JAX-WS SOAP Example – RPC Style

AX-WS is bundled with JDK 1.6, which makes Java web service development easier to develop. This tutorial shows you how to do the following tasks:

1. Create a SOAP-based RPC style web service endpoint by using JAX-WS.
2. Create a Java web service client manually.

Note
In general words, "*web service endpoint*" is a service which published outside for user to access; where "*web service client*" is the party who access the published service.

1. Create a Web Service Endpoint Interface

File : HelloWorld.java

```
package com.york.ws;

import javax.jws.WebMethod;

import javax.jws.WebService;

import javax.jws.soap.SOAPBinding;

import javax.jws.soap.SOAPBinding.Style;

//Service Endpoint Interface

@WebService

@SOAPBinding(style = Style.RPC)

public interface HelloWorld{

        @WebMethod String getHelloWorldAsString(String name);

}
```

2. Create a Web Service Endpoint Implementation

File : HelloWorldImpl.java

```
package com.york.ws;

import javax.jws.WebService;

//Service Implementation

@WebService(endpointInterface = "com.york.ws.HelloWorld")

public class HelloWorldImpl implements HelloWorld{

        @Override
        public String getHelloWorldAsString(String name) {

                return "Hello World JAX-WS " + name;

        }

}
```

3. Create a Endpoint Publisher

File : HelloWorldPublisher.java

```
package com.york.endpoint;

import javax.xml.ws.Endpoint;

import com.mkyong.ws.HelloWorldImpl;

//Endpoint publisher

public class HelloWorldPublisher{

        public static void main(String[] args) {

           Endpoint.publish("http://localhost:9999/ws/hello", new HelloWorldImpl());

    }

}
```

Run the endpoint publisher, and your "**hello world web service**" is deployed in URL "**http://localhost:9999/ws/hello**".

4. Test It

You can test the deployed web service by accessing the generated WSDL (Web Service Definition Language) document via this URL "**http://localhost:9999/ws/hello?wsdl**" .

Web Service Clients

Ok, web service is deployed properly, now let's see how to create web service client to access to the published service.

1. Java Web Service Client

Without tool, you can create a Java web service client like this :

```
package com.york.client;

import java.net.URL;

import javax.xml.namespace.QName;
import javax.xml.ws.Service;

import com.mkyong.ws.HelloWorld;

public class HelloWorldClient{

        public static void main(String[] args) throws Exception {

        URL url = new URL("http://localhost:9999/ws/hello?wsdl");

        //1st argument service URI, refer to wsdl document above

        //2nd argument is service name, refer to wsdl document above

        QName qname = new QName("http://ws.mkyong.com/", "HelloWorldImplService");

        Service service = Service.create(url, qname);

        HelloWorld hello = service.getPort(HelloWorld.class);

        System.out.println(hello.getHelloWorldAsString("York Chen"));

    }

}
```

2. Java Web Service Client via wsimport tool

Alternative, you can use “**wsimport**” tool to parse the published wsdl file, and generate necessary client files (stub) to access the published web service.

Where is wsimport?
This **wsimport** tool is bundle with the JDK, you can find it at “*JDK_PATH/bin*” folder.

Issue “**wsimport**” command.

```
wsimport -keep http://localhost:9999/ws/hello?wsdl
```

It will create *HelloWorld.java, HelloWorldImplService.java,*

Now, create a Java web service client which depends on the above generated files.

```
package com.york.client;

import com.york.ws.HelloWorld;

import com.york.ws.HelloWorldImplService;

public class HelloWorldClient{

        public static void main(String[] args) {

                HelloWorldImplService helloService = new HelloWorldImplService();
                HelloWorld hello = helloService.getHelloWorldImplPort();

                System.out.println(hello.getHelloWorldAsString("york chen"));

    }

}
```