



---

# Lecture course notes

# Web services Computing

Neila BEN LAKHAL (PhD @ Tokyo Institute of Technology)  
By [Neila.benlakhal@enicarthage.rnu.tn](mailto:Neila.benlakhal@enicarthage.rnu.tn)

## Course Introduction

neila.benlakhal@gmail.com





## Outils de travail à préparer pour le cours SOA

---

### ● Outils de tests :

- SOAPUI (<http://www.soapui.org/>)
- POSTMAN(<https://www.getpostman.com/>)

### ● Environnement de développement

- Wamp Server (<http://www.wampserver.com/en/>)
- Nusoap Library (<http://sourceforge.net/projects/nusoap/>)
- PhP5+

### ● Éditeur qui supporte php et XML/WSDL comme Notepad++, Eclipse, Brackets, atom, sublimeText, etc.



## Class website

<https://sites.google.com/view/neila/>



## *Implement my first web service*

[neila.benlakhal@gmail.com](mailto:neila.benlakhal@gmail.com)



## **Approaches to building services**

○ There are two approaches to building services:

Approach 1: Bottom-up approach

Approach 2: Top-down approach

## **TOP-DOWN Approach**

- In top-down or contract-first development, the service interface is the starting point for developing services.
- A framework is used to generate a code skeleton based on the service WSDLs and XSDs. The generated implementation can then be completed.
- This can be a good approach when there is no service implementation yet in place.

## **Bottom-up Approach**

- Bottom-up development works the other way around; the interface is generated based on an existing implementation thereby exposing its functionality.
- Bottom-up development can be a fast way of creating interfaces but gives you less control over the exact definition of the interface.

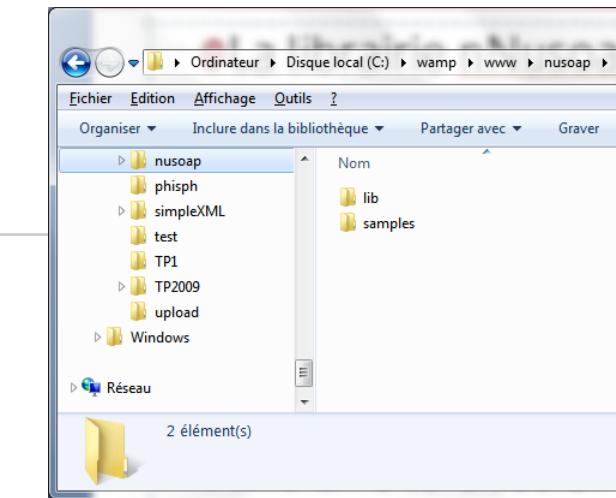


## **Implementing a bottom-up Service**



## Approche 1 : Bottom-up

- Téléchargement de Nusoap (NuSOAP - SOAP Toolkit for PHP) : nusoap-0.9.5.zip  
<http://sourceforge.net/projects/nusoap/>
- Extraire le dossier compressé dans le dossier www de votre serveur WAMP
- Renommer le dossier compressé ; nusoap
- Une fois la librairie téléchargée et placée dans un sous répertoire où va se trouver votre fichier webservice, nous allons pouvoir commencer à voir comment créer votre web service.



- La librairie **Nusoap** est un ensemble de classes Php qui permettent la création et la consommation de WS basé sur SOAP 1.1, WSDL 1.1 et HTTP 1.0/1.1
- La librairie NUSOAP est sous license GNU en PHP4. Elle a été développée par NuSphere et Dietrich Ayala.

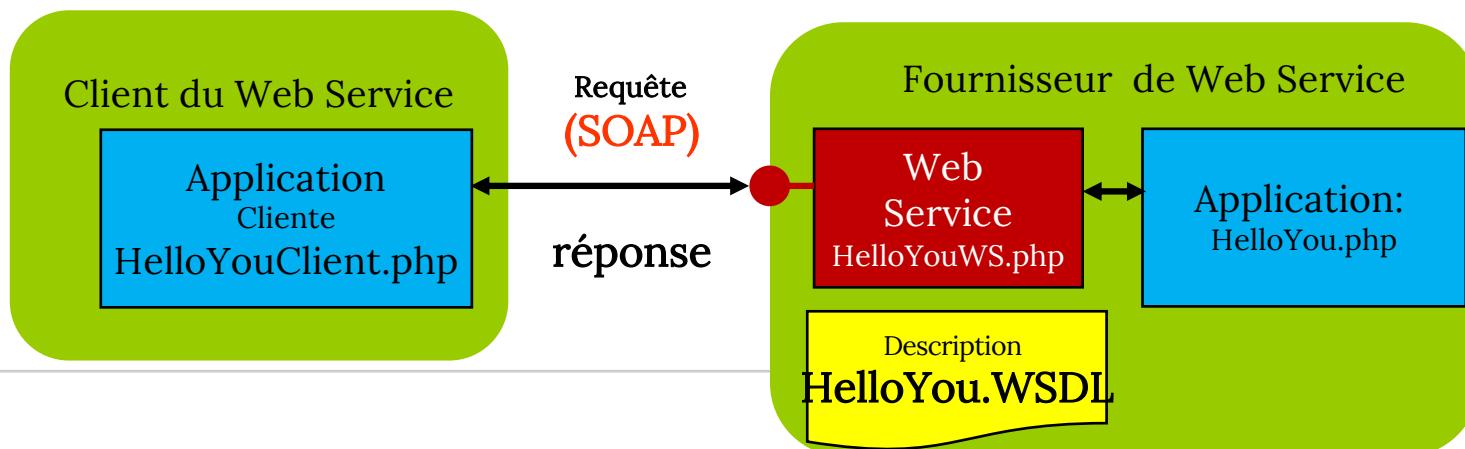
## L'approche bottom-up

○ Initialement, on a :

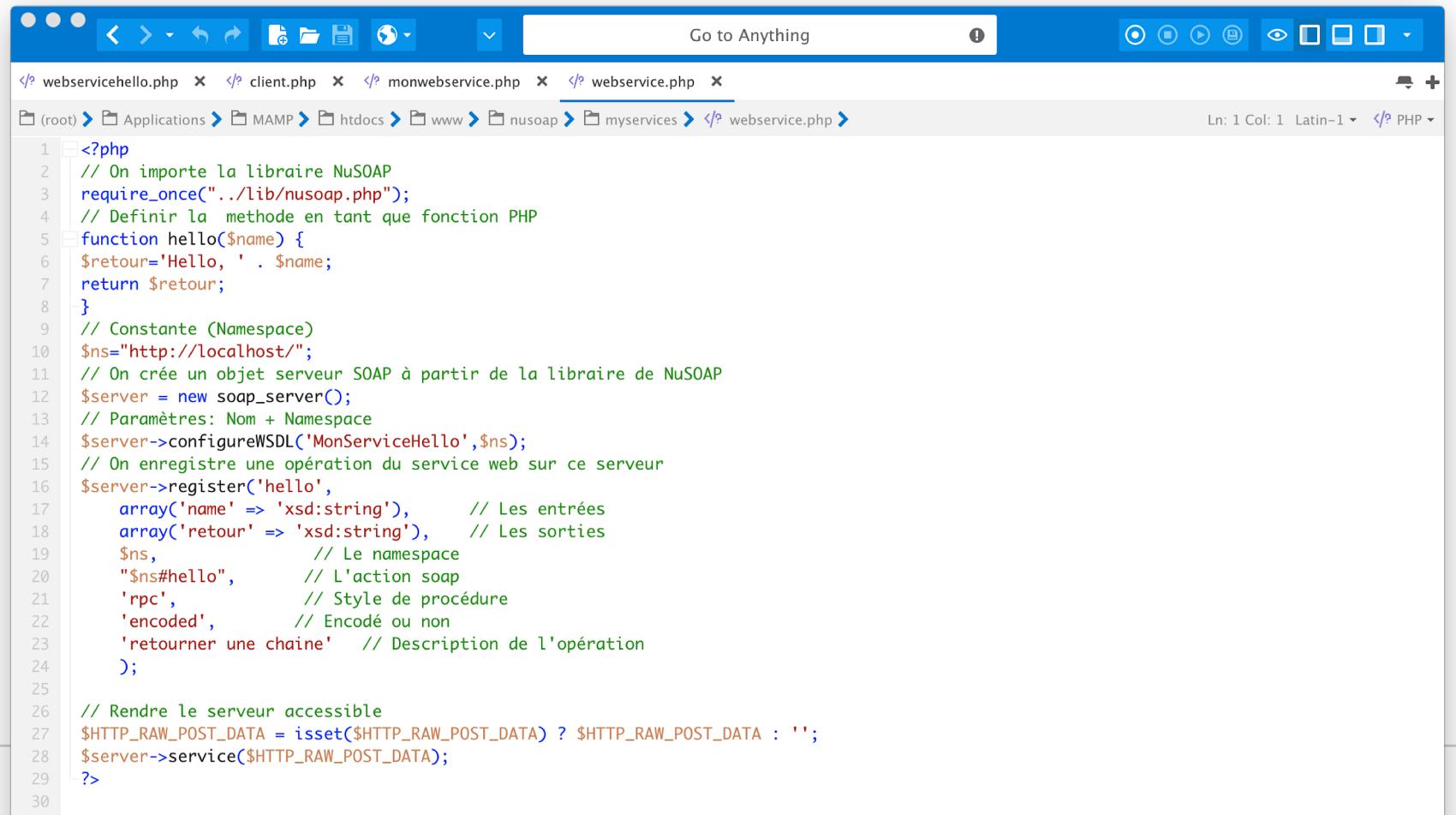
- Un script .php (une fonction par exemple)

○ À préparer:

- Web service: webservice.php
- Application cliente: Client.php
- .WSDL: il est généré automatiquement par Nusoap



# Webservice.php



```
1 <?php
2 // On importe la librairie NuSOAP
3 require_once("../lib/nusoap.php");
4 // Definir la methode en tant que fonction PHP
5 function hello($name) {
6     $retour='Hello, ' . $name;
7     return $retour;
8 }
9 // Constante (Namespace)
10 $ns="http://localhost/";
11 // On crée un objet serveur SOAP à partir de la librairie de NuSOAP
12 $server = new soap_server();
13 // Paramètres: Nom + Namespace
14 $server->configureWSDL('MonServiceHello',$ns);
15 // On enregistre une opération du service web sur ce serveur
16 $server->register('hello',
17     array('name' => 'xsd:string'),      // Les entrées
18     array('retour' => 'xsd:string'),    // Les sorties
19     $ns,                                // Le namespace
20     "$ns#hello",                         // L'action soap
21     'rpc',                               // Style de procédure
22     'encoded',                            // Encodé ou non
23     'retourner une chaîne'   // Description de l'opération
24 );
25
26 // Rendre le serveur accessible
27 $HTTP_RAW_POST_DATA = isset($HTTP_RAW_POST_DATA) ? $HTTP_RAW_POST_DATA : '';
28 $server->service($HTTP_RAW_POST_DATA);
29 ?>
30
```

neila.benlakhal@gmail.com

12

## l'approche 'bottom-up' : création du Web service à partir d'une fonction php

- Vous devez créer un fichier pour votre web service, nous allons le nommer par exemple webservice.php :

- Syntaxe:

```
<?php  
//on inclut la librairie nécessaire pour mettre en place le webservice  
require_once("../lib/nusoap.php");  
//on initialise un nouvel objet serveur  
$server = new soap_server();  
// on configure en donnant un nom et un Namespace  
$server -> configureWSDL('nomDuWebservice','Namespace');  
?>
```

webservice.php

- Votre web service est créée, il vous faut maintenant ajouter des méthodes, et le faire communiquer avec les différents clients.

## Syntaxe: Crédit des l'approche 'bottom-up' : création du Web service à partir d'une fonction php

```
<?php //nous créons ici la fonction - à rajouter à la suite du code
      //de la page précédente
function nomfonction($parametre){
    //---code de la fonction ici---
    Return $retour;
    //on enregistre la méthode grâce à register
    $server->register('nomfonction', //nom de la méthode
                        array('parametre'=>'xsd:string'), //entrée(s) et type(s)
                        array('retour'=>'xsd:string'), //valeur de retour et type
                        'Namespace'); //espace de nommage
    // Rendre le serveur accessible
    $HTTP_RAW_POST_DATA = isset($HTTP_RAW_POST_DATA) ?
    $HTTP_RAW_POST_DATA : "";
    $server->service($HTTP_RAW_POST_DATA);
?>
```

webservice.php



## Exemple

- Exemple : l'approche bottom-up pour créer un web service contenant la fonction suivante **hello.php** qui prend en argument une chaîne de caractères et qui la renvoie:

hello.php

```
<?php
function hello($name) {
$retour='Hello, ' . $name;
return $retour;
}?>
```

- Étape 1: Développer le web service (**webservice.php**) qui contient la fonction **hello.php**.
- Étape 2: Déployer le WS dans le serveur d'application ( l'enregistrer dans le dossier WWW de WAMP Server)
- Étape 3: Développer un client (**client.php**) qui l'appellera.



## Webservice.php : explication 1/5

- Importer la librairie NuSOAP. Celle-ci permettra à PHP de communiquer à travers le protocole SOAP :

```
require_once('../lib/nusoap.php');
```

- Ajouter la fonction **hello** qui prend en argument une chaîne de caractères et qui la renvoie:

```
function hello($name)
{
    $retour='Hello, ' . $name;
    return $retour;}
```



## Webservice.php : explication 2/5

Configurer le serveur SOAP : Cette étape fait appelle à la librairie NuSOAP que nous avons importé :

- Créer une constante \$ns pour le namespace pour des fins de clarté (on pourrait aussi réécrire le namespace au complet à chaque fois qu'il est nécessaire) :

```
$ns="http://localhost/";
```

- Créer un objet soap\_server que nous appellerons \$server:

```
$server = new soap_server();
```



## Webservice.php : explication 3/5

- Configuration de notre serveur SOAP:
- Configurer l'objet \$server pour qu'il offre une interface WSDL afin que l'on puisse communiquer avec lui et donc accéder aux services que nous ajouterons à la prochaine étape:
- Pour ce faire, nous devons donner un nom au WSDL et son namespace (référencé par \$ns):

```
// Paramètres: Nom du service + Namespace  
$server->configureWSDL('MonServiceHello',$ns);
```

- Ainsi, NUSOAP va en fait générer automatiquement pour nous le fichier .wsdl de ce Web service.



## Webservice.php : explication 4/5

- On enregistre une opération du service web sur ce serveur :

```
$server->register('hello',
    array('name' => 'xsd:string'), // Les entrées
    array('retour' => 'xsd:string'), // Les sorties
    $ns, // Le namespace
    "$ns#hello", // L'action soap
    'rpc', // Style de procédure
    'encoded', // Encodé ou non
    'retourner une chaîne' // Description de l'opération
);
```



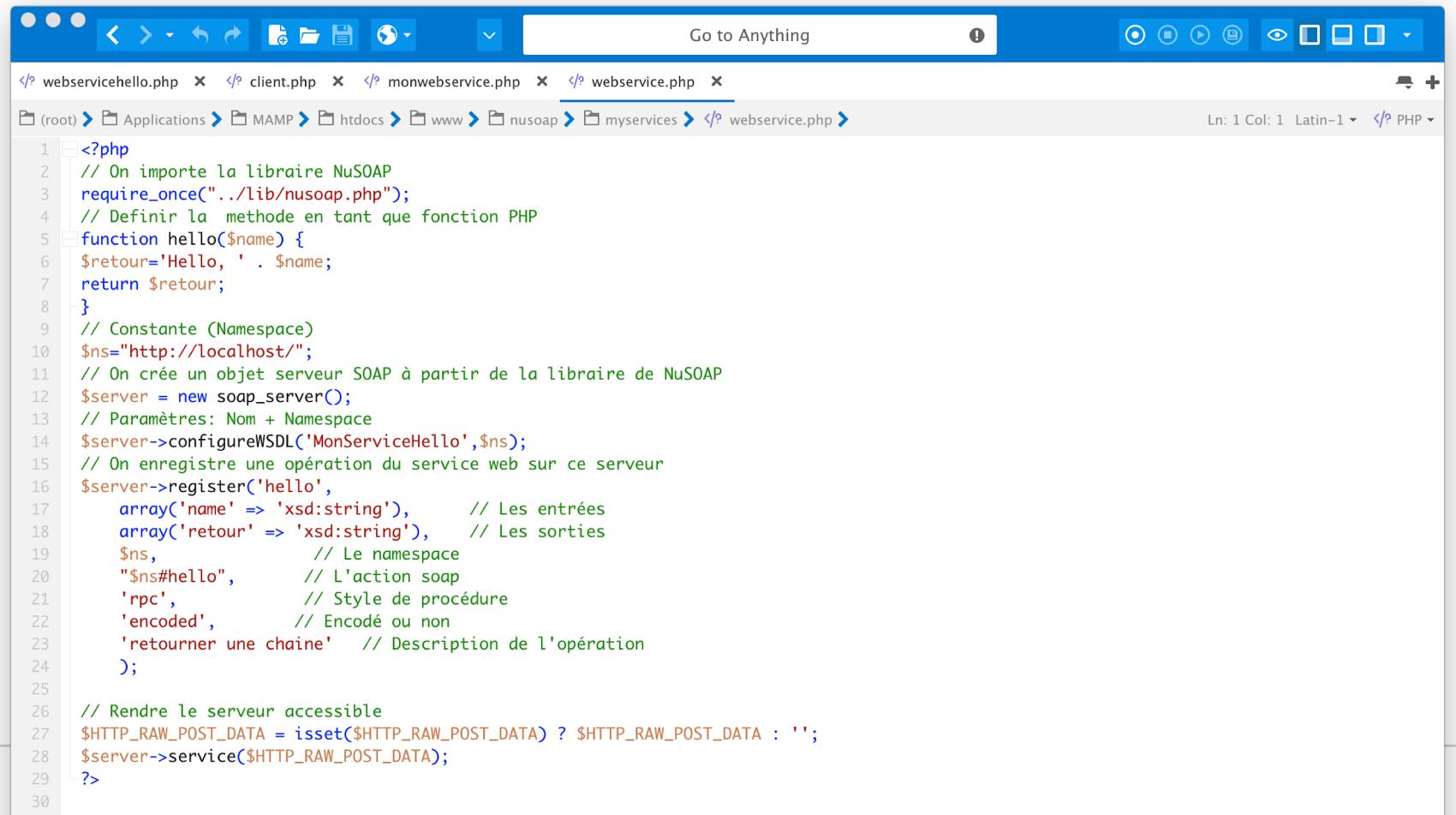
## Webservice.php : explication 5/5

- Rendre le service accessible sur le serveur :

```
$HTTP_RAW_POST_DATA =  
isset($HTTP_RAW_POST_DATA) ?  
$HTTP_RAW_POST_DATA : '';  
$server->service($HTTP_RAW_POST_DATA);
```

- Tout ce que PHP ne peut parser est considéré comme une variable contenue dans la méthode GET ou POST et est placé en RAW format(tel qu'il est) dans la variable super global \$HTTP\_RAW\_POST\_DATA Le XML en fait partie.

# Webservice.php



```
1 <?php
2 // On importe la librairie NuSOAP
3 require_once("../lib/nusoap.php");
4 // Definir la methode en tant que fonction PHP
5 function hello($name) {
6     $retour='Hello, ' . $name;
7     return $retour;
8 }
9 // Constante (Namespace)
10 $ns="http://localhost/";
11 // On crée un objet serveur SOAP à partir de la librairie de NuSOAP
12 $server = new soap_server();
13 // Paramètres: Nom + Namespace
14 $server->configureWSDL('MonServiceHello',$ns);
15 // On enregistre une opération du service web sur ce serveur
16 $server->register('hello',
17     array('name' => 'xsd:string'),      // Les entrées
18     array('retour' => 'xsd:string'),    // Les sorties
19     $ns,                                // Le namespace
20     "$ns#hello",                         // L'action soap
21     'rpc',                               // Style de procédure
22     'encoded',                            // Encodé ou non
23     'retourner une chaîne'   // Description de l'opération
24 );
25
26 // Rendre le serveur accessible
27 $HTTP_RAW_POST_DATA = isset($HTTP_RAW_POST_DATA) ? $HTTP_RAW_POST_DATA : '';
28 $server->service($HTTP_RAW_POST_DATA);
29 ?>
```

## Étape 2: Déploiement du service Web

- Le déploiement d'un service web en PHP est très simple, il suffit de placer le fichier webservice.php dans le répertoire contenant toute le contenu web de notre serveur HTTP (le dossier WWW de WAMP dans notre cas).
  
- Le chemin d'accès du web service déployé: par exemple:  
<c:/wamp/www/nusoap/myservices/webservice.php>  
correspond à:  
<http://localhost/nusoap/myservices/webservice.php>



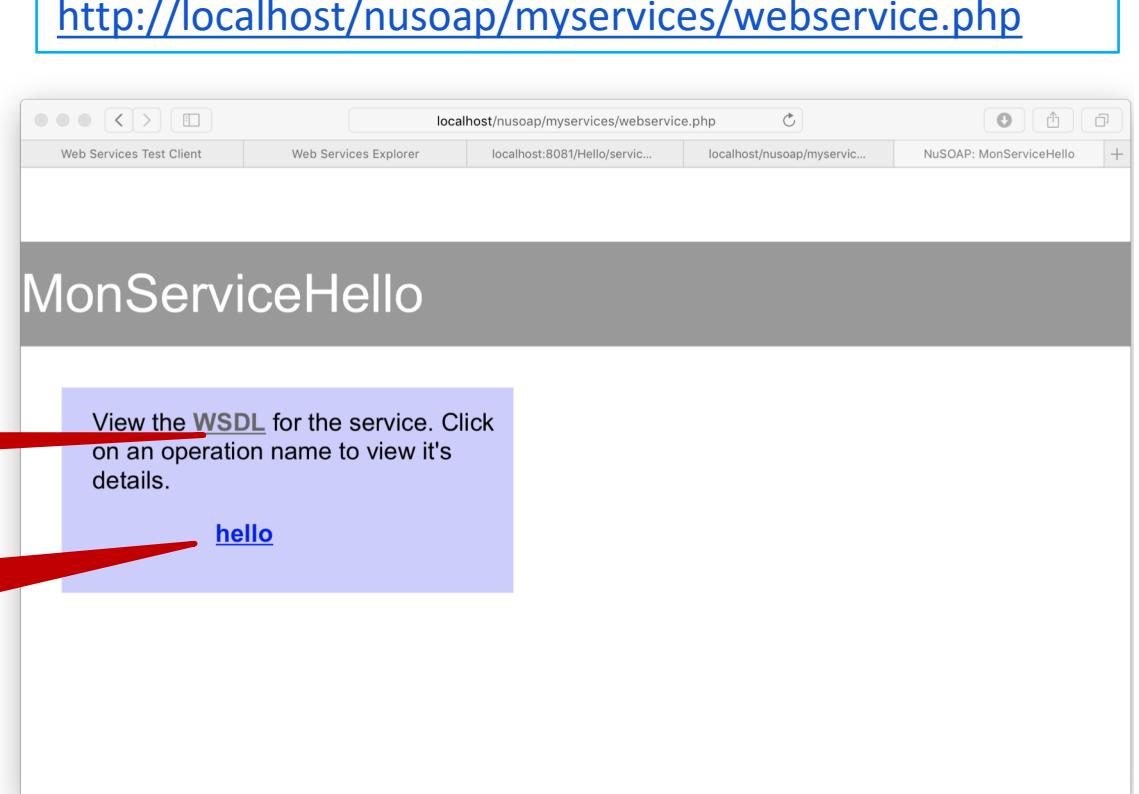
## Visionnement dans un explorateur web

○ Lorsqu'on lance ce fichier de service (PAS une application) dans un explorateur web (simplement en tapant son adresse), nous obtenons la page d'en face, générée automatiquement par la librairie NuSOAP.

