Create a web service from a WSDL using JAX-WS

The wsimport tool reads local WSDL file and generates the following portable artifacts:

* **Service Endpoint Interface (SEI)** - The SEI is the annotated Java representation of the WSDL file for the web service. This interface is used for implementing JavaBeans endpoints or creating dynamic proxy client instances.
* **javax.xml.ws.Service** extension class - This is a generated class that extends the javax.xml.ws.Service class. This class is used to configure and create both dynamic proxy and dispatch instances.
* **Java Architecture for XML Binding (JAXB)** beans that are required to model the web service data.
* Exception class mapped from **wsdl:fault**

# Using Wsimport Ant Task:

Required Software:

* JDK 1.6 or above
* JAX-WS Metro web service stack 2.1 or above
* Apache ant 1.7 or above

To generate artifacts, using wsimport ant task:

1. Download following files and put it in local directory (recommended right\_to\_lift directory):

* RightToLift.wsdl
* RightToLift.xsd
* PIDXLibDS.xsd
* pidxlibds-binding.xsd (JAX-WS custom binding XSD-to-Java mappings)
* build.properties
* build.xml

1. Modify build.properties file to reflect your software & development environment:

* jaxws.metro.home - Metro web service stack installation dir
* wsdl.file - Local WSDL file location
* wsdllocation.file - Local wsdl URI, will be used to set wsdlLocation value of @WebService & @WebServiceClient annotation elements on generated SEI and Service Interface
* class.dest.dir - Directory for generated class files
* source.dest.dir - Directory for generated source files
* source.package.name - Target package (defaults to namespace defined in WSDL)
* binding.filename - External JAXB binding file

1. Run Ant using command-line

# Using Wsimport:

Required Software:

* JDK 1.6 or above
* JAX-WS Metro web service stack 2.1 or above

To generate artifacts:

Make sure that you are using JAX-WS Metro web service stack.

wsimport -d target\classes -extension -keep -s src\main\java -verbose RightToLift.wsdl -b pidxlibds-binding.xsd -wsdllocation RightToLift.wsdl

* -d Directory for generated class files
* -b External JAXB binding file
* -keep Keep generated files
* -s Directory for generated source files
* -wsdllocation Local wsdl URI

# Build the web service implementation using artifacts:

## Deploy in Servlet Container:

Once the endpoint interface has been generated, perform following to deploy this web service in apache tomcat 6.X container:

1. Write a java class that implements the RightToLift interface. Make sure that following web service values are defined:

* name - name of the java class
* portName - port name of the web service
* endpointInterface - complete name of the service endpoint interface
* wsdlLocation - wsdl location

For ex:

*@WebService(name = "RightToLiftService",*

*portName="RightToLiftPort",*

*endpointInterface="com.dtn.righttolift.RightToLift",*

*targetNamespace = "http://www.pidx.org/RightToLift.wsdl",*

*wsdlLocation="WEB-INF/wsdl/RightToLift.wsdl")*

*@BindingType(SOAPBinding.SOAP12HTTP\_BINDING)*

*public class RightToLiftWS implements RightToLift {*

*// implementmethods*

*}*

1. Create sun-jaxws.xml (needed to deploy as standard WAR on non-java EE5 servlet container using SUN's reference implementation )

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

*<endpoint name="RightToLiftServiceEndPoint"*

*service="{http://www.pidx.org/RightToLift.wsdl}RightToLiftService"*

*port="{http://www.pidx.org/RightToLift.wsdl}RightToLiftPort"*

*implementation="Java class created in step 1"*

*url-pattern="/RightToLift"*

*wsdl="WEB-INF/wsdl/RightToLift.wsdl" >*

*</endpoints>*

1. Modify web.xml file to utilize Sun's RI, WSServletContextListener as the listener for servlet context events andWSServlet as the dispatcher servlet

<listener>

<listener-class>

com.sun.xml.ws.transport.http.servlet.WSServletContextListener

</listener-class>

</listener>

<servlet>

<servlet-name>RightToLift</servlet-name>

<servlet-class>com.sun.xml.ws.transport.http.servlet.WSServlet</servlet-class>

</servlet>

1. Web Service is ready to be deployed in tomcat
2. Simply load the web service using SOAP UI tools.

SOAP UI (<http://www.soapui.org/>)

## Use EndPoint API to configure & deply in Java SE 6:

The EndPoint class can be used to create and publish web service endpoints. An endpoint consists of an object that acts as the web service implementation (called here implementor) plus some configuration information, e.g. a Binding. Implementor and binding are set when the endpoint is created and cannot be modified later. Their values can be retrieved using the getImplementor and getBinding methods respectively. Other configuration information may be set at any time after the creation of an Endpoint but before its publication.

