

Contract last approach for Soap Web Services with JAX_WS

Web services (from wikipedia):

A **Web service** is a method of communication between two electronic devices over [World Wide Web](#). A **web service** is a software function provided at a network address over the web or the cloud; it is a service that is "always on" as in the concept of [utility computing](#).

The [W3C](#) defines a "Web service" as:

[...] a software system designed to support [interoperable](#) machine-to-machine interaction over a [network](#). It has an interface described in a machine-processable format (specifically [WSDL](#)). Other systems interact with the Web service in a manner prescribed by its description using [SOAP](#) messages, typically conveyed using [HTTP](#) with an [XML serialization](#) in conjunction with other Web-related standards.^[1]

XML web services use Extensible Markup Language ([XML](#)) messages that follow the [SOAP](#) standard and have been popular with the traditional enterprises. In such systems, there is often a machine-readable description of the operations offered by the service written in the [Web Services Description Language](#) (WSDL). The latter is not a requirement of a SOAP *endpoint*, but it is a prerequisite for automated [client-side](#) code generation in many [Java](#) and [.NET](#) SOAP frameworks (frameworks such as [Apache Axis2](#), [Apache CXF](#), [Spring](#), [gSOAP](#) being notable exceptions). Some industry organizations, such as the [WS-I](#), mandate both SOAP and WSDL in their definition of a web service.

Lab work objectives

Interoperability regarding web services allows using different languages, operating system..., on both sides (server and client). Nevertheless, here, only JAX_WS is used: a Java library that makes Web Service development easier.

Furthermore, the contract last approach is used (code first then generate de WSDL). See the course for the contract first approach.

Question: what is the difference between the contract last approach and the contract first one?

The JDK version of Java includes a web server, so no external web server like Apache is required.

Required material

- Java JDK only version 1.6 or higher, ... (not JRE)
- Eclipse version JEE developer
- Gradle

- JAX WS

JAX WS can be managed (download, integrated into the project...) automatically with Gradle: simply create a Gradle project and use the following build.gradle file (see previous lab work about Gradle).

```
apply plugin: 'java'
apply plugin: 'eclipse'

repositories {
    mavenCentral()
}

dependencies {
    compile group: 'javax.xml.ws', name: 'jaxws-api', version: '2.2.11'
    testCompile group: 'junit', name: 'junit', version: '4.+'
}
```

Java project creation

Use the Spring Initializer to create a Gradle project:

<https://start.spring.io/>

Build the project with:

- gradlew build under windows
- ./gradlew build under Linux

Convert the project in an Eclipse project:

- gradlew eclipse under windows
- ./gradlew eclipse under Linux

Then import the project inside Eclipse.

JAX WS is provided as a library. To add the library to the project edit the build.gradle file and add the following line to the dependency part:

```
compile group: 'javax.xml.ws', name: 'jaxws-api', version: '2.2.11'
```

Update the Gradle project with:

- gradlew build under windows
- ./gradlew build under Linux

Update the Eclipse project with:

- gradlew eclipse under windows
- ./gradlew eclipse under Linux

Then refresh Eclipse (File -> Refresh).

Server side code

Three files are required:

- A java interface for the web service definition (a list of methods)
- The class that implements the interface
- A main program that starts the web server and publishes the web service.

Here is an example of an interface:

```
package servicesWeb;

import javax.jws.WebMethod;
import javax.jws.WebService;

@WebService
public interface MyServiceWeb {

    @WebMethod
    public void createEntity();

}
```

Annotations (`WebService` and `WebMethod`) are used by JAX WS to generate automatically additional code for the web service publishing (use as import `javax.jws...`). The class implementing the web service can be deduced from the interface (take care at the package name in the `WebService` annotation):

```
package servicesWeb;

import javax.jws.WebService;

@WebService(endpointInterface = "servicesWeb.MyServiceWeb")
public class MyServiceWebImpl implements MyServiceWeb{
    // ...

}
```

Publishing a web service is very simple with JAX_WS (add those lines into the main program):

```
MyServiceWeb myServiceWeb = new MyServiceWebImpl();
Endpoint.publish("http://localhost:8080/WS/MonServiceWeb",myServiceWeb);
```

