**What is a Web Service**

In simple words, services that can be accessed over network are called web services. So how does it differ from web application, they are also services that are accessed over network. There are few attributes that clarifies this difference.

* Web applications are meant for users and to be accessed in browser having human readable format whereas web services are meant for applications to access data in the format of XML, JSON etc.
* Web applications always use HTTP/HTTPS protocol whereas traditional web services use SOAP protocol. Recently REST is getting popularity that is an architecture style and almost all times run on HTTP/HTTPS protocol.
* Web applications are not meant for reusability whereas this is one of the benefit of web services. A single web service can be used by different kinds of applications.
* Web application can access web services to access some data or to perform some tasks, web services can’t access web applications to fetch some data.
* Web applications are capable to maintain user session, web services are stateless.

I hope above differences are good enough to clear any confusion with web applications and web services. Both are different concepts and meant for different purpose.

**Types of Web Services**

There are two types of web services.

1. SOAP: SOAP stands for Simple Object Access Protocol. SOAP is an XML based industry standard protocol for designing and developing web services. Since it’s XML based, it’s platform and language independent. So our server can be based on JAVA and client can be on .NET, PHP etc. and vice versa.
2. REST: REST is an architectural style for developing web services. It’s getting popularity recently because it has small learning curve when compared to SOAP. Resources are core concepts of Restful web services and they are uniquely identified by their URIs.

**Java Web Services**

Java provides it’s own API to create both SOAP as well as REST web services.

1. JAX-WS: JAX-WS stands for Java API for XML Web Services. JAX-WS is XML based Java API to build web services server and client application.
2. JAX-RS: Java API for RESTful Web Services (JAX-RS) is the Java API for creating REST web services. JAX-RS uses annotations to simplify the development and deployment of web services.

Both of these APIs are part of standard JDK installation, so we don’t need to add any jars to work with them. Both of these APIs use annotations very heavily.

1. **What is a Web Service?**

Web Services work on client-server model where client applications can access web services over the network. Web services provide endpoint URLs and expose methods that can be accessed over network through client programs written in java, shell script or any other different technologies.  
Web services are stateless and doesn’t maintain user session like web applications.

1. **What are the advantages of Web Services?**

Some of the advantages of web services are:

* + Interoperability: Web services are accessible over network and runs on HTTP/SOAP protocol and uses XML/JSON to transport data, hence it can be developed in any programming language. Web service can be written in java programming and client can be PHP and vice versa.
  + Reusability: One web service can be used by many client applications at the same time.
  + Loose Coupling: Web services client code is totally independent with server code, so we have achieved loose coupling in our application.
  + Easy to deploy and integrate, just like web applications.
  + Multiple service versions can be running at same time.

1. **What are different types of Web Services?**

There are two types of web services:

* + SOAP Web Services: Runs on SOAP protocol and uses XML technology for sending data.
  + Restful Web Services: It’s an architectural style and runs on HTTP/HTTPS protocol almost all the time. REST is a stateless client-server architecture where web services are resources and can be identified by their URIs. Client applications can use HTTP GET/POST methods to invoke Restful web services.

1. **What is SOAP?**

SOAP stands for Simple Object Access Protocol. SOAP is an XML based industry standard protocol for designing and developing web services. Since it’s XML based, it’s platform and language independent. So our server can be based on JAVA and client can be on .NET, PHP etc. and vice versa.

1. **What are advantages of SOAP Web Services?**

SOAP web services have all the advantages that web services has, some of the additional advantages are:

* + WSDL document provides contract and technical details of the web services for client applications without exposing the underlying implementation technologies.
  + SOAP uses XML data for payload as well as contract, so it can be easily read by any technology.
  + SOAP protocol is universally accepted, so it’s an industry standard approach with many easily available open source implementations.

1. **What are disadvantages of SOAP Web Services?**

Some of the disadvantages of SOAP protocol are:

* + Only XML can be used, JSON and other lightweight formats are not supported.
  + SOAP is based on the contract, so there is a tight coupling between client and server applications.
  + SOAP is slow because payload is large for a simple string message, since it uses XML format.
  + Anytime there is change in the server side contract, client stub classes need to be generated again.
  + Can’t be tested easily in browser

1. **What is WSDL?**

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.

1. **What are different components of WSDL?**

Some of the different tags in WSDL xml are:

* + xsd:import namespace and schemaLocation: provides WSDL URL and unique namespace for web service.
  + message: for method arguments
  + part: for method argument name and type
  + portType: service name, there can be multiple services in a wsdl document.
  + operation: contains method name
  + soap:address for endpoint URL.

