**Web Services**

**What Are Web Services**

1. Web Service is a software system for communicating two devices over the network.
2. Service is made available over the web.
3. Websites is used for user consumption but web services is used for application consumption.

**What are the advantages of web services?**

1. **Interoperability**: By the help of web services, an application can communicate with other application developed in any language.
2. **Reusability**: We can expose the web service so that other applications can use it.
3. **Modularity**: By the help of web service, we can create a service for a specific task such as tax calculation etc.

**What are the different types of web services?**

There are two types of web services:

SOAP

RESTful

**What is SOAP?**

* SOAP stands for Simple Object Access Protocol. It is a XML-based protocol for accessing web services.
* Specification of SOAP is JAX-WS.
* Used to communicate between client and server with message(xml format).

**What are the advantages of SOAP web services?**

WS Security-SOAP defines its own security known as WS Security.

Language Independent- SOAP web services can be written in any programming language.

Platform Independent-SOAP web services can be written in any programming language and executed in any platform.

**Disadvantages of Soap Web Services**

**Slow**: SOAP uses XML format that must be parsed to be read. It defines many standards that must be followed while developing the SOAP applications. So it is slow and consumes more bandwidth and resource.

**WSDL dependent**: SOAP uses WSDL and doesn't have any other mechanism to discover the service.

**What is WSDL?**

WSDL stands for Web Services Description Language. It is a xml document containing information about web services such as method name, method parameter etc.

**Elements of WSDL**

1. **Description->**It is the root element of a WSDL 2.0 file. It usually contains a set of name space declarations which are used throughout the WSDL file.
2. **Types** ->The WSDL types element describes the data types used by your web service.Data types are usually specified by XML schema.
3. **Binding->**The WSDL binding element describes how your web service is bound to a protocol.
4. **Interface ->**The WSDL interface element describes the operations supported by your web service.It is similar to methods in programming language.Client can only call one opertion per request.
5. **Service->**It describes the endpoint of your web service. In other words, the address where the web service can be reached.
6. **Endpoint ->**The endpoint element describes the address of the web service.
7. **Message**->The message element describes the data being exchanged between the Web service providers and consumers.

SERVICE

PORT

PORT TYPE

BINDING

OPERATIONS

INPUT MESSAGE

OUTPUT MESSAGE

TYPES

TYPES

TYPES

TYPES

**What is UDDI?**

* UDDI stands for Universal Description, Discovery and Integration where publishers publish the web services.It contains a list of available web services. WSDL is the part of UDDI.
* This is just like a yellow pages of web services.

**Basic Annotation**

@WebService, @SOAPBinding ,@WebMethod, @WebParam,@WebResult

**Top-down approach: Start with WSDL**

* When you start with WSDL you first define your services in the WSDL document. You can generate java classes from WSDL, if the document contains logical interfaces. When you create java classes using a WSDL the following conversions happens.
* The logical interface, defined by the wsdl:portType element, is mapped to a SEI (service endpoint interface ) • The complex types are mapped to Java Clasess following JAXB sepecification.
* The endpoint defined by the wsdl:service element is generated into a Java class. This java class is used by consumers to access endpoints implementing the service.
* You can use [wsimport](http://docs.oracle.com/javase/6/docs/technotes/tools/share/wsimport.html) tool to generate the java classes from WSDL document.

**Bottom-up approach: Start with Java Classes**

* In bottom-up approach we create the service end point interface (SEI) first. You define the methods in SEI that you expose as services. SEI typically is a standard Java interface. This is going to be the “*wsdl:portType* “ element in your WSDL document. The methods defined in SEI will become the “*wsdl:operation*” elements in the “*wsdl:portType element*” in WSDL. You need to annotate the SEI interface with required JAX-WS annotations.

**What is SEI?**

* Service endpoint interface(SEI) is used to convert java object to xml(SOAP) message.

**How to create client to acces webservices using SOAP**

1. For creating the java file from WSDL file use the command wsimport from command prompt.

D:\vikash\_docs\java\_notes\sei>wsimport -keep –s src <http://www.webservicex.net/geoipservice.asmx?WSDL>

-d

-quiet

-verbose

-p org.vikash

-b binding\_file.xml wsdl

2) Creating jaxb classes for input and output types using wsgen command line.

