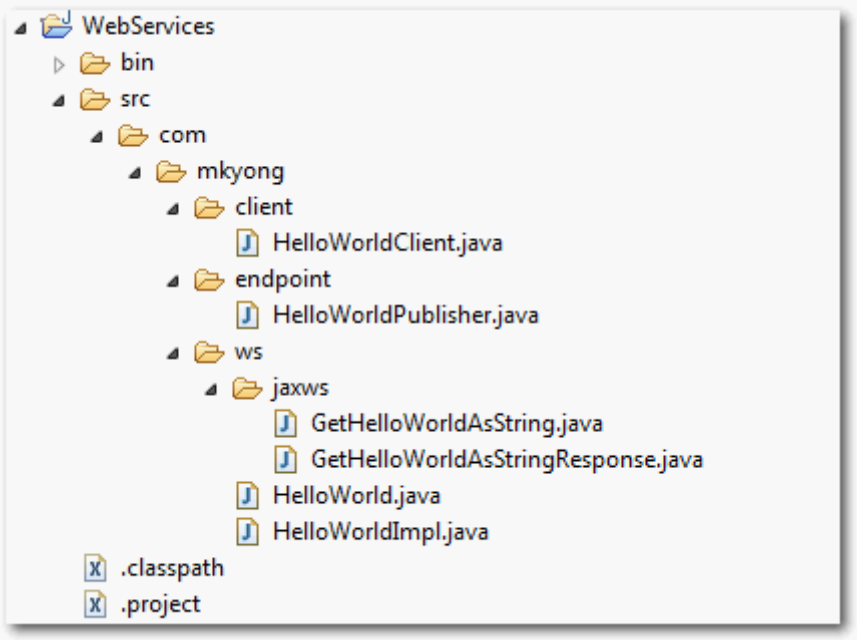
JAX-WS Hello World Example – Document Style

By mkyong | November 20, 2010 | Updated : August 29, 2012 | Viewed : 467,829 | +601 pv/w

In this tutorial, we show you how to use JAX-WS to create a SOAP-based web service (document style) endpoint. Compare with RPC style, it need some extra efforts to get it works.

Directory structure of this example



JAX-WS Web Service End Point

Here are the steps to create a document style web service in JAX-WS.

1. Create a Web Service Endpoint Interface

Actually, annotated with `@SOAPBinding` is optional, because the default style is document.

File : HelloWorld.java

```
package com.mkyong.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.DOCUMENT, use=Use.LITERAL) //optional
public interface HelloWorld{

    @WebMethod String getHelloWorldAsString(String name);

}
```

Note
In JAX-WS development, convert from "*RPC style*" to "*Document style*" is very easy, just change the `@SOAPBinding` style option.

2. Create a Web Service Endpoint Implementation

File : HelloWorldImpl.java

```java
package com.mkyong.ws;

import javax.jws.WebService;

//Service Implementation
@WebService(endpointInterface = "com.mkyong.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

```java
package com.mkyong.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());
    }

}
```

Wait, when you run the end point publisher, you will hits following error message :

```
Wrapper class com.mkyong.ws.jaxws.GetHelloWorldAsString is not found.
    Have you run APT to generate them?
```

See this article. You need to use "**wsgen**" tool to generate necessary JAX-WS portable artifacts. Let move to next step.

4. wsgen command

Document style requires extra classes to run, you can use "**wsgen**" to generate all necessary Java artifacts (mapping classes, wsdl or xsd schema). The "**wsgen**" command is required to read a service endpoint implementation class :

```
wsgen -keep -cp . com.mkyong.ws.HelloWorldImpl
```

It will generate two classes, copy it to your "**package.jaxws**" folder.

File : GetHelloWorldAsString.java

```java
package com.mkyong.ws.jaxws;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlType;

@XmlRootElement(name = "getHelloWorldAsString", namespace = "http://ws.mkyong.com/")
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "getHelloWorldAsString", namespace = "http://ws.mkyong.com/")
public class GetHelloWorldAsString {

    @XmlElement(name = "arg0", namespace = "")
    private String arg0;

    /**
     *
     * @return
     *     returns String
     */
    public String getArg0() {
        return this.arg0;
    }

    /**
     *
     * @param arg0
     *     the value for the arg0 property
     */
    public void setArg0(String arg0) {
        this.arg0 = arg0;
    }

}
```

File : GetHelloWorldAsStringResponse.java

```java
package com.mkyong.ws.jaxws;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;
import javax.xml.bind.annotation.XmlType;

@XmlRootElement(name = "getHelloWorldAsStringResponse", namespace = "http://ws.mkyong.com/")
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "getHelloWorldAsStringResponse", namespace = "http://ws.mkyong.com/")
public class GetHelloWorldAsStringResponse {