1. **What is UDDI?**

UDDI is acronym for Universal Description, Discovery and Integration. UDDI is a directory of web services where client applications can lookup for web services. Web Services can register to the UDDI server and make them available to client applications.

1. **What is difference between Top Down and Bottom Up approach in SOAP Web Services?**

In Top Down approach first WSDL document is created to establish the contract between web service and client and then code is written, it’s also termed as contract first approach. This is hard to implement because classes need to be written to confirm the contract established in WSDL. Benefit of this approach is that both client and server code can be written in parallel.

In Bottom Up approach, first web service code is written and then WSDL is generated. It’s also termed as contract last approach. This approach is easy to implement because WSDL is generated based on code. In this approach client code have to wait for WSDL from server side to start their work.

1. **What is REST Web Services?**

REST is the acronym for REpresentational State Transfer. REST is an architectural style for developing applications that can be accessed over the network. REST architectural style was brought in light by Roy Fielding in his doctoral thesis in 2000.

REST is a stateless client-server architecture where web services are resources and can be identified by their URIs. Client applications can use HTTP GET/POST methods to invoke Restful web services. REST doesn’t specify any specific protocol to use, but in almost all cases it’s used over HTTP/HTTPS. When compared to SOAP web services, these are lightweight and doesn’t follow any standard. We can use XML, JSON, text or any other type of data for request and response.

1. **What are advantages of REST web services?**

Some of the advantages of REST web services are:

* + Learning curve is easy since it works on HTTP protocol
  + Supports multiple technologies for data transfer such as text, xml, json, image etc.
  + No contract defined between server and client, so loosely coupled implementation.
  + REST is a lightweight protocol
  + REST methods can be tested easily over browser.

1. **What are disadvantages of REST web services?**

Some of the disadvantages of REST are:

* + Since there is no contract defined between service and client, it has to be communicated through other means such as documentation or emails.
  + Since it works on HTTP, there can’t be asynchronous calls.
  + Sessions can’t be maintained.

1. **What is a Resource in Restful web services?**

Resource is the fundamental concept of Restful architecture. A resource is an object with a type, relationship with other resources and methods that operate on it. Resources are identified with their URI, HTTP methods they support and request/response data type and format of data.

1. **What are different HTTP Methods supported in Restful Web Services?**

Restful web services supported HTTP methods are – GET, POST, PUT, DELETE and HEAD.

1. **Compare SOAP and REST web services?**

|  |  |
| --- | --- |
| SOAP | REST |
| SOAP is a standard protocol for creating web services. | REST is an architectural style to create web services. |
| SOAP is acronym for Simple Object Access Protocol. | REST is acronym for REpresentational State Transfer. |
| SOAP uses WSDL to expose supported methods and technical details. | REST exposes methods through URIs, there are no technical details. |
| SOAP web services and client programs are bind with WSDL contract | REST doesn’t have any contract defined between server and client |
| SOAP web services and client are tightly coupled with contract. | REST web services are loosely coupled. |
| SOAP learning curve is hard, requires us to learn about WSDL generation, client stubs creation etc. | REST learning curve is simple, POJO classes can be generated easily and works on simple HTTP methods. |
| SOAP supports XML data format only | REST supports any data type such as XML, JSON, image etc. |
| SOAP web services are hard to maintain, any change in WSDL contract requires us to create client stubs again and then make changes to client code. | REST web services are easy to maintain when compared to SOAP, a new method can be added without any change at client side for existing resources. |
| SOAP web services can be tested through programs or software such as Soap UI. | REST can be easily tested through CURL command, Browsers and extensions such as Chrome Postman. |

1. **What are different ways to test web services?**

SOAP web services can be tested programmatically by generating client stubs from WSDL or through software such as Soap UI.

REST web services can be tested easily with program, curl commands and through browser extensions. Resources supporting GET method can be tested with browser itself, without any program.

1. **Can we maintain user session in web services?**

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

1. **What is difference between SOA and Web Services?**

Service Oriented Architecture (SOA) is an architectural pattern where applications are designed in terms of services that can be accessed through communication protocol over network. SOA is a [design pattern](https://www.journaldev.com/1827/java-design-patterns-example-tutorial) and doesn’t go into implementation.

Web Services can be thought of as Services in SOAP architecture and providing means to implement SOA pattern.

1. **What is the use of Accept and Content-Type Headers in HTTP Request?**

These are important headers in Restful web services. Accept headers tells web service what kind of response client is accepting, so if a web service is capable of sending response in XML and JSON format and client sends Accept header as “application/xml” then XML response will be sent. For Accept header “application/json”, server will send the JSON response.