**un lien vers le WSDL du web**

**un lien vers une page de documentation pour chacune des opérations offertes par le service**

<http://localhost/nusoap/myservices/webservice.php>



## http://localhost/nusoap/myservices/webservice.php?wsdl

The screenshot shows a web-based interface for testing web services. At the top, there's a blue header bar with the URL 'http://localhost/nusoap/myservices/webservice.php?wsdl'. Below the header is a toolbar with standard browser controls (back, forward, search). The main content area has tabs: 'Web Services Test Client' (selected), 'Web Services Explorer', and several others like 'localhost:8081/Hello/services/Hello?w...', 'localhost/nusoap/myservices/Axiscli...', and 'localhost/nusoap/myservices/webser...'. A '+' button is also visible.

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<definitions xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:tns="http://localhost/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://localhost/">
  <types>
    <xsd:schema targetNamespace="http://localhost/">
      <xsd:import namespace="http://schemas.xmlsoap.org/soap/encoding/" />
      <xsd:import namespace="http://schemas.xmlsoap.org/wsdl/" />
    </xsd:schema>
  </types>
  <message name="helloRequest">
    <part name="name" type="xsd:string"/>
  </message>
  <message name="helloResponse">
    <part name="retour" type="xsd:string"/>
  </message>
  <portType name="MonServiceHelloPortType">
    <operation name="hello">
      <documentation>retourner une chaine</documentation>
      <input message="tns:helloRequest"/>
      <output message="tns:helloResponse"/>
    </operation>
  </portType>
  <binding name="MonServiceHelloBinding" type="tns:MonServiceHelloPortType">
    <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
    <operation name="hello">
      <soap:operation soapAction="http://localhost/#hello" style="rpc" />
      <input>
        <soap:body use="encoded" namespace="http://localhost/" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
      </input>
      <output>
        <soap:body use="encoded" namespace="http://localhost/" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
      </output>
    </operation>
  </binding>
  <service name="MonServiceHello">
    <port name="MonServiceHelloPort" binding="tns:MonServiceHelloBinding">
      <soap:address location="http://localhost/nusoap/myservices/webservice.php" />
    </port>
  </service>
</definitions>
```

## **Build a bottom-up Web service in Java**

- Download Java 2EE (WTP+Axis plugin)
- Download Apache tomcat server (9.0)
  
- Resources :
  - [https://www.eclipse.org/webtools/community/tutorials/BottomUpAxis2WebService/bu\\_tutorial.html](https://www.eclipse.org/webtools/community/tutorials/BottomUpAxis2WebService/bu_tutorial.html)
  - <https://crunchify.com/create-and-deploy-simple-web-service-and-web-service-client-in-eclipse/>
- PDF available on course web site.

## Steps

- Start by creating a dynamic Web project
- Choose apache tomcat (already installed and running) as a server
- Create a Java class
- Select class src file and Create a Web service;choose Axis as a implementation; part of WTP in eclipse j2EE.
- You can also choose to test and publish the web service by creating and a client (java servlet to test the web service) :
- The web service contract (WSDL) will be generated.

# Create a Java class

```
package enicarthage.rnu.tn;

public class Math {

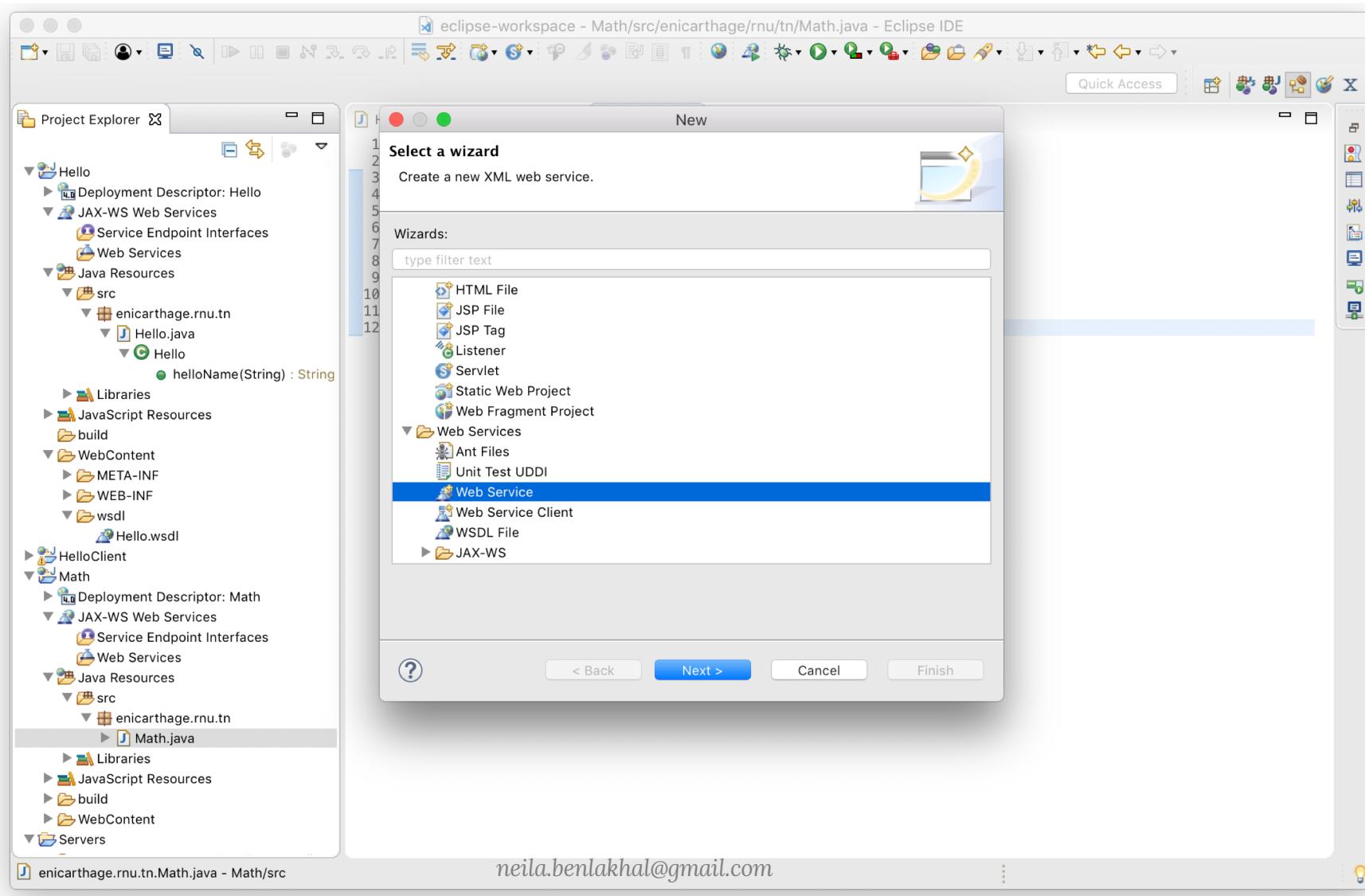
    public float addValue(float value) {
        return (value + 10);
    }

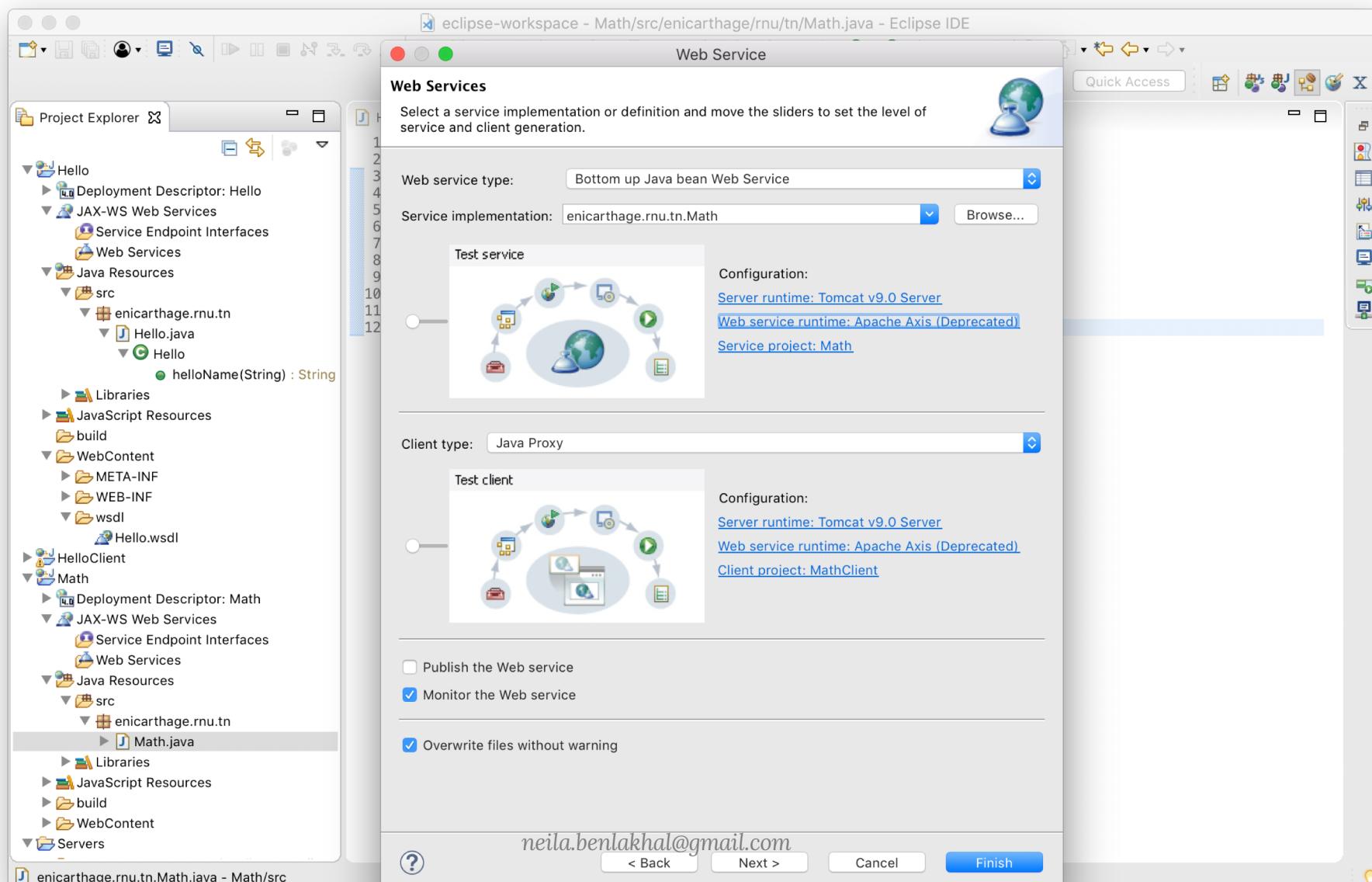
    public float subtractValue(float value) {
        return (value - 10);
    }
}
```

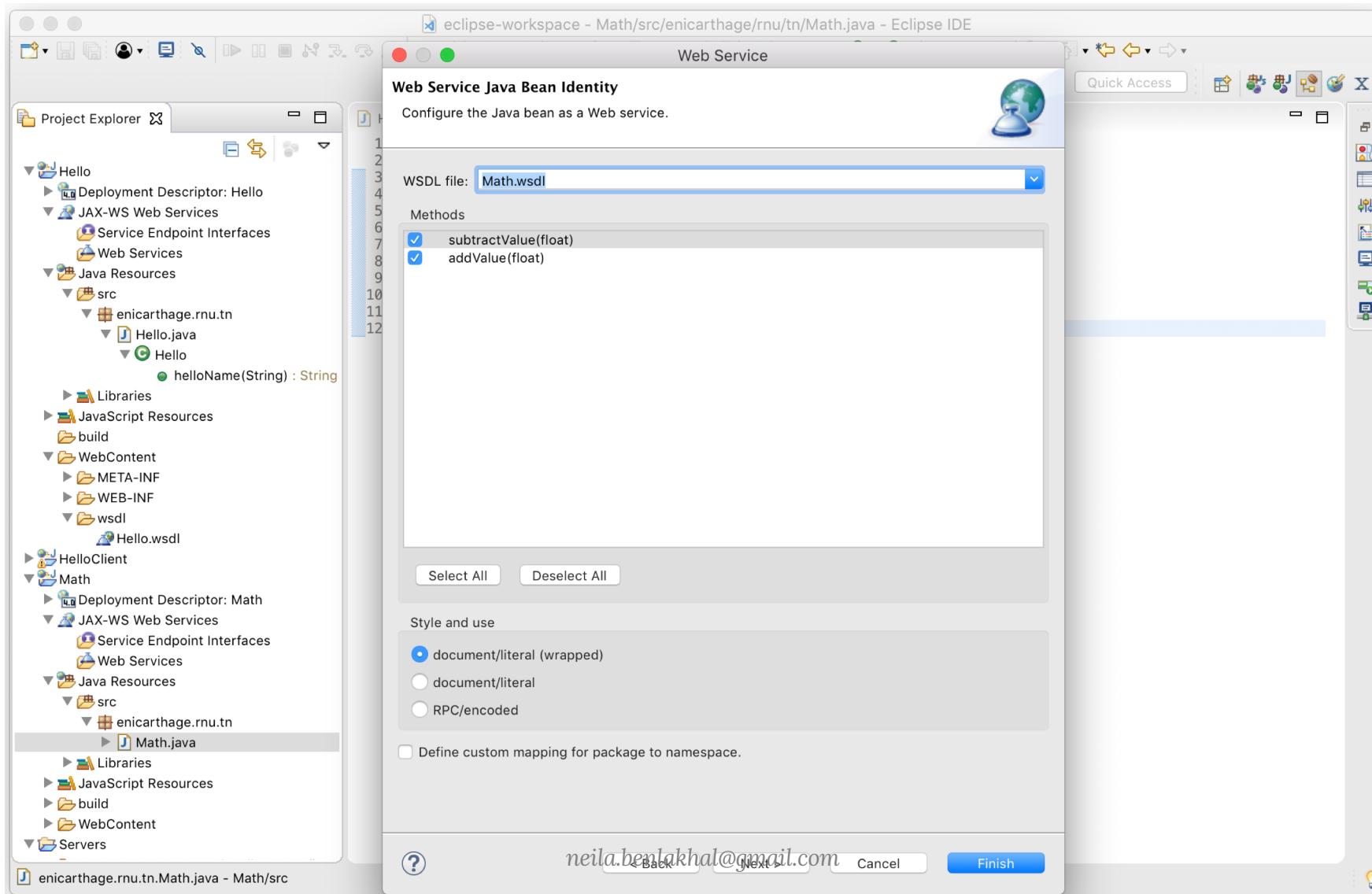
The screenshot shows the Eclipse IDE interface with the title bar "eclipse-workspace - Math/src/enicarthage/rnu/tn/Math.java - Eclipse IDE". The left side features the "Project Explorer" view, which displays a hierarchical tree of project components. The "Hello" project contains a "src" folder with "Hello.java" and "Hello.wsdl". The "Math" project also contains a "src" folder with "Math.java". The "Math.java" file is currently selected in the editor and is displayed in the main window. The code in "Math.java" is as follows:

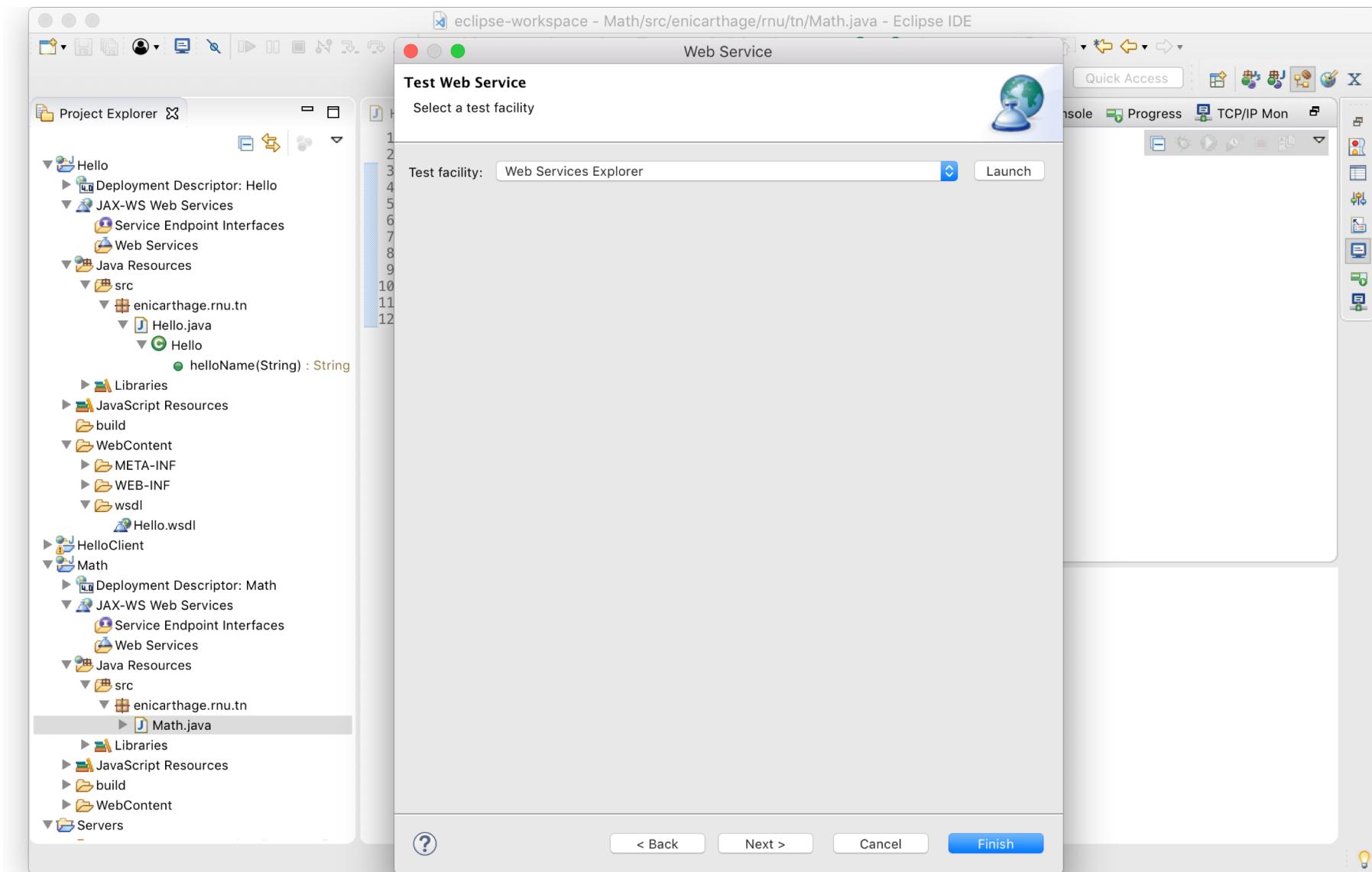
```
1 package enicarthage.rnu.tn;
2
3 public class Math {
4
5     public float addValue(float value) {
6         return (value + 10);
7     }
8
9     public float subtractValue(float value) {
10        return (value - 10);
11    }
12 }
```

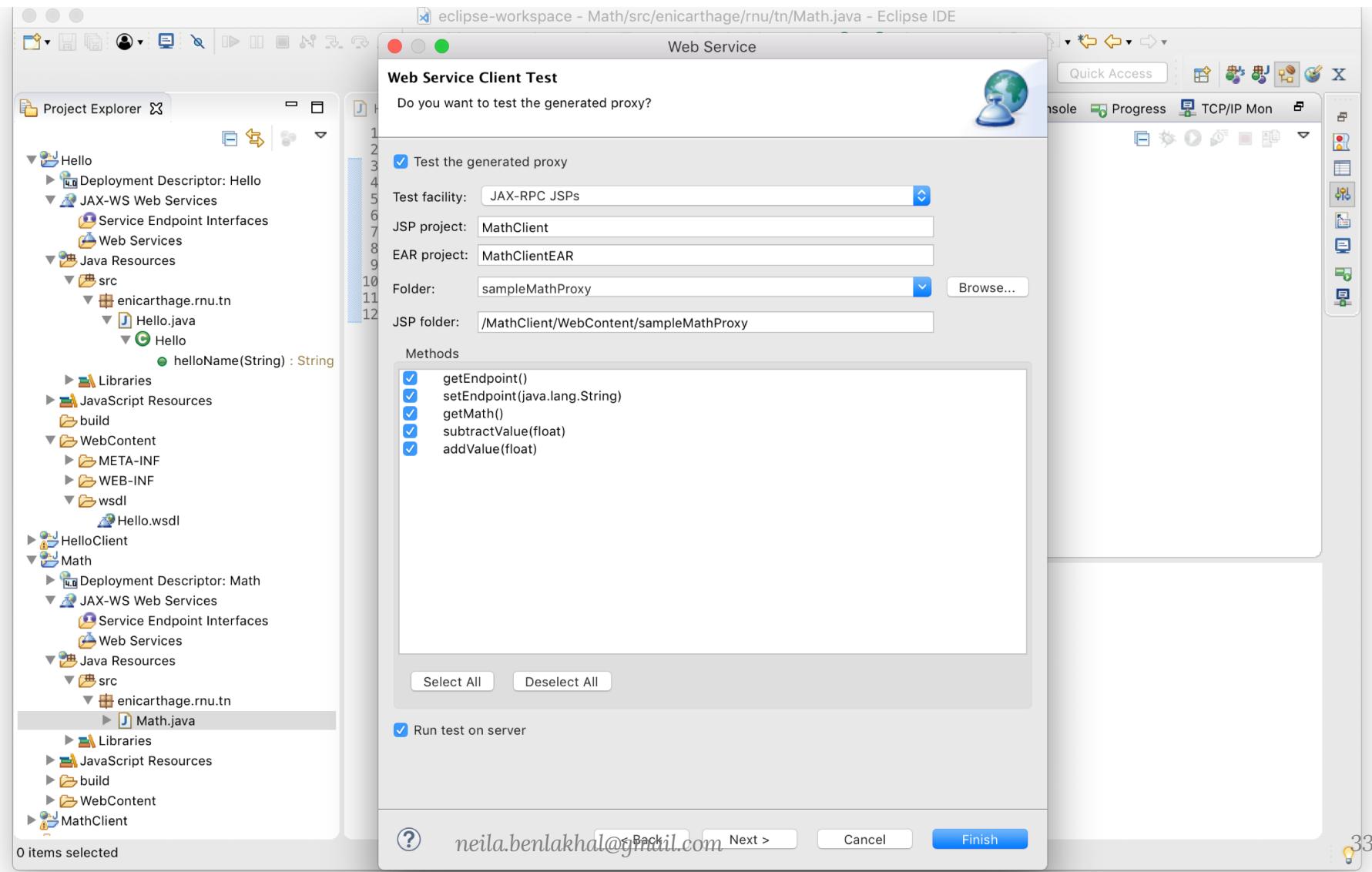
The status bar at the bottom indicates "Writable" and "Smart Insert" modes, along with a line number "12 : 2".











## Web service test

- Java servlet is deployed to test the service :

<http://localhost:8081/MathClient/sampleMathProxy/TestClient.jsp?endpoint=http://localhost:6480/Math/services/Math>

- Web service end point :

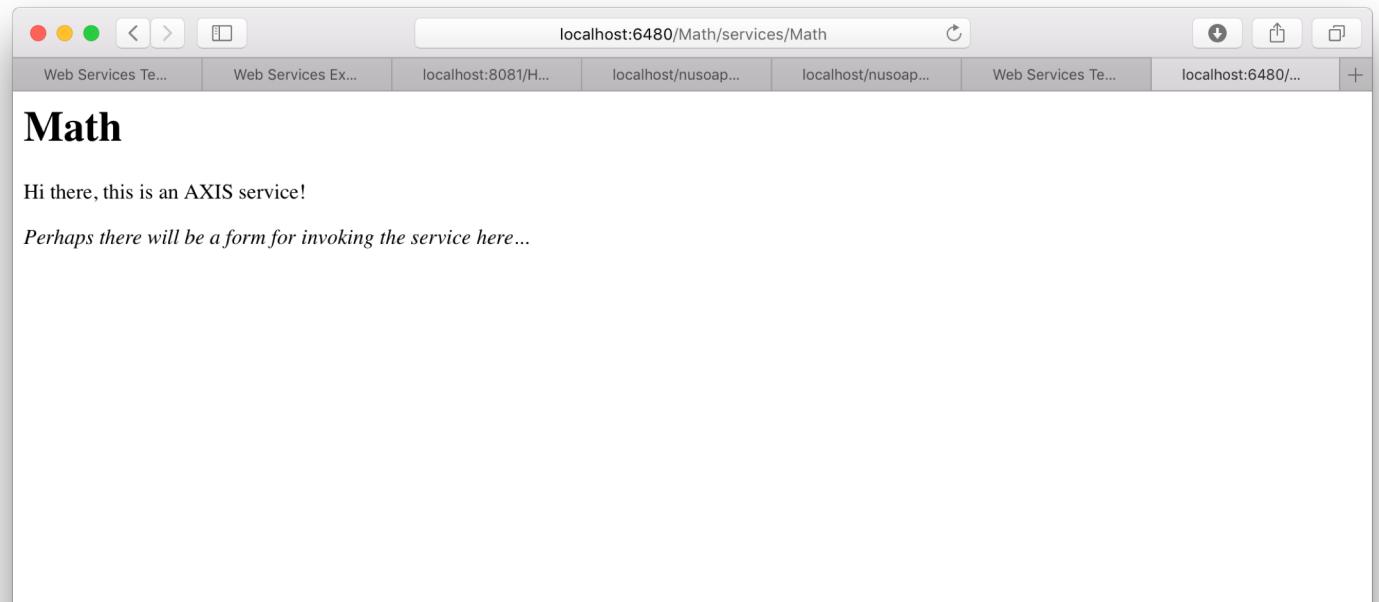
<http://localhost:6480/Math/services/Math>

The screenshot shows a web-based test client interface for a web service. At the top, there are navigation icons and a URL bar displaying `localhost:8081/MathClient/sampleMathProxy/TestClient.jsp?endpoint=http://localhost:8081/MathClient/sampleMathProxy`. Below the URL bar is a toolbar with icons for download, upload, and copy/paste. The main area is divided into sections:

- Methods**: A list of service methods:
  - [getEndpoint\(\)](#)
  - [setEndpoint\(java.lang.String\)](#)
  - [getMath\(\)](#)
  - [subtractValue\(float\)](#)
  - [addValue\(float\)](#)
- Inputs**: A form field labeled "value:" containing the value "20". Below the input field are two buttons: "Invoke" and "Clear".
- Result**: The result of the operation, showing the value "10.0".

At the bottom right of the interface, the email address `neila.benlakhal@gmail.com` is displayed.

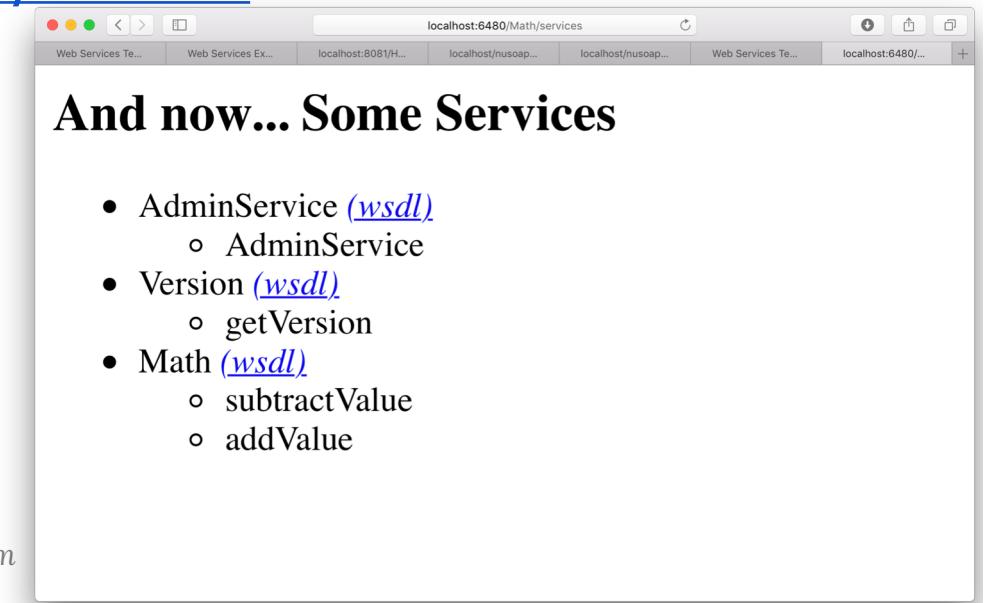
# Get the web services endpoint and check it is running in axis



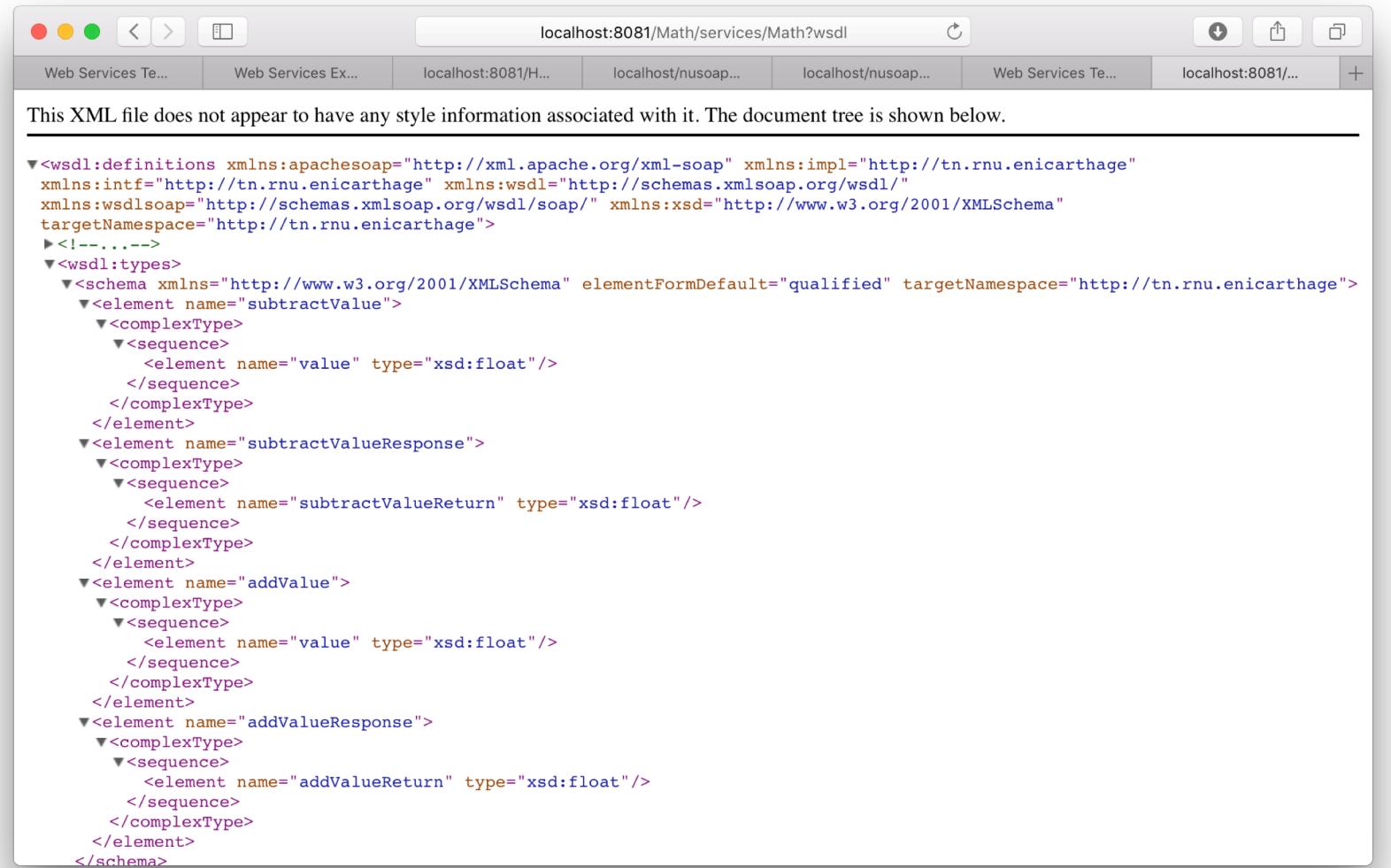
## Get the WSDL of the web services

○ Access :

<http://localhost:6480/Math/services>



# Generated Web service Contract



This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
<wsdl:definitions xmlns:apachesoap="http://xml.apache.org/xml-soap" xmlns:impl="http://tn.rnu.enicarthage"
    xmlns:intf="http://tn.rnu.enicarthage" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://tn.rnu.enicarthage">
    <!--...-->
    <wsdl:types>
        <schema xmlns="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" targetNamespace="http://tn.rnu.enicarthage">
            <element name="subtractValue">
                <complexType>
                    <sequence>
                        <element name="value" type="xsd:float"/>
                    </sequence>
                </complexType>
            </element>
            <element name="subtractValueResponse">
                <complexType>
                    <sequence>
                        <element name="subtractValueReturn" type="xsd:float"/>
                    </sequence>
                </complexType>
            </element>
            <element name="addValue">
                <complexType>
                    <sequence>
                        <element name="value" type="xsd:float"/>
                    </sequence>
                </complexType>
            </element>
            <element name="addValueResponse">
                <complexType>
                    <sequence>
                        <element name="addValueReturn" type="xsd:float"/>
                    </sequence>
                </complexType>
            </element>
        </schema>
    </wsdl:types>

