Interview Q&A on Restful Web Services

-By Sivareddy

1) ***What are restful web services and which framework you used to develop Restful Web services?***
   - REST (Representational State Transfer)
   - REST is an architectural style for designing distributed systems
   - It is not a standard but a set of constraints, such as being stateless, having a client/server relationship, and a uniform interface
   - REST works on HTTP Methods like GET, POST, PUT, DELETE
   - REST Provides status code for each HTTP request like 200, 400,404 etc.

   REST Principles:
   - **Resources** expose easily understood directory structure URIs.
   - **Representations** transfer JSON or XML to represent data objects and attributes.
   - **Messages** use HTTP methods explicitly (for example, GET, POST, PUT, and DELETE).
   - **Stateless** interactions store no client context on the server between requests. State dependencies limit and restrict scalability. The client holds session state.

   JAX-RS – Jersey restful Web services
2) ***What are different methods available in Restful web services?***
   `GET`, `POST`, `PUT`, `DELETE`, `HEAD`, PATCH,
3) ***What is idempotent and which HTTP methods are idempotent in Restful web services?***
   An idempotent HTTP method is a HTTP method that can be called many times without different outcomes. If you follow the REST design principles, then by default
   Idempotent methods:**GET, PUT, DELETE, HEAD, OPTIONS and TRACE HTTP**
   Non-Idempotent Method**- POST**
4) ***What is different between POST vs. PUT?***
   POST is used to create resource and it is non-idempotent HTTP method
   PUT is used to update/modify the resource and it is idempotent HTTP method
5) ***What is URI and how do you define in URI for restful web services?***
   REST APIs use Uniform Resource Identifiers (URIs) to address resources
   Below is the Syntax to define the URI:

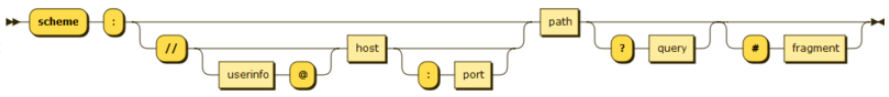
The *URI generic syntax* consists of a hierarchical sequence of five *components*:[8]

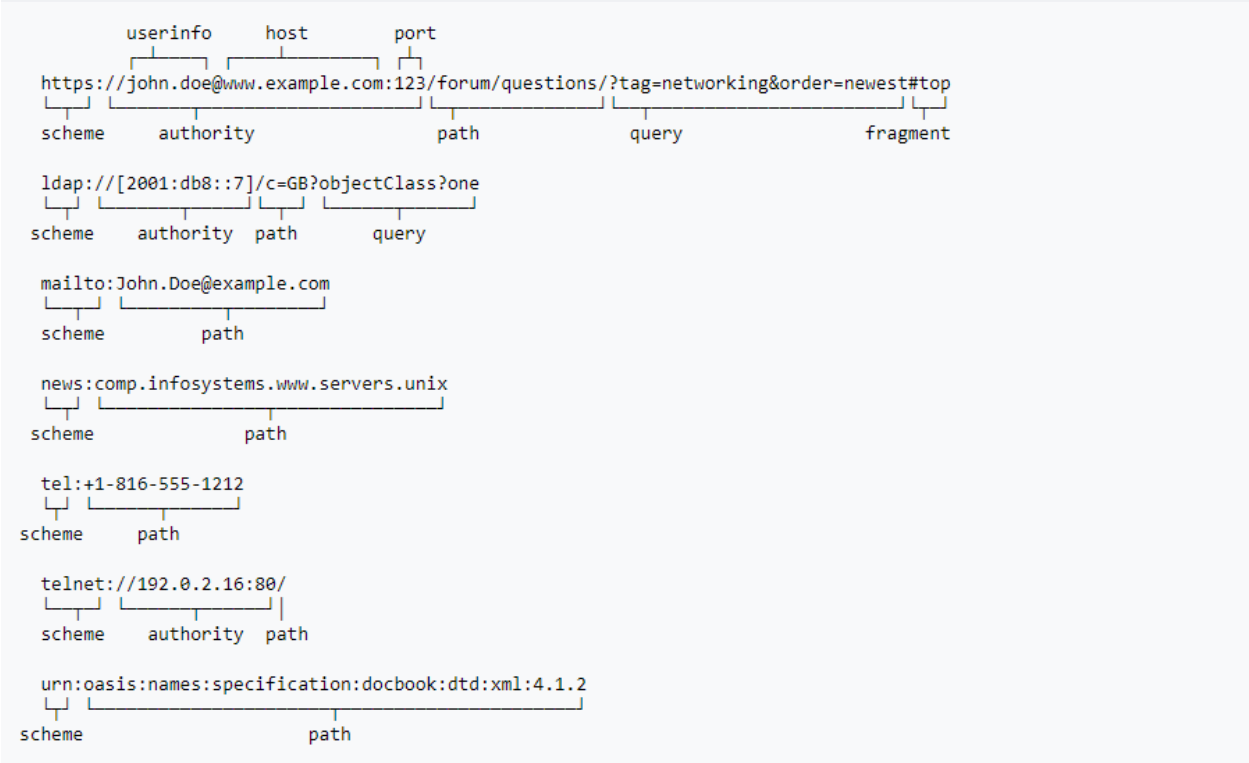
```
URI = scheme:[//authority]path[?query][#fragment]
```

where the authority component divides into three *subcomponents*:

```
authority = [userinfo@]host[:port]
```

This is represented in a syntax diagram as:





Source: Wiki

6) ***How to upload document using Restful Web Services?***

```
@POST
@Consumes(MediaType.MULTIPART_FORM_DATA)
public Response uploadFile(
@FormDataParam("file") InputStream uploadedInputStream,
@FormDataParam("file") FormDataContentDisposition fileDetail) {
-----
}
```

7) ***What is the difference between @QueryParam and @PathParam?***

@QueryParam - Binds the value(s) of a HTTP query parameter to a resource method parameter, resource class field, or resource class bean property.

