Creating SOAP Web Services using JAX-WS

jaxenter.com/creating-soap-web-services-using-jax-ws-117689.html

There are various ways of creating web services. In this post we are going to create a SOAP based web service using JAX-WS, which is Java API for XML Web Services and we will deploy it under Tomcat.

One important point to remember is, both SOAP and REST style web services can be built using JAX-WS. There is a common misconception that JAX-WS is used for creating SOAP based web services and JAX-RS is used for creating REST style web services.

JAX-WS API is very rich and provides a handful of annotations to make developers life easy.

Different styles of SOAP based Web Service

SOAP based web services can be categorized as

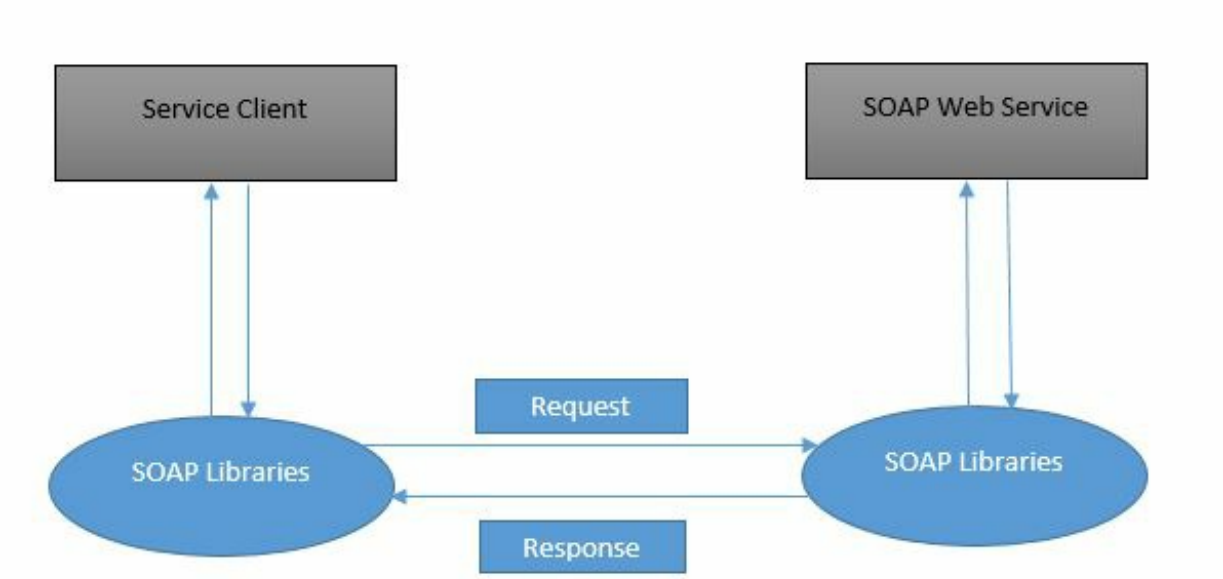
- **RPC Style –** RPC style is used for creating SOAP web services which includes simple data types (Built-in types)
- **Document Style –** This is the default style and used to create SOAP web services containing complex data types

DiscoverIndia Web Service

We are going to create a SOAP based web service named discoverIndia. What this web service is all about is, you pass a name of an Indian state to this service and you can retrieve

- capital of that state
- number of districts in that state
- local languages of that state
- airports in that state
- places to visit in that state
- some interesting facts about the state

Before getting into implementation lets understand the architecture of a SOAP based web service.



So without any further ado let's get started with the code.

Implementing the Web Service

Here is the Service Endpoint Interface (SEI) for the service

```
@WebService
public interface Country {

 @WebMethod
 String getCapital(String s);

 @WebMethod
 String getLanguages(String s);

 @WebMethod
 String getAirports(String s);

 @WebMethod
 int getDistricts(String s);

 @WebMethod
 String getPlacesToVisit(String s);

 @WebMethod
 String getInterestingFacts(String
s);
}
```

CountryImpl is the Service Implementation Bean (SIB) that implements all the methods of the SEI.

```
@WebService
public class CountryImpl implements Country {
 private Utility states;
 public CountryImpl(){
  states=new Utility();
  states.loadData();
 }
 @WebMethod
 public String getCapital(String stateName) {
  return states.getState(stateName).getCapital();
 }
 @WebMethod
 public String getLanguages(String stateName) {
  return states.getState(stateName).getLanguages();
 }
 @WebMethod
 public String getAirports(String stateName) {
  return states.getState(stateName).getAirports();
 }
 @WebMethod
 public int getDistricts(String stateName) {
  return states.getState(stateName).getDistricts();
 }
 @WebMethod
 public String getPlacesToVisit(String stateName) {
  return states.getState(stateName).getPlacesToVisit();
 }
 @WebMethod
 public String getInterestingFacts(String stateName) {
  return
states.getState(stateName).getInterestingFacts();
 }
}
```

Note that in CountryImpl class we have a member of Utility class which stores the information of states in a Map.