```



# Web services testing



## Étape 3: Tests d'invocation du service Web et résultats

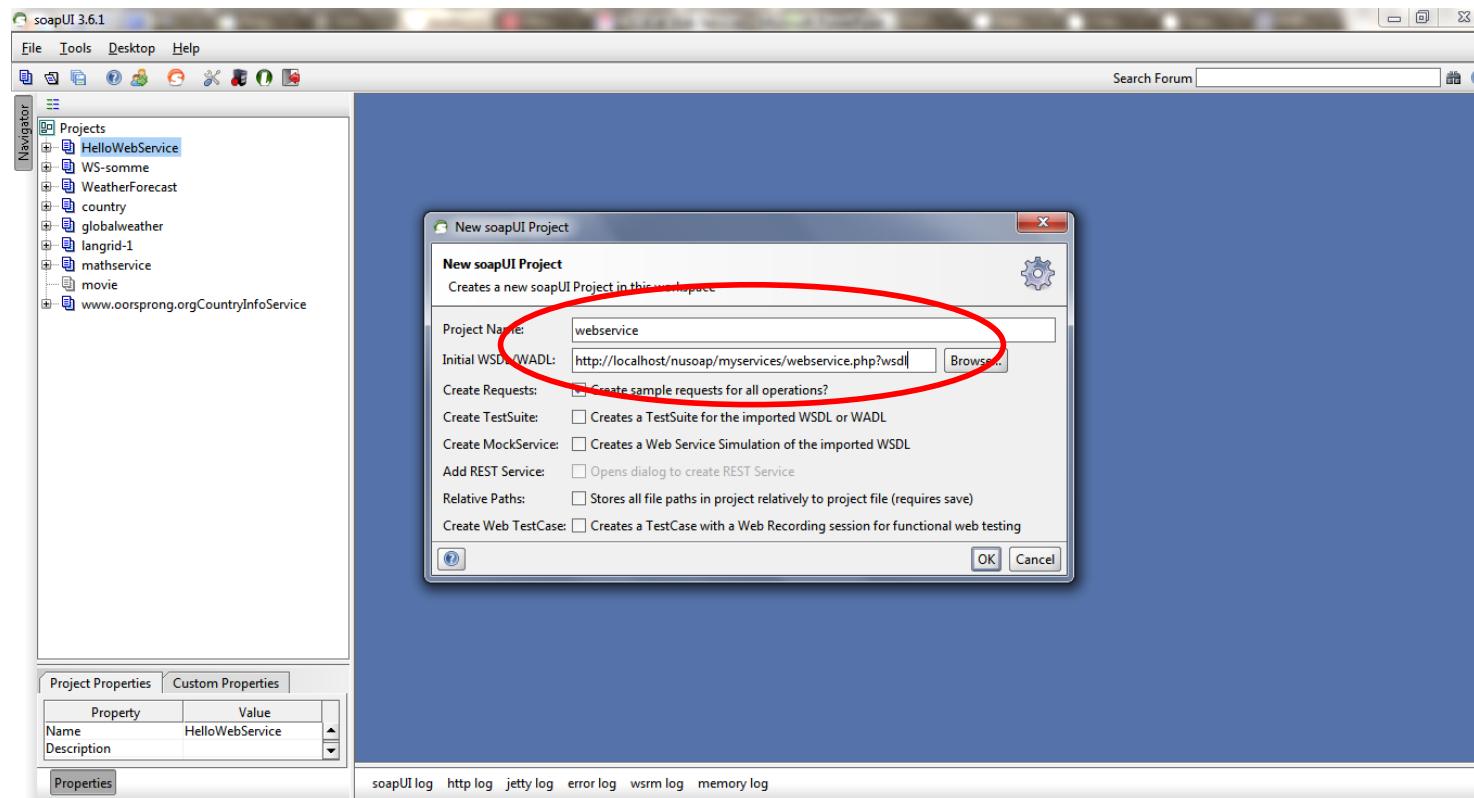
◉ Pour tester le service Web, :

- Méthode 1: Créer une application (un client) qui fait appel au service que nous venons de créer,
- Méthode 2: Utiliser un logiciel tel que soapUI pour tester l'invocation de ce service :

◉ <http://www.soapui.org/>.

- SOAPUI est un outil pour tester des Web Services
  - Fonctionnalités de SOAPUI
  - Supporte les Web Services étendus (WSDL + SOAP + UDDI) ou REST
  - Inspecter des Web Services
  - Invoquer des Web Services
  - Développer des Web Services
  - Simuler des Web Services via des bouchons (mocks)
  - Effectuer des tests qualités (temps de réponse, ...)

## Méthode 1: soapUI



soapUI 3.6.1

File Tools Desktop Help

Navigator

- Projects
  - HelloWebService
    - MonServiceHelloBinding
      - hello
        - Request1
  - WS-Security
  - WeatherForecast
  - country
  - globalweather
  - langrid-1
  - mathservice
  - movie
  - www.oorsprong.orgCountryInfo

Request Properties

Property	Value
Name	Request1
Description	
Message Size	441

Properties

Request 1

http://localhost/nusoap/myservices/webservice.php

Raw XML

```
<soapenv:Envelope
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:loc="http://localhost/">
  <soapenv:Header/>
  <soapenv:Body>
    <loc:hello soapenv:encodingStyle=
      "http://schemas.xmlsoap.org/soap/encoding/">
      <name xsi:type="xsd:string">sarah</name>
    </loc:hello>
  </soapenv:Body>
</soapenv:Envelope>
```

