TP Web services (SOAP)

Les Web services

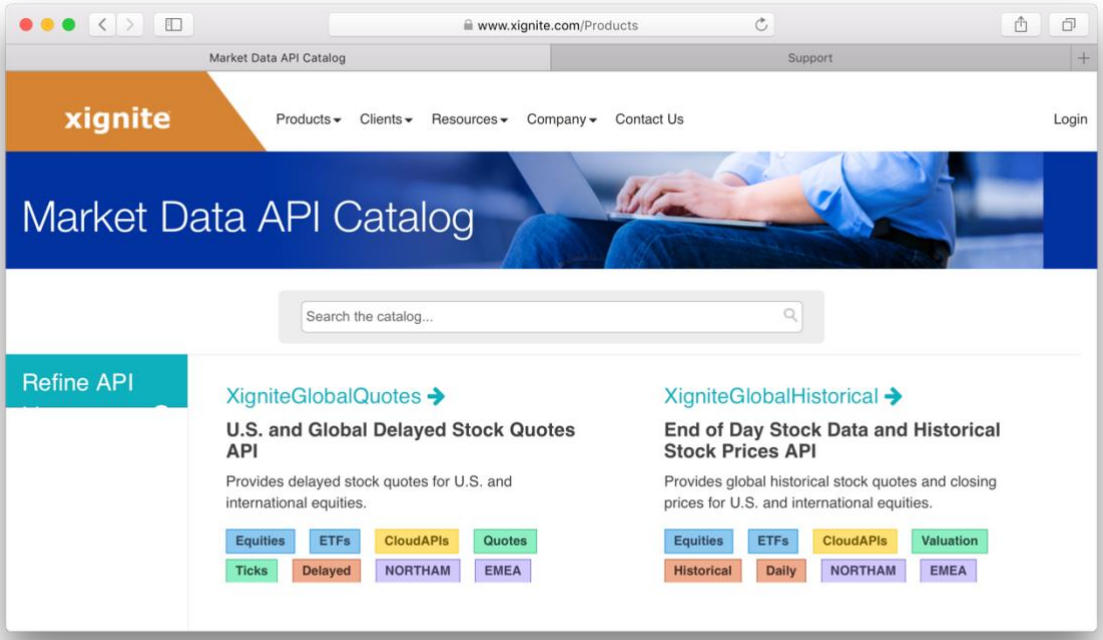
Les outils :

- La librairie Nusoap
- SOAP for PHP5
- Notedpad++ /SublimeText
- .Net Web Service Studio
- SOAPUI
- Eclipse java EE IDE

Créé par : Neila Ben Lakhal

TP Web services (SOAP)

[Exercice 1.] **Accéder à http://www.xignite.com/Products/Catalog.aspx : c'est un annuaire de Web services financiers payant :**



- Accéder à :

  https://www.xignite.com/product/global-stock-quote-data#/DeveloperResources/Code/ListExchanges

- Aller dans l'onglet « code »,

- Parcourir les différents onglets : (différents protocoles d'invocation REST /SOAP), Ouvrir l'onglet : **Sample code**

- C'est des codes sources avec différents langages de programmation pour un client qui veut utiliser cette fonction de ce web service.
- Récupérer le Endpoint du service et l'adresse de son contrat et tester le via SOAPUI.

[Exercice 2.] Développement de Web services -L'approche Bottom-up :

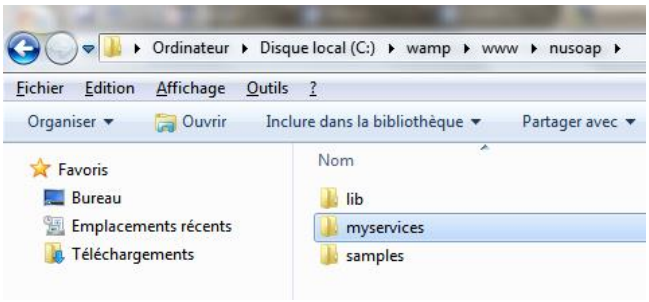
- Soit le script php suivant :

```
<?php
 function mySum($X, $Y) {
                    $total = $X + $Y;
                    return $total; }
$myNumber = 0;
echo "Before the function, myNumber = ". $myNumber ."<br />";
$myNumber = mySum(3,4);
echo "After the function, myNumber = " . $myNumber ."<br />";
?>
```

- Que fait ce script ?
- Ouvrez Notedpad++ et enregistrer le script php dans un fichier somme.php dans un dossier somme sous le dossier WWW de l'environnement WAMP.

