Developing Web services

{scrollbar}

This article will move through basics of Web Services, the various Web Services platform elements SOAP (Simple Object Access Protocol), UDDI (Universal Description, Discovery and Integration) and WSDL (Web Services Description Language).

What is Web Services?

INLINE

Web Services is a platform to build loosely coupled applications.

A web service is piece of code that can be remotely invoked using HTTP, that means an HTTP request is enough to invoke a web service. Moving back in time, remote access to binary data has been either platform specific or vendor specific. For example:

1. Microsoft's DCOM (Distributed Component Object Model) for communication between networked computers access remote COM (Component Object Model) interface through remote procedure calls.
2. Corba uses IIOP (Internet Inter-ORB Protocol) to access remote objects.
3. In Java RMI (Remote Method Invocation) is used to access an EJB (Enterprise Java Bean) object which is again language specific.

All of the above examples suggest that remotely accessing an object required proprietary technologies that were tightly coupled to the remote code.

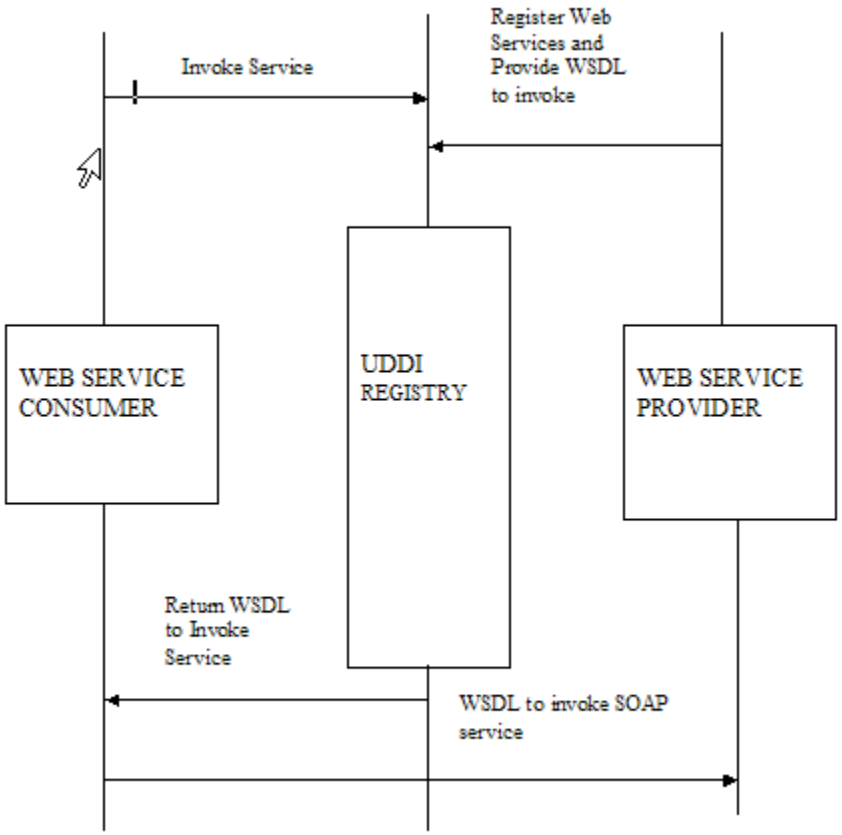
Web Services on the other hand, is basically a platform to built a distributed architecture where there are numerous computers connected to each other in a network. The various computers in the network will be running applications developed using different tools, different technologies from different vendors. Each of these applications can be exposed as a Web Service using SOAP, UDDI and WSDL wherein a service running on one computer can access a service running on other irrespective of difference in technologies. So, if there is a .Net application running on one computer it can be accessed using a Java application from other, only condition is both should be exposed as a Web service. Web Service clients can be console based as well as browser based.

Why Web Services?

There are several reasons why Web Services has become a need:

1. One of the biggest reasons is interoperability wherein different vendor specific applications can interact with each other. For Example a .Net application accessing a Java web service.
2. Using Web Services you can expose your application and its functionality globally.
3. Using Web Services, an enterprise would not not depend on one particular vendor for all the solutions. It can move to different vendors for different functionality and can optimally choose out of several options.
4. Web Services use standardized protocols SOAP, UDDI, WSDL and HTTP for implementation.
5. Web Services has support for most of the communication protocol and it can be implemented using FTP as well as HTTP.
6. Web Services follow a loosely coupled architecture.

Web Services Architecture



As shown in the architecture diagram, the Web Service Provider registers the Web Service to UDDI registry and provides WSDL for invoking service. The Web Service consumer, which can be an application client or any other Web Service, queries the UDDI registry and finds WSDL. Next the consumer uses WSDL to invoke the SOAP service.

Web Services Description Language (WSDL)

WSDL is an XML based way of describing a Web Service. It specifies the location of Web Service and methods available with the service. A WSDL document has **<portType>**, **<message>**, **<types>** and **<binding>** as the elements.

