**WebService**

* The standard protocol which makes the communication between the request generator and the request provider is nothing but webservice. E.g. Makemytrip site. Exposed webservices.
* A web service is used to communicate among various applications by using open standards such as HTML, XML, WSDL, and SOAP.( Platform independent communication)

**Basic Working of Webservice:**

* Service provider will develop webservices and make it available on internet : server
* Some who consumes the webservices : Client

2 basic things required for communication:

1. Medium : Http/Internet
2. Format: HTML, JSON, XML and WSDL.

**WSDL and UDDI:**

* **W**eb **S**ervice **D**escription **L**anguage :

An interface which will provide the description of all the attributes of the web services. (XML based for machine readability)

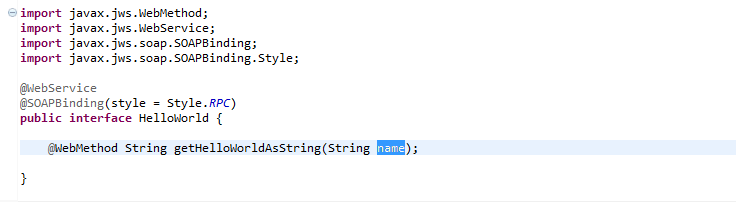
* If the client is known to server, the wsdl is directly handed over to the clients and they can use it.
* In other case, there is online registry or directory which will publish the wsdl for the service provider and those would be **UDDI (U**niversal **D**escription **D**iscovery **and I**ntegration**).**
* Query and get hold of WSDL and then use the webservice.

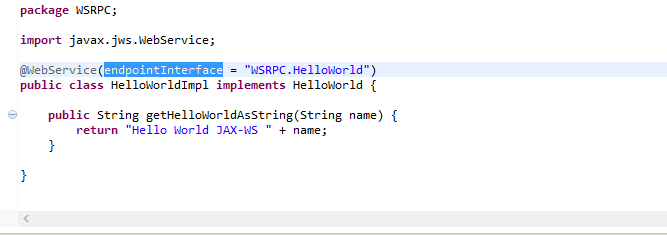
**SOAP :**

* **S**imple **O**bject **A**ccess **P**rotocol basically consist of 3 main things :
  1. **Envelope:** root element of SOAP message which will consist of header and body.
  2. **Header:** Provides the information of the message itself (Authentication, routing information). It is optional to define a header.
  3. **Body:** Contains information to be provided.