Raw XML

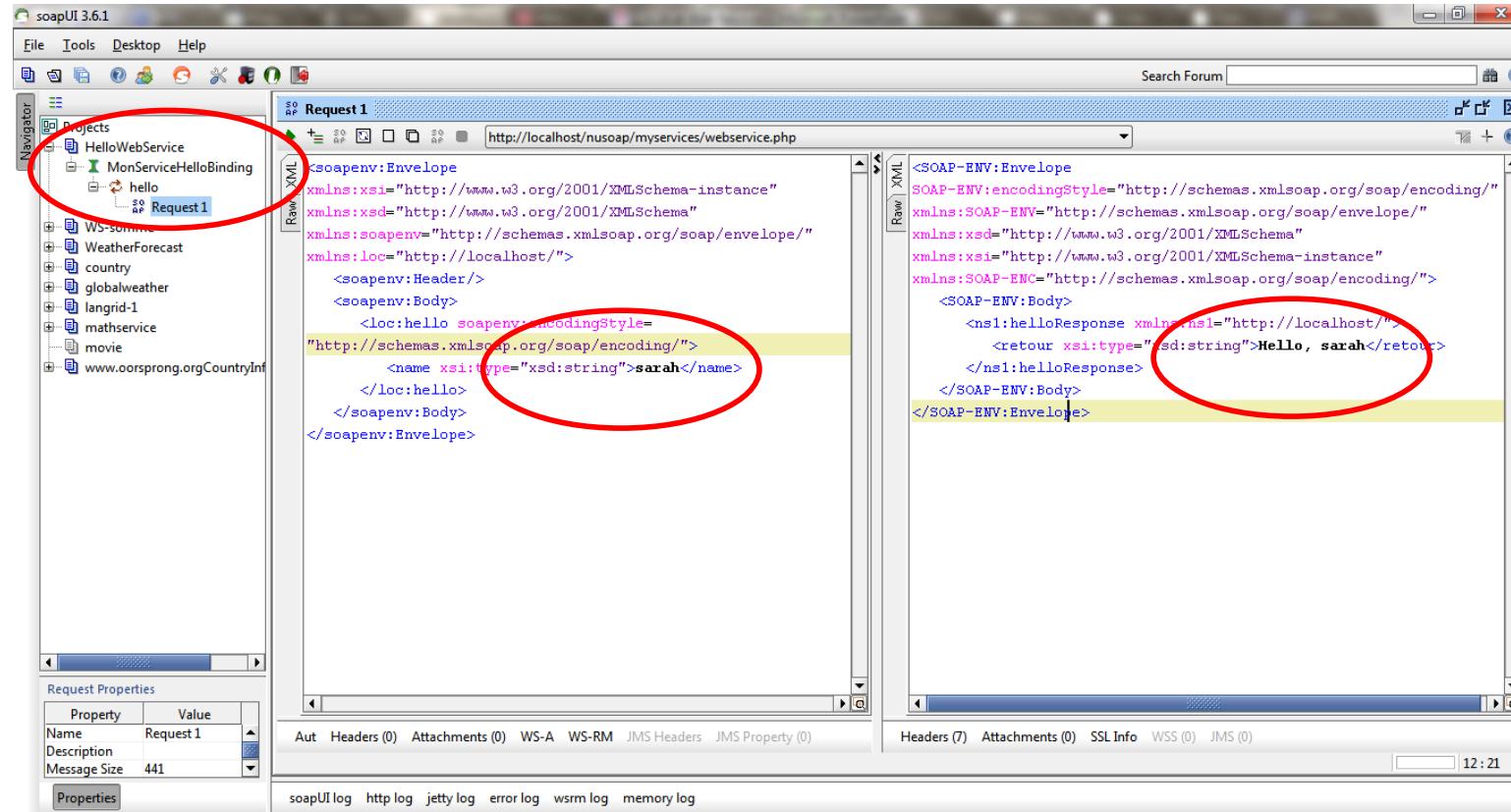
```
<SOAP-ENV:Envelope
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <ns1:helloResponse xmlns:ns1="http://localhost/">
      <retour xsi:type="xsd:string">Hello, sarah</retour>
    </ns1:helloResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Aut Headers (0) Attachments (0) WS-A WS-RM JMS Headers JMS Property (0)

Headers (7) Attachments (0) SSL Info WSS (0) JMS (0)

12:21

soapUI log http log jetty log error log wsrm log memory log

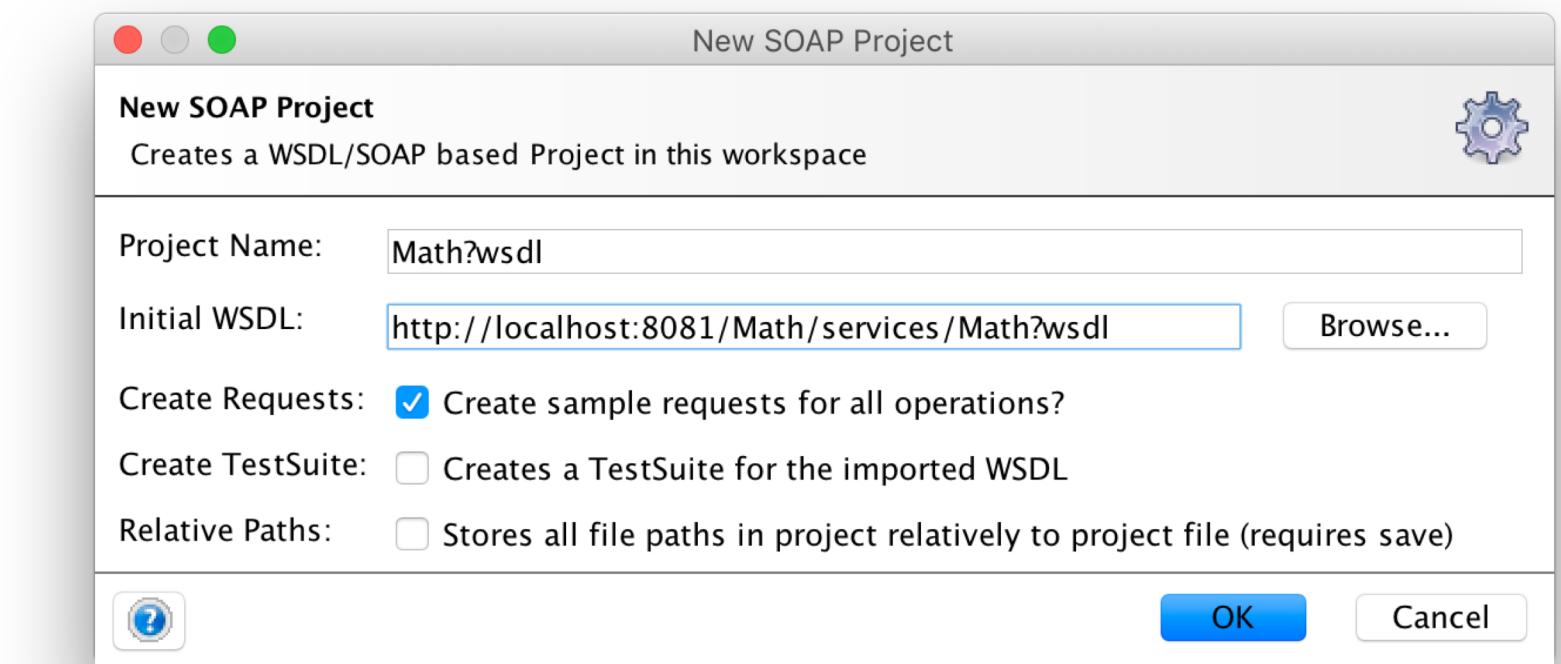


neila.benlakhal@gmail.com

42

# Java axis web service Testing

○ <http://localhost:8081/Math/services/Math?wsdl>



SoapUI 5.4.0

Empty SOAP REST Import Save All Forum Trial Preferences Proxy Search Forum Online Help

Projects

- >Hello?wsdl
  - HelloSoapBinding
    - helloName
- HelloAxis
  - HelloSoapBinding
    - helloName
- Math?wsdl
  - MathSoapBinding
    - addValue
      - Request 1
    - subtractValue
- Sample SOAP Project Core
  - ServiceSoapBinding
    - login
      - login rq
      - logout
      - search
        - search rq
      - buy
        - buy rq
- Simple TestSuite
- TesTsuite fail if we don't get f

Request Properties

Property	Value
Name	Request 1
Description	

neila.benlakhal@gmail.com

Request 1

http://localhost:8081/Math/services/Math

Request 1

1<soapenv:Envelope  
2 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope"  
3 xmlns:tn="http://tn.rnu.enicarthage"  
4 <soapenv:Header/>  
5 <soapenv:Body>  
6 <tn:addValue>  
7 <tn:value>25.10</tn:value>  
8 </tn:addValue>  
9 </soapenv:Body>  
10 </soapenv:Envelope>

1<soapenv:Envelope  
2 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope"  
3 xmlns:xsd="http://www.w3.org/2001/XMLSchema"  
4 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
5 <soapenv:Body>  
6 <addValueResponse xmlns="http://tn.rnu.enicarthage":  
7 <addValueReturn>35.1</addValueReturn>  
8 </addValueResponse>  
9 </soapenv:Body>  
10 </soapenv:Envelope>

Raw XML Raw XML

Two Oct 09 12:26:18 CET 2018 - DFRIC - 47

SoapUI 5.4.0

Empty SOAP REST Import Save All Forum Trial Preferences Proxy Search Forum Online Help

**Navigator**

- Empty
- SOAP
- REST
- Import
- Save All
- Forum
- Trial
- Preferences
- Proxy
- Search Forum
- Online Help

**Request 1**

POST http://localhost:8081/Math/services/Math HTTP/1.1  
Accept-Encoding: gzip,deflate  
Content-Type: text/xml;charset=UTF-8  
SOAPAction: "  
Content-Length: 268  
Host: localhost:8081  
Connection: Keep-Alive  
User-Agent: Apache-HttpClient/4.1.1 (java 1.5)

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoa<br><soapenv:Header/><soapenv:Body><tn:addValue><tn:value>25.10</tn:value></tn:addValue></soapenv:Body></soapenv:Envelope>

**Raw**

HTTP/1.1 200  
Content-Type: text/xml;charset=utf-8  
Transfer-Encoding: chunked  
Date: Tue, 09 Oct 2018 12:36:18 GMT  
<?xml version="1.0" encoding="UTF-8"?><soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">

**Request Properties**

Property	Value
Name	Request 1
Description	
Message Size	270
Encoding	UTF-8
Endpoint	http://localhost:8081/Math/services/Math

**Properties**

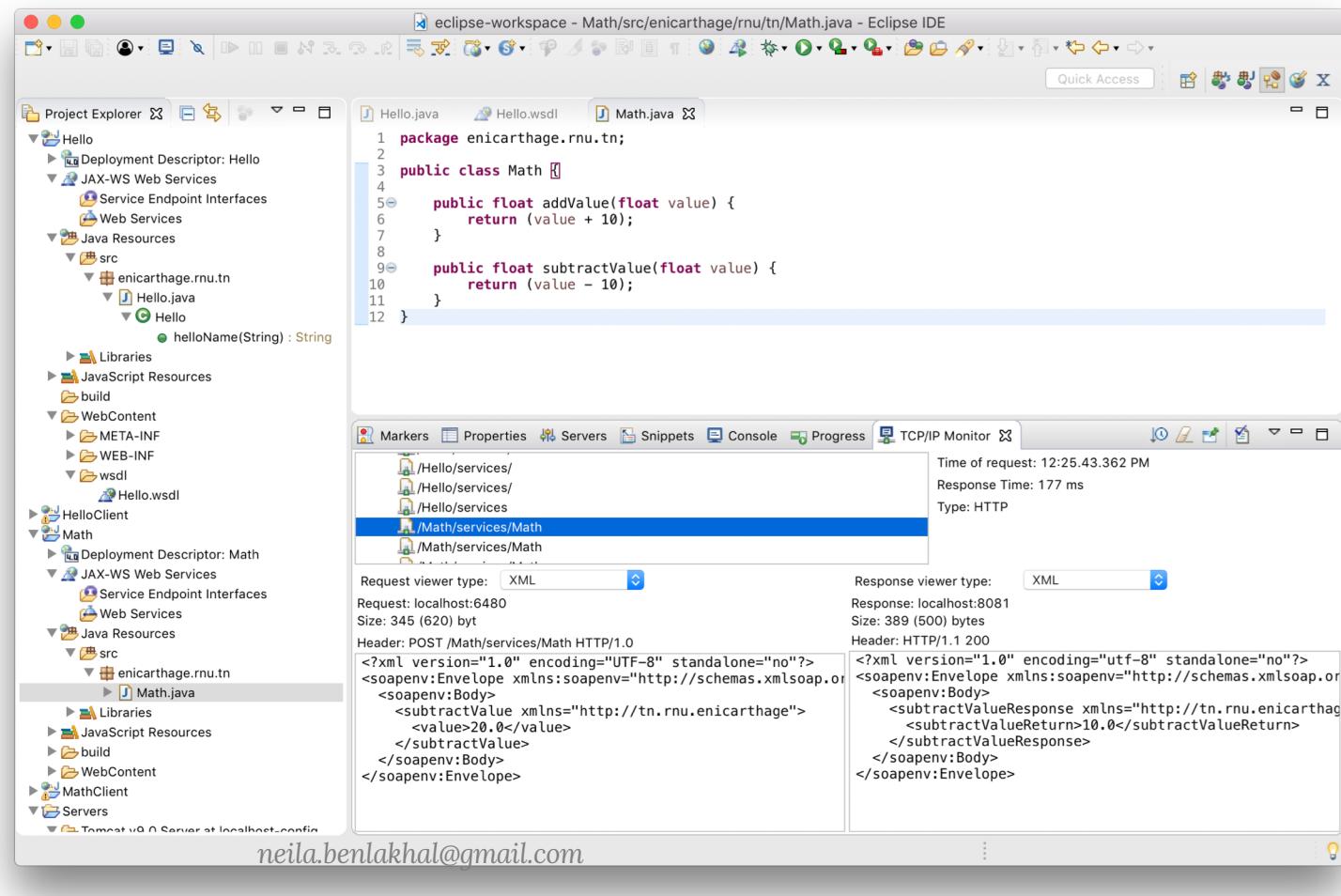
Properties

Tue Oct 09 14:36:43 CET 2018:DEBUG:<< "Content-Length: 531[\r][\n]"  
Tue Oct 09 14:36:43 CET 2018:DEBUG:<< "[\r][\n]"  
Tue Oct 09 14:36:43 CET 2018:DEBUG:<< "<?xml version="1.0" encoding="utf-8"?><soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">"

SoapUI log http log jetty log error log wsrm log memory log script log

neila.benlakhal@gmail.com

# Eclipse TCP/IP Monitor (Soap request/response)





## Consommation du Service Web

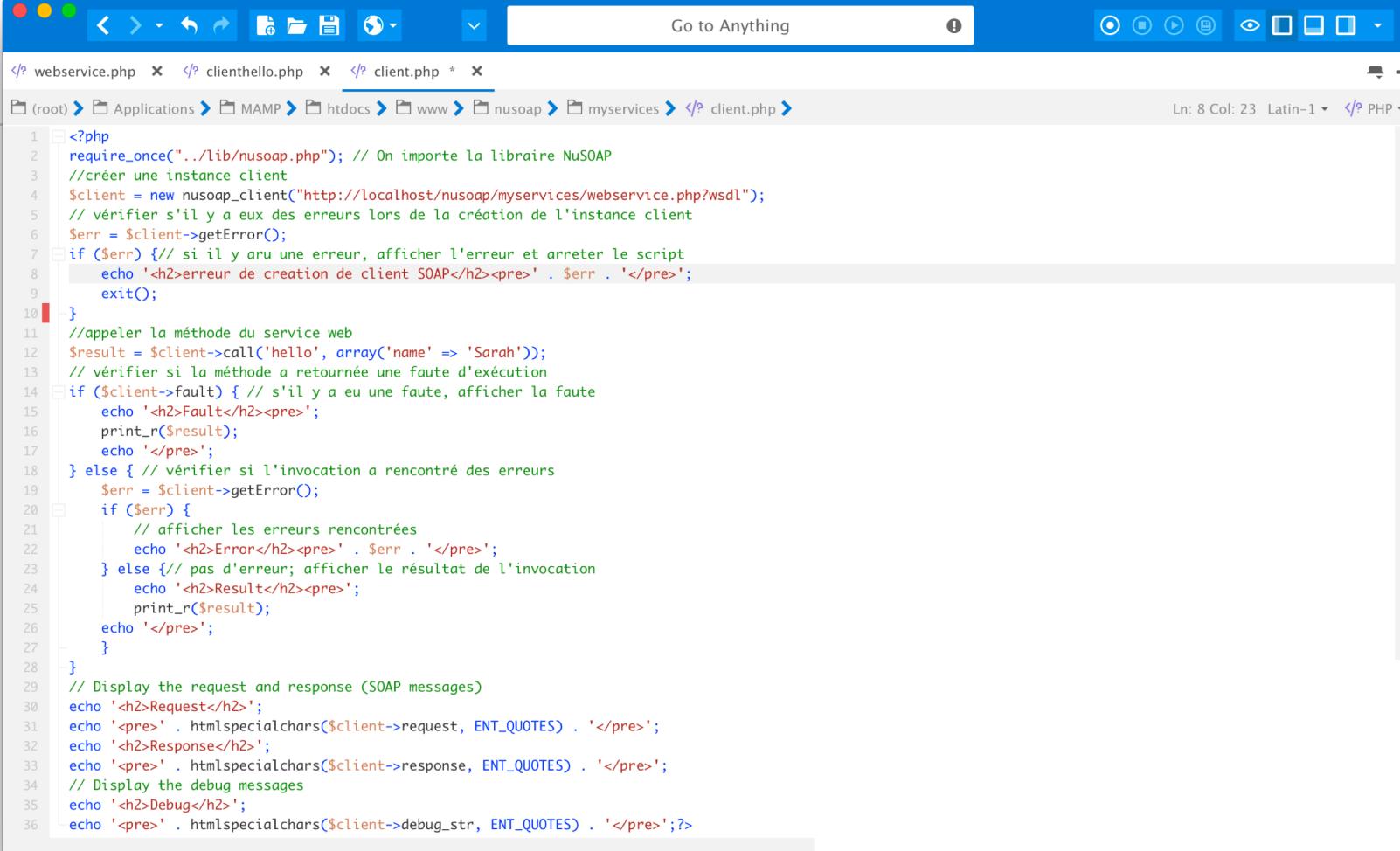
● Méthode 2: Créer une application (un client) qui fait appel au service web que nous venons de créer.

- Le Web service est disponible sur le serveur et L'Interface du service est générée (le WSDL)
  - N'importe qui peut l'invoquer à travers son WSDL peu importe la plateforme choisie.



## Client.php

- Développer une application **client.php** qui appelle le service fourni par **webservice.php**.
  
- Développer une application **client.php** qui appelle le service **Math(classe en Java)**



The screenshot shows a Mac OS X terminal window with a blue header bar. The title bar says "Go to Anything". The menu bar includes standard OS X icons like file, edit, and search, along with a "PHP" dropdown. The main pane displays a PHP script named "client.php". The script imports the NuSOAP library, creates a client object, and calls the "hello" method with the parameter "Sarah". It handles errors and prints the result. The code is color-coded for syntax highlighting.

```
<?php
require_once("../lib/nusoap.php"); // On importe la librairie NuSOAP
//créer une instance client
$client = new nusoap_client("http://localhost/nusoap/myservices/webservice.php?wsdl");
// vérifier s'il y a eux des erreurs lors de la création de l'instance client
$err = $client->getError();
if ($err) { // si il y a eu une erreur, afficher l'erreur et arreter le script
    echo '<h2>erreur de creation de client SOAP</h2><pre>' . $err . '</pre>';
    exit();
}
//appeler la méthode du service web
$result = $client->call('hello', array('name' => 'Sarah'));
// vérifier si la méthode a retournée une faute d'exécution
if ($client->fault) { // s'il y a eu une faute, afficher la faute
    echo '<h2>Fault</h2><pre>';
    print_r($result);
    echo '</pre>';
} else { // vérifier si l'invocation a rencontré des erreurs
    $err = $client->getError();
    if ($err) {
        // afficher les erreurs rencontrées
        echo '<h2>Error</h2><pre>' . $err . '</pre>';
    } else { // pas d'erreur; afficher le résultat de l'invocation
        echo '<h2>Result</h2><pre>';
        print_r($result);
        echo '</pre>';
    }
}
// Display the request and response (SOAP messages)
echo '<h2>Request</h2>';
echo '<pre>' . htmlspecialchars($client->request, ENT_QUOTES) . '</pre>';
echo '<h2>Response</h2>';
echo '<pre>' . htmlspecialchars($client->response, ENT_QUOTES) . '</pre>';
// Display the debug messages
echo '<h2>Debug</h2>';
echo '<pre>' . htmlspecialchars($client->debug_str, ENT_QUOTES) . '</pre>';?>
```



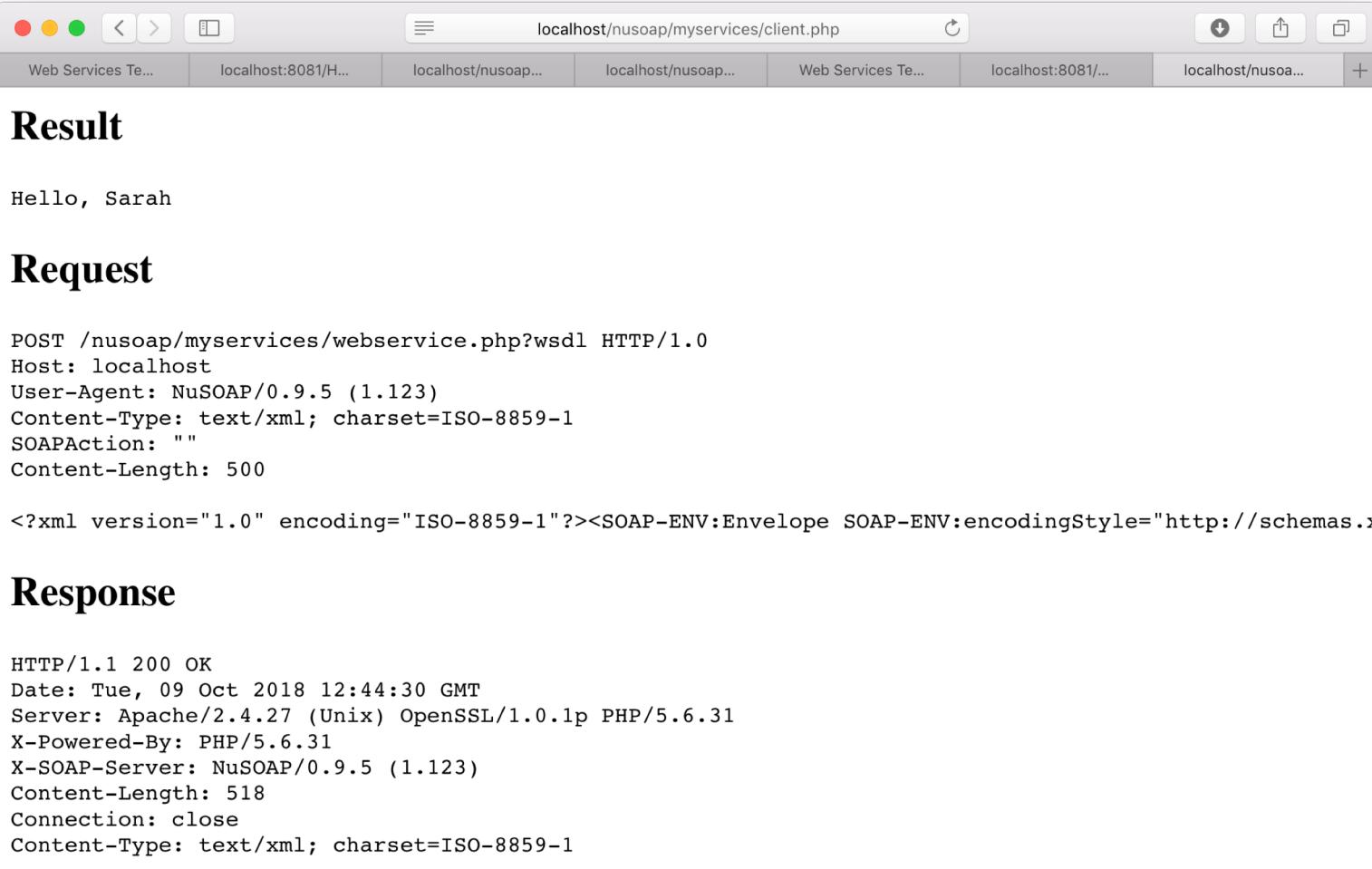
## Client.php 1/2

```
<?php
require_once("../lib/nusoap.php"); // On importe la librairie NuSOAP
// crÈer une instance client
$client = new nusoap_client('http://localhost/nusoap/myservices/webservice.php?wsdl');
// vÈrifier s'il y a eux des erreurs lors de la crÈation de l'instance client
$err = $client->getError();
if ($err) { // si il y a eu une erreur, afficher l'erreur et arreter le script
    echo '<h2>erreur de creation de client SOAP</h2><pre>' . $err . '</pre>';
    exit();}
// appeler la mÈthode du service web
$result = $client->call('hello', array('name' => 'Sarah'));
// vÈrifier si la mÈthode a retournÈe une faute d'exÈcution
if ($client->fault) { // s'il y a eu une faute, afficher la faute
    echo '<h2>Fault</h2><pre>';
    print_r($result);
    echo '</pre>';
} else { // vÈrifier si l'invocation a rencontrÈ des erreurs
    $err = $client->getError();
    if ($err) {
        // afficher les erreurs rencontrÈes
        echo '<h2>Error</h2><pre>' . $err . '</pre>';
    } else { // pas d'erreur; afficher le rÈsultat de l'invocation
        echo '<h2>Result</h2><pre>';
        print_r($result);
        echo '</pre>';
    }
}
```



## Client.php 2/2

```
// Display the request and response (SOAP messages)
echo '<h2>Request</h2>';
echo '<pre>' . htmlspecialchars($client->request,
ENT_QUOTES) . '</pre>';
echo '<h2>Response</h2>';
echo '<pre>' . htmlspecialchars($client->response,
ENT_QUOTES) . '</pre>';
// Display the debug messages
echo '<h2>Debug</h2>';
echo '<pre>' . htmlspecialchars($client->debug_str,
ENT_QUOTES) . '</pre>';?>
```



A screenshot of a Mac OS X desktop environment showing a web browser window. The browser's title bar reads "localhost/nusoap/myservices/client.php". The address bar also shows "localhost/nusoap/myservices/client.php". The browser has multiple tabs open, with titles like "Web Services Te...", "localhost:8081/H...", "localhost/nusoap...", "localhost/nusoap...", "Web Services Te...", "localhost:8081/...", and "localhost/nusoap...".

The main content area of the browser displays the following text:

## Result

Hello, Sarah

## Request