Créer le Service Web :

- Le but de cet exercice est de déployer le script somme.php en tant que service Web nommé WS-somme.php :
- Télécharger la librairie NuSOAP : Téléchargement de Nusoap (NuSOAP - SOAP Toolkit for PHP) :nusoap-0.9.5.zip http://sourceforge.net/projects/nusoap/ : la librairie Nusoap est un ensemble de classes Php qui permet la création et la consommation de WS basés sur SOAP 1.1, WSDL 1.1 et HTTP 1.0/1.1
- Extraire le dossier compressé dans le dossier WWW de votre serveur WAMP
- Renommer le dossier compressé : nusoap
- Créer un dossier myservices sous le dossier nusoap et enregistrer le fichier somme.php dans ce dossier sous le nom WS-somme.php



- Appliquer l'approche bottom-up (Bottom-Up : création d'un programme dans un langage donnée (php), puis génération du WSDL et des classes nécessaires. Cette approche masque la partie définition WSDL, elle permet surtout de réutiliser du code existant) pour créer un web service à partir de la fonction mySum de somme.php : vous pouvez vous aider de syntaxe-Webservice.php qui suit :

```
<?php
// On importe la libraire NuSOAP
require_once("../lib/nusoap.php");
// On crée un objet serveur SOAP à partir de la libraire de NuSOAP
$server = new soap_server();
// Paramètres: Nom + Namespace
$server->configureWSDL('MonService','Namespace');
//nous créons ici la fonction –
```

```
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'), si sans valeur de retour
        array('retour' => 'xsd:string'),  // Les sorties
   'Namespace',                 // Le namespace
   "Namespace#nomfonction",         // L'action soap
   'rpc',                       // Style de procédure
   'encoded',                   // Encodé ou non
   'description de la fonction'      // documentation
   );
// Rendre le serveur accessible
$HTTP_RAW_POST_DATA = isset($HTTP_RAW_POST_DATA) ? $HTTP_RAW_POST_DATA : '';
$server->service($HTTP_RAW_POST_DATA);
?>
```

Comment se fait le déploiement d'un Web service ?
Vérifier que votre web service est correctement déployé dans le serveur WAMP
Comment accéder le fichier .WSDL du web service WS-somme.php ?
Analyser le contenu du fichier .WSDL

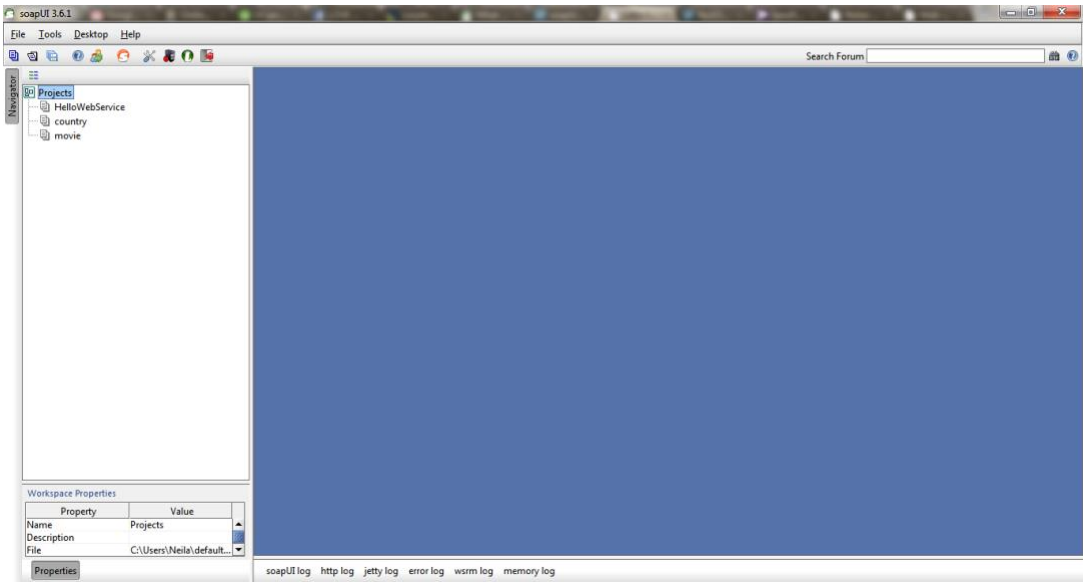
[Exercice 3.] Tester le Service Web : L'outil graphique SOAPUI

Le but de cet exercice est de tester le Web service WS-somme.php.
Il y a deux méthodes possibles pour tester un Web service :