1. Write a java class that implements the RightToLift interface. Make sure that following web service values are defined:

* serviceName - name of the java class
* portName - port name of the web service
* endpointInterface - complete name of the service endpoint interface
* wsdlLocation - wsdl location

*@WebService(serviceName = "RightToLiftService",*

*portName = "RightToLiftPort",*

*endpointInterface = "org.pidx.righttolift.RightToLift",*

*targetNamespace = "http://www.pidx.org/RightToLift.wsdl",*

*wsdlLocation = "RightToLift.wsdl")*

*public class MyRTLService implements RightToLift {*

*@WebMethod(action = "* *AuthorizeLoad")*

*@WebResult(name = "PIDXAuthResponse", targetNamespace = "http://www.pidx.org/schema/ds/v5.01", partName = "parameter")*

*public PIDXAuthResponse AuthorizeLoad(PIDXAuthRequest parameter) throws RightToLiftFault\_Exception {*

*// implement*

*}*

*@WebMethod(action = "* SubmitBOL*")*

*@WebResult(name = "PIDXBOLResponse", targetNamespace = "http://www.pidx.org/schema/ds/v5.01", partName = "parameter")*

*public PIDXBOLResponse* SubmitBOL*(PIDXBOL parameter) throws RightToLiftFault\_Exception {*

*// implement*

*}*

*}*

1. Create and publish endpoint for the given implementer:

*HttpServer server = HttpServer.create(new InetSocketAddress(8001), 10);*

*server.setExecutor(new Executor() {*

*public void execute(Runnable r) {*

*r.run();*

*}*

*});*

*server.start();*

*HttpContext context = server.createContext("/rightToLift");*

*endPoint = Endpoint.create(SOAPBinding.SOAP12HTTP\_BINDING, rightToLift);*

*List<Source> metadata = new ArrayList<Source>();*

*URL xsdURL = ClassLoader.getSystemResource("RightToLift.xsd");*

*metadata.add(new StreamSource(xsdURL.openStream(), xsdURL.toExternalForm()));*

*URL xsdURL = ClassLoader.getSystemResource("PIDXLibDS.xsd");*

*metadata.add(new StreamSource(xsdURL.openStream(), xsdURL.toExternalForm()));*

*endpoint.setMetadata(metadata);*

*endPoint.publish(context);*

1. Run as standard java application and web service is ready. Use SOAP UI to load web service

(as per code outlined in 2: http://localhost:8001/rightToLift?wsdl)

# Build client side implementation using artifacts:

* Create artifacts using one of the mechanism defined in Step [1](#_Using_Wsimport_Ant) or [2](#_Using_Wsimport:)
* Create web service features to represent addressing features (WS-Addressing with either SOAP 1.1/HTTP or SOAP 1.2/HTTP binding), enable ws-addressing and requires its use

*WebServiceFeature[] enableAddressing = {new AddressingFeature(true, false)};*

* Use RightLiftToService to retrieve the PortType

*RightToLift portType = new RightToLiftService().getRightToLiftPort(enableAddressing);*

* Cast portType to BindingProvider to access the protocol binding and associated objects for request, response

*BindingProvider bp = (BindingProvider) portType;*

*bp.getRequestContext().put(BindingProvider.USERNAME\_PROPERTY, "USERNAME");*

*bp.getRequestContext().put(BindingProvider.PASSWORD\_PROPERTY, "PASSWORD");*

*bp.getRequestContext().put(BindingProvider.ENDPOINT\_ADDRESS\_PROPERTY, "SERVICE\_PROVIDER\_URL");*

* Create appropriate data and call portType interface's corresponding method.

*PIDXAuthRequest request = new PIDXAuthRequest();*

*.........*

*PIDXAuthResponse response = portType. AuthorizeLoad(request);*

Please refer to The [Java EE 5 Tutoria](http://docs.oracle.com/javaee/5/tutorial/doc/)l for more information regarding building web services.