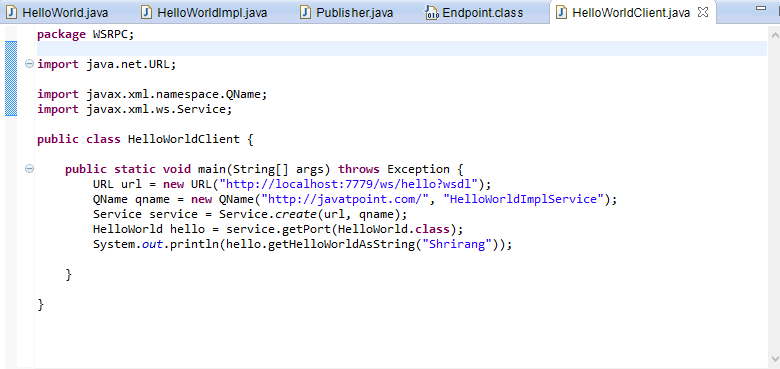
JAX\_WS\_RPC Example

* 3 Server side files and 1 client side files





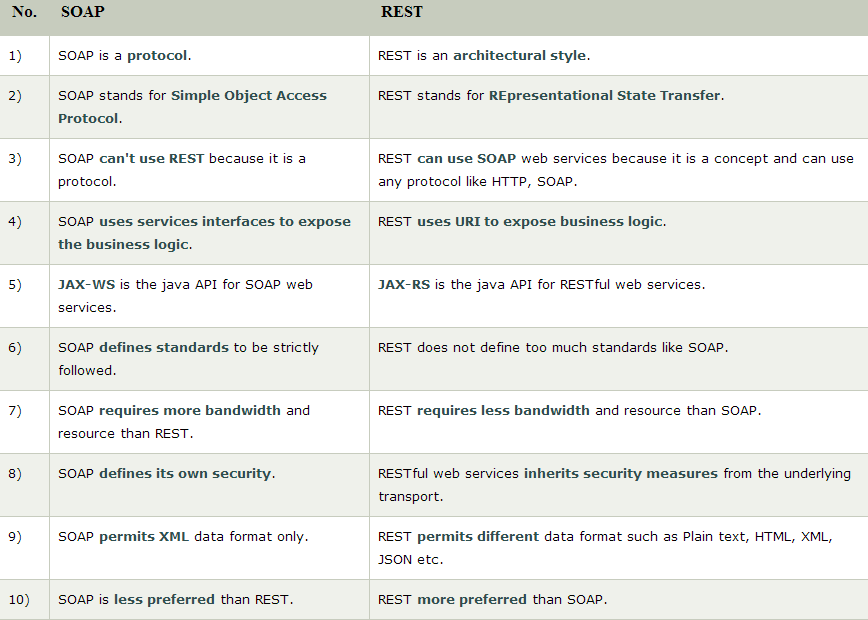




**RESTful Web services**

* REST stands for REpresentational State Transfer.
* REST is an architectural style not a protocol.
* **Fast**: RESTful Web Services are fast because there is no strict specification like SOAP. It consumes less bandwidth and resource.
* **Permits different data format**: RESTful web service permits different data format such as Plain Text, HTML, XML and JSON.

**Difference between SOAP and REST services:**



**Java Web Services Tutorial:**

* Two main java web services api:

JAX-WS and JAX-RS.

* The java web service application can be accessed by other programming languages such as .Net and PHP.

**Java Web Services API:**

1) **JAX-WS**: for SOAP web services. There are two ways to write JAX-WS application code: by **RPC** style and **Document** style.

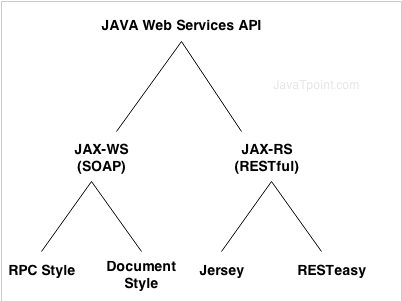
**Medium:** HTTP (POST)

**Format:** XML

2) **JAX-RS**: for RESTful web services. There are mainly 2 implementation currently in use for creating JAX-RS application: **Jersey** and **RESTeasy**.

**Medium:** HTTP (POST, GET, PUT, DELETE)

**Format:** XML,JSON,HTML,etc



4 basic parameters which are required for RESTful web services:

1. Resource based URI:

Resource based URI : Weatherapp.com/zipcodes/12345

Normal URI : weatherapp.com/weatherdetails.do?zipcode=12345

1. HTTP status code:

200 – successful; 500-Failure; 404-Not found

1. HTTP methods:

GET,POST,PUT,DELETE

1. Message Header

Header should contain context-type which will help understand which format client is transmitting the info.

**How does REST API work?**

1. Consumers need to know about the URI. They should be simple enough so that it is not difficult to remember.
2. Every page will have its unique URI.

Xyz.com/profiles/Kaushik.html - 1st level entities

Xyz.com/profiles/Kaushik

Xyz.com/profiles/{profileName}

Think of resources and create the URI’s as shown above.

1. URI contains nouns and not verbs and plurals.
2. Not dependent on the framework, since there is nothing like .do or .in
3. xyz.com/messages/{messageID}/comments/{commentId}

/messages/{messageID}/likes/{likeId}

* Instance Resource URI : /messages/{messageID}/likes/{likeId}
* Collection resource URI : /messages/{messageID}/likes

Clear enough to understand the difference.

* We have parameterized resource based URI which will help in paging and filtering the data.

Xyz.com/messages?offset=30&limit=10

Limit: number of elements which are going to be pulled

Offset: the threshold from which the pulling of the data will start

Xyz.com/messages?date=2017&offset=30&limit=10

**Operations on the Resources (Http methods):**

Your resource based URI do not define the action which needs to be performed as in regular URLs so we need Http methods to perform the action.

1. **GET:**

Getting a message.

This is a read only method.

Method Idempotent. – Can initiate any number of times still the state will be same. Repeatable calls.

1. **PUT:**

Updating the message for a request which has already been submitted.

Request body will have the updated message.

This is a write method.

Method Idempotent.

1. **DELETE:**

Deleting a message for a messageId.

This is a write method

Method Idempotent.

1. **POST:**

Create a new message.

Always issued to a collection resource since there is no ID available to fetch. This is a write method

/messages/

Resquest body will have the new message created.

Response will have the messageID of the new message created.

Not a Method Idempotent.

**Responses:**

Standard formats: XML and JSON

<MessageEntity>

<id>10</id>

<message>Hello world</message>

</MessageEntity>

{

“id”:”10”;

“message”:”Hello world”

}

When you are sending or receiving the calls using REST you are basically sending or receiving representation of the states.

**HTTP Headers:**

The headers consist of the ‘content-type’ parameter which defines the format in which the response is expected.

**HTTP Status Codes:**

1XX – Informational codes  
 200 – Success codes

3XX – Redirection codes

404 - Not Found

5XX – Server Error codes

**HATEOAS:**

**H**ypermedia **A**s **T**he **E**ngine **O**f **A**pplication **S**tate.