Here is the *State* class which represents a state of the country with getters and setters.

```
public class State {
 private String name;
 private String capital;
 private int districts;
 private String languages;
 private String airports;
 private String placesToVisit;
 private String interestingFacts;

 State(String name,String capital,int districts,String languages,String
airports,String placesToVisit,String interestingFacts){
  this.name=name;
  this.capital=capital;
  this.districts=districts;
  this.languages=languages;
```

```
  this.airports=airports;
  this.placesToVisit=placesToVisit;
  this.interestingFacts=interestingFacts;
 }

 public String getName(){
  return name;
 }
 public void setName(String name){
  this.name=name;
 }
 public String getAirports() {
  return airports;
 }
 public void setAirports(String airports) {
  this.airports = airports;
 }
 public String getCapital() {
  return capital;
 }
 public void setCapital(String capital) {
  this.capital = capital;
 }
 public String getLanguages() {
  return languages;
 }
 public void setLanguages(String languages) {
  this.languages = languages;
 }
 public int getDistricts() {
  return districts;
 }
 public void setDistricts(int districts) {
  this.districts = districts;
 }
 public String getPlacesToVisit() {
  return placesToVisit;
 }
 public void setPlacesToVisit(String placesToVisit) {
  this.placesToVisit = placesToVisit;
 }
 public String getInterestingFacts() {
  return interestingFacts;
 }
 public void setInterestingFacts(String interestingFacts) {
  this.interestingFacts = interestingFacts;
 }

}
```

We have a utility class which have a *Map state_map* to store a State information with state name as key. The *Utility* class also has a *loadData()* method which populates the *state_map*.

```
public class Utility {

 Map<String, State> state_map;

 Utility() {
  state_map = new HashMap<String, State>();
 }

 void loadData(){

  State rajasthan=new State("Rajasthan","Jaipur",33,"Hindi, Rajasthani","Jaipur,
Jodhpur, Udaipur","Ajmer, Udaipur, Jaipur","Rajasthan is famous for its marbles");
  State punjab=new State("Punjab","Chandigarh",22,"Hindi,
Punjabi","Amritsar","Amritsar, Ludhiana","Punjab is popular for its joyful punjabi
people");
  State mp=new State("Madhya Pradesh","Bhopal",51 ,"Hindi","Bhopal, Gwalior","Bhopal,
Gwalior","Madhya Pradesh is well known for its wildlife and monuments");
  State haryana=new State("Haryana","Chandigarh",21 ,"Hindi, Haryanvi","","Hisar,
Rohtak, Fridabad","Haryana is popular for its haryanvi language");
  State gujrat=new State("Gujrat","Gandhinagar",33 ,"Hindi, Gujrati","Ahmedabad,
Vadodara, Gandhinagar","Ahmedabad, Vadodara","Gujrat is famous for its rich culture
and dandiya nights");

  state_map.put("Rajasthan", rajasthan);
  state_map.put("Punjab", punjab);
  state_map.put("Madhya Pradesh", mp);
  state_map.put("Haryana", haryana);
  state_map.put("Gujrat", gujrat);

  }

 State getState(String stateName) {
  State state = null;
  state = state_map.get(stateName);
  return state;
 }

}
```

Configuring the Web Service

Since we are going to deploy this web service in a web container in this case Apache Tomcat, we have to configure our web service in web.xml file as below.

