Web Service labs
(with JAVA JAX-WS)

Chantal Taconet

Revision : 283

ASR/CSC5002
This lab is to discover Web Services, mainly in java.
— You have to get ready with JAX-WS (included in the JDK), for this purpose several steps are proposed.
1. Test a simple java application (client and server side).
2. Write a client for an existing remote server.
3. Run the printer application presented in the Lecture notes.
4. Test the multi-language facility of web service with clients in php.
5. Test the distribution of the client and server inside the classroom.
6. Finally, implement a new service and a new client from scratch

1 Simple application

The time service returns the current time of a system. The interface has two methods, the first method returns the time in string and the second one in integer format (number of seconds since the 1st of january 1970).

The server side
1. Create a directory for the SOAP examples.
2. From this directory unzip the example SOAP archive
3. Create two terminals, one for the server and one for the client
4. On the server terminal, go to the TimeServer directory:
   ■ Compile the server
     ```
     cd
     cd CSC5002/TimeService/TimeServiceServer
     mvn clean install
     ```
   ■ Take a look at the generated classes and wsdl
     ```
     tree target/generated-sources/
     target/generated-sources/
     --annotations
     --wsl
     --TimeServiceImplService_schema1.xsd
     --TimeServiceImplService.wsdl
     --wsgen
     --time
     --server
     --jaxws
     --GetTimeAsElapsed.java
     --GetTimeAsElapsedResponse.java
     --GetTimeAsString.java
     --GetTimeAsStringResponse.java
     ```
   ■ Start the service
     ```
     mvn exec:java@server
     ```
   ■ Verify in a navigator (or with curl) that the service is started and obtain the WSDL with this URL (you can change the URL in the TimeServerPublisher.java file)
     ```
     http://localhost:7777/TimeService?wsdl
     ```
   ■ You can use the lsof command to get information on the process listening to a specific TCP port
Web Service labs (with JAVA JAX-WS)

lsoci -i :7777

Which provides information such as

<table>
<thead>
<tr>
<th>COMMAND</th>
<th>PID</th>
<th>USER</th>
<th>FD</th>
<th>TYPE</th>
<th>DEVICE</th>
<th>SIZE/OFF</th>
<th>NODE</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>java</td>
<td>12225</td>
<td>taconet</td>
<td>8</td>
<td>IPv6</td>
<td>915222</td>
<td>0</td>
<td>TCP</td>
<td>localhost:7777 LISTEN</td>
</tr>
</tbody>
</table>

You can use this command to see if you have a process on a given port.

You can also use

netstat -an | grep 7777

Which provides

<table>
<thead>
<tr>
<th>Proto Recv-Q Send-Q Adresse locale Adresse distante Etat</th>
</tr>
</thead>
<tbody>
<tr>
<td>tcp6 0 0 127.0.0.1:7777 :::* LISTEN</td>
</tr>
</tbody>
</table>

The client side

We are going to test a simple java client using the time service.

1. With the knowledge of the time service WSDL URL, we can generate with the wsimport tool the Time Service stub, which represents the service on the client side.

   cd CSC5002/TimeService/TimeServiceClient
   mvn clean install

   tree target/generated-sources/
   target/generated-sources/
      --annotations
      --jaxws-wsimport
      --time
         --generated
            --GetTimeAsElapsed.java
            --GetTimeAsElapsedResponse.java
            --GetTimeAsString.java
            --GetTimeAsStringResponse.java
            --ObjectFactory.java
            --package-info.java
            --TimeServiceImplService.java
            --TimeService.java

2. Test the client

   mvn exec:java@client

   You should see the result of the Web Service : the time obtained from the server.
   It’s a great result : one operational client/server web service.

