RESTful JAX-RS Annotations Example

JAX-RS API provides following annotations to develop RESTful applications in java. We are using jersey implementation for developing JAX-RS examples.

Click me to download jersey jar files.

JAX-RS Annotations

The **javax.ws.rs** package contains JAX-RS annotations.

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.

JAX-RS @Path, @GET and @PathParam Annotations

File: HelloService.java

```
package com.javatpoint.rest;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.core.Response;
@Path("/hello")
public class HelloService{
    @GET
    @Path("/{param}")
    public Response getMsg(@PathParam("param") String msg) {
```

```
        String output = "Jersey say : " + msg;
        return Response.status(200).entity(output).build();
    }
}
```

File: 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">
 <servlet>
   <servlet-name>Jersey REST Service</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.javatpoint.rest</param-value>
   </init-param>
   <load-on-startup>1</load-on-startup>
 </servlet>
 <servlet-mapping>
   <servlet-name>Jersey REST Service</servlet-name>
   <url-pattern>/rest/*</url-pattern>
 </servlet-mapping>
</web-app>
```

File: index.html

```
<a href="rest/hello/javatpoint">Click Here</a>
```

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

Output:

```
Jersey say : javatpoint
```

Click me to download this example

JAX-RS Multiple @PathParam Annotation

File: HelloService.java

```
package com.javatpoint.rest;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.core.Response;
@Path("/hello")
public class HelloService{
   @GET
   @Path("{year}/{month}/{day}")
   public Response getDate(
         @PathParam("year") int year,
         @PathParam("month") int month,
         @PathParam("day") int day) {

      String date = year + "/" + month + "/" + day;

      return Response.status(200)
```

```
        .entity("getDate is called, year/month/day : " + date)
        .build();
    }
}
```

File: web.xml

It is same as above example.

File: index.html

```
<a href="rest/hello/2014/12/05">Click Here</a>
```

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

Output:

```
getDate is called, year/month/day : 2014/12/5
```

Click me to download this example

JAX-RS @FormParam and @POST Annotation

File: HelloService.java

```
package com.javatpoint.rest;
import javax.ws.rs.FormParam;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.core.Response;
@Path("/product")
public class ProductService{
    @POST
    @Path("/add")
    public Response addUser(
        @FormParam("id") int id,
        @FormParam("name") String name,
        @FormParam("price") float price) {

        return Response.status(200)
            .entity(" Product added successfuly!<br> Id: "+id+"<br> Name: " + name+"
<br> Price: "+price)
            .build();
    }
}
```

File: web.xml

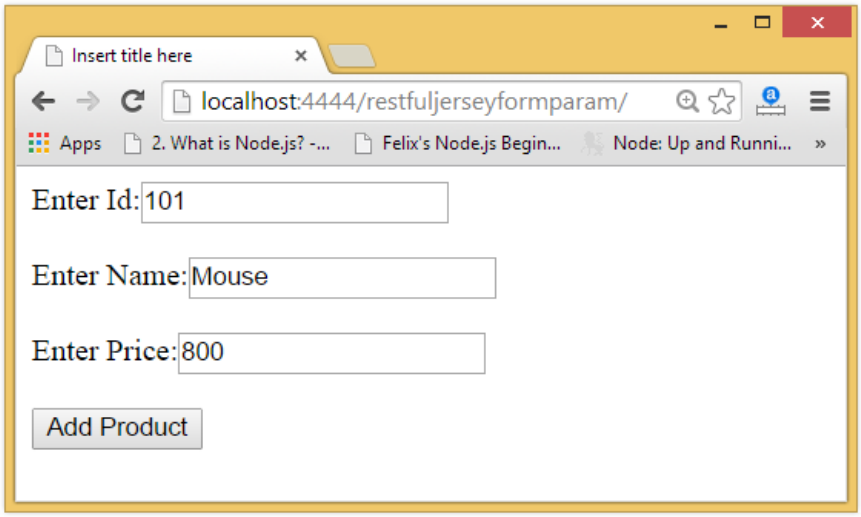
It is same as above example.

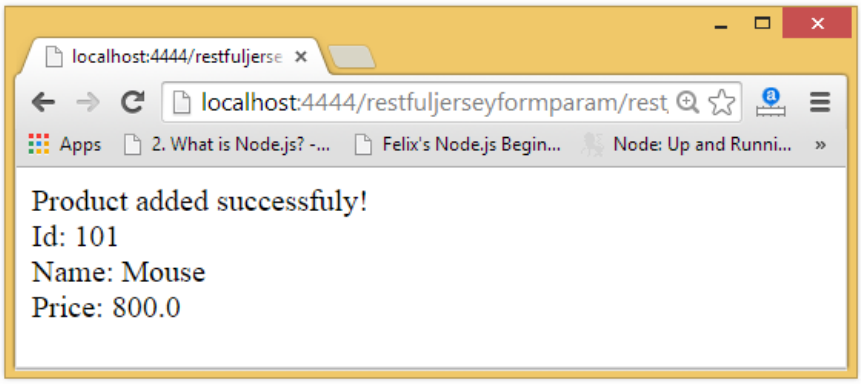
File: index.html

```
<form action="rest/product/add" method="post">
Enter Id:<input type="text" name="id"/><br/><br/>
Enter Name:<input type="text" name="name"/><br/><br/>
Enter Price:<input type="text" name="price"/><br/><br/>
<input type="submit" value="Add Product"/>
</form>
```

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

Output:





Click me to download this example

← prev

next →