Resource : Data/response from getting server

 We can get the resource from RESTful service in different formats like, HTML,XML,JSON,TEXT,PDF and in the Image formats as well ,most in real time we are getting JSON format.

REST guidelines always talks about stateless communication between client and the Server.

Stateless means, every single request from client to server will be considered as a fresh request.

Because of this reason REST always prefers to choose HTTP as it a stateless protocol.

RESTful used 4 main HTTP methods…

* **GET** - Retrieve Data
* **POST**- Create/Insert Data
* **PUT**- Update Data
* **DELETE**- Delete Data

Generally we will prefer RESTful Services in these scenarios…

* If clients **require *caching***, means if you have limited bandwidth
* If you want every thing to **be stateless** [ I have already explained about stateless ]

REST annotations

@Path: it is used class or method level it will check the path next to the base URI

@GET :it is only method level only response should be HTTP get only.

@POST : it is only method level only response should be HTTP POST only

@PUT :it is method level response HTTP PUT

@DELETE :it is method level response HTTP delete

@Produces : it is a method or field level annotation, This tells which MIME type is delivered by the method annotated with @GET.

I mean when ever we send a HTTP GET request to our RESTful service, it will invokes particular method and produces the output in different formats.  There you can specifies in what are all formats (MIME) your method can produce the output, by using @produces annotation.

**Remember**: We will use @Produces annotation for GET requests only.

**@Consumes**

This is a class and method level annotation, this will define which MIME type is consumed by the particular method. I mean in which format the method can accept the input from the client.

 In JAX-RS we can use the following annotations to extract the input values sent by the client.

i)@PathParm

ii)@QueryParam

iii)@MatrixParam

iv)@FormParam

@PathParam:

**http://localhost:7001/<Rest Service Name>/rest/customers/100/Java4s**

@QueryParm :

http://localhost:7001/…/rest/customers?custNo=100&custName=Java4s

@MatrixParam:

http://localhost:7001/…/rest/customers;custNo=100;custName=Java4s

1. **@Produces("text/plain")**: for downloading text file.
2. **@Produces("image/png")**: for downloading png image file.
3. **@Produces("application/pdf")**: for downloading PDF file.
4. **@Produces("application/vnd.ms-excel")**: for downloading excel file.
5. **@Produces("application/msword")**: for downloading ms word file.

**Client Code :**

1. **import** java.net.URI;
2. **import** javax.ws.rs.client.Client;
3. **import** javax.ws.rs.client.ClientBuilder;
4. **import** javax.ws.rs.client.WebTarget;
5. **import** javax.ws.rs.core.MediaType;
6. **import** javax.ws.rs.core.UriBuilder;
7. **import** org.glassfish.jersey.client.ClientConfig;
8. **public** **class** ClientTest {
9. **public** **static** **void** main(String[] args) {
10. ClientConfig config = **new** ClientConfig();
11. Client client = ClientBuilder.newClient(config);
12. WebTarget target = client.target(getBaseURI());
13. //Now printing the server code of different media type
14. System.out.println(target.path("rest").path("hello").request().accept(MediaType.TEXT\_PLAIN).get(String.**class**));   // rest URI in web.xml ;hello is path uri at class level
15. System.out.println(target.path("rest").path("hello").request().accept(MediaType.TEXT\_XML).get(String.**class**));
16. System.out.println(target.path("rest").path("hello").request().accept(MediaType.TEXT\_HTML).get(String.**class**));
17. }
18. **private** **static** URI getBaseURI() {
19. //here server is running on 4444 port number and project name is restfuljersey
20. **return** UriBuilder.fromUri("http://localhost:4444/restfuljersey").build();
21. }
22. }

|  |  |
| --- | --- |
| **Annotation** | **Description** |
| Path | It identifies the URI path. It can be specified on class or method. |
| PathParam | represents the parameter of the URI path. |
| GET | specifies method responds to GET request. |
| POST | specifies method responds to POST request. |
| PUT | specifies method responds to PUT request. |
| HEAD | specifies method responds to HEAD request. |
| DELETE | specifies method responds to DELETE request. |
| OPTIONS | specifies method responds to OPTIONS request. |
| FormParam | represents the parameter of the form. |
| QueryParam | represents the parameter of the query string of an URL. |
| HeaderParam | represents the parameter of the header. |
| CookieParam | represents the parameter of the cookie. |
| Produces | defines media type for the response such as XML, PLAIN, JSON etc. It defines the media type that the methods of a resource class or MessageBodyWriter can produce. |
| Consumes | It defines the media type that the methods of a resource class or MessageBodyReader can produce. |

Upload File Example:

The @FormDataParam("file") annotation is used to mention file parameter in the service class. The @Consumes(MediaType.MULTIPART\_FORM\_DATA) is used to provide information of the file upload.