```
POST /nusoap/myservices/webservice.php?wsdl HTTP/1.0
Host: localhost
User-Agent: NuSOAP/0.9.5 (1.123)
Content-Type: text/xml; charset=ISO-8859-1
SOAPAction: ""
Content-Length: 500

<?xml version="1.0" encoding="ISO-8859-1"?><SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.x
```

## Response

```
HTTP/1.1 200 OK
Date: Tue, 09 Oct 2018 12:44:30 GMT
Server: Apache/2.4.27 (Unix) OpenSSL/1.0.1p PHP/5.6.31
X-Powered-By: PHP/5.6.31
X-SOAP-Server: NuSOAP/0.9.5 (1.123)
Content-Length: 518
Connection: close
Content-Type: text/xml; charset=ISO-8859-1
```



## Explication client.php 1/4

- Importer la librairie NuSOAP. Celle-ci permettra à PHP de communiquer à travers le protocole SOAP :

```
Require_once('..../lib/nusoap.php');
```

- Créer une instance de la classe client en utilisant le constructeur soapclient: ce constructeur prend comme paramètre obligatoire l'URL du fichier WSDL du web service:

```
$client = new  
soapclient('http://localhost/nusoap/myservices/webservice.php?ws  
dl');
```



## Explication client.php 2/4

- Vérifier s'il y a eu des erreurs lors de la création du client : l'instance client n'a pu être créée

```
// vérifier s'il y a eux des erreurs lors de la création de l'instance client
$err = $client->getError();
if ($err) { // s'il y a une erreur, afficher l'erreur et arrêter le script
    echo '<h2>erreur de création de client SOAP</h2><pre>' . $err .
'</pre>';
    exit();
```

- Appeler la méthode du service web : la méthode hello appelé avec Sarah comme valeur du paramètre name:

```
$result = $client->call('hello', array('name' => 'Sarah'));
```

## Explication client.php 3/4

```
// vérifier si la méthode a retournée une faute (ex: type du paramètre)
if ($client->fault) { // s'il y a eu une faute, afficher la faute
    echo '<h2>Fault</h2><pre>';
    print_r($result);
    echo '</pre>';
} else { // vérifier si l'invocation a rencontré des erreurs
    $err = $client->getError();
    if ($err) { // afficher les erreurs rencontrées
        echo '<h2>Error</h2><pre>' . $err . '</pre>';
    } else { // pas d'erreur; afficher le résultat de l'invocation
        echo '<h2>Result</h2><pre>';
        print_r($result);
        echo '</pre>';
    }
}
```

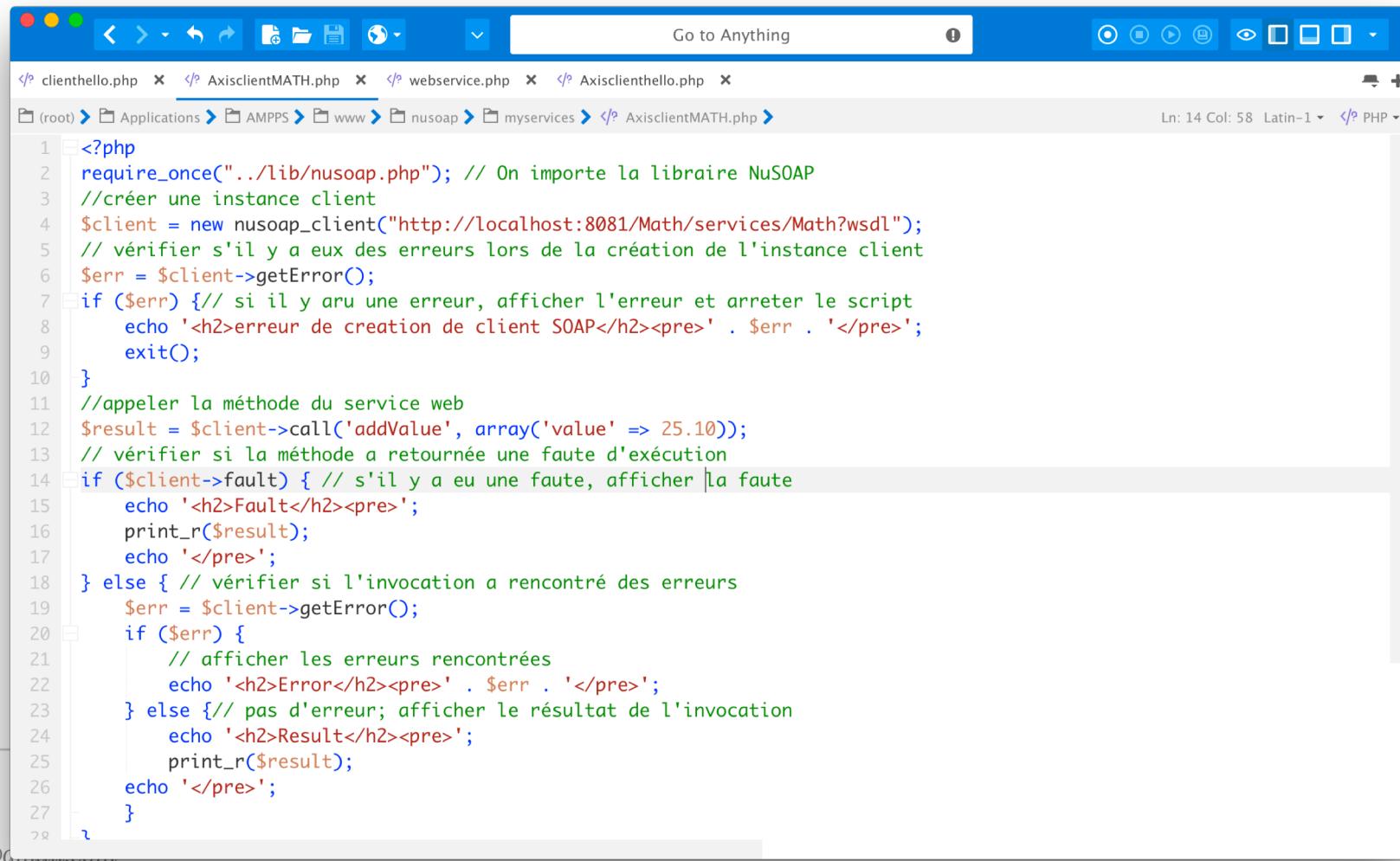


## Explication client.php 4/4

- Fonctions facultatives: Afficher le contenu de la requête et de la réponse SOAP ainsi que le debug de l'invocation:
- Convertit tous les caractères spéciaux en entité HTML.
- htmlspecialchars(\$chaine, ENT\_QUOTES) converti tous les caractères spéciaux en entités HTML (ex: '<' devient '&lt;' etc.)

```
// Display the request and response (SOAP messages)
echo '<h2>Request</h2>';
echo '<pre>'. htmlspecialchars($client->request, ENT_QUOTES) . '</pre>';
echo '<h2>Response</h2>';
echo '<pre>'. htmlspecialchars($client->response, ENT_QUOTES) . '</pre>';
// Display the debug messages
echo '<h2>Debug</h2>';
echo '<pre>'. htmlspecialchars($client->debug_str, ENT_QUOTES) . '</pre>';?>
```

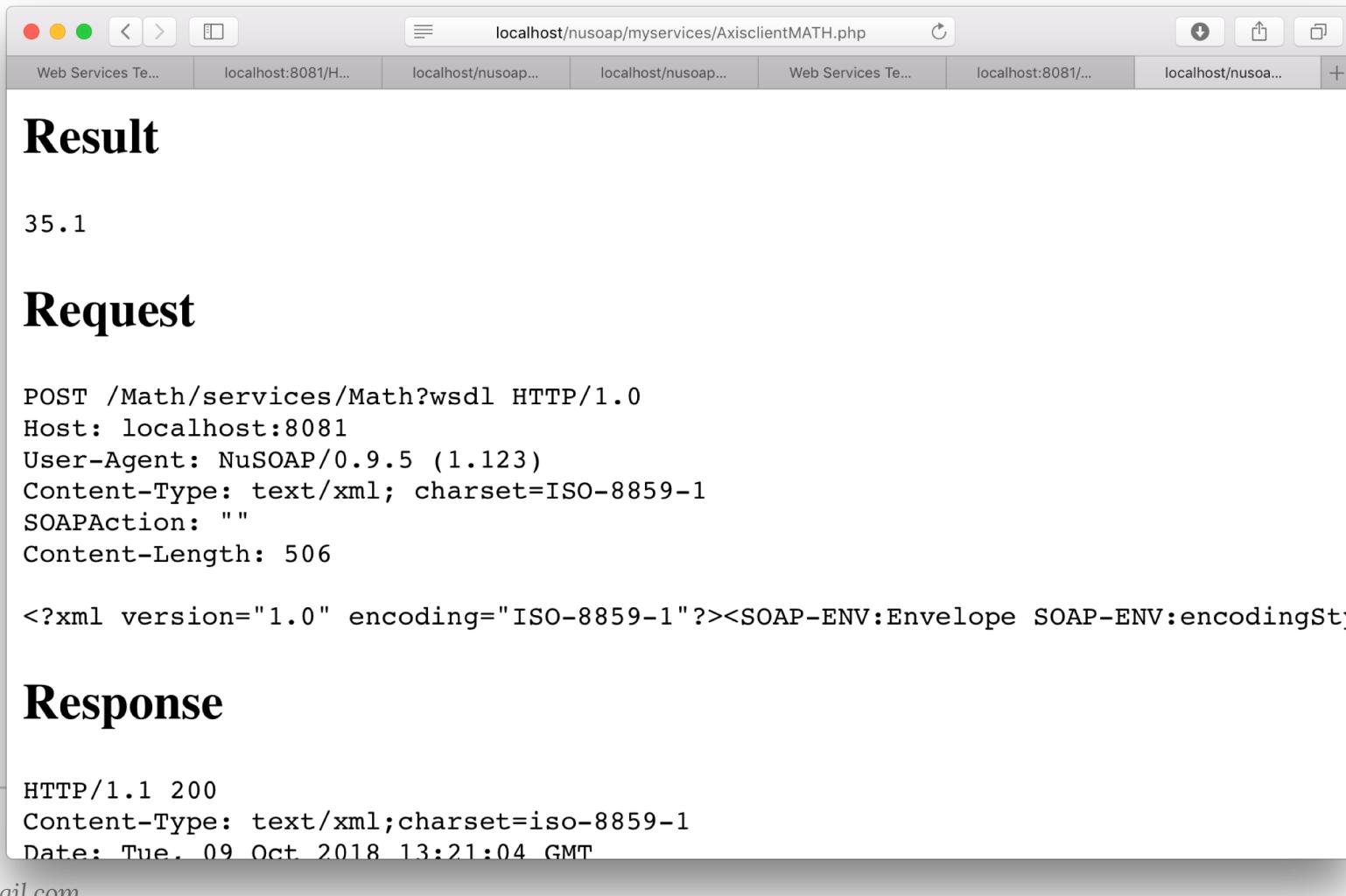
# Client.php service Java



The screenshot shows a Mac OS X desktop environment with a terminal window open. The title bar of the terminal window says "AxisclientMATH.php". The window displays a PHP script named "AxisclientMATH.php" which interacts with a Java web service. The script uses the NuSOAP library to create a client, call a method ("addValue") with a value of 25.10, and then prints the result or any errors.

```
<?php
require_once("../lib/nusoap.php"); // On importe la librairie NuSOAP
//créer une instance client
$client = new nusoap_client("http://localhost:8081/Math/services/Math?wsdl");
// vérifier s'il y a eux des erreurs lors de la création de l'instance client
$err = $client->getError();
if ($err) { // si il y aru une erreur, afficher l'erreur et arreter le script
    echo '<h2>erreur de creation de client SOAP</h2><pre>' . $err . '</pre>';
    exit();
}
//appeler la méthode du service web
$result = $client->call('addValue', array('value' => 25.10));
// vérifier si la méthode a retournée une faute d'exécution
if ($client->fault) { // s'il y a eu une faute, afficher la faute
    echo '<h2>Fault</h2><pre>';
    print_r($result);
    echo '</pre>';
} else { // vérifier si l'invocation a rencontré des erreurs
    $err = $client->getError();
    if ($err) {
        // afficher les erreurs rencontrées
        echo '<h2>Error</h2><pre>' . $err . '</pre>';
    } else { // pas d'erreur; afficher le résultat de l'invocation
        echo '<h2>Result</h2><pre>';
        print_r($result);
        echo '</pre>';
    }
}
```

neila.benlakhal@...:~\$



A screenshot of a Mac OS X desktop environment showing a web browser window. The browser's title bar reads "localhost/nusoap/myservices/AxisclientMATH.php". The address bar also shows "localhost/nusoap/myservices/AxisclientMATH.php". The browser has multiple tabs open, with titles like "Web Services Te...", "localhost:8081/H...", "localhost/nusoap...", "localhost/nusoap...", "Web Services Te...", "localhost:8081/...", and "localhost/nusoap...".

The main content area of the browser displays the following text:

## Result

35.1

## Request

```
POST /Math/services/Math?wsdl HTTP/1.0
Host: localhost:8081
User-Agent: NuSOAP/0.9.5 (1.123)
Content-Type: text/xml; charset=ISO-8859-1
SOAPAction: ""
Content-Length: 506

<?xml version="1.0" encoding="ISO-8859-1"?><SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
```

## Response

---

```
HTTP/1.1 200
Content-Type: text/xml; charset=iso-8859-1
Date: Tue, 09 Oct 2018 13:21:04 GMT
```

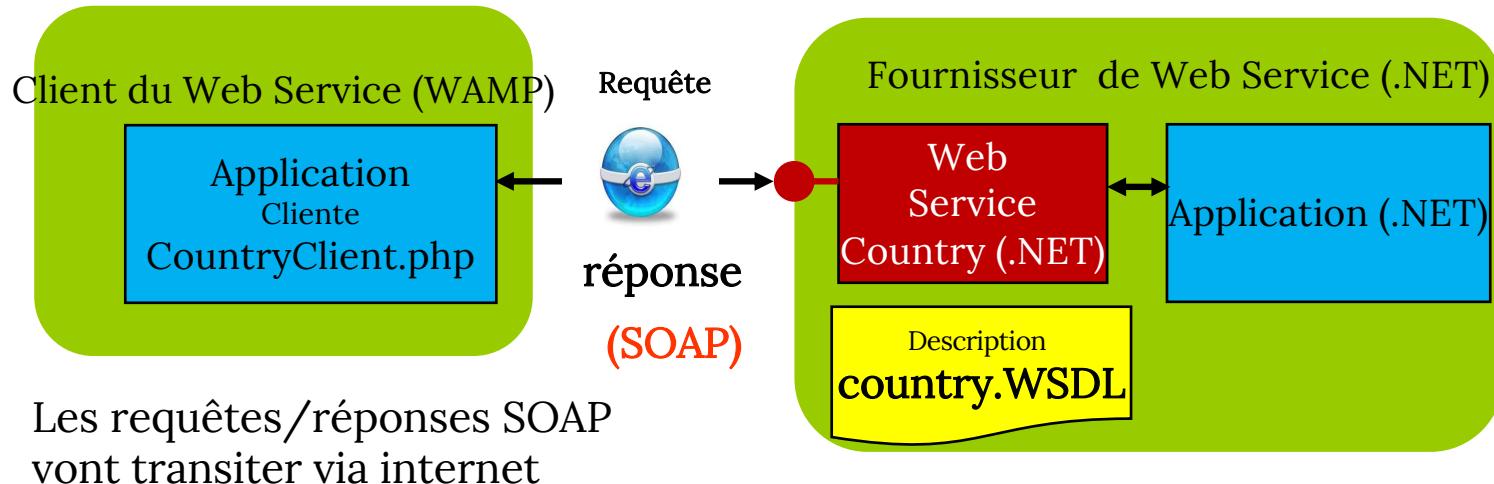


## **Interopérabilité entre des environnements hétérogènes:**

- Beaucoup de plateformes d'implémentation de Web Services:
  - Microsoft . Net
  - Apache Axis
  - Sun Java-WS
  - Etc.
  
- Consommer un web service développé sous la plateforme .NET avec un client PhP.

## Interopérabilité entre des environnements hétérogènes

- Consommer un web service développé sous la plateforme .NET avec un client PHP :
- Le web service country fourni par l'annuaire public WebservicesX.net :
  - <http://www.webservicex.net/country.asmx?WSDL>





## Le Web service Country

○ Le web service Country défini plusieurs fonctions :

- **GetGMTbyCountry**  
Get greenwich mean time(GMT) by country name
- **GetCountryByCountryCode**  
Get country name by country code
- **GetCountryByCurrencyCode**  
Get country by currency code
- **GetCurrencies**  
Get all currency,currency code for all countries
- **GetCurrencyByCountry**  
Get currency by country name
- **GetCurrencyCodeByCurrencyName**  
Get currency by currency name
- **GetISD**  
Get International Dialing Code by country name
- Etc.

soapUI 3.6.1

File Tools Desktop Help

Navigator

Projects

- HelloWebService
  - MonServiceHelloBinding
    - hello
      - Request 1
  - WS-somme
  - WeatherForecast
  - country
    - countrySoap
      - GetCountries
      - GetCountryByCountryCode
      - GetCountryByCurrency
      - GetCurrencies
      - GetCurrencyByCountry
      - GetCurrencyCode
      - GetCurrencyCodeByCountry
      - GetGMTbyCountry
      - GetISD
      - GetISOCountryCode
      - countrySoap12
      - countrySoap TestSuite
      - countrySoap12 TestSuite
      - countrySoap MockService
      - countrySoap12 MockService
  - globalweather
  - langrid-1
  - mathservice

Interface Properties

Property	Value
Name	countrySoap
Description	
Definition URL	<a href="http://www.webservicex.net/country.asmx?WSDL">http://www.webservicex.net/country.asmx?WSDL</a>

Properties

countrySoap

WS-I Compliance

WSDL Definition

WSDL URL: <http://www.webservicex.net/country.asmx?WSDL>

Namespace: http://www.webserviceX.NET

Binding: countrySoap

SOAP Version: SOAP 1.1

Style: Document

WS-A version: NONE

Definition Parts

country.asmx?WSDL: <http://www.webservicex.net/country.asmx?WSDL>

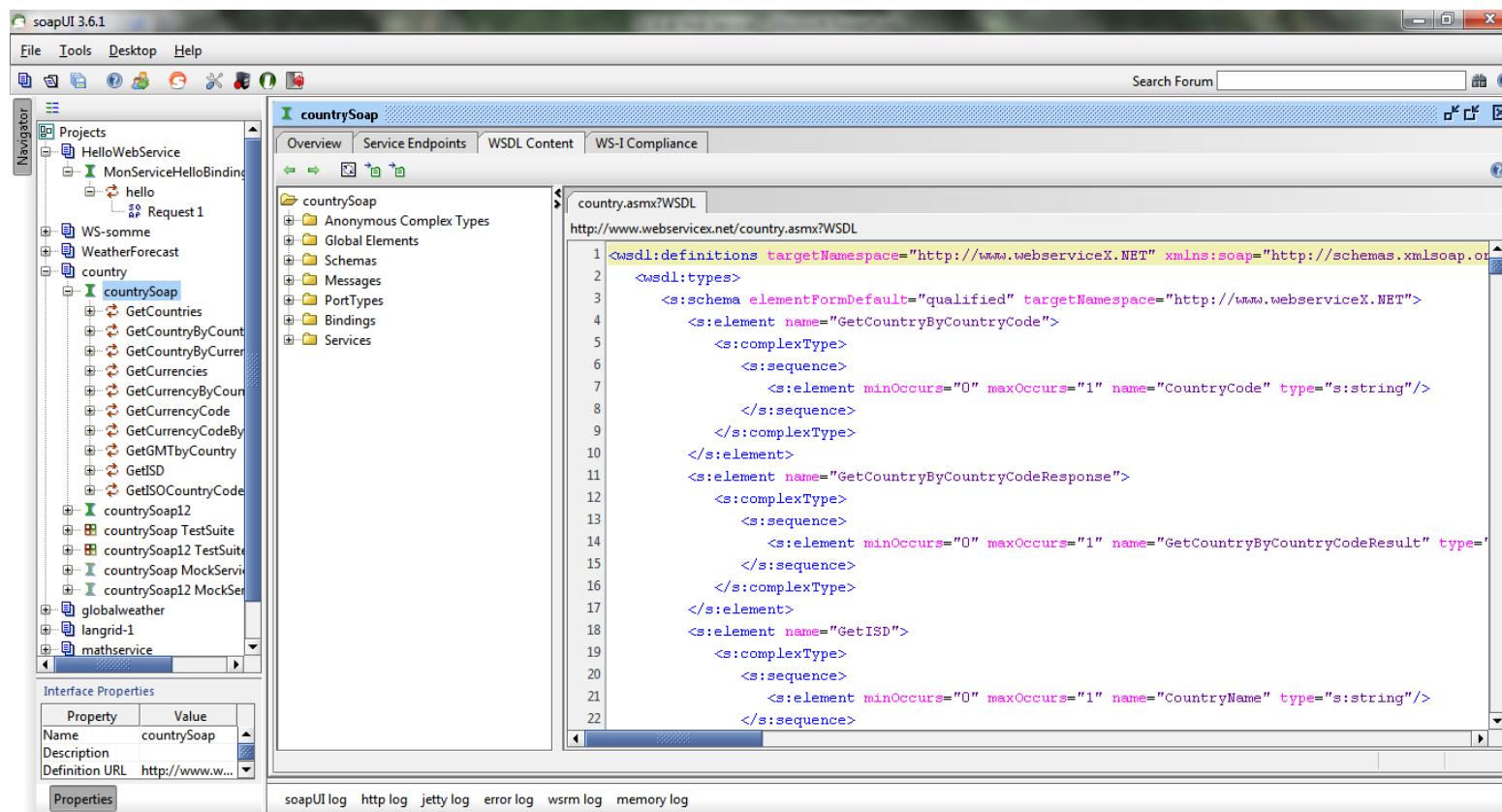
Operations

Name	Use	One-Way	Action
GetCountries	Literal	false	<a href="http://www.webserviceX.NET/GetCountries">http://www.webserviceX.NET/GetCountries</a>
GetCountryByCountryCode	Literal	false	<a href="http://www.webserviceX.NET/GetCountryByCountryCode">http://www.webserviceX.NET/GetCountryByCountryCode</a>
GetCountryByCurrency	Literal	false	<a href="http://www.webserviceX.NET/GetCountryByCurrency">http://www.webserviceX.NET/GetCountryByCurrency</a>
GetCurrencies	Literal	false	<a href="http://www.webserviceX.NET/GetCurrencies">http://www.webserviceX.NET/GetCurrencies</a>
GetCurrencyByCountry	Literal	false	<a href="http://www.webserviceX.NET/GetCurrencyByCountry">http://www.webserviceX.NET/GetCurrencyByCountry</a>
GetCurrencyCode	Literal	false	<a href="http://www.webserviceX.NET/GetCurrencyCode">http://www.webserviceX.NET/GetCurrencyCode</a>
GetCurrencyCodeByCurrencyName	Literal	false	<a href="http://www.webserviceX.NET/GetCurrencyCodeByCurrencyName">http://www.webserviceX.NET/GetCurrencyCodeByCurrencyName</a>
GetGMTbyCountry	Literal	false	<a href="http://www.webserviceX.NET/GetGMTbyCountry">http://www.webserviceX.NET/GetGMTbyCountry</a>
GetISD	Literal	false	<a href="http://www.webserviceX.NET/GetISD">http://www.webserviceX.NET/GetISD</a>
GetISOCountryCodeByCountryName	Literal	false	<a href="http://www.webserviceX.NET/GetISOCountryCodeByCountryName">http://www.webserviceX.NET/GetISOCountryCodeByCountryName</a>

soapUI log http log jetty log error log wsrm log memory log

neila.benlakhal@gmail.com

# SOAPUI: Web service Country: WSDL



neila.benlakhal@gmail.com

## L'appel SOAP du service et de sa méthode GetCurrencyByCountry avec le paramètre CountryName =Japan:

The screenshot shows the soapUI 3.6.1 interface. The left pane displays the project structure under 'HelloWebService' with a tree view of 'MonServiceHelloBinding' and 'country' services, including various methods like 'GetCountries', 'GetCountryByCountryCode', and 'GetCurrencies'. The main workspace shows two tabs: 'Request 1' and 'Response 1'. The 'Request 1' tab contains the XML code for a SOAP message to the 'GetCurrencyByCountry' method, specifying 'Japan' as the country name. The 'Response 1' tab shows the XML response, which includes a table with two rows, each representing a currency entry for Japan (JPY). A green box highlights the response area with the text 'La réponse SOAP du service'. At the bottom, there are log tabs for 'soapUI log', 'http log', 'jetty log', 'error log', 'wsm log', and 'memory log'. The status bar indicates a response time of 1954ms (858 bytes).

Request Properties

Property	Value
Entitize Properti...	false
Pretty Print	true
Dump File	

Properties

Response time: 1954ms (858 bytes)

Headers

La réponse SOAP du service

3:1



## **Invocation du web service country via un script .php:clientcountry.php 1/5**

- Importer la librairie NuSOAP. Celle-ci permettra à PHP de communiquer à travers le protocole SOAP :