Content-Type header is used to tell server what is the format of data being sent in the request. If Content-Type header is “application/xml” then server will try to parse it as XML data. This header is useful in HTTP Post and Put requests.

1. **How would you choose between SOAP and REST web services?**

Web Services work on client-server model and when it comes to choose between SOAP and REST, it all depends on project requirements. Let’s look at some of the conditions affecting our choice:

* + Do you know your web service clients beforehand? If Yes, then you can define a contract before implementation and SOAP seems better choice. But if you don’t then REST seems better choice because you can provide sample request/response and test cases easily for client applications to use later on.
  + How much time you have? For quick implementation REST is the best choice. You can create web service easily, test it through browser/curl and get ready for your clients.
  + What kind of data format are supported? If only XML then you can go with SOAP but if you think about supporting JSON also in future then go with REST.

1. **What is JAX-WS API?**

JAX-WS stands for Java API for XML Web Services. JAX-WS is XML based Java API to build web services server and client application. It’s part of standard Java API, so we don’t need to include anything else which working with it. Refer to [JAX-WS Tutorial](https://www.journaldev.com/9123/jax-ws-tutorial) for a complete example.

1. **Name some frameworks in Java to implement SOAP web services?**

We can create SOAP web services using JAX-WS API, however some of the other frameworks that can be used are Apache Axis and Apache CXF. Note that they are not implementations of JAX-WS API, they are totally different framework that work on Servlet model to expose your business logic classes as SOAP web services. Read more at [Java SOAP Web Service Eclipse](https://www.journaldev.com/9131/soap-webservices-in-java-example-eclipse) example.

1. **Name important annotations used in JAX-WS API?**

Some of the important annotations used in JAX-WS API are:

* + @WebService
  + @SOAPBinding
  + @WebMethod

1. **What is use of javax.xml.ws.Endpoint class?**

Endpoint class provides useful methods to create endpoint and publish existing implementation as web service. This comes handy in testing web services before making further changes to deploy it on actual server.

1. **How to get WSDL file of a SOAP web service?**

WSDL document can be accessed by appending ?wsdl to the SOAP endoint URL. In above example, we can access it at http://localhost:8888/testWS?wsdl location.

1. **What is sun-jaxws.xml file?**

This file is used to provide endpoints details when JAX-WS web services are deployed in servlet container such as Tomcat. This file is present in WEB-INF directory and contains endpoint name, implementation class and URL pattern. For example;

sun-jaxws.xml

<?xml version="1.0" encoding="UTF-8"?>

<endpoints xmlns="http://java.sun.com/xml/ns/jax-ws/ri/runtime" version="2.0">

<endpoint

name="PersonServiceImpl"

implementation="com.journaldev.jaxws.service.PersonServiceImpl"

url-pattern="/personWS"/>

</endpoints>

1. **What is JAX-RS API?**

Java API for RESTful Web Services (JAX-RS) is the Java API for creating REST web services. JAX-RS uses annotations to simplify the development and deployment of web services. JAX-RS is part of JDK, so you don’t need to include anything to use it’s annotations.

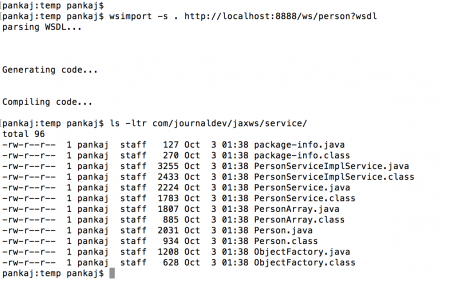
1. **Name some implementations of JAX-RS API?**

There are two major implementations of JAX-RS API.

* 1. Jersey: Jersey is the reference implementation provided by Sun. For using Jersey as our JAX-RS implementation, all we need to configure its servlet in web.xml and add required dependencies. Note that JAX-RS API is part of JDK not Jersey, so we have to add its dependency jars in our application.
  2. RESTEasy: RESTEasy is the JBoss project that provides JAX-RS implementation.

1. **What is wsimport utility?**

We can use wsimport utility to generate the client stubs. This utility comes with standard installation of JDK. Below image shows an example execution of this utility for one of JAX-WS project.

[](https://cdn.journaldev.com/wp-content/uploads/2015/10/wsimport-utility-parse-wsdl.png)

1. **Name important annotations used in JAX-RS API?**

Some of the important JAX-RS annotations are:

* 1. @Path: used to specify the relative path of class and methods. We can get the URI of a webservice by scanning the Path annotation value.
  2. @GET, @PUT, @POST, @DELETE and @HEAD: used to specify the HTTP request type for a method.
  3. @Produces, @Consumes: used to specify the request and response types.
  4. @PathParam: used to bind the method parameter to path value by parsing it.