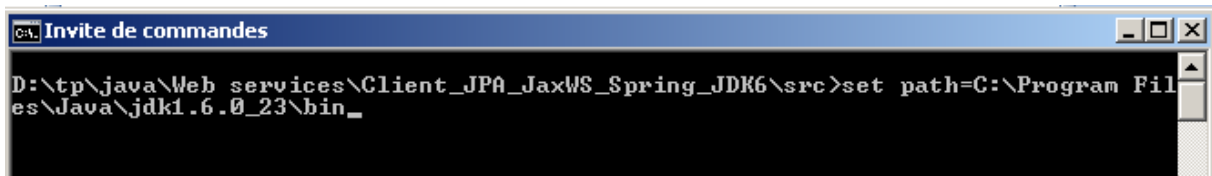
It will start a web server and install the web service. Put those lines into a main program (add `javax.xml.ws.Endpoint` as import). The service is accessible at the given URL. A WSDL description generated in XML is accessible at:

<http://localhost:8080/WS/MyServiceWeb?wsdl>

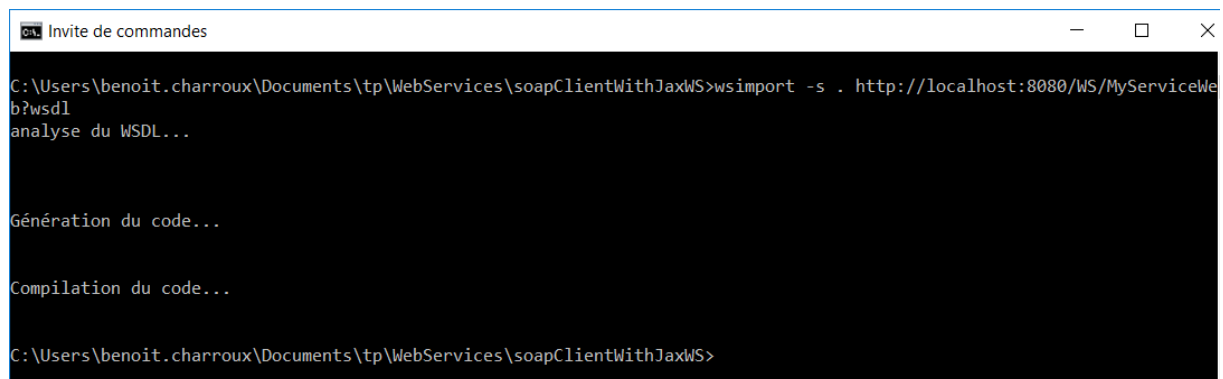
Question: analyze the wsdl file and understand the different parts.

Client side coding

JAX WS is able to parse remotely the WSDL and to generate helper classes that make client side code writing easier. Indeed, the JDK includes a command wsimport.



```
C:\> Invite de commandes
D:\tp\java\Web_services\Client_JPA_JaxWS_Spring_JDK6\src>set path=C:\Program Files\Java\jdk1.6.0_23\bin_
```



```
C:\Users\benoit.charroux\Documents\tp\WebServices\soapClientWithJaxWS>wsimport -s . http://localhost:8080/WS/MyServiceWeb?wsdl
analyse du WSDL...

Génération du code...

Compilation du code...

C:\Users\benoit.charroux\Documents\tp\WebServices\soapClientWithJaxWS>
```

If the network uses a proxy, an option should be added to the wsimport command:

-httpproxy:host:port

Add the generated code to a new Gradle project.

Then use the following program to call the Web Service.

```
public class Main {

    public static void main(String[] args) {
        MyServiceWebImplService serviceImpl = new MyServiceWebImplService();
        MyServiceWeb myServiceWeb = serviceImpl.getMyServiceWebImplPort();

        myServiceWeb. ... // method call
    }
}
```

Client test with SOAPUI

There are many tools to test web services.

SOAPUI is one of them. It allows to generate SOAP messages from the WSDL file, and to exchange messages with the Web Service.

Work to be done

Define, write and test your own WebService.