>wsgen –cp . –keep com.jspider.webservice. CalServiceImp

1. Java classes is the stub then create the object of service class and call the port. Then on port only we have to call webservices.
2. Two types

->Service first

->WSDL(contract) first

**Schema Types and Binding Styles**

* Used to change the wsdl style. Two types that is RPC and DOCUMENT.
* Document is used to place schema location or types in other xsd file.
* RPC is used to include everything in wsdl file only.
* **@SOAPBinding**(style=Style.***RPC***),@SOAPBinding(style=Style.***DOCUMENT***)

**Service Interface and Custom Types**

* Used for service end point interface(SEI).
* For that we have to create one interface that contains all the annotation that is used in wsdl file and we have to give one annotation(@WebService(endpointInterface="com.vikash.javabrains.ProductCatalogInterface")) in our impl class.

**Using JAXB Annotations**

* Using to convert the java object to xml.

**Handling Faults**

* If any method is throwing any exception then that method will not return any output. So wsdl file will contain any fault message tag.

**Testing Client**

* Soap UI,Web service explorer

**Using EndPoint**

* Without any server we can publish our webservices using EndPoint class through Metro.Glassfish also contains the bundle of Metro.
* Publisher class is a single thread model. single thread can use this class.

**wsimport**

**What is RESTful web services?**

REST stands for REpresentational State Transfer. It is a architectural style. It is not a protocol like SOAP.

**What are the advantages of RESTful web services?**

* Fast
* Language Independent
* Platform Independent
* Can use SOAP.
* Allows different data format.

**What is the difference between SOAP and REST web services?**

|  |  |  |
| --- | --- | --- |
| **No.** | **SOAP** | **REST** |
| 1) | SOAP is a **protocol**. | REST is an **architectural style**. |
| 2) | SOAP stands for **Simple Object Access Protocol**. | REST stands for **REpresentational State Transfer**. |
| 3) | SOAP **can't use REST** because it is a protocol. | REST **can use SOAP** web services because it is a concept and can use any protocol like HTTP, SOAP. |
| 4) | SOAP **uses services interfaces to expose the business logic**. | REST **uses URI to expose business logic**. |
| 5) | SOAP **permits XML** data format only. | REST **permits different** data format such as Plain text, HTML, XML, JSON etc. |

**What is SOA?**

SOA stands for Service Oriented Architecture. It is a design pattern to provide services to other application through protocol.

**What tools are used to test web services?**

* **SoapUI tool** for testing SOAP and RESTful web services
* **Poster** for firefox browser
* **Postman** extension for Chrome

**Web Service Security**

The following code sets authentication for use with the Call and Stub interfaces, respectively.

import javax.xml.rpc.Stub;

Stub stub = (Stub) ...;

stub.\_setProperty(Stub.USERNAME\_PROPERTY, "wsuser");

stub.\_setProperty(Stub.PASSWORD\_PROPERTY, "wspwd");

import javax.xml.rpc.Call;

Call call = (Call) ...;

call.setProperty(Call.USERNAME\_PROPERTY, "wsuser");

call.setProperty(Call.PASSWORD\_PROPERTY, "wspwd");

**Types of Operations available in WSDL**

**1) One-way** ->The operation can receive a message but will not return a response

**2) Request-response** ->The operation can receive a request and will return a response

**3) Solicit-response**->The operation can send a request and will wait for a response

**4) Notification** ->The operation can send a message but will not wait for a response

**SOAP Webservice**

* Simple Object Access Protocol (SOAP) is a standard protocol specification for message exchange based on XML. Communication between the web service and client happens using XML messages.
* SOAP provides a way to communicate between applications running on different operating systems, with different technologies and programming languages.

->A simple web service architecture have two components

1. Client
2. Service provider

CLIENT

SERVICE PROVIDER

INTERNET

Fun1() n fun2() are avail

{fun1();fun2();}

So as in above diagram,how client will communicate to service provider.So in order to communicate client must know some information for e.g.

1. Location of webservices server
2. Functions available,signature and return types of function.
3. Communication protocol
4. Input output formats

Service provider will create a standard XML file which will have all above information.So If this file is given to client then client will be able to access web service. This XML file is called WSDL.

#### Ways to access web service

There are two ways to access web service.