    @XmlElement(name = "return", namespace = "")
    private String _return;

    /**
     *
     * @return
     *     returns String
     */
    public String getReturn() {
        return this._return;
    }

    /**
     *
     * @param _return
     *     the value for the _return property
     */
    public void setReturn(String _return) {
        this._return = _return;
    }

}
```

Note
The “wsgen” tool is available in the “JDK_Path\bin\” folder. For detail, please read this JAX-WS : wsgen tool example article.

5. Done

Done, publish it and test it via URL : *http://localhost:9999/ws/hello?wsdl*.

Web Service Client

Create a web service client to access your published service.

File : HelloWorldClient.java

```
package com.mkyong.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");
        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("mkyong"));

    }

}
```

Output

```
Hello World JAX-WS mkyong
```

Tracing SOAP Traffic

From top to bottom, showing how SOAP envelope flows between client and server in this document style web service.

1. Request a WSDL file

First, client send a wsdl request to service endpoint :

Client send request :

```
GET /ws/hello?wsdl HTTP/1.1
User-Agent: Java/1.6.0_13
Host: localhost:9999
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive
```

Server send response :

```
HTTP/1.1 200 OK
Transfer-encoding: chunked
Content-type: text/xml;charset=utf-8

<?xml version="1.0" encoding="UTF-8"?>
    <!-- Published by JAX-WS RI at http://jax-ws.dev.java.net.
        RI's version is JAX-WS RI 2.1.1 in JDK 6. -->
    <!-- Generated by JAX-WS RI at http://jax-ws.dev.java.net.
        RI's version is JAX-WS RI 2.1.1 in JDK 6. -->
<definitions
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:tns="http://ws.mkyong.com/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    targetNamespace="http://ws.mkyong.com/"
    name="HelloWorldImplService">
<types>
<xsd:schema>
    <xsd:import namespace="http://ws.mkyong.com/"
        schemaLocation="http://localhost:9999/ws/hello?xsd=1"></xsd:import>
</xsd:schema>
</types>

<message name="getHelloWorldAsString">
    <part name="parameters" element="tns:getHelloWorldAsString"></part>
</message>
<message name="getHelloWorldAsStringResponse">
    <part name="parameters" element="tns:getHelloWorldAsStringResponse"></part>
</message>

<portType name="HelloWorld">
    <operation name="getHelloWorldAsString">
        <input message="tns:getHelloWorldAsString"></input>
        <output message="tns:getHelloWorldAsStringResponse"></output>
    </operation>
</portType>

<binding name="HelloWorldImplPortBinding" type="tns:HelloWorld">

    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document">
    </soap:binding>
    <operation name="getHelloWorldAsString">
        <soap:operation soapAction=""></soap:operation>
        <input>
            <soap:body use="literal"></soap:body>
        </input>
        <output>
            <soap:body use="literal"></soap:body>
        </output>
    </operation>

</binding>

<service name="HelloWorldImplService">

<port name="HelloWorldImplPort" binding="tns:HelloWorldImplPortBinding">

<soap:address location="http://localhost:9999/ws/hello"></soap:address>

</port>
</service>
</definitions>
```

2. getHelloWorldAsString(String name)

A second call, client put method invoke request in SOAP envelope and send it to service endpoint. At the service endpoint, call the requested method and put the result in a SOAP envelope and send it back to client.

Client send request :

```
POST /ws/hello HTTP/1.1
SOAPAction: ""
Accept: text/xml, multipart/related, text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Content-Type: text/xml; charset=utf-8
User-Agent: Java/1.6.0_13
Host: localhost:9999
Connection: keep-alive
Content-Length: 224

<?xml version="1.0" ?>
    <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
        <S:Body>
            <ns2:getHelloWorldAsString xmlns:ns2="http://ws.mkyong.com/">
                <arg0>mkyong</arg0>
            </ns2:getHelloWorldAsString>
        </S:Body>
    </S:Envelope>
```

Server send response :

```
HTTP/1.1 200 OK
Transfer-encoding: chunked
Content-type: text/xml; charset=utf-8

<?xml version="1.0" ?>
    <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
        <S:Body>
            <ns2:getHelloWorldAsStringResponse xmlns:ns2="http://ws.mkyong.com/">
                <return>Hello World JAX-WS mkyong</return>
            </ns2:getHelloWorldAsStringResponse>
        </S:Body>
    </S:Envelope>
```

Download Source Code

Download It – JAX-WS-HelloWorld-Document-Example.zip (10KB)

hello world | jax-ws | web services

About the Author

mkyong

Founder of Mkyong.com, love Java and open source stuff. Follow him on Twitter, or befriend him on Facebook or Google Plus. If you like my tutorials, consider make a donation to these charities.