**RESTful Web Services**

REpresentational state transfer, or REST, is a set of architectural principles with which you can design web services.

In the REST architectural style, data and functionality = resources

and are accessed using Uniform Resource Identifiers (URIs), typically links on the Web.

Resources can be pictures, video files, Web pages, business information, or anything that can be represented in a computer-based system.

The focus of a RESTful service is on resources and how to provide access to these resources.

A resource can easily be thought of as an object as in OOP.

Uses the HTTP headers to hold meta information (although it is protocol-agnostic)

Can be used with XML, JSON or whatever necessary

Usually used with JSON due to the easily parsable content

REST only supports SSL security.

WADL (Web Application Description Language) is a XML description of a deployed RESTful web application.

RESTful web services are light weight and faster than SOAP

**Create REST Based Web Services**

1. Java API for RESTful Web Services (**JAX-RS**), is a set of APIs to developer REST service.

JAX-RS is part of the Java EE6, and make developers to develop REST web application easily

1. Using **Jersey**

**Exception Handling**

1. Using @ResponseEntity and HttpStatus codes
2. Using @ResponseStatus on the custom exception class
3. Using custom method to handle error on the controller (@ExceptionHandler and @ResponseStatus).

Return error representation instead of default html error page

**Consume REST Services** - How do you consume a restful web service?

Clients

1. **Net Client**

While post need to use OutputStream

URL

HttpURLConnection

setRequestMethod

setRequestProperty

1. **Jersey Client**

**Testing -** Use SOAPUI or POSTMASTER

**GET**

Retrieve information. GET requests must be safe and **idempotent**, meaning regardless of how many times it repeats with the same parameters, the results are the same.

Retrieve an address with an ID of 1:

GET /addresses/1

**POST**

Request that the resource at the URI do something with the provided entity.

Often POST is used to create a new entity, but it can also be used to update an entity.

Create a new address:

POST /addresses

**PUT**

Store an entity at a URI. PUT can create a new entity or update an existing one.

**PUT is idempotent, while POST is not. It means if use PUT an object twice, it has no effect.**

Modify the address with an ID of 1:

PUT /addresses/1

**Securing RestFul Services**

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

1. Securing RESTful Web Services Using web.xml
2. Securing RESTful Web Services Using SecurityContext
3. Securing RESTful Web Services Using Annotations

DeclareRoles, PermitAll, DenyAll, RolesAllowed