1. **If Service provider knows client** ->If service provider knows its client then it will provide its wsdl to client and client will be able to access web service.

INTERNET

CLIENT

SERVICE PROVIDER

Fun1() n fun2() are avail

Using WSDL client access {fun1();fun2();}

Web service

Service provide give WSDL to client

1. **Service provider register its WSDL to UDDI and client can access it from UDDI ->**UDDI stands for Universal Description, Discovery and Integration.It is a directory service. Web services can register with a UDDI and make themselves available through it for discovery.So following steps are involved.
2. Service provider registers with UDDI.
3. Client searches for service in UDDI.
4. UDDI returns all service providers offering that service.
5. Client chooses service provider.
6. UDDI returns WSDL of chosen service provider.
7. Using WSDL of service provider,client accesses web service.

**UDDI**

* 1. UDDI stands for Universal Description, Discovery and Integration.
  2. UDDI is a directory for storing information about web services , like yellow pages.
  3. UDDI is a directory of web service interfaces described by WSDL.

**INTERNET**

**CLIENT**

**SERVICE PROVIDER**

Fun1() n fun2() are avail

{fun1();fun2();}

6

2 1

**UDDI**

6 4 3

**Disadvantages of Soap Web Services**

Slow: SOAP uses XML format that must be parsed to be read. It defines many standards that must be followed while developing the SOAP applications. So it is slow and consumes more bandwidth and resource.

**RESTful web service**

->REST is an architectural style which was brought in by Roy Fielding in 2000 in his doctoral thesis.

->In the web services terms, REpresentational State Transfer (REST) is  a stateless client-server architecture in which the web services are viewed as resources and can be identified by their URIs. Web service clients that want to use these resources access via globally defined set of remote methods that describe the action to be performed on the resource.

->It consists of two components .

1) REST server which provides access to the resources.

2) REST client which accesses and modify the REST resources.

3) Everything in REST is considered as a resource.

4)Every resource is identified by an URI.

5)Rest APIs are stateless.

Two types of URI

->Instance resource UI : e.g: webapi/messages/1

->Collection resource URI : webapi/messages

5)Uses uniform interfaces. Resources are handled using POST, GET, PUT, DELETE operations which are similar to Create, Read, update and Delete(CRUD) operations.

6)Be stateless. Every request is an independent request. Each request from client to server must contain all the information necessary to understand the request.

7)Communications are done via representations. E.g. XML, JSON

Request a resource using URI

**RESOURCE**

**CLIENT**

Res=Representation of resource

(XML,JSON,plain text,png etc)**RESTful web service server**

->REST allows that resources have different representations, e.g.xml, json etc. The rest client can ask for specific representation via the HTTP protocol.

**HTTP Methods**

RESTful web services use HTTP protocol methods for the operations they perform.Methods are.

1. **GET**:It defines a reading access of the resource without side-effects.This operation is idempotent i.e.they can be applied multiple times without changing the result. idempotent
2. **PUT** :  It update a new resource. It must also be idempotent.
3. **DELETE** : It removes the resources. The operations are idempotent i.e. they can get repeated without leading to different results. Non-idempotent
4. **POST** : creates a new resource. idempotent

| **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 a HTTP POST request |
| @GET | Indicates that the following method will answer to a HTTP GET request |
| @PUT | Indicates that the following method will answer to a HTTP PUT request |
| @DELETE | Indicates that the following method will answer to a 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. |
| @QueryParam(“year”) int year | Use for query parameter in filtering.e.g: /messages?start=10&size=20 |
| @MatrixParam | Use for matrix parameter in filtering.e.g: /messages;start=10&size=20 |
| @HeaderParam | Use to get header param of client from postman. |
| @CookieParam | Use to get the cookie value |
| @FormParam | Use to get the form parameter. |

### Features of RESTful web services

1. **Resource identification through URI**

->Resources are identified by their URIs . So, a client can directly access a RESTful Web Services using the URIs of the resources.

2) **Uniform interface**

-> Resources are manipulated using a fixed set of four create, read, update, delete operations: PUT, GET, POST, and DELETE.

**3) Client-Server**

-> A clear separation concerns is the reason behind this constraint. Separating concerns between the Client and Server helps improve portability in the Client and Scalability of the server components.