```
require_once('..../lib/nusoap.php');
```

- Créer une instance de la classe client en utilisant le constructeur soapclient: ce constructeur prend comme paramètre obligatoire l'URL du fichier WSDL du web service:

```
$wsdl = "http://www.webservicex.net/country.asmx?WSDL ";
$client = new soapclient($wsdl, 'wsdl');
```

## Explication clientcountry.php 2/5

- ➊ Vérifier s'il y a eu des erreurs lors de la création du client : l'instance client n'a pu être créée

```
// vérifier s'il y a eux des erreurs lors de la création de l'instance client
$err = $client->getError();
if ($err) {
    // s'il y a une erreur, afficher l'erreur et arrêter le script
    echo '<h2>erreur de création de client SOAP</h2><pre>' . $err . '</pre>';
    exit();
```



## Explication clientcountry.php 3/5

Appeler la méthode du service web : la méthode **GetCurrencyByCountry** appelé avec *Japan* comme valeur du paramètre **CountryName**

- Méthode 1: passer les paramètre en tant que éléments XML
- Méthode 2: passer les paramètres en tant que tableau php

```
//appeler la méthode du service web : méthode 1
$param = '<GetCurrencyByCountry
xmlns="http://www.webserviceX.NET">
    <CountryName>Japan</CountryName>
    </GetCurrencyByCountry>';
//appeler et afficher le résultat
print_r($client->call('GetCurrencyByCountry', $param));
```



## Explication clientcountry.php 4/5

- Méthode 2: passer les paramètres en tant que tableau php

```
//appeler la méthode du service web : méthode 2
print_r($client->call('GetCurrencyByCountry',
    array('CountryName' => 'Japan')));
```



## Explication clientcountry.php 5/5

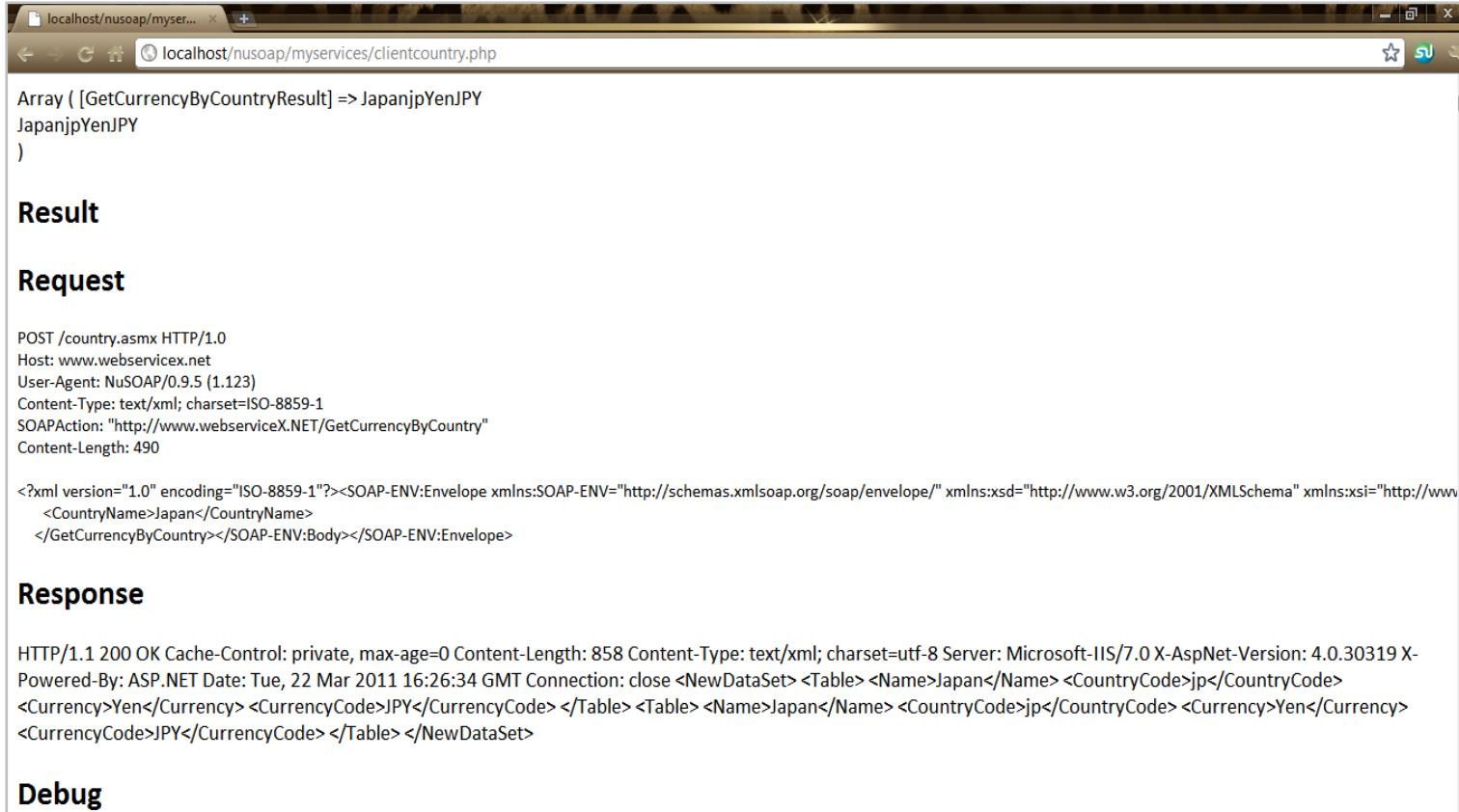
- Fonctions facultatives: Afficher le contenu de la requête et de la réponse SOAP ainsi que le debug de l'invocation:
- Convertit tous les caractères spéciaux en entité HTML.
- htmlspecialchars(\$chaine, ENT\_QUOTES) converti tous les caractères spéciaux en entités HTML (ex: '<' devient '&lt;' etc.)

```
// Display the request and response (SOAP messages)
echo '<h2>Request</h2>';
echo '<pre>' . htmlspecialchars($client->request, ENT_QUOTES) . '</pre>';
echo '<h2>Response</h2>';
echo '<pre>' . htmlspecialchars($client->response, ENT_QUOTES) . '</pre>';
// Display the debug messages
echo '<h2>Debug</h2>';
echo '<pre>' . htmlspecialchars($client->debug_str, ENT_QUOTES) . '</pre>';?>
```

The screenshot shows a Notepad++ window with the following details:

- Title Bar:** C:\wamp\www\nusoap\myservices\clientcountryver2.php - Notepad++
- Menu Bar:** Fichier, Edition, Recherche, Affichage, Encodage, Langage, Paramétrage, Macro, Exécution, TextFX, Compléments, Documents, ?
- Toolbar:** Standard file operations (New, Open, Save, Print, Find, Replace, Cut, Copy, Paste, etc.)
- Tab Bar:** clientcountryver2.php (selected), weightconvertclient.php, quoteclient.php, hellsdl12.php, hellsdl12client.php, new\_15.xml, clientcount
- Code Area:** PHP code for a SOAP client. The code imports NuSOAP, creates a client instance, checks for errors, calls a service method, displays requests and responses, and outputs debug messages.

```
1 <?php // On importe la librairie NuSOAP
2 require_once("../lib/nusoap.php");
3 //créer une instance client
4 $wsdl = "http://www.webservicex.net/country.asmx?WSDL ";
5     $client = new soapclient($wsdl, 'wsdl');
6
7 // vérifier s'il y a eux des erreurs lors de la création de l'instance client
8 $err = $client->getError();
9 if ($err) { // si il y aru une erreur, afficher l'erreur et arreter le script
10     echo '<h2>erreur de creation de client SOAP</h2><pre>' . $err . '</pre>';
11     exit();
12 }
13 //appeler la méthode du service web
14 print_r($client->call('GetCurrencyByCountry', array('CountryName' => 'Japan')));
15
16 // Display the request and response (SOAP messages)
17 echo '<h2>Request</h2>';
18 echo '<pre>' . htmlspecialchars($client->request, ENT_QUOTES) . '</pre>';
19 echo '<h2>Response</h2>';
20 echo $client->response;
21 // Display the debug messages
22 echo '<h2>Debug</h2>';
23 echo '<pre>' . htmlspecialchars($client->debug_str, ENT_QUOTES) . '</pre>';
24 ?>
25
```



The screenshot shows a web browser window with the URL `localhost/nusoap/myservices/clientcountry.php`. The page content displays the following PHP code:

```
Array ( [GetCurrencyByCountryResult] => JapanjpYenJPY  
JapanjpYenJPY  
)
```

**Result**

**Request**

```
POST /country.asmx HTTP/1.0  
Host: www.webservicex.net  
User-Agent: NuSOAP/0.9.5 (1.123)  
Content-Type: text/xml; charset=ISO-8859-1  
SOAPAction: "http://www.webservicex.NET/GetCurrencyByCountry"  
Content-Length: 490  
  
<?xml version="1.0" encoding="ISO-8859-1"?><SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"><SOAP-ENV:Body><GetCurrencyByCountry></GetCurrencyByCountry></SOAP-ENV:Body></SOAP-ENV:Envelope>
```

**Response**

```
HTTP/1.1 200 OK Cache-Control: private, max-age=0 Content-Length: 858 Content-Type: text/xml; charset=utf-8 Server: Microsoft-IIS/7.0 X-AspNet-Version: 4.0.30319 X-Powered-By: ASP.NET Date: Tue, 22 Mar 2011 16:26:34 GMT Connection: close <NewDataSet> <Table> <Name>Japan</Name> <CountryCode>jp</CountryCode> <Currency>Yen</Currency> <CurrencyCode>JPY</CurrencyCode> </Table> <Table> <Name>Japan</Name> <CountryCode>jp</CountryCode> <Currency>Yen</Currency> <CurrencyCode>JPY</CurrencyCode> </Table> </NewDataSet>
```

**Debug**

### Request

```
POST /country.asmx HTTP/1.0
Host: www.webservicex.net
User-Agent: NuSOAP/0.9.5 (1.123) Content-Type: text/xml; charset=ISO-8859-1
SOAPAction: "http://www.webserviceX.NET/GetCurrencyByCountry"
Content-Length: 490
<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:ns9659="http://tempuri.org">
<SOAP-ENV:Body>
<GetCurrencyByCountry
xmlns="http://www.webserviceX.NET">
<CountryName>Japan</CountryName>
</GetCurrencyByCountry>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

### Response

```
HTTP/1.1 200 OK Cache-Control: private, max-age=0
Content-Length: 858 Content-Type: text/xml; charset=utf-8
Server: Microsoft-IIS/7.0 X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
Date: Tue, 22 Mar 2011 20:52:45 GMT Connection: close
<NewDataSet>
<Table> <Name>Japan</Name>
<CountryCode>jp</CountryCode> <Currency>Yen</Currency>
<CurrencyCode>JPY</CurrencyCode> </Table> <Table>
<Name>Japan</Name> <CountryCode>jp</CountryCode>
<Currency>Yen</Currency>
<CurrencyCode>JPY</CurrencyCode>
```

## To do

- Write a client for a available Web service on Internet.

# **WEB SERVICE AVEC UN TYPE COMPLEXE ET PLUSIEURS ENDPOINTS**



## Web service avec type complexe

- Créer un web service avec les deux fonctions suivantes :

```
//fonction simple (first endpoint)
function hello($username) {
    return 'Howdy, '.$username.'!';
}

//fonction qui retourne un type complexe XML(tableau)
function login($username, $password) {
    //peut par exemple accéder à une BD mySQL
    return array( 'id_user'=>1,
        'fullname'=>'WEB',
        'email'=>web@soa.com, 'level'=>100 ); }
```

# Ajout d'un type complexe

```
require_once("../lib/nusoap.php");
$server = new nusoap_server;
$server->configureWSDL('web service avec type complexe', 'urn:localhost');
$server->wsdl->schemaTargetNamespace = 'urn:localhost';
//SOAP complex type return type (an array/struct)
$server->wsdl->addComplexType(
    'Person', //nom du type complexe
    'complexType', //le type est un type complexe
    'struct', // le type est un tableau associatif
    'all', //doit contenir tous ces champs
    '',
    array(
        'id_user' => array('name' => 'id_user', 'type' => 'xsd:int'),
        'fullname' => array('name' => 'fullname', 'type' => 'xsd:string'),
        'email' => array('name' => 'email', 'type' => 'xsd:string'),
        'level' => array('name' => 'level', 'type' => 'xsd:int')
    ) //tableau des éléments
);
```

# Enregistrer les fonctions

```
$server->register('hello',
    array('username' => 'xsd:string'), //paramètres
    array('return' => 'xsd:string'), //output
    'urn:localhost', //namespace
    'urn:localhost#helloServer', //soapaction
    'rpc', // style
    'encoded', // use
    'Just say hello'); //description

//this is the second webservice entry point/function
$server->register('login',
    array('username' => 'xsd:string', 'password'=>'xsd:string'), //paramètres
    array('return' => 'tns:Person'), //output
    'urn:localhost', //namespace
    'urn:localhost#loginServer', //soapaction
    'rpc', // style
    'encoded', // use
    'Check user login'); //description
```

C:\wamp\www\nusoap\myservices\clientComplexEndpoints.php - Notepad++

Fichier Edition Recherche Affichage Encodage Langage Paramétrage Macro Exécution TextFX Compléments Documents ?

wswithcomplextype.php TP JS1.htm webservicescomplexEndpoints.php clientComplexEndpoints.php

```
1 <?php
2 require_once('../lib/nusoap.php');
3
4 //This is your webservice server WSDL URL address
5 $wsdl = "http://localhost/nusoap/myservices/webservicescomplexEndpoints.php?wsdl";
6 //create client object
7 $client = new nusoap_client($wsdl, 'wsdl');
8 $err = $client->getError();
9 if ($err) {
10     // Display the error
11     echo '<h2>Constructor error</h2>' . $err;
12     // At this point, you know the call that follows will fail
13     | exit();
14 }
15
16 //calling our first simple entry point
17 $result1=$client->call('hello', array('username'=>'Web'));
18 print_r($result1);
19 echo "<br/>";
20
21 //call second function which return complex type
22 $result2 = $client->call('login', array('username'=>'Web', 'password'=>'SOA') );
23 // $result2 would be an array/struct
24 print_r($result2);
25 ?>
```

PHP Hypertext Preprocessor file length : 776 lines : 25 Ln : 24 Col : 9

neila.benlakhal@gmail.com

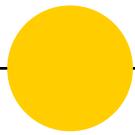
## La méthode AddComplexType

- La méthode **AddComplexType** prédéfini dans NUSOAP permet de définir ses propres types complexes :
- cette méthode prend comme paramétré un nom, et indique à WSDL qu'il s'agit d'un type complexe avec le (struct) etc..
- Signature de AddComplexType:

```
Wsdl->addComplexType ($name,  
$typeClass = 'complexType',  
$phpType = 'array',  
$compositor = '',  
$restrictionBase = '',  
$elements = array(),  
$attrs = array(),  
$arrayType = ""  
  
Parameters:  
string      name  
string      typeClass (complexType|simpleType|attribute)  
string      phpType: array and struct (php assoc array)  
string      compositor (all|sequence|choice)  
string      restrictionBase namespace:name  
(http://schemas.xmlsoap.org/soap/encoding/:Array)  
array       elements = array ( name => array(name=>" ,type=>" ) )  
array       attrs = array(array('ref=>'SOAP-  
ENC:arrayType','wsdl:arrayType='=>'xsd:string[]'))  
string      arrayType: namespace:name (xsd:string)
```

neila.benlaabid@gmail.com

# **Implémentation des Web services Approche Top-down**



## L'approche top-down

- À l'inverse de l'approche bottom-up utilisée pour « SOAiser » un système d'information informatique déjà développé, où on fait des compositions de services web, l'approche top-down est utilisé pour informatiser un processus métier à travers la décomposition des processus et l'identification des différents services qui le compose.
- Application de cette approche :

Nous utiliserons une librairie SOAP qui fait partie de PhP5 et qui est préinstallé sur vos machines avec l'environnement WAMP.

## SOAP pour PHP5

- Soap library (soap.dll) est une extension qui est par défaut désactivé sur WAMP.
- Pour l'activer :
- Lancez WAMP server
- Cliquez sur l'icône du serveur WAMP, choisissez **PHP** puis **PHP extensions** puis **php\_soap**.
- Voilà, l'extension SOAP est active sur votre serveur.

## L'approche top-down

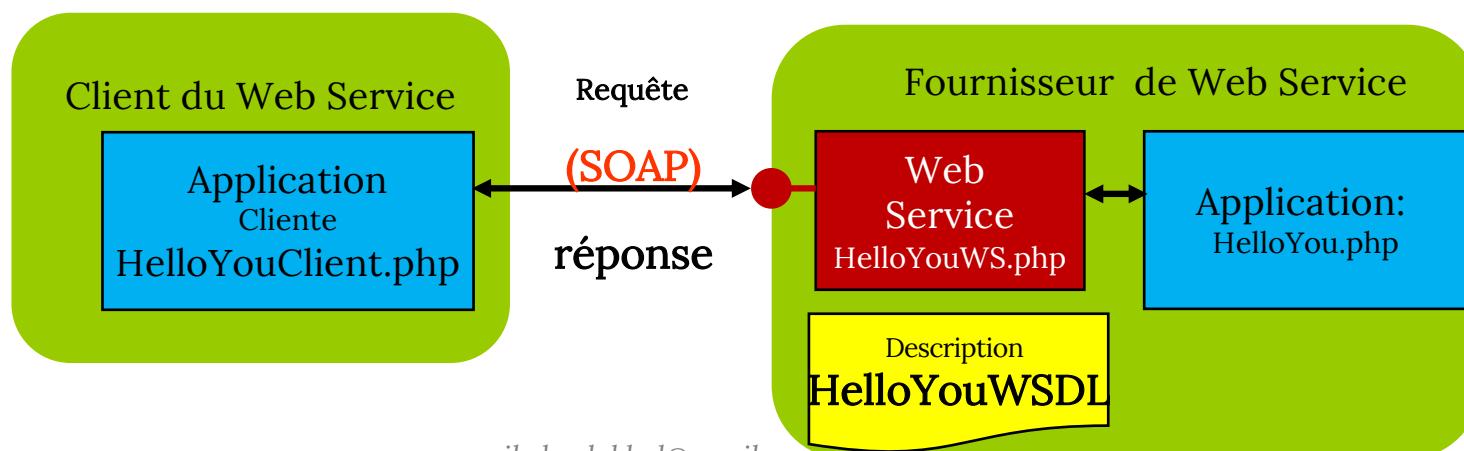
- Cet approche exige qu'on a le fichier .WSDL du web service!
- Dans certains cas on a juste une application php mais pas de fichier WSDL !
- La librairie SOAP incluse dans php5 ne permet pas l'invocation de WS qui n'ont pas de descriptif WSDL.
- Solution:

Approche 2: Bottom-up: commence seulement avec un script .php

Utiliser NUSOAP: une librairie open-source qui génère .wsdl d'un script php.