1. **What is the use of @XmlRootElement annotation?**

XmlRootElement annotation is used by JAXB to transform java object to XML and vice versa. So we have to annotate model classes with this annotation.

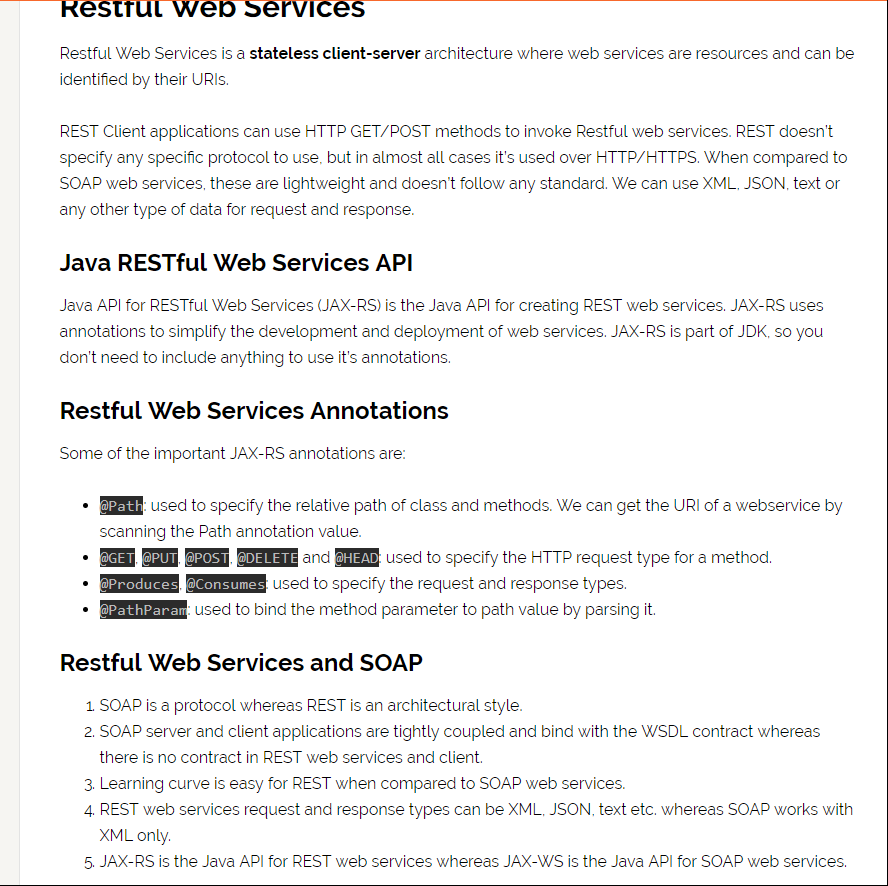
1. **How to set different status code in HTTP response?**

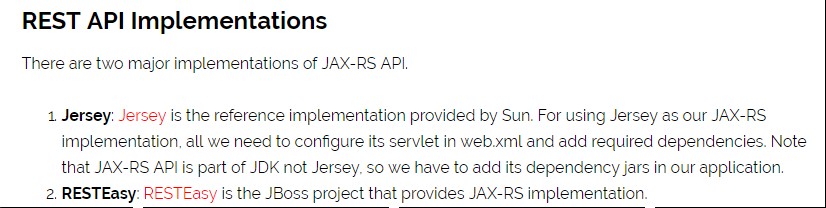
For setting HTTP status code other than 200, we have to use javax.ws.rs.core.Response class for response. Below are some of the sample return statements showing it’s usage.

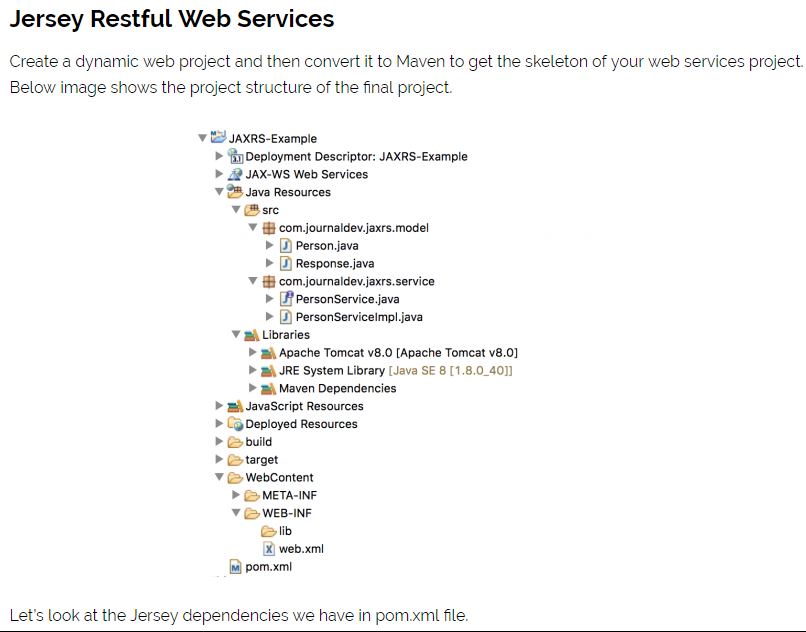
return Response.status(422).entity(exception).build();