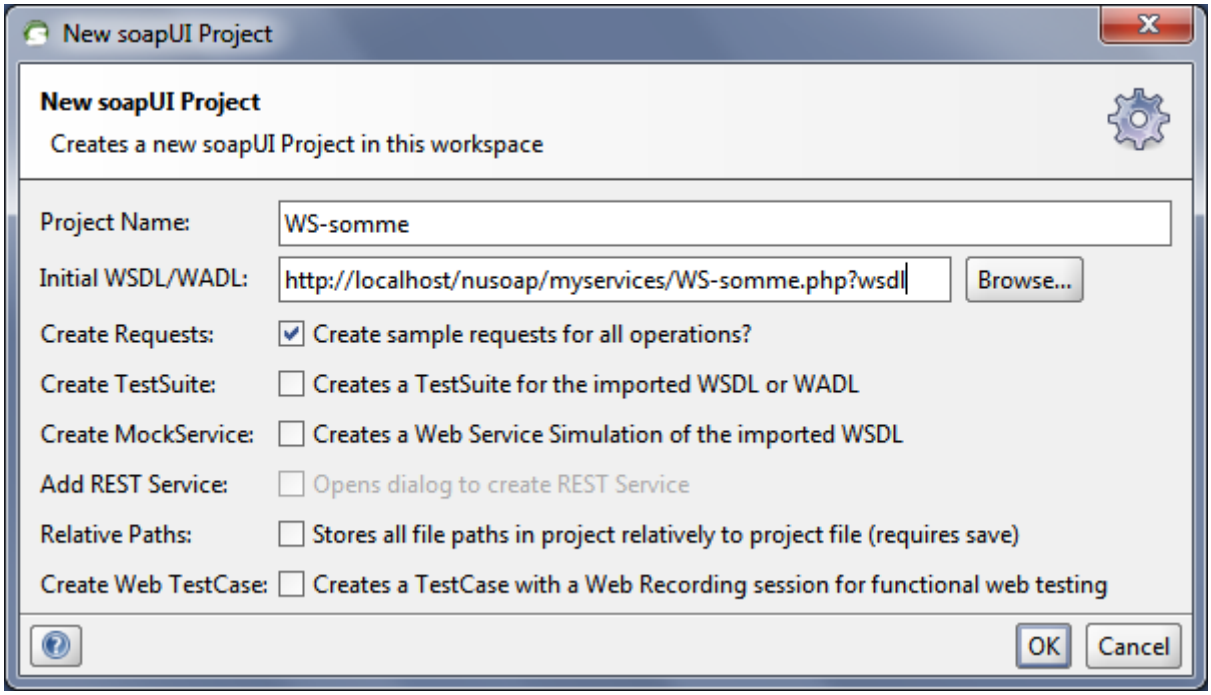
- L'outil graphique SOAPUI
- En invoquant le Web service par un client Client-somme.php

L'outil SOAPUI :
Permet de faire des tests unitaires sur des Web Services déployés.
SOAPUI est un outil Open Source (donc gratuit) développé par la société Eviware. Il est à noter qu'une version commerciale existe.
SOAPUI est disponible ici : http://www.soapui.org/

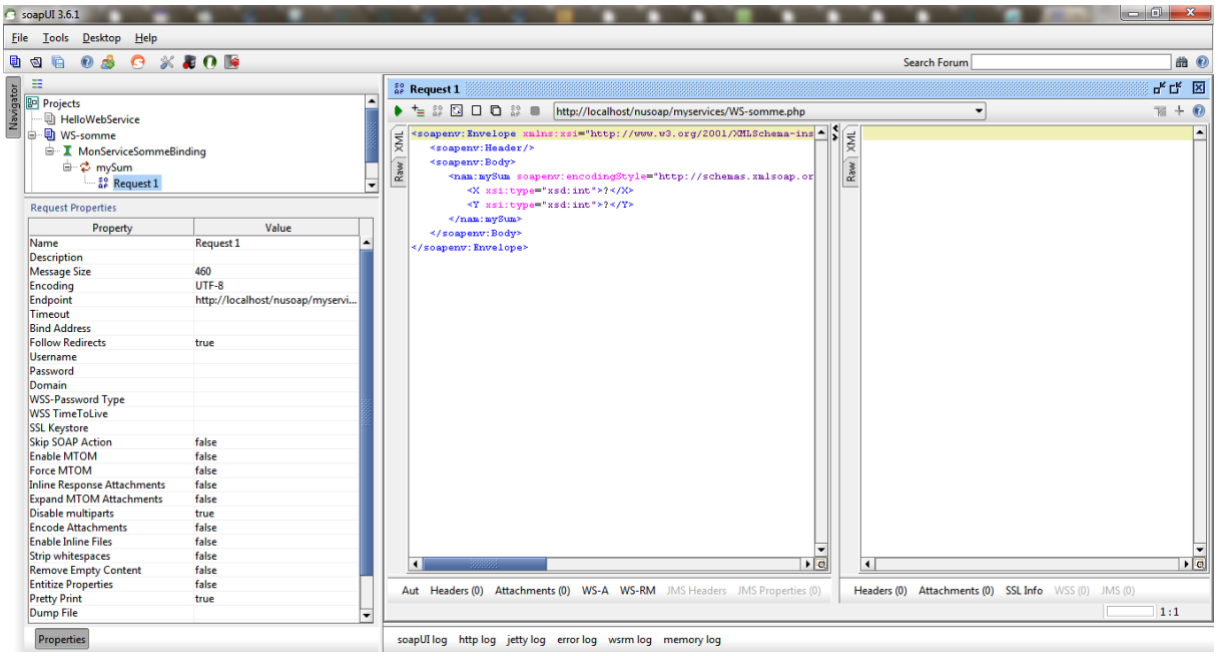
Télécharger l'outil SOAPUI et lancer son installation. Une fois installé, lancez SOAPUI.



La première chose à faire pour tester notre web service est de créer un nouveau projet:
Aller au Menu et cliquer "File", sélectionner "New SoapUI Project".