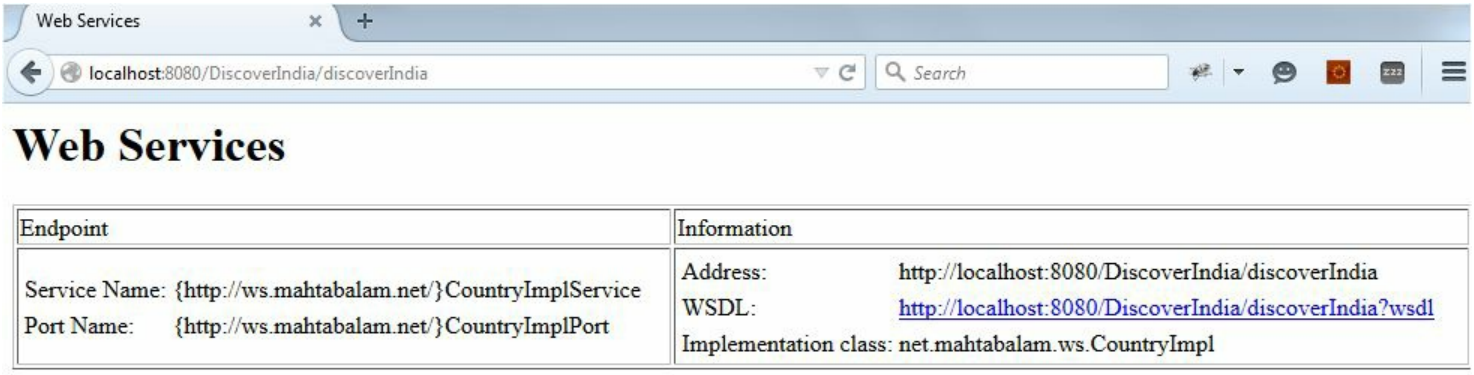
```
<listener>
  <listener-class>

com.sun.xml.ws.transport.http.servlet.WSServletContextListener
  </listener-class>
</listener>
<servlet>
  <servlet-name>xyz</servlet-name>
  <servlet-class>
   com.sun.xml.ws.transport.http.servlet.WSServlet
  </servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>xyz</servlet-name>
  <url-pattern>/discoverIndia</url-pattern>
</servlet-mapping>
```

One last configuration is required – creating a file sun-jaxws.xml in WEB-INF folder which defines Service implementation class.

```
<endpoints xmlns="http://java.sun.com/xml/ns/jax-ws/ri/runtime"
 version="2.0">
 <endpoint name="WebServicImpl"
  implementation="net.mahtabalam.ws.CountryImpl"
  url-pattern="/discoverIndia" />
</endpoints>
```

Note that you will require extra jax-ws libraries to publish SOAP based web service under Tomcat. You can download these libraries from jax-ws.net and put them in the WEB-INF/lib folder.

Now run the project and your web service will be deployed and available at whatever URL you mentioned in web.xml file (in this case /discoverIndia).



You can access the WSDL (Service Contract) by appending *?wsdl* at the end of deployed web service URL.

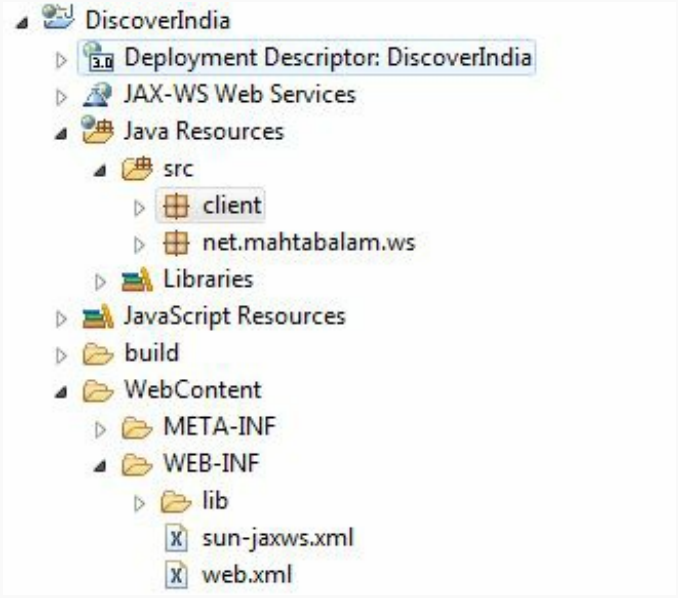
localhost:8080/DiscoverIndia/discoverIndia?wsdl

```
- <definitions targetNamespace="http://ws.mahtabalam.net/" name="CountryImplService">
  - <types>
    - <xsd:schema>
        <xsd:import namespace="http://ws.mahtabalam.net/" schemaLocation="http://localhost:8080/DiscoverIndia/discoverIndia?xsd=1"/>
      </xsd:schema>
    </types>
  - <message name="getInterestingFacts">
      <part name="parameters" element="tns:getInterestingFacts"/>
    </message>
  - <message name="getInterestingFactsResponse">
      <part name="parameters" element="tns:getInterestingFactsResponse"/>
    </message>
  - <message name="getCapital">
      <part name="parameters" element="tns:getCapital"/>
    </message>
  - <message name="getCapitalResponse">
      <part name="parameters" element="tns:getCapitalResponse"/>
    </message>
  - <message name="getLanguages">
      <part name="parameters" element="tns:getLanguages"/>
    </message>
```

