**Web Services**

Steps in Creating a Web Service Application

1. Code the implementation class.
2. Compile the implementation class and generate needed files for servicing using wsgen.
3. Deploy the WAR file. The tie classes (which are used to communicate with clients) are generated by the Application Server during deployment. Alternatively, you can use the application sever that is part of J2SE 6 albeit limited in features.
4. Code the client class.
5. Use wsimport to generate and compile the stub files.
6. Compile the client class.
7. Run the client.

**Web Service Server**

**create a folder named "webexer1". inside it, create a folder called "server" -**

**this will be the root folder of the server. inside the folder "server" create a "hello" folder.**

**copy and paste the following program to a text file and save it as HelloWorld.java**

**(place it inside the "hello" folder)**

// all web services (implementations)

// must be placed in a package

package hello;

import javax.jws.WebService;

import javax.jws.WebMethod;

// annotation; a j2se 5 feature

@WebService

public class HelloWorld {

@WebMethod

public String sayHello(String n) {

System.out.println("sending message: "+n);

return "hello, "+n+"!";

}

}

*if the command line shows an error in the compilation,* [*click this link to help you set your java/jdk path/classpath*](http://javarevisited.blogspot.com/2013/02/windows-8-set-path-and-classpath-java-windows-7.html)

**Compilation -- go to the server folder in the command line and compile the implementation code using this command**

> javac hello\HelloWorld.java

*if the command line shows an error in the web service source generation (wsgen),* [*click this link to help you set your java/jdk path/classpath*](http://javarevisited.blogspot.com/2013/02/windows-8-set-path-and-classpath-java-windows-7.html)

**Build Service -- generate class files to make the implementation "web serviceable" :)**

> wsgen –cp . hello.HelloWorld

*Note: use the –cp parameter to include the current directory to the class path*

**create a server that will host your class implementation; save this at the root folder of your server**

import javax.xml.ws.Endpoint;

import hello.\*;

public class WSServer {

public static void main(String[] args) {

Endpoint.publish("http://localhost:8080/hello/myhello",new HelloWorld());

}

}

*if the command line shows an error in the compilation,* [*click this link to help you set your java/jdk path/classpath*](http://javarevisited.blogspot.com/2013/02/windows-8-set-path-and-classpath-java-windows-7.html)

**compile the server using the command:**

> javac -cp . WSServer.java

**then run the server using this command:**

> java -cp . WSServer

**You can test the generation of WSDL by pointing your browser to the URI:**

<http://localhost:8080/hello/myhello?wsdl>

**Web Service Client**

**inside the folder "webexer1", create a folder named "client". go to that folder in the command line and issue the following command to generate stub files**

**for the client application.**

> wsimport http://localhost:8080/hello/myhello?wsdl

*the command will generate the needed files to run the client (including the package "hello" and all necessary files inside it)*

**create a text file and copy the following program and save it as Client.java:**

import hello.\*;

import javax.xml.ws.WebServiceRef;

public class Client {

@WebServiceRef(wsdlLocation="http://localhost:8080/hello/myhello")

public static void main(String[] args) {

try {

HelloWorldService hws = new HelloWorldService();

HelloWorld hw = hws.getHelloWorldPort();

System.out.println("");

System.out.println("from the server: "+hw.sayHello("ali"));

} catch(Exception e) {

e.printStackTrace();

}

}

}

**Compile the client**

> javac -cp . Client.java

**Run the client**

> java -cp . Client