Examine the wsdl

Take a look (for example with the eclipse wsdl editor available with the eclipse Web Tool Platform) at the wsdl (TimeServiceServer/target/generated-sources/wsdl directory). You find :

   • The service description : TimeServiceImplService.wsdl (look at the message, portType, operation and bindings XML elements) and
The associated types in TimeServiceImplService_schema1.xsd, notice that the input parameters and return are all defined as complex types.
And understand the generated java classes (in target/generated-sources/jaxws-wsimport/time/generated/ directory), there is one class by message and complex type defined in the schema for input and output parameters.

```java
--GetTimeAsElapsedTime.java
--GetTimeAsElapsedTimeResponse.java
--GetTimeAsString.java
--GetTimeAsStringResponse.java
```

2 Make the printer example work

From the printer example directory ((CSC5002)/PrinterService) the following files are available.

```text
--PrinterClient
--pom.xml
--src/main/java
  --PrinterClient.java
--PrinterClientPhp
  --PrinterClient.php
--PrinterServer
  --pom.xml
  --src
    --main
      --java
        --printer
          --JobInfo.java
          --PrintDenied.java
          --PrinterInfo.java
          --PrinterServer.java
          --PrinterServiceImpl.java
          --PrinterService.java
--README.txt
```

Both sides (client and server) include a pom.xml to

- Produce automatically all the files that can be generated

`mvn clean install`

- Run the client and the server.

`mvn exec:java@server`

`mvn exec:java@client`

The first lines of those build files may be modified for your own web services and clients.

The server side

```text
cd
cd CSC5002/PrinterService/PrinterServer
mvn clean install
mvn exec:java@server
```
Web Service labs (with JAVA JAX-WS)

The client side

mvn clean install
mvn exec:java@client

3 Distribution of web service

We are going to test the distribution of the web service: Client and server on different computers (due to
firewall this facility is only possible inside a classroom).

With ssh connect yourself to another computer. Start your PrinterServer. From the initial computer start a
PrinterClient which connects itself to the remote PrinterServer. Be careful, the server is published at another URL,
modify the pom.xml accordingly.

4 Multi-language facility of web service

Web Service is multi-language. Client and servers may be written in different languages.

To illustrate this capability, we are going to:

- Test a php version of the time client (sorry there is a problem with the - character in the php listings
  below) TimeClient.php

```php
<?php
try {
    $webservice = new SoapClient("http://127.0.0.1:7777/TimeService?wsdl");
    $result = $webservice->getTimeAsString();
    print "getTimeAsString():\n" . $result . "\n;
} catch (SoapFault $fault) {
    echo "<h2>Error :</h2><pre>
    print_r($fault);
    echo "</pre>";
    die;
}
?>
```

php TimeClient.php

- Test a php version of the printer PrinterClient.php

```php
<?php
try {
    $count = 0;
    $webservice = new SoapClient("http://127.0.0.1:9000/PrinterService?wsdl");
    // Be careful, parameter names have to be defined with the @WebParam annotation if not it is arg0
    // only arg0 is working as far as now ...
    $result = $webservice->submitPrint(array("arg0" => "/etc/hosts"));
    print "submitPrint(): Current Job=\n" . $result . "\n;"
    $result = $webservice->getPrinterInfo();
    print "getPrinterInfo(): Current Job=\n" . $result . "\n;"
    $result = $webservice->getPrinterInfo();
    print "getPrinterInfo(): Current Job=\n" . $result . "\n;"
} catch (SoapFault $fault) {
    echo "<h2>Error :</h2><pre>
    print_r($fault);
    echo "</pre>";
    die;
}
?>
```

php PrinterClient.php

5 Implement a new service and a new client from scratch

To verify that your experiences allow you to implement a WebService you are asked to implement a calculator
with a memory. The operations to declare and implement are the following:

- store: store a float in memory
- add: add a float to the value in memory
Web Service labs (with JAVA JAX-WS)

- **multiply**: multiply a float with the value in memory
- **power**: put the value in memory to the power of the integer in parameter

The organization of the files should look like:

```
.
--CalculatorClient
  --pom.xml
  `-- src/main/java
    `-- CalculatorClient.java

--CalculatorServer
  --pom.xml
  `-- src/main/java
    `-- calculator
      `--Calculator.java
      `--CalculatorServer.java
      `-- CalculatorService.java
```

Note that the Client and Server are in two separate directories. It highlights that it may be implemented by two different teams.