To upload file using JAX-RS API, we are using jersey implementation.

[Click me to download jersey jar files.](http://www.javatpoint.com/webservicepages/download/jerseyjars.zip)

To upload file through jersey implementation, you need to provide extra configuration entry in web.xml file.

1. **<init-param>**
2. **<param-name>**jersey.config.server.provider.classnames**</param-name>**
3. **<param-value>**org.glassfish.jersey.filter.LoggingFilter;
4. org.glassfish.jersey.media.multipart.MultiPartFeature**</param-value>**
5. **</init-param>**

Let's see the complete code to upload file using RESTful JAX-RS API.

JAX-RS File Upload

*File: FileUploadService.java*

1. **package** com.javatpoint.rest;
2. **import** java.io.File;
3. **import** java.io.FileOutputStream;
4. **import** java.io.IOException;
5. **import** java.io.InputStream;
6. **import** javax.ws.rs.Consumes;
7. **import** javax.ws.rs.POST;
8. **import** javax.ws.rs.Path;
9. **import** javax.ws.rs.core.MediaType;
10. **import** javax.ws.rs.core.Response;
11. **import** org.glassfish.jersey.media.multipart.FormDataContentDisposition;
12. **import** org.glassfish.jersey.media.multipart.FormDataParam;
13. @Path("/files")
14. **public** **class** FileUploadService {
15. @POST
16. @Path("/upload")
17. @Consumes(MediaType.MULTIPART\_FORM\_DATA)
18. **public** Response uploadFile(
19. @FormDataParam("file") InputStream uploadedInputStream,
20. @FormDataParam("file") FormDataContentDisposition fileDetail) {
21. String fileLocation = "e://" + fileDetail.getFileName();
22. //saving file
23. **try** {
24. FileOutputStream out = **new** FileOutputStream(**new** File(fileLocation));
25. **int** read = 0;
26. **byte**[] bytes = **new** **byte**[1024];
27. out = **new** FileOutputStream(**new** File(fileLocation));
28. **while** ((read = uploadedInputStream.read(bytes)) != -1) {
29. out.write(bytes, 0, read);
30. }
31. out.flush();
32. out.close();
33. } **catch** (IOException e) {e.printStackTrace();}
34. String output = "File successfully uploaded to : " + fileLocation;
35. **return** Response.status(200).entity(output).build();
36. }
37. }

*File: web.xml*

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<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
3. http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd" id="WebApp\_ID" version="3.0"**>**
4. **<servlet>**
5. **<servlet-name>**Jersey REST Service**</servlet-name>**
6. **<servlet-class>**org.glassfish.jersey.servlet.ServletContainer**</servlet-class>**
7. **<init-param>**
8. **<param-name>**jersey.config.server.provider.packages**</param-name>**
9. **<param-value>**com.javatpoint.rest**</param-value>**
10. **</init-param>**
11. **<init-param>**
12. **<param-name>**jersey.config.server.provider.classnames**</param-name>**
13. **<param-value>**org.glassfish.jersey.filter.LoggingFilter;
14. org.glassfish.jersey.media.multipart.MultiPartFeature**</param-value>**
15. **</init-param>**
16. **<load-on-startup>**1**</load-on-startup>**
17. **</servlet>**
18. **<servlet-mapping>**
19. **<servlet-name>**Jersey REST Service**</servlet-name>**
20. **<url-pattern>**/rest/\***</url-pattern>**
21. **</servlet-mapping>**
22. **</web-app>**

*File: index.html*

1. **<h2>**File Upload Example**</h2>**
2. **<form** action="rest/files/upload" method="post" enctype="multipart/form-data"**>**
3. **<p>**
4. Select a file : **<input** type="file" name="file" size="45" **/>**
5. **</p>**
6. **<input** type="submit" value="Upload File" **/>**
7. **</form>**

Now run this application on server, you will see the following output:

Output:

1. File successfully uploaded to e://myimage.png