**REST & RESTFul Web Services**

**What is REST?**

REST is an architectural style which is based on web standards and HTTP protocol.

It’s a style/pattern to implement Web Services.

In REST architecture, everything on the server is considered as resource and access to this resource is supported by a common interface based on HTTP standard methods.

REST architecture has a REST server that provides the resources and the REST Client accesses and modifies these resources.

All resources should support the HTTP common operations. Resources are identified by the global ID (URI).

REST allows that resources have different representations like text, xml, html, JSON etc. REST Clients can ask for specific representation via the HTTP protocol.

**What is a Web Service?**

Web Service is basically a method of communication between two electronic devices over the network.

It is a software function provided at a network address over the web with the service always running.

W3C defines Web Service as a software system designed to support interoperable machine to machine interaction over a network.

Many organizations use multiple software systems for management. Different software systems often need to exchange data with each other, and a Web service is a method of communication that allows two software systems to exchange this data over the internet. The software system that requests data is called a *service requester*, whereas the software system that would process the request and provide the data is called a *service provider*.

Different software might be built using different programming languages, and hence there is a need for a method of data exchange that doesn't depend upon a particular programming language. Most types of software can, however, interpret XML tags. Thus, Web services can use XML files for data exchange.

Rules for communication between different systems need to be defined, such as:

* How one system can request data from another system
* Which specific parameters are needed in the data request
* What would be the structure of the data produced? Normally, data is exchanged in XML files, and the structure of the XML file is validated against an .xsd file.
* What error messages to display when a certain rule for communication is not observed, to make troubleshooting easier

All of these rules for communication are defined in a file called WSDL (Web Services Description Language), which has the extension .wsdl.

A directory called UDDI (Universal Description, Discovery and Integration) defines which software system should be contacted for which type of data. So when one software system needs one particular report/data, it would go to the UDDI and find out which other system it can contact for receiving that data. Once the software system finds out which other system it should contact, it would then contact that system using a special protocol called SOAP (Simple Object Access Protocol). The service provider system would first of all validate the data request by referring to the WSDL file, and then process the request and send the data under the SOAP protocol.

**What is RESTful Web Service?**

Web Service which is implemented using the REST architectural style that is based on HTTP methods is called RESTful Web Service.

RESTful Web Service defines the base URI for the services, the supported MIME types (xml, html, text, JSON etc.) and the set of operations (PUT, POST, DELETE, GET) which are supported.

***The primary purpose of RESTful web service is it provides to manipulate representations of web resources using a uniform set of stateless operations.***

**HTTP Methods**

The PUT, GET, POST and DELETE methods are typical used in REST based architectures. The following table gives an explanation of these operations.

* GET defines a reading access of the resource without side-effects. The resource is never changed via a GET request, e.g., the request has no side effects (idempotent).
* PUT creates a new resource. It must also be idempotent.
* DELETE removes the resources. The operations are idempotent. They can get repeated without leading to different results.
* POST updates an existing resource or creates a new resource.

JAX-RS with Jersey

Java provides support for REST via the Java Specification Request (JSR) 311. This specification is called JAX-RS (Java API for Restful Web Service).JAX-RS makes use of annotations to mark REST relevance to java classes.

**Jersey**

Jersey is a reference implementation of the JSR 311 specification. Jersey provides library to implement the Restful Web service in a Java Servlet container.

On the server side, Jersey provides a servlet implementation that scans predefined classes to identify restful resources. In the web.xml file we register the servlet for our application.

Jersey also provides a Client Library to communicate with a RESTful web service.

The base URL of this servlet is:

http://your\_domain:port/display-name/url-pattern/path\_from\_rest\_class

This servlet analyzes the incoming HTTP request and selects the correct class and method to respond to this request. This selection is based on annotations in the class and methods.

JAX-RS supports the creation of XML and JSON via the Java Architecture for XML Binding (JAXB).

**JAX-RS annotations**

The most important annotations in JAX-RS are listed in the following table.

| **Annotation** | **Description** |
| --- | --- |
| @PATH(your\_path) | Sets the path to base URL + /your\_path. The base URL is based on your application name, the servlet and the URL pattern from the web.xml configuration file. |
| @POST | Indicates that the following method will answer to an HTTP POST request. |
| @GET | Indicates that the following method will answer to an HTTP GET request. |
| @PUT | Indicates that the following method will answer to an HTTP PUT request. |
| @DELETE | Indicates that the following method will answer to an HTTP DELETE request. |
| @Produces(MediaType.TEXT\_PLAIN[, more-types]) | @Produces defines which MIME type is delivered by a method annotated with @GET. In the example text ("text/plain") is produced. Other examples would be "application/xml" or "application/json". |
| @Consumes(type[, more-types]) | @Consumes defines which MIME type is consumed by this method. |
| @PathParam | Used to inject values from the URL into a method parameter. This way you inject, for example, the ID of a resource into the method to get the correct object. |