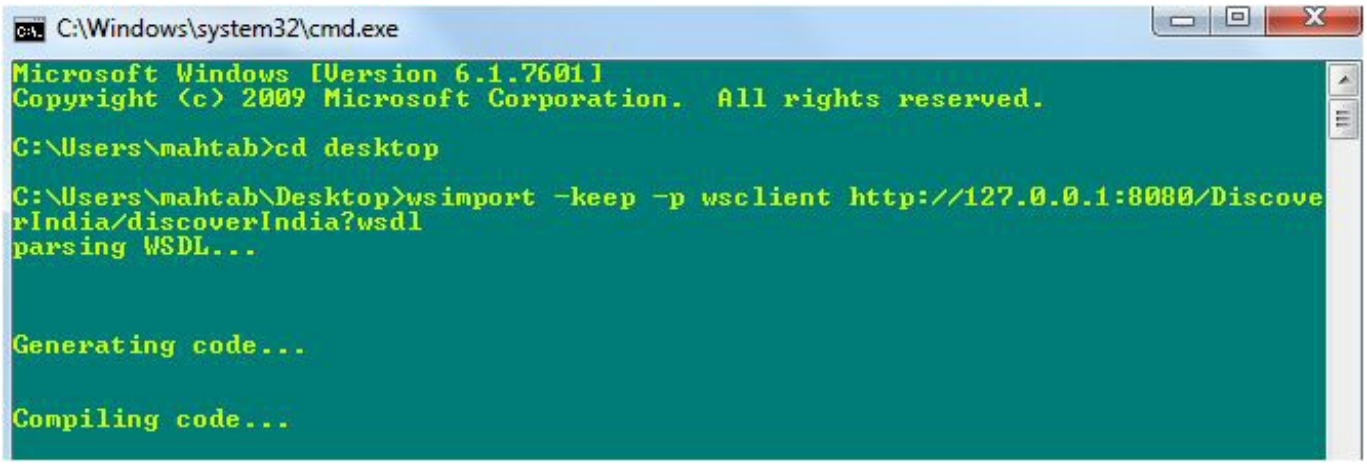
Once the web service is published you can write a client against web service in any language Python, Perl, C#, Java etc.

That's the beauty of web services, they are language independent. Let's write a Java client to consume the web service.

We will use the *wsimport* utility which comes with JDK6 to generate client side artifacts to make it easy to write a client against the discoverIndia web service.



A snapshot of the project structure in Eclipse IDE.



The above command will generate artefacts that we will use to write a simple client.

Here is the Client.java which uses *wsimport* generated classes to write a simple web service client.

```
package wsclient;
 public class Client {

 public static void main(String[] args) {
  CountryImplService service=new CountryImplService();
  CountryImpl country=service.getCountryImplPort();
  System.out.println("Capital of Kerala : "+country.getCapital("Kerala"));
  System.out.println("No. of districts in Telangna :
"+country.getDistricts("Telangna"));
  System.out.println("Local languages of Gujrat : "+country.getLanguages("Gujrat"));
  System.out.println("Airports in Uttrakhand : "+country.getAirports("Uttrakhand"));
  System.out.println("Places to visit in Rajasthan :
"+country.getPlacesToVisit("Rajasthan"));
  System.out.println("Interesting Facts about Meghalaya :
"+country.getInterestingFacts("Meghalaya"));
 }
 }
```

Now compile Client.java and run it.

```
C:\Users\mahtab\Desktop>javac wsclient/Client.java

C:\Users\mahtab\Desktop>java wsclient.Client
Capital of Kerala : Thiruvananthapuram
No. of districts in Telangna : 10
Local languages of Gujrat : Hindi, Gujrati
Airports in Uttrakhand : Dehradun
Places to visit in Rajasthan : Ajmer, Udaipur, Jaipur
Interesting Facts about Meghalaya : Meghalaya is the only state in India which h
ave English as its official language

C:\Users\mahtab\Desktop>
```

Remember that if you call web service with a state name that is not handled by web service you will get SOAP Fault as response.

Get the source code: If you are interested in trying the code yourself, you can get it from GitHub.

Play with the live Web Service: This web service is deployed live at mathabalam.net and WSDL file is available here.

Author

Mahtab Alam

My name is Mahtab Alam. I'm a tea lover and love unplanned travelling.
I learn things and implement in my own way.
I am Oracle Certified Java Programmer, Web Component Developer and SQL Expert.
Mostly I code in Java but open to all other technologies.
I speak my mind at www.mahtabalam.net