return Response.ok(response).build(); //200

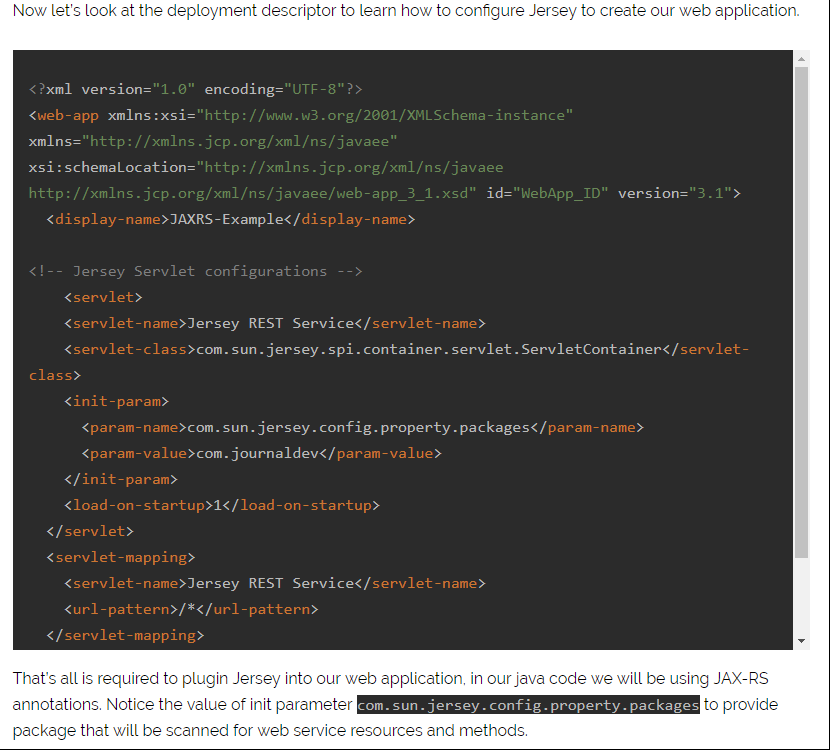
For a complete example, please read [RESTEasy Tutorial](https://www.journaldev.com/9189/resteasy-tutorial-eclipse-tomcat).

















@Path("/person")

@Consumes(MediaType.APPLICATION\_XML)

@Produces(MediaType.APPLICATION\_XML)

public class PersonServiceImpl implements PersonService {

private static Map<Integer,Person> persons = new HashMap<Integer,Person>();

@Override

@POST

@Path("/add")

public Response addPerson(Person p) {

Response response = new Response();

if(persons.get(p.getId()) != null){

response.setStatus(false);

response.setMessage("Person Already Exists");

return response;

}

persons.put(p.getId(), p);

response.setStatus(true);

response.setMessage("Person created successfully");

return response;

}

@Override

@GET

@Path("/{id}/delete")

public Response deletePerson(@PathParam("id") int id) {

Response response = new Response();

if(persons.get(id) == null){

response.setStatus(false);

response.setMessage("Person Doesn't Exists");

return response;

}

persons.remove(id);

response.setStatus(true);

response.setMessage("Person deleted successfully");

return response;

}

@Override

@GET

@Path("/{id}/get")

public Person getPerson(@PathParam("id") int id) {

return persons.get(id);

}

@GET

@Path("/{id}/getDummy")

public Person getDummyPerson(@PathParam("id") int id) {

Person p = new Person();

p.setAge(99);

p.setName("Dummy");

p.setId(id);

return p;

}

@Override

@GET

@Path("/getAll")

public Person[] getAllPersons() {

Set<Integer> ids = persons.keySet();

Person[] p = new Person[ids.size()];

int i=0;

for(Integer id : ids){

p[i] = persons.get(id);

i++;

}

return p;

}

}