URI : users/query?from=100

```
@Path("/users")
publicclassUserService{

    @GET
    @Path("/query")
    publicResponse getUsers(
    @QueryParam("from") int from){
  }}
```

@PathParam - Binds the value of a URI template parameter or a path segment containing the template parameter to a resource method parameter, resource class field, or resource class bean property.

```
@Path("/users/{username}")
public class UserResource {

    @GET
    @Produces("text/xml")
    public String getUser(@PathParam("username") String userName) {
       ...
    }
  }
```

8) What is difference between @Produces vs. @Consumes?

The @Produces annotation is used to specify the MIME media types or representations a resource can produce and send back to the client

The value of @Produces is an array of String of MIME types. For example: @Produces({"image/jpeg,image/png"})

```
@Path("/myResource")
@Produces("text/plain")
public class SomeResource {
        @GET
        public String doGetAsPlainText() {
                ...
        }
        @GET
        @Produces("text/html")
        public String doGetAsHtml() {
                ...
        }
}
```

The @Consumes annotation is used to specify which MIME media types of representations a resource can accept, or consume, from the client.

The value of @Consumes is an array of String of acceptable MIME types. For example:

```
@Consumes({"text/plain,text/html"})
```

The following example shows how to apply @Consumes at both the class and method levels:

```
@Path("/myResource")
@Consumes("multipart/related")
public class SomeResource {
   @POST
   public String doPost(MimeMultipart mimeMultipartData) {                    ...
   }

   @POST
   @Consumes("application/x-www-form-urlencoded")
   public String doPost2(FormURLEncodedProperties formData) {
                  ...
   }
}
```

9) What are the annotations you used in restful web services?

@GET,@Produces,@Path,@PathParam,@QueryParam,@POST,@Consumes,@FormParam, @PUT,@DELETE

```
@GET
@Produces("application/json")
@Consumes("application/json")
@Path("json/employeeList")
public EmployeeList getJSON(@PathParam("firstName") String firstName,
@QueryParam("start") int start, @QueryParam("limit") int limit) {
 EmployeeList list = new EmployeeList(EmployeeService.listEmployees(start, limit));
 return list;
}

@POST
@Consumes("application/json")
@Produces("application/json")
public RestResponse<Employee> create(Employee employee) {
...
}

@PUT
@Consumes("application/json")
@Produces("application/json")
@Path("{empID}")
public RestResponse<Employee> update(@PathParam("empId") int empId, Employee
employee) {
...
```

}

10) What is HATEOAS in REST?

- **HATEOAS -** Hypermedia As The Engine Of Application State
- With HATEOAS, a client interacts with a network application whose application servers provide information dynamically through hypermedia
- The way that the HATEOAS constraint decouples client and server enables the server functionality to evolve independently.

```
GET/accounts/12345/HTTP/1.1
Host: bank.example.com
Accept: application/xml

Response 1:

HTTP/1.1200OK
Content-Type:application/xml
Content-Length:...

<?xml version="1.0"?>
<account>
<account_number>12345</account_number>
<balancecurrency="usd">100.00</balance>
<linkrel="deposit"href="/accounts/12345/deposit"/>
<linkrel="withdraw"href="/accounts/12345/withdraw"/>
<linkrel="transfer"href="/accounts/12345/transfer"/>
<linkrel="close"href="/accounts/12345/close"/>
</account>

Response 2:

HTTP/1.1200OK
Content-Type:application/xml
Content-Length:...

<?xml version="1.0"?>
<account>
<account_number>12345</account_number>
<balancecurrency="usd">-25.00</balance>
<linkrel="deposit"href="/accounts/12345/deposit"/>
</account>
```

11) How do you test your REST APIs?

To test REST APIs, we have multiple REST API clients like Postman, SoapUI, Advanced Rest Client etc.

12) Which framework do you use to write integration test cases?

Rest Assured, Cucumber, Spring Test

13) What is the importance of HTTP status codes in REST?

Status codes indicate the result of the HTTP request.

1XX – informational - 100 Continue, 102 Processing, etc..

2XX – success - 200 OK, 201 Created,202 Accepted etc..

3XX – redirection - 301 Moved Permanently, 303 See Other etc..

4XX - client error

5XX - server error

14) How to secure Restful web services?

1) Basic Authentication: credentials will be encoded in Base 64
2) HMAC - digest = base64encode(hmac("sha256", "secret", "GET+/users/username/account"))", "secret", "GET+/users/username/account"))
3) OAuth/SSO