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. |

**Part 3: RESTful Web Service - JAX-RS Annotations**

This tutorial is part 3 of 5-part tutorial on JEE annotations. We recommend that you read [Prerequisite](http://www.techferry.com/articles/JEE-annotations.html#Prerequisite) section first, review the [abstract](http://www.techferry.com/articles/JEE-annotations.html#abstract) and [Example Application](http://www.techferry.com/articles/JEE-annotations.html#exampleApp) to understand the context. You can also jump to other parts by clicking on the links below.

| **Annotation Tutorial: Contents** |
| --- |
| [JEE Annotations](http://www.techferry.com/articles/JEE-annotations.html)  ( [Abstract](http://www.techferry.com/articles/JEE-annotations.html#abstract)  [Example Application](http://www.techferry.com/articles/JEE-annotations.html#exampleApp)  [References](http://www.techferry.com/articles/JEE-annotations.html#references))   * [Part 1: Spring Annotations](http://www.techferry.com/articles/spring-annotations.html) * [Part 2: Hibernate - JPA Annotations](http://www.techferry.com/articles/hibernate-jpa-annotations.html) * [Part 3: RESTful Web Service - JAX-RS Annotations](http://www.techferry.com/articles/RESTful-web-services-JAX-RS-annotations.html) * [Part 4: JAXB Annotations](http://www.techferry.com/articles/jaxb-annotations.html) * [Part 5: Spring - jUnit Annotations](http://www.techferry.com/articles/spring-jUnit-annotations.html) |

**RESTful Web Service - JAX-RS Annotations - Contents:**

|  |  |
| --- | --- |
| **Annotation** | **Package Detail/Import statement** |
| [@GET](http://www.techferry.com/articles/RESTful-web-services-JAX-RS-annotations.html#GET) | import javax.ws.rs.GET; |
| [@Produces](http://www.techferry.com/articles/RESTful-web-services-JAX-RS-annotations.html#Produces) | import javax.ws.rs.Produces; |
| [@Path](http://www.techferry.com/articles/RESTful-web-services-JAX-RS-annotations.html#Path) | import javax.ws.rs.Path; |
| [@PathParam](http://www.techferry.com/articles/RESTful-web-services-JAX-RS-annotations.html#PathParam) | import javax.ws.rs.PathParam; |
| [@QueryParam](http://www.techferry.com/articles/RESTful-web-services-JAX-RS-annotations.html#QueryParam) | import javax.ws.rs.QueryParam; |
| [@POST](http://www.techferry.com/articles/RESTful-web-services-JAX-RS-annotations.html#POST) | import javax.ws.rs.POST; |
| [@Consumes](http://www.techferry.com/articles/RESTful-web-services-JAX-RS-annotations.html#Consumes) | import javax.ws.rs.Consumes; |
| [@FormParam](http://www.techferry.com/articles/RESTful-web-services-JAX-RS-annotations.html#FormParam) | import javax.ws.rs.FormParam; |
| [@PUT](http://www.techferry.com/articles/RESTful-web-services-JAX-RS-annotations.html#PUT) | import javax.ws.rs.PUT; |
| [@DELETE](http://www.techferry.com/articles/RESTful-web-services-JAX-RS-annotations.html#DELETE) | import javax.ws.rs.DELETE; |

As stated earlier in [Example Application](http://www.techferry.com/articles/JEE-annotations.html#exampleApp), we are using Jersey for RESTful Web services and JAX-RS annotations. 

|  |  |
| --- | --- |
| Jersey Annotation Tip | REST follows one-to-one mapping between create, read, update, and delete (CRUD) operations and HTTP methods.   * To create a resource on the server, use POST. * To retrieve a resource, use GET. * To change the state of a resource or to update it, use PUT. * To remove or delete a resource, use DELETE. |

**@GET**

Annotate your Get request methods with @GET.

|  |  |
| --- | --- |
| 1  2  3  4 | @GET  public String getHTML() {    ...  } |

**@Produces**

@Produces annotation specifies the type of output this method (or web service) will produce.

|  |  |
| --- | --- |
| 1  2  3  4  5 | @GET  @Produces("application/xml")  public Contact getXML() {    ...  } |
| 1  2  3  4  5 | @GET  @Produces("application/json")  public Contact getJSON() {    ...  } |

**@Path**

@Path annotation specify the URL path on which this method will be invoked.

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | @GET  @Produces("application/xml")  @Path("xml/{firstName}")  public Contact getXML() {    ...  } |

**@PathParam**

We can bind REST-style URL parameters to method arguments using @PathParam annotation as shown below.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | @GET  @Produces("application/xml")  @Path("xml/{firstName}")  public Contact getXML(@PathParam("firstName") String firstName) {    Contact contact = contactService.findByFirstName(firstName);    return contact;  } |
| 1  2  3  4  5  6  7 | @GET  @Produces("application/json")  @Path("json/{firstName}")  public Contact getJSON(@PathParam("firstName") String firstName) {    Contact contact = contactService.findByFirstName(firstName);    return contact;  } |

**@QueryParam**

Request parameters in query string can be accessed using @QueryParam annotation as shown below.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | @GET  @Produces("application/json")  @Path("json/companyList")  public CompanyList getJSON(@QueryParam("start") int start, @QueryParam("limit") int limit) {    CompanyList list = new CompanyList(companyService.listCompanies(start, limit));    return list;  } |

The example above returns a list of companies (with server side pagination) which can be displayed with rich clients implemented using Ext-js or jQuery. You can read more more about setting up [ExtJS grid panel with remote sorting and pagination using Hibernate](http://blog.techferry.com/2012/01/25/extjs-grid-panel-with-remote-sorting-and-pagination-using-hibernate/).

**@POST**

Annotate POST request methods with @POST.

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | @POST  @Consumes("application/json")  @Produces("application/json")  public RestResponse<Contact> create(Contact contact) {  ...  } |

**@Consumes**

The @Consumes annotation is used to specify the MIME media types a REST resource can consume.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | @PUT  @Consumes("application/json")  @Produces("application/json")  @Path("{contactId}")  public RestResponse<Contact> update(Contact contact) {  ...  } |

**@FormParam**

The REST resources will usually consume XML/JSON for the complete Entity Bean. Sometimes, you may want to read parameters sent in POST requests directly and you can do that using @FormParam annotation. GET Request query parameters can be accessed using [@QueryParam](http://www.techferry.com/articles/RESTful-web-services-JAX-RS-annotations.html#QueryParam) annotation.

|  |  |
| --- | --- |
| 1  2  3  4  5 | @POST  public String save(@FormParam("firstName") String firstName,      @FormParam("lastName") String lastName) {        ...    } |

**@PUT**

Annotate PUT request methods with @PUT.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | @PUT  @Consumes("application/json")  @Produces("application/json")  @Path("{contactId}")  public RestResponse<Contact> update(Contact contact) {  ...  } |

**@DELETE**

Annotate DELETE request methods with @DELETE.

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | @DELETE  @Produces("application/json")  @Path("{contactId}")  public RestResponse<Contact> delete(@PathParam("contactId") int contactId) {  ...  } |

**References**

1. [Jersey JAX-RS Annotations](https://wikis.oracle.com/display/Jersey/Overview+of+JAX-RS+1.0+Features): https://wikis.oracle.com/display/Jersey/Overview+of+JAX-RS+1.0+Features