## L'approche top-down

- Comme pour le développement d'application Web on va simuler le fournisseur de service et le client du service sur une même machine où WAMP est installé et la librairie SOAP est activée :



# L'approche top-down

## ● Initialement, on a :

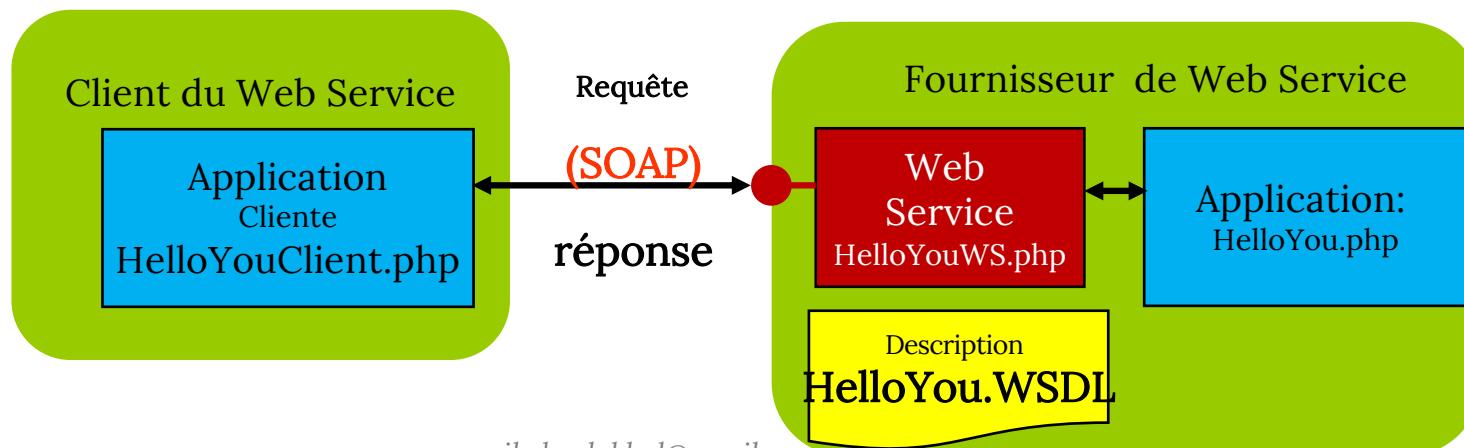
HelloYou.php, un script php

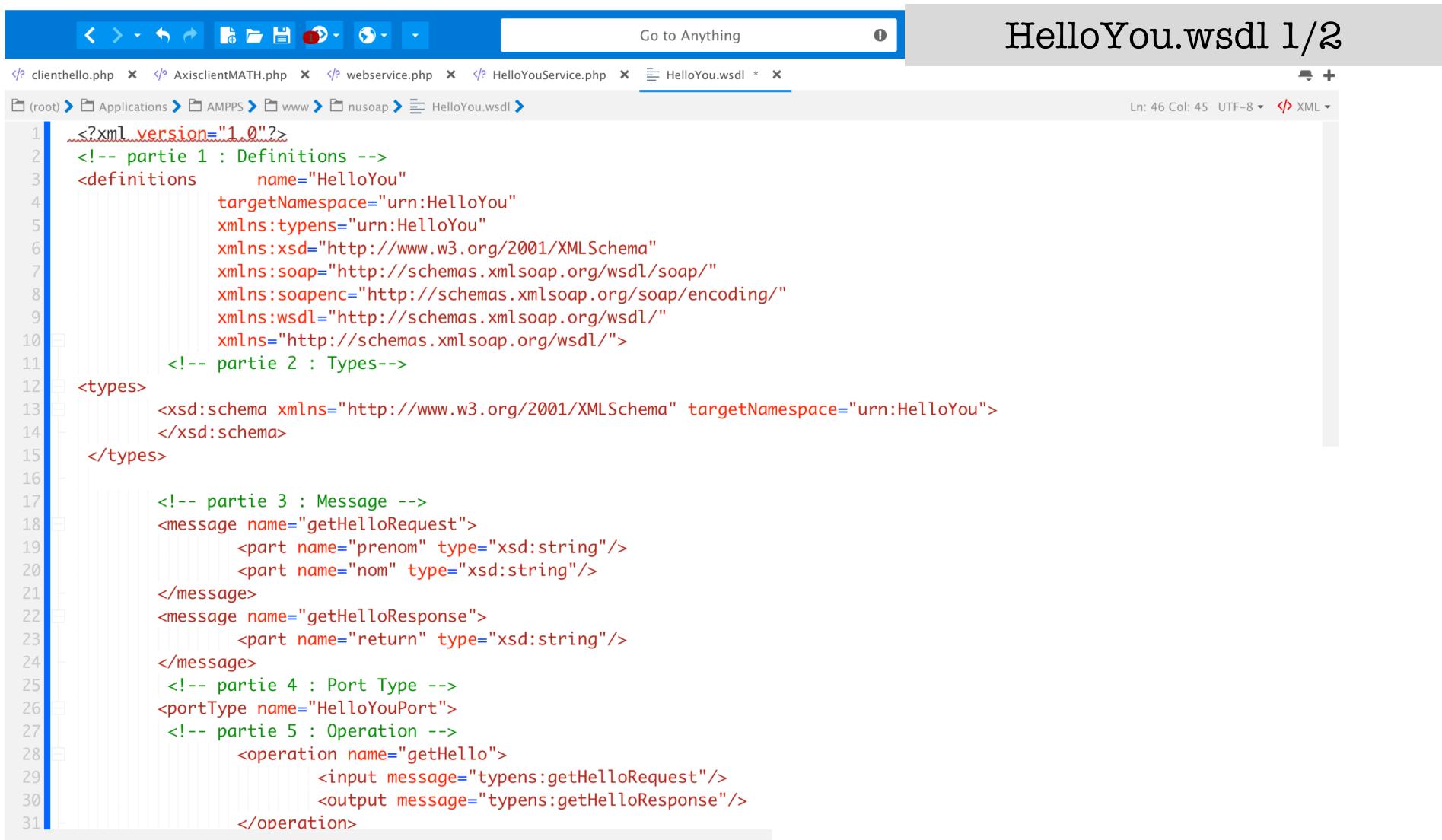
Helloyou.wsdl, le fichier description du Web service

## ● À préparer:

Web service: HelloYouWS.php

Application cliente: HelloyouClient.php





The screenshot shows a web browser window with the title "HelloYou.wsdl 1/2". The address bar indicates the file is located at "HelloYou.wsdl". The page content displays the XML code of the WSDL file, which defines a service named "HelloYou" with a single operation "getHello".

```
<?xml version="1.0"?>
<!-- partie 1 : Definitions -->
<definitions name="HelloYou"
    targetNamespace="urn:HelloYou"
    xmlns:typens="urn:HelloYou"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns="http://schemas.xmlsoap.org/wsdl/">
    <!-- partie 2 : Types-->
<types>
    <xsd:schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="urn:HelloYou">
        </xsd:schema>
    </types>

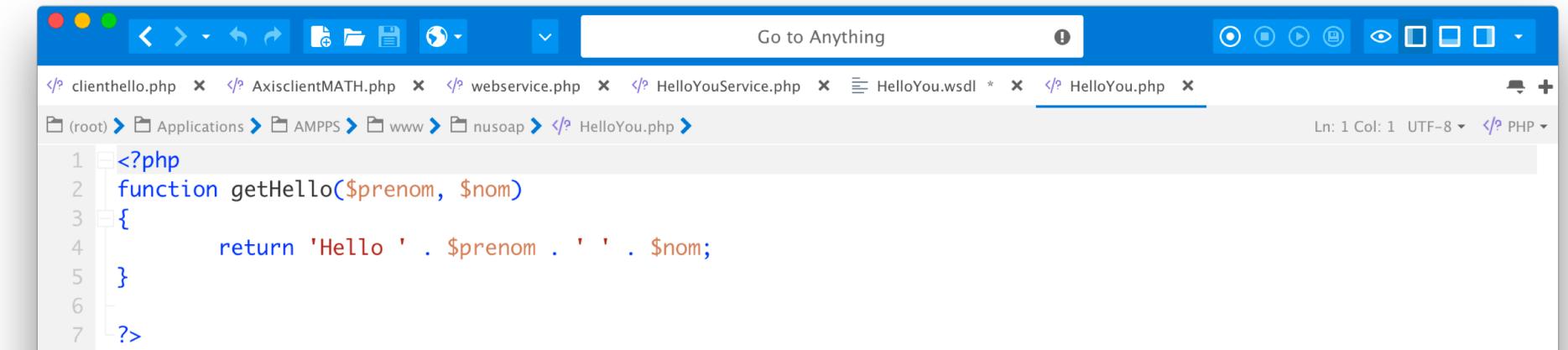
    <!-- partie 3 : Message -->
    <message name="getHelloRequest">
        <part name="prenom" type="xsd:string"/>
        <part name="nom" type="xsd:string"/>
    </message>
    <message name="getHelloResponse">
        <part name="return" type="xsd:string"/>
    </message>
    <!-- partie 4 : Port Type -->
<portType name="HelloYouPort">
    <!-- partie 5 : Operation -->
        <operation name="getHello">
            <input message="typens:getHelloRequest"/>
            <output message="typens:getHelloResponse"/>
        </operation>
    </portType>
</definitions>
```

## HelloYou.wsdl 2/2

```
</> clientHello.php </> AxisclientMATH.php </> webservice.php </> HelloYouService.php </> HelloYou.wsdl * </>
File Edit View Insert Tools Window Help Go to Anything
File Applications AMPPS www nusoap HelloYou.wsdl
Ln: 46 Col: 45 UTF-8 </> XML
<!-- partie 4 : Port Type -->
<portType name="HelloYouPort">
<!-- partie 5 : Operation -->
<operation name="getHello">
    <input message="typens:getHelloRequest"/>
    <output message="typens:getHelloResponse"/>
</operation>
</portType>
<!-- partie 6 : Binding -->
<binding name="HelloYouBinding" type="typens:HelloYouPort">
    <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="getHello">
        <soap:operation soapAction="HelloYouAction"/>
        <input name="getHelloRequest">
            <soap:body use="encoded"
                       namespace="urn:HelloYou"
                       encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
        </input>
        <output name="getHelloResponse">
            <soap:body use="encoded"
                       namespace="urn:HelloYou"
                       encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
        </output>
    </operation>
</binding>
<!-- partie 7 : Service -->
<service name="HelloYouService">
    <documentation>Retourne une phrase simple </documentation>
    <!-- partie 8 : Port -->
    <port name="HelloYouPort" binding="typens:HelloYouBinding"><!-- modifier ce chemin vers server.php -->
        <soap:address location="http://localhost/nusoap/HelloYouService.php"/>
    </port>
</service>
</definitions>
```

# Préparation du Web service

- On a le script PHP: HelloYou.php

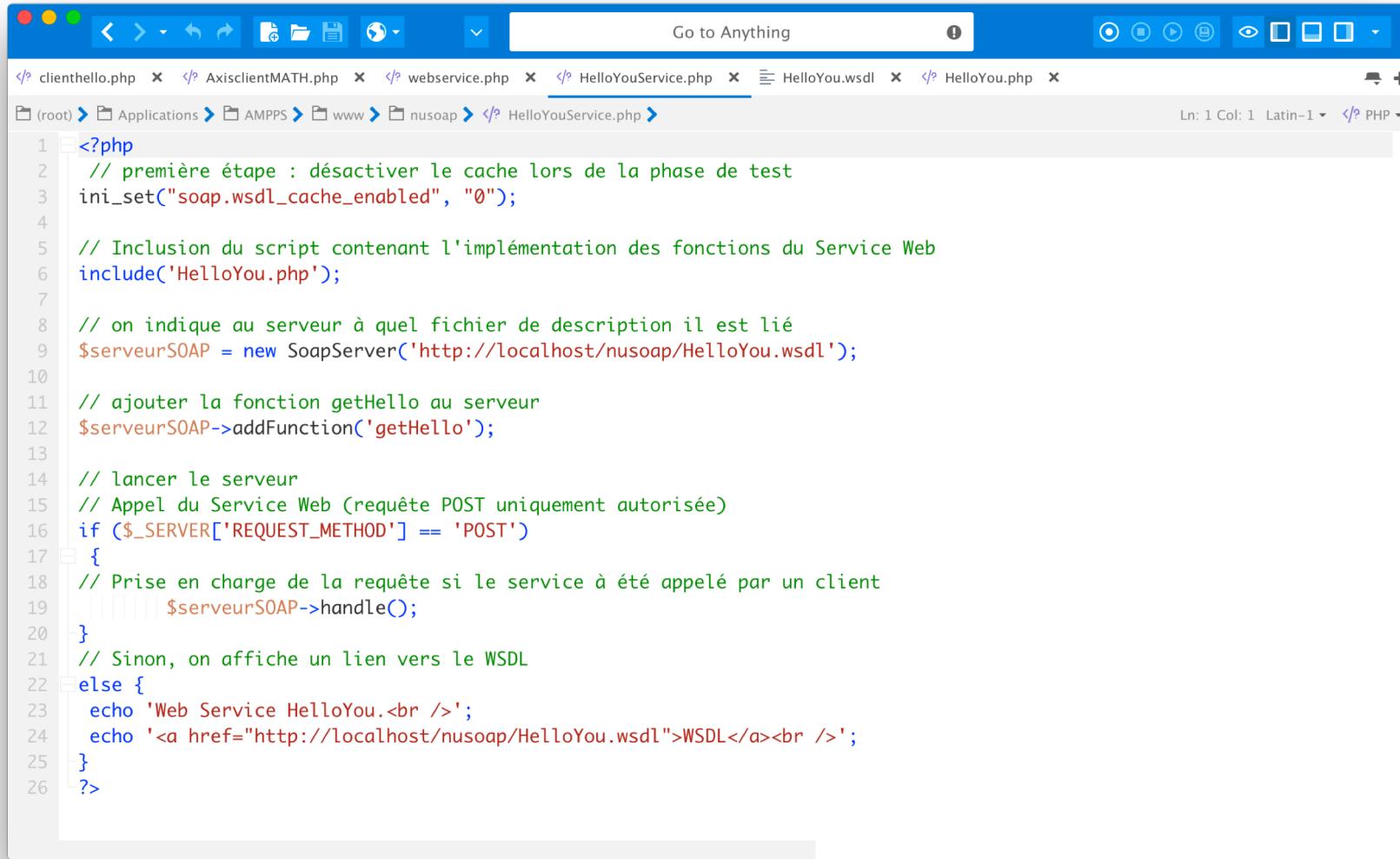


```
<?php
function getHello($prenom, $nom)
{
    return 'Hello ' . $prenom . ' ' . $nom;
}
?>
```

- Inclure le script php dans un web service: HelloYouWS.php

# Le web service

## HelloYouWS.php



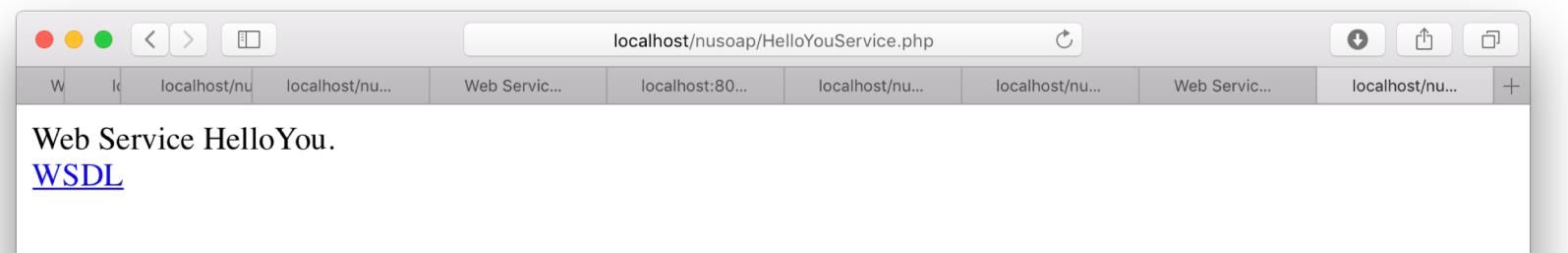
```
1 <?php
2 // première étape : désactiver le cache lors de la phase de test
3 ini_set("soap.wsdl_cache_enabled", "0");
4
5 // Inclusion du script contenant l'implémentation des fonctions du Service Web
6 include('HelloYou.php');
7
8 // on indique au serveur à quel fichier de description il est lié
9 $serveurSOAP = new SoapServer('http://localhost/nusoap/HelloYou.wsdl');
10
11 // ajouter la fonction getHello au serveur
12 $serveurSOAP->addFunction('getHello');
13
14 // lancer le serveur
15 // Appel du Service Web (requête POST uniquement autorisée)
16 if ($_SERVER['REQUEST_METHOD'] == 'POST')
17 {
18 // Prise en charge de la requête si le service a été appelé par un client
19 // $serveurSOAP->handle();
20 }
21 // Sinon, on affiche un lien vers le WSDL
22 else {
23 echo 'Web Service HelloYou.<br />';
24 echo '<a href="http://localhost/nusoap/HelloYou.wsdl">WSDL</a><br />';
25 }
26 ?>
```



## Déploiement du Web service

● Coté serveur:

- HelloYou.php
- HelloYou.wsdl
- HelloYouWS.php
- Le Web service est déployé par le fournisseur de service et il est prêt à l'utilisation :





## Consommation du Web service

### ○ HelloyouClient.php:

HelloYouClient.php

The screenshot shows a code editor window with the file `HelloYouClient.php` open. The code is a PHP script that consumes a SOAP web service. It includes comments in French explaining the steps: disabling the cache, linking the client to the WSDL file, executing the `getHello` method, and displaying SOAP requests and responses for debugging. The code uses the `SoapClient` class and the `__getLastRequest()` and `__getLastResponse()` methods to inspect the generated SOAP messages.

```
<?php
// première étape : désactiver le cache lors de la phase de test
ini_set("soap.wsdl_cache_enabled", "0");

// lier le client au fichier WSDL
$clientSOAP = new SoapClient('HelloYou.wsdl',array('trace' => 1));

// executer la méthode getHello
echo $clientSOAP->getHello('SARAH', 'BEN');

// Affichage des requêtes et réponses SOAP (pour debug)
// il faut que Trace soit à TRUE pour que ça marche
echo '<br />Requête SOAP : '.htmlspecialchars($clientSOAP->__getLastRequest()).'<br />';
echo '<br />Réponse SOAP : '.htmlspecialchars($clientSOAP->__getLastResponse()).'<br />';

?>
```

A screenshot of a web browser window titled "localhost/nusoap/HelloYouClient.php". The browser has multiple tabs open, all showing "localhost/nu...". The main content area displays two SOAP messages.

**Hello SARAH BEN**

Requete SOAP : <?xml version="1.0" encoding="UTF-8"?> <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ns1="urn:HelloYou" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/" SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"><SOAP-ENV:Body><ns1:getHello><prenom xsi:type="xsd:string">SARAH</prenom><nom xsi:type="xsd:string">BEN</nom></ns1:getHello></SOAP-ENV:Body></SOAP-ENV:Envelope>

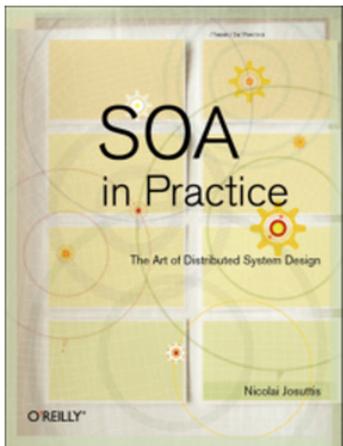
Reponse SOAP : <?xml version="1.0" encoding="UTF-8"?> <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ns1="urn:HelloYou" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/" SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"><SOAP-ENV:Body><ns1:getHelloResponse><return xsi:type="xsd:string">Hello SARAH BEN</return></ns1:getHelloResponse></SOAP-ENV:Body></SOAP-ENV:Envelope>

neila.benlakhal@gmail.com 92

## Bottom-up vs. Top-down

- Extrait du livre SOA in practice
- Voir site du cours pour une version électronique du livre.

Nicolai M. Josuttis: **SOA in Practice**



[Nicolai M. Josuttis](#)

**SOA in Practice  
The Art of Distributed System Design**

O'Reilly

ISBN-10: 0-596-52955-4  
ISBN-13: 978-0-596-52955-0

In general, there are two different approaches to dealing with these kinds of problems:

- In the *top-down* approach, you decompose a problem, system, or process into smaller chunks until you reach the level of (basic) services.
- In the *bottom-up* approach, you build business processes by composing services into more general chunks.

All experts agree that in practice, a pure application of either of these approaches does not work. Of course, you can design your processes from the top down, which will help you to understand what is needed and what might be a “natural” separation of activities. However, ignoring existing realities might result in high costs compared with an approach that also takes the existing bottom layer into account. On the other hand, designing business processes from the bottom up introduces the danger of proposing technical details and constraints to very high levels, making the processes inflexible and unintuitive.

[BloombergSchmelzer06] puts it as follows:

Your approach to SOA should be both top-down (through process decomposition) and bottom-up (exposing existing functionality as Services and composing them into processes). If you take only a top-down approach, you’re likely to recommend building Services that are technically difficult or complex to implement. Taking solely a bottom-up approach can yield Services you don’t need or, even worse, Services that don’t fulfill the requirements of the business.

So, in practice, a mixture of both approaches is appropriate. (There are many different names for such a mixed approach, such as “middle-out,” “middle-ground,” “meet in the middle,” or just “agile.”)

