.java

```
package test.jersey.series.formparam;

import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Entity;
import javax.ws.rs.client.Invocation.Builder;
import javax.ws.rs.client.WebTarget;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.MultivaluedHashMap;
import javax.ws.rs.core.MultivaluedMap;
import javax.ws.rs.core.Response;

import org.glassfish.jersey.client.ClientConfig;

public class TestBookService {

    public static void main(String[] args) {

        // setting & invoking first request
        System.out.println("Executing POST request for @FormParameter"
        String url = "http://localhost:8080/Jersey-FormParam/rest/book
        String responseString = testFormParameterForPost(url);
        System.out.println("Response String for @FormParam : " + respo
    }

    public static String testFormParameterForPost(String httpURL) {

        // local variables
        ClientConfig clientConfig = null;
        Client client = null;
        WebTarget webTarget = null;
        MultivaluedMap<String, String> formParameters = null;
        Builder builder = null;
        Response response = null;
        int responseCode;
        String responseMessageFromServer = null;
        String responseString = null;

        try{
            // invoke service after setting necessary parameters
            clientConfig = new ClientConfig();
            client =  ClientBuilder.newClient(clientConfig);
            //          client.property("Content-Type", MediaType.TEXT
            //          client.property("accept", MediaType.TEXT_PLAIN
            webTarget = client.target(httpURL);

            //
            formParameters = new MultivaluedHashMap<String, String>();
            formParameters.add("name", "Diagnostic Microbiology");
            formParameters.add("author", "Patricia Tille");
            formParameters.add("category", "Microbiology");

            // invoke service
            builder = webTarget.request(MediaType.TEXT_PLAIN).accept(M
            response = builder.post(Entity.form(formParameters));

            // get response code
            responseCode = response.getStatus();
            System.out.println("Response code: " + responseCode);

            if (response.getStatus() != 200) {
                throw new RuntimeException("Failed with HTTP error cod
            }

            // get response message
            responseMessageFromServer = response.getStatusInfo().getRe
            System.out.println("ResponseMessageFromServer: " + respons

            // get response string
            responseString = response.readEntity(String.class);
        }
        catch(Exception ex) {
            ex.printStackTrace();
        }
        finally{
            // release resources, if any
            response.close();
            client.close();
        }
        return responseString;
    }
}
```

Output in console

```
Executing POST request for @FormParameter
Response code: 200
ResponseMessageFromServer: OK
Response String for @FormParam : Book [Name=Diagnostic Microbiology, Autho
```

Conclusion: When the requirement is to send the data from the form, then this approach would be appropriate

Download project

```
Jersey-2x-FormParam (8kB)
```

Happy Coding !!

Happy Learning !!

RELATED

Jersey 2.x web service using @QueryParam annotation
October 10, 2014
In "Jersey 2.x"

Jersey 2.x web service using @MatrixParam annotation
October 10, 2014
In "Jersey 2.x"

Jersey 2.x web service using @HeaderParam/@Context annotations
October 11, 2014
In "Jersey 2.x"

Tags: **Java**, **JAX-RS**, **Jersey**, **REST**, **Web Services**

Leave A Reply

Comment

Name *Name (required)*

Email *Email (required)*

Website *Website*

Post Comment

☐ Notify me of follow-up comments by email.

☐ Notify me of new posts by email.