The typical syntax of a WSDL document is shown in the following example.

solidSyntax of a WSDL document <definitions> <types> definition of types.... </types> <message> definition of message.... </message> <portType> definition of portType.... </portType> <binding> definition of binding.... </binding> <service> definition of service.... </service> </definitions>

The syntax of a WSDL suggests that it is a set of definitions where the definition element is at the root.

Let us try to understand each element in a WSDL document using a `HelloWorld.wsdl` document. This WSDL document is automatically generated by the Geronimo Eclipse Plugin (GEP). With the tutorials listed at the end of this document you will learn how to generate it using Eclipse and GEP.

solidHelloWorld.wsdl <!--WSDL created by Apache Axis version: 1.4 Built on Apr 22, 2006 (06:55:48 PDT)--> <?xml version="1.0" encoding="UTF-8"?> <wsdl:definitions> <wsdl:types> <schema elementFormDefault="qualified" targetNamespace="http://webservices" xmlns="http://www.w3.org/2001 /XMLSchema"> <element name="hello"> <complexType/> </element> <element name="helloResponse"> <complexType/> </element> </schema> </wsdl: types> <wsdl:message name="helloRequest"> <wsdl:part element="intf:hello" name="parameters"/> </wsdl:message> <wsdl:message name=" helloResponse"> <wsdl:part element="intf:helloResponse" name="parameters"/> </wsdl:message> <wsdl:portType name="HelloWorld"> <wsdl:operation name="hello"> <wsdl:input message="intf:helloRequest" name="helloRequest"/> <wsdl:output message="intf:helloResponse" name="helloResponse"/> < /wsdl:operation> </wsdl:portType> <wsdl:binding name="HelloWorldSoapBinding" type="intf:HelloWorld"> <wsdlsoap:binding style="document" transport=" http://schemas.xmlsoap.org/soap/http"/> <wsdl:operation name="hello"> <wsdlsoap:operation soapAction=""/> <wsdl:input name="helloRequest"> <wsdlsoap:body use="literal"/> </wsdl:input> <wsdl:output name="helloResponse"> <wsdlsoap:body use="literal"/> </wsdl:output> </wsdl:operation> < /wsdl:binding> <wsdl:service name="HelloWorldService"> <wsdl:port binding="intf:HelloWorldSoapBinding" name="HelloWorld"> <wsdlsoap:address location="http://localhost:8080/SimpleWeb/services/HelloWorld"/> </wsdl:port> </wsdl:service> </wsdl:definitions>

- **<types>**
  Web Service is all about sending and receiving messages. The <types> element describes the various messages which will be used by the service. Basically it defines the various data types. As can be seen, HelloWorld service uses hello and helloResponse as the two messages.

- **<message>**
  This element defines the various messages used by the Web service. In our example we have two message name, *helloRequest* which is associated with hello data type and *helloResponse* which is associated with helloResponse data type as defined in the <types> element. Each message type has unique name which is suggested by <message name=""> tag. The <part element="" ...> suggests the data type associated with each message. <part> element can be considered as a parameter to a function call.

- **<portType>**
  This basically involves a set of operation and messages involved in each operation. <portType name=""> defines a unique name. In our example <wsdl:portType name="HelloWorld"> suggest a unique name. Each portType has an associated <operation name=""> element which suggest an operation name. In our example <wsdl:operation name="hello"> suggests an operation name. Each operation element can have <input>, <output> or both the tags accordingly it will be one-way, notification or request-response operation. Our example has <wsdl:input message="intf:helloRequest" name="helloRequest"/> which uses the helloRequest as the input message and <wsdl:output message="intf:helloResponse" name="helloResponse"/> which uses the helloResponse as the output message.

- **<bindings>**
  A binding defines message format and protocol details for operation and message defined by each portType. Each binding element is identified by a unique **name** attribute which can be any user defined name. The **type** element defines the portType referenced by binding. In our case <wsdl:binding name= "HelloWorldSoap Binding" type="intf:HelloWorld"> defines a binding with a unique name and **HelloWorld** portType associated with it. The **soap:binding** element has **style** attribute which can be either RPC-oriented(messages containing parameters and return values) or document-oriented(message containing documents). The default value for this element is document. The other element is **transport** which suggests the transmission protocol used. In our example we have <wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/> document as the style and http as the transport protocol. *<wsdl:operation name="hello">* name defines operation associated with each portType. Each operation will have a a soapAction associated with it. *<wsdl:input name="helloRequest">* define the input for the operation and <wsdlsoap:body use="literal"/> defines the encoding standard. In our example we are using literal as the encoding standard.

- **<service>**
  Service element defines where the service can be accessed. Each service is associated with a unique name which is suggested by <wsdl:service name="HelloWorldService"> in our example. The next element *<wsdl:port binding="intf:HelloWorldSoapBinding" name="HelloWorld">* suggests the previous binding which will be used by service. The element *<wsdlsoap:address location="http://localhost:8080/SimpleWeb/services/HelloWorld"/>* suggests the physical address through which a service can be accessed.

Web Services tutorials