Usually, your aim should be to fulfill a real business need. That is, a new or modified business process or a new functionality should come into play. However, you should also respect what you already have (adapting IT reality).

neila.benlakhal@gmail.com



# WSDL

Web Service Description Language

## Web Service Description Language

- Les Web Services offrent un moyen de décrire leurs interfaces suffisamment en détail pour permettre à un utilisateur de créer une application cliente capable de converser avec eux.
- Cette description est fournie dans un document **WSDL** : l'acronyme de **Web Service Description Language**

# WSDL

- Un document WSDL :

Est basé sur le langage XML et ayant l'extension **.wsdl**  
Fournit une description suffisamment en détail d'un WS pour permettre à un utilisateur de créer une application cliente capable de converser avec lui

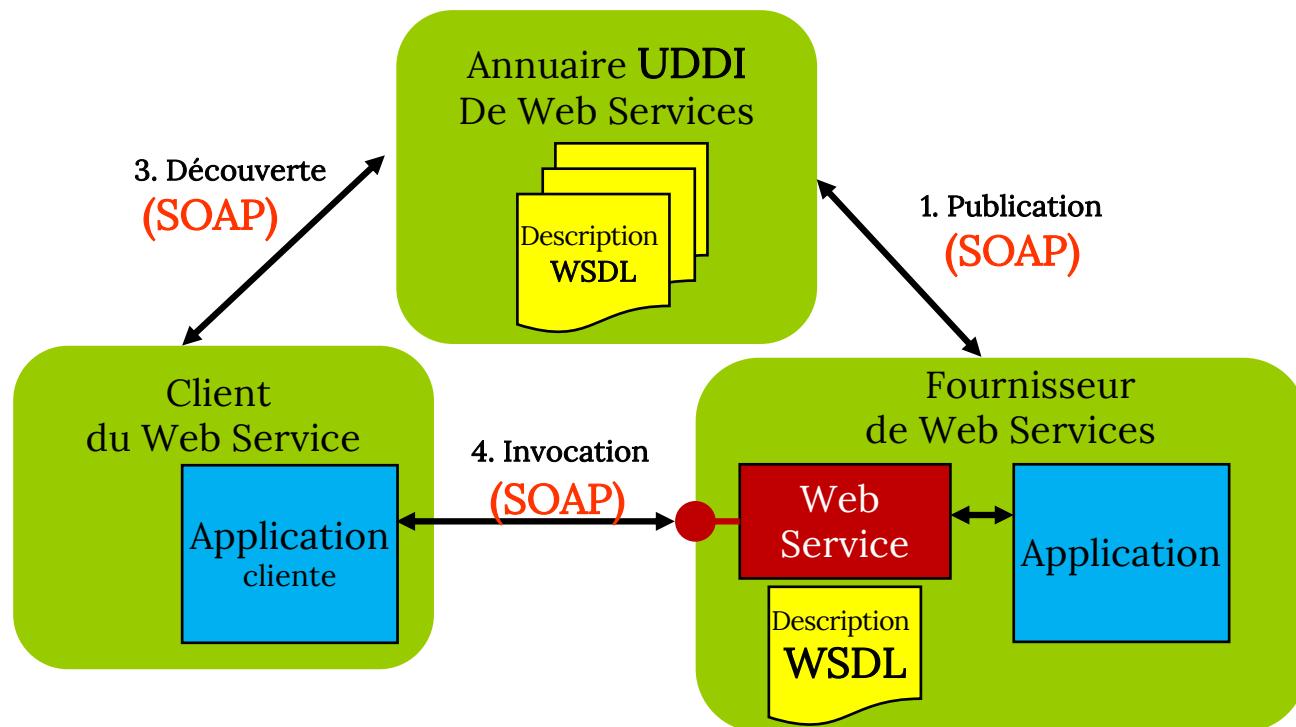
Une description exacte du Web Service par les méthodes avec les types de paramètres requises et des réponses avec les types de paramètres renvoyés

Fournit une description indépendante du langage et de la plate-forme

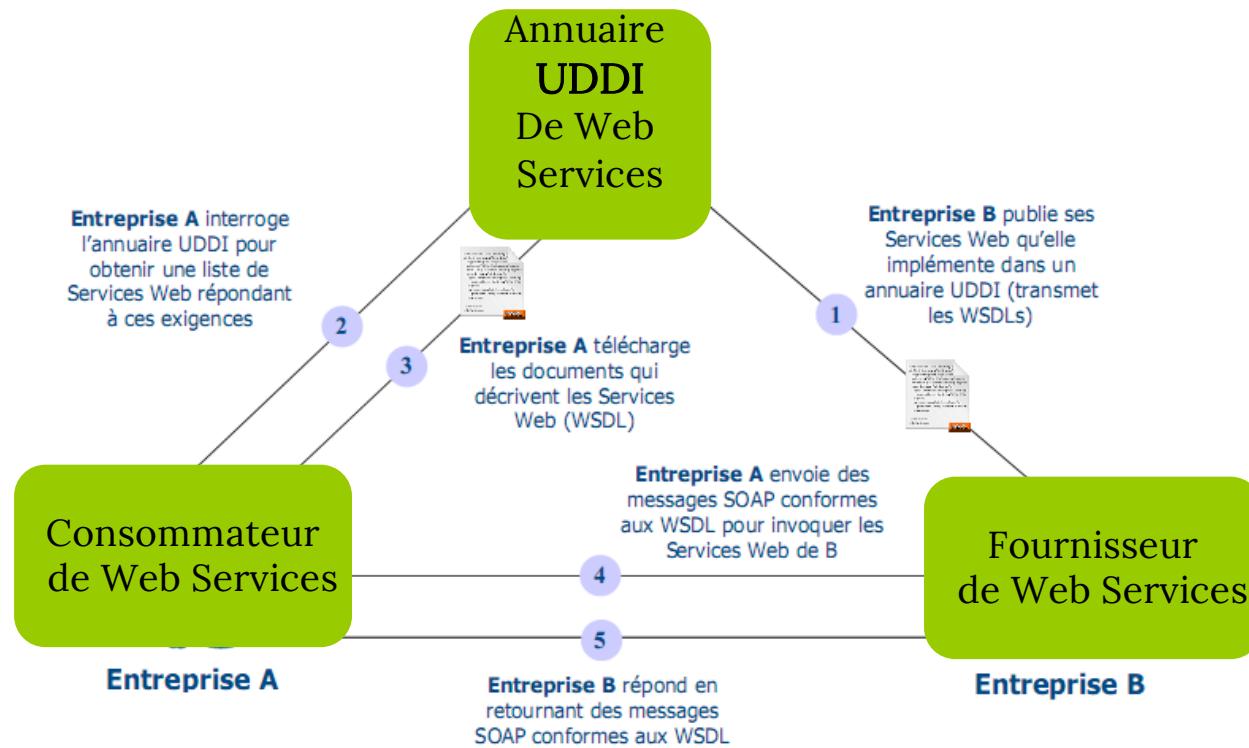
## **Web Service Description Language**

- Initiative de Ariba, IBM et Microsoft
- Spécification standard géré par le W3C
  - WSDL 1.1, (2001) : <http://www.w3.org/TR/wsdl>
  - WSDL 2.0 (2003) : <http://www.w3.org/TR/wsdl20/>
- A partir d'un document WSDL il est possible :
  - Générer un client pour invoquer un Web Service
  - Générer le code pour implémenter un Web Service

## Où est utilisé WSDL ?



# Où est utilisé WSDL ?



## Structure d'un Document WSDL

- Un document WSDL ayant l'extension .wsdl est structuré comme suit:

Il commence par la balise **<definitions>**

```
<definitions>
    <types>... </types>
    <message>... </message>
    <porttype>...</porttype>
    <binding>...</binding>
    <service>...>/service>
</definitions>
```



## Organisation d'un document WSDL

○ Un document WSDL est décomposé en deux parties :

○ **Partie abstraite** : qui décrit les messages et les opérations disponibles

- Types (**<types>**)
- Messages (**<message>**)
- Types de port (**<portType>**)

○ **Partie concrète** : qui décrit le protocole à utiliser et le type d'encodage à utiliser pour les messages

- Bindings (**<binding>**)
- Services (**<service>**)
- Plusieurs parties concrètes peuvent être proposées pour la partie abstraite

○ Motivation de cette séparation ?

- Réutilisabilité de la partie abstraite

## Exemple

- Un web service : Hello World
- Comporte une fonction string sayHello(string)

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="HelloService"
  targetNamespace="http://www.ecerami.com/wsdl/HelloService.wsdl"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns="http://www.ecerami.com/wsdl/HelloService.wsdl"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <message name="SayHelloRequest">
    <part name="firstName" type="xsd:string"/>
  </message>
  <message name="SayHelloResponse">
    <part name="greeting" type="xsd:string"/>
  </message>

  <portType name="Hello_PortType">
    <operation name="sayHello">
      <input message="tns:SayHelloRequest"/>
      <output message="tns:SayHelloResponse"/>
    </operation>
  </portType>
  ...
</definitions>
```

neila.benlakhal@gmail.com

104

```
... <binding name="Hello_Binding" type="tns:Hello_PortType">
  <soap:binding style="rpc"
    transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="sayHello">
    <soap:operation soapAction="sayHello"/>
    <input>
      <soap:body
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="urn:examples:helloservice"
        use="encoded"/>
    </input>
    <output>
      <soap:body
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="urn:examples:helloservice"
        use="encoded"/>
    </output>
  </operation>
</binding>
<service name="Hello_Service">
  <documentation>WSDL File for HelloService</documentation>
  <port binding="tns:Hello_Binding" name="Hello_Port">
    <soap:addresslocation="http://localhost:8080/soap/servlet/rpcrouter"/>
  </port>
</service>
</definitions>
```

neila.benlakhal@gmail.com

## L'élément <definitions>

<definitions> ... </definitions>

C'est l'élément racine du document WSDL,

Il donne le nom du service (optionnel, attribut **name**),  
déclare les espaces de noms utilisés, et contient les  
éléments du service.

L'attribut **targetNamespace** permet au document wsdl de  
se référer à lui-même .

Il est ***obligatoire***.

```
<definitions name="HelloService"
  targetNamespace="http://www.ecerami.com/wsdl/HelloService.wsdl"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns="http://www.ecerami.com/wsdl/HelloService.wsdl"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

## L'élément **<types>**

- **<types>...</types>**
- (optionnel et 1 seul autorisé)
  - Contient la définition des types des données importées sous forme de Schéma XML
  - Ou encore les types de données à transmettre

## L'élément <message>

### 🟡<message> ...</message> (plusieurs autorisés)

Décrit des messages à transmettre (paramètre d'une opération, valeur de retour, exception, ...)

```
<message name="SayHelloRequest">
    <part name="firstName" type="xsd:string"/>
</message>
<message name="SayHelloResponse">
    <part name="greeting" type="xsd:string"/>
</message>
```

neila.benlakhal@gmail.com

108

## L'élément **<portType>**

### ● **<portType>...</portType>(plusieurs autorisés)**

Décrit un ensemble d'opérations où chacune à 0 ou plusieurs messages en entrée, 0 ou plusieurs messages de sortie ou de fautes

```
<portType name="Hello_PortType">
    <operation name="sayHello">
        <input message="tns:SayHelloRequest"/>
        <output message="tns:SayHelloResponse"/>
    </operation>
</portType>
```

*neila.benlakhal@gmail.com*

## L'élément <binding>

### ○ <binding>...</binding> (plusieurs autorisés)

Spécifie comment les messages vont être transmis:  
une liaison entre un portType à un protocole (SOAP,  
HTTP)

```
<binding name="Hello_Binding" type="tns:Hello_PortType">
    <soap:binding style="rpc"
        transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="sayHello">
        <soap:operation soapAction="sayHello"/>
        <input><soap:body
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            namespace="urn:examples:helloservice" use="encoded"/>
        </input>
        <output><soap:body
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            namespace="urn:examples:helloservice" use="encoded"/>
        </output>
    </operation>
</binding>
```

## L'élément <service>

● <service>...</service> (plusieurs autorisés)

Regroupe l'ensemble des ports (relation entre binding et URL)

```
<service name="Hello_Service">
  <documentation>WSDL File for HelloService</documentation>
  <port binding="tns:Hello_Binding" name="Hello_Port">
    <soap:address location="http://localhost:8080/soap/servlet/rpcrouter"/>
  </port>
</service>
```



## WSDL par l'exemple : Carnet d'adresse

- Le service Notebook fournit 4 opérations :

- Une opération **addPerson** qui prend en paramètre un objet Person et retourne un booléen pour indiquer l'état de création:

boolean **addPerson**(Person p)

- Une opération **addPerson** qui prend paramètres 3 chaînes de caractères (name, address et birthyear) sans retour :

void **addPerson**(string name,string address,string birthyear)



## WSDL par l'exemple : Carnet d'adresse

- Une opération **getPersonByName** qui prend en paramètre une chaîne de caractère et retourne un objet Person :

Person **getPersonByName(string name)**

- Une opération **getPersons** sans paramètre en entrée et qui retourne un tableau d'objets Person :

Person[] **getPersons()**

- L'accès au service est réalisé par de message SOAP
- Le protocole utilisé pour l'échange des messages SOAP est HTTP et le style utilisé est du RPC

## <type>

- L'élément <type> contient la définition des types utilisés pour décrire la structure des messages échangés par le SW
- Le système de typage est généralement un Schéma XSD
- Cet élément peut être facultatif si les types utilisés par les messages sont des types de bases (Integer, Boolean, ... )
- Dans le cas de structures complexes (Person par exemple) un Schéma XML est alors employé

```
<definitions name="Notebook" targetNamespace="http://notebookwebservice.esti.rnu.tn/
  xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="http://notebookwebservice.esti.rnu.tn/
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
  <types>
    <xsd:schema targetNamespace="http://
      <xsd:complexType name="person">
        <xsd:sequence>
          <xsd:sequence>
            <xsd:element name="address" type="xs:string" minOccurs="0"/>
            <xsd:element name="birthyear" type="xs:string" minOccurs="0"/>
            <xsd:element name="name" type="xs:string" minOccurs="0"/>
          </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="personArray" f
          <xsd:sequence>
            <xsd:element name="item" type="tns:person" minOccurs="0" maxOccurs="unbounded"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:schema>
    </types>
  ...</definitions>
```

Une personne est définie par une adresse, une année de naissance et un nom

Définition d'un type tableau de personne

## <message>

- <message> permet de décrire les messages échangés par les SW :

Paramètres d'entrées des opérations + Paramètres de sorties  
+Exception

- Chaque <message> est identifié par un nom (attribut name) et est constitué d'un ensemble d'éléments <part> :

Un élément <part> correspond à un paramètre d'une opération

L'élément <part> est défini par : attribut **name** et un attribut **type**

```
<definitions targetNamespace="http://notebookwebservice.esti.rnu.tn/"  
    name="Notebook" ...>  
    <types>...</types>  
    <message name="addPersonWithComplexType">  
        <part name="newPerson" type="tns:person"/>  
    </message>  
    <message name="addPersonWithComplexTypeResponse">  
        <part name="addPersonWithComplexTypeResult" type="xsd:boolean"/>  
    </message>
```

Message utilisé pour : boolean addPerson(Person p)

neila.benlakhal@gmail.com

## Si une opération est décrit par plusieurs paramètres, plusieurs éléments <part> seront à définir

```
<definitions targetNamespace="http://notebookwebservice.esti.rnu.tn/"  
    name="Notebook" ...>  
<types>...</types>  
<message name="addPersonWithComplexType">  
    <part name="newPerson" type="tns:person"/>  
    </message>  
<message name="addPersonWithComplexTypeResponse">  
    <part name="addPersonWithComplexTypeResult" type="xsd:boolean"/>  
    </message>  
<message name="addPersonWithSimpleType">  
    <part name="name" type="xsd:string"/>  
    <part name="address" type="xsd:string"/>  
    <part name="birthyear" type="xsd:string"/>  
    </message>  
<message name="getPerson">  
    <part name="personName" type="xsd:string"/>  
    </message>  
<message name="getPersonResponse">  
    <part name="getPersonResult" type="tns:person"/>  
    </message>  
<message name="getPersons"/>  
<message name="getPersonsResponse">  
    <part name="getPersonsResult" type="tns:personArray"/>  
    </message>  
</definitions>
```

The diagram illustrates the mapping of WSDL parts to Java methods. It consists of four colored boxes:

- Blue Box:** Contains the part "newPerson". To its right is the method **boolean addPerson(Person p)**.
- Pink Box:** Contains the parts "name", "address", and "birthyear". To its right is the method **void addPerson(string name, string address, string birthyear)**.
- Purple Box:** Contains the part "personName". To its right is the method **Person getPersonByName(string name)**.
- Green Box:** Contains the part "getPersonsResult". To its right is the method **Person[] getPersons()**.

## **<portType>**

- Un élément **<portType>** est un regroupement d'opérations :

Identifiable par un nom (attribut name)

Composé de sous élément **<operation>**

- Une opération est comparable à une méthode Java :

Identifiable par un nom (attribut name)

La description des paramètres est obtenue par une liste de messages

## Élément <portType> et sous élément <Operation>

- ➊ Une opération exploite les messages via les sous éléments :

<input> : message transmis au service

<output> : message produit par le service

<fault> : message d'erreur (très proche des exceptions)

- ➋ Chaque sous élément possède les attributs suivants :

```
<definitions targetNamespace="http://notebookwebservice.esti.rnu.tn/"  
    name="Notebook" ...>  
<types>...</types>  
<message>...</message>  
<portType name="Notebook">  
    <operation name="addPerson">  
        <input message="tns:addPersonWithComplexType"/>  
        <output message="tns:addPersonWithComplexTypeResponse"/>  
    </operation> ...</portType></definitions> </definitions>
```

# Élément <portType> et sous élément <Operation>

```
<definitions targetNamespace="http://notebookwebservice.esti.rnu.tn/"  
    name="Notebook" ...>  
<types>...</types>  
<message>  
    ...  
</message>  
<portType name="Notebook">  
    <operation name="addPerson">  
        <input message="tns:addPersonWithComplexType"/>  
        <output message="tns:addPersonWithComplexTypeResponse"/>  
    </operation>  
    <operation name="addPerson" parameterOrder="name address birthyear">  
        <input message="tns:addPersonWithSimpleType"/>  
    </operation>  
    <operation name="getPerson">  
        <input message="tns:getPerson"/>  
        <output message="tns:getPersonResponse"/>  
    </operation>  
    <operation name="getPersons">  
        <input message="tns:getPersons"/>  
        <output message="tns:getPersonsResponse"/>  
    </operation>  
</portType>  
</definitions> </definitions>
```

# Les types de ports

- Il y a 4 types d'opérations : messages + les balises <input>, <output>

Opération one-way

Le service reçoit un message <input>.



Opération request-response

Le service reçoit un message requête <input>  
puis renvoie au client un message réponse  
<output> ou un message erreur <fault>.



Opération solicit-response

Le service envoie un message sollicitation <output>  
puis reçoit du client un message réponse <output>  
ou un message d'erreur <fault>.



Opération notification

Le service envoie un message notification <output>.



## Élément <portType> et sous élément <Operation>

- Possibilité de définir une opération suivant 4 modèles :

1. **One-way** : envoie de messages

Le client du service envoie un message à l'opération et n'attend pas de réponse

- Uniquement un seul message utilisé <input>

```
<definitions targetNamespace="http://notebookwebservice.esti.rnu.tn/" name="Notebook" ...>
<types>...</types>
<message>...</message>
<portType name="Notebook">
...
<operation name="addPerson" parameterOrder="name address birthyear">
<input message="tns:addPersonWithSimpleType"/>
</operation>
...
</portType>
</definitions></definitions>
```

## Élément <portType> et sous élément <Operation>

### 2. Request/Response : question – réponse

Le client du service envoie un message à l'opération et un message est retournée au client

- Un message <input>, un message <output> et un message <fault>

```
<definitions targetNamespace="http://notebookwebservice.esti.rnu.tn/" name="Notebook" ...>
<types>...</types>
<message>...</message>
<portType name="Notebook">
...
<operation name="addPerson">
<input message="tns:addPersonWithComplexType"/>
<output message="tns:addPersonWithComplexTypeResponse"/>
</operation> ...
</portType>
</definitions></definitions>
```

## Élément <portType> et sous élément <Operation>

### 3. Notification : notification

- Le service envoie un message au client
- Uniquement un seul message utilisé <output>

```
<operation name="personStatus">
<output message="trackingInformation"/>
</operation>
```

### 4. Solicit - response : sollicitation - réponse

- Le client reçoit un message du service et répond au service
- Un message <output>, un message <input> et un message <fault>

```
<operation name="clientQuery">
<output message="bandWithRequest"/>
<input message="bandwidthInfo" />
<fault message="faultMessasge" />
</operation>
```

## L'élément <binding>

- Un élément <binding> permet de réaliser la partie concrète d'un élément <portType> (décrit avec : un nom (attribut name) et un portType (attribut type))
- Il décrit précisément le protocole à utiliser pour manipuler un élément <portType> :
  - SOAP 1.1 et 1.2
  - HTTP GET & Post (pour le transfert d'images par exemple)
- Plusieurs éléments <binding> peuvent être définis de sorte qu'un élément portType peut être appelé de différentes manières

## Structure générale de l'élément <binding>

- Le schéma XML de WSDL ne décrit pas les sous éléments de binding, operation, input, ouput et fault
- Ces éléments sont spécifiques aux protocoles utilisés

```
<definitions>...
<binding name="NamePortBinding" type="tns:portType">
<!-- Décrit le protocole à utiliser --&gt;
&lt;operation name="operation1"&gt;
<!-- Action du protocole sur l'opération --&gt;
&lt;input&gt;&lt;!-- Action du protocole sur les messages d'entrés (input) --&gt;
&lt;/input&gt;
&lt;output&gt;&lt;!-- Action du protocole sur les messages de sorties (ouput) --&gt;
&lt;/output&gt;
&lt;fault&gt;&lt;!-- Action du protocole sur les messages d'erreurs (fault) --&gt;
&lt;/fault&gt;
&lt;/operation&gt;
&lt;/binding&gt;
...&lt;/definitions&gt;</pre>
```

## Exemple : Définition d'un Binding SOAP 1.1

```
<definitions ...><!-- Définition de la partie Abstraite du WSDL -->
<binding name="NoteBookPortBinding" type="tns:Notebook">
<soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc"/>
<operation name="addPersonWithComplexType">
<soap:operation soapAction="" />
<input>
<soap:body use="literal" namespace="http://notebookwebservice.esti.rnu.tn/" />
</input><output><soap:body use="literal"
namespace="http://notebookwebservice.esti.rnu.tn/" /></output>
</operation>...</binding>
...</definitions>
```

L'attribut transport permet de préciser le protocole à utiliser pour le transport des messages SOAP : HTTP : http://schemas.xmlsoap.org/soap/http  
SMTP , FTP, ...

neila.benlakhal@gmail.com

## Elément Service et Port

- Le Port Type Notebook est accessible en SOAP/HTTP via cette URL :

<http://localhost:8080/NotebookWebService/notebook>

```
<definitions ...>
<!-- Définition de la partie Abstraite du WSDL -->
<binding ...>
</binding>
<service name="Notebook">
<port name="NoteBookPort" binding="tns:NoteBookPortBinding">
<soap:address location="http://localhost:8080/NotebookWebService/notebook"/>
</port>
</service>
</definitions>
```