**4) Stateless**

-> each request from client to server must contain all the information necessary to understand the request, and cannot take advantage of any stored context on the server.

**5) Cache**:

-> to improve network efficiency responses must be capable of being labeled as cacheable or non-cacheable.

**6) Named resources**

-> the system is comprised of resources which are named using a URL.

**7) Interconnected resource representations**

-> the representations of the resources are interconnected using URLs, thereby enabling a client to progress from one state to another.

**8) Layered components**

-> intermediaries, such as proxy servers, cache servers, gateways, etc, can be inserted between clients and resources to support performance, security, etc.

**9) Self-descriptive messages**

-> Resources are decoupled from their representation so that their content can be accessed in a variety of formats, such as HTML, XML, plain text, PDF, JPEG, JSON, and others.

Example-

package com.webservices.server;

import javax.ws.rs.GET;

import javax.ws.rs.Path;

import javax.ws.rs.PathParam;

import javax.ws.rs.Produces;

import javax.ws.rs.core.MediaType;

//@Path here defines class level path. Identifies the URI path that

// a resource class will serve requests for.

@Path("UserInfoService")

public class UserInfo {

@GET

@Path("/name/{i}")

@Produces(MediaType.TEXT\_XML)

public String userName(@PathParam("i") String i) {

String name = i;

return "<User>" + "<Name>" + name + "</Name>" + "</User>";

}

@GET

@Path("/age/{j}")

@Produces(MediaType.TEXT\_XML)

public String userAge(@PathParam("j") int j) {

int age = j;

return "<User>" + "<Age>" + age + "</Age>" + "</User>";

}}

For Running the program-

<http://localhost:8080/RESTfulWS/rest/UserInfoService/name/Pavithra>

## What Is JSON?

[JSON](http://en.wikipedia.org/wiki/JSON) is short for **JavaScript Object Notation**, and is a way to store information in an organized, easy-to-access manner.

var jason = {

"age" : "24",

"hometown" : "Missoula, MT",

"gender" : "male"

};

Bottom-up means you start with a Java method, and generate the WSDL from it.

Top-down means you start with a WSDL and then create all the necessary scaffolding in Java all the way down.

**Filtering and pagination**

1)/messages?year=2015 or /messages?start=10&size=20

**Using Context annotations**

->If we are not sure that what is the header param name or matrix param name or cookie param name then we can use @Context annotation.e.g:

**BeanParam annotations**

->Use to replace multiple query param in a method to a bean classs and access through @BeanParam.

**Sub-Resources**

->Use to send the request from one resource to another resource.

**Sending Status Codes and Location Headers**

->use to send custom status and location value in the header in our response.

**Handling Exceptions**

->use to handle custom exception.

->@Provider is used to register any mapper class of custom exception to jersy.

->Jersy alredy know about the @WebApplicationException.

**Content Negotiation**

->Used to tell the server that I am sending request in xml and need response in xml only.

->For that ,I have to add name-value pair in header through postman that is (Accept-text/xml) or (Accept-application/json). (Content-Type-text/xml)

->For Accept(which format client will accept) add format into @produces annotation and Content-Type(which format server will get) add format into @consumes annotation.  
@Produces(value={MediaType.APPLICATION\_JSON,MediaType.APPLICATION\_XML})

@ consumes (value={MediaType.APPLICATION\_JSON,MediaType.APPLICATION\_XML})

**REST API Authentication Mechanisms**

1)**Basic auth**

Add username and password in request header. And used **filters** will use to get the username and password to get authentication.

**Filters and Interceptors**

|  |  |
| --- | --- |
| Filters | interceptors |
| used to manipulate header information or metadata information. | used to manipulate actual body of request and response. |
| Two types  1)ContainerRequestFilter  2) ContainerResponseFilter | Two types  1)ReaderInterceptors  2)WriterInterceptor |
| e.g. Ligging ,security etc | e.g. encoding an entity response |
|  |  |
|  |  |
|  |  |
|  |  |

->Filters and Interceptors are used at server and client side both.

**Client Side**

->Filters

Two types

1)ClientRequestFilter

2) ClientResponseFilter

->Interceptors will be same for both server and client side.

->One more entities that is MessageBody

Two types:MessageBodyReader and MessageBodywriter

