**RestFull WebServices Using Jersey**

**Q :- RestFull Web Service ?**

Ans : REST is an web base architectural that used HTTP Protocol for Communication. REST is a stateless client-server architecture. In REST architecture there is a REST Server and Client, a REST Server simply provides access to resources and the REST client accesses and presents the resources.these resource may by xml,json,html etc… Here each resource is identified by URIs/ Global IDs.

### HTTP Methods

The following HTTP methods are most commonly used in a REST based architecture.

* **GET** − Provides a read only access to a resource.
* **PUT** − Used to create a new resource.
* **DELETE** − Used to remove a resource.
* **POST** − Used to update an existing resource or create a new resource.
* **OPTIONS** − Used to get the supported operations on a resource.

**Q : what is idempotent ?**

1. **Ans :** Idempotent operations means their result will always same no matter how many times these operations are invoked. if the method is called only once, or ten times over. The result should be the same. If you invoke a PUT API N times, very first request will update the resource; then rest N-1 requests will just overwrite the same resource state again and again – effectively not changing anything It essentially means that the result of a successful performed request is independent of the number of times. PUT and DELETE operations are idempotent , But POST is NOT idempotent.

### Q: Can we maintain user session in web services?

Ans : Web services are stateless so we can’t maintain user sessions in web services.

**Q: what is AcceptHeader ?**

Ans : Accept is a client-side header which tells the server what type of content you want back.like JSON,XML like if client sends Accept header as “application/xml” then XML response will be sent

**Q: What is WSDL ?**

Ans : WSDL stands for Web Service Description Language. WSDL is an XML based document that provides technical details about the web service. Some of the useful information in WSDL document are: method name, port types, service end point, binding, method parameters etc.

Q: what is Resource in REST ?

Ans: REST architecture treats every content as a resource. These resources can be text files, html pages, images, videos or dynamic business data. REST Server simply provides access to resources and REST client accesses and modifies the resources.

Q: -What are the core components of a HTTP Request?

Ans : A HTTP Request has five major parts −

* **Verb** − Indicate HTTP methods such as GET, POST, DELETE, PUT etc.
* **URI** − Uniform Resource Identifier (URI) to identify the resource on server.
* **HTTP Version** − Indicate HTTP version, for example HTTP v1.1 .
* **Request Header** − Contains metadata for the HTTP Request message as key-value pairs. For example, client ( or browser) type, format supported by client, format of message body, cache settings etc.
* **Request Body** − Message content or Resource representation.

Q : What are the core components of a HTTP response?

A HTTP Response has four major parts −

* **Status/Response Code −** Indicate Server status for the requested resource. For example 404 means resource not found and 200 means response is ok.
* **HTTP Version**− Indicate HTTP version, for example HTTP v1.1 .
* **Response Header −** Contains metadata for the HTTP Response message as key-value pairs. For example, content length, content type, response date, server type etc.
* **Response Body −** Response message content or Resource representation.

**Diff between** **@ResponseBody VS ResponseEntity : -**

* if you put @ResponseBody annotation in the method level, Spring will convert the return object in to the http response body.
* **ResponseEntity** works similar as @ResponseBody annotation. But when you create ResponseEntity object, you can add the response header to the http response as well.

**Q : - How to handle Exception in RestFull API Using Spring ?**

**Ans : there are three way to handle the exception**

* 1. **@ResponseEntity : -**This ResponseEntity  class takes two arguments, one is the returning object itself and other the status code.
  2. **@ResponseStatus : -**There is another way to handle this, using @ResponseStatus on the custom exception class. Here we have a custom exception class called CustomerNotFoundException . This annotation takes two arguments, one for value which defines the return HttpStatus code and another for reason.
* @ResponseStatus(value=HttpStatus.NOT\_FOUND, reason="No such Order") // 404
* public class OrderNotFoundException extends RuntimeException {
* // ...
* }

And here is a controller method using it:

@RequestMapping(value="/orders/{id}", method=GET)

public String showOrder(@PathVariable("id") long id, Model model) {

Order order = orderRepository.findOrderById(id);

if (order == null) throw new OrderNotFoundException(id);

model.addAttribute(order);

return "orderDetail";

}

**Security With RestFull : -**As RESTful Web Services work with HTTP URL Paths, it is very important to safeguard a RESTful Web Service

You can secure your RESTful Web services using one of the following methods to support authentication, authorization, or encryption:

* 1. Updating the **web.xml** deployment descriptor to define security configuration. See [Securing RESTful Web Services Using web.xml.](https://docs.oracle.com/cd/E24329_01/web.1211/e24983/secure.htm#BABDGCJD)
  2. Applying annotations to your JAX-RS classes. See [Securing RESTful Web Services Using Annotations.](https://docs.oracle.com/cd/E24329_01/web.1211/e24983/secure.htm#BABGBJAC).

The **javax.annotation.security** package provides annotations, defined in Table that you can use to secure your RESTful Web services



Example:

package samples.helloworld;

import javax.ws.rs.GET;

import javax.ws.rs.Path;

import javax.ws.rs.Produces;

**import javax.annotation.Security.RolesAllowed;**

@Path("/helloworld")

**@RolesAllowed({"ADMIN", "ORG1"})**

public class helloWorld {

@GET

@Path("sayHello")

@Produces("text/plain")

@**RolesAllows("ADMIN")**

public String sayHello() {

return "Hello World!";

}

}

Q: pagination to get large amount of data using REST :

Ans: http://localhost:8080/confluence/rest/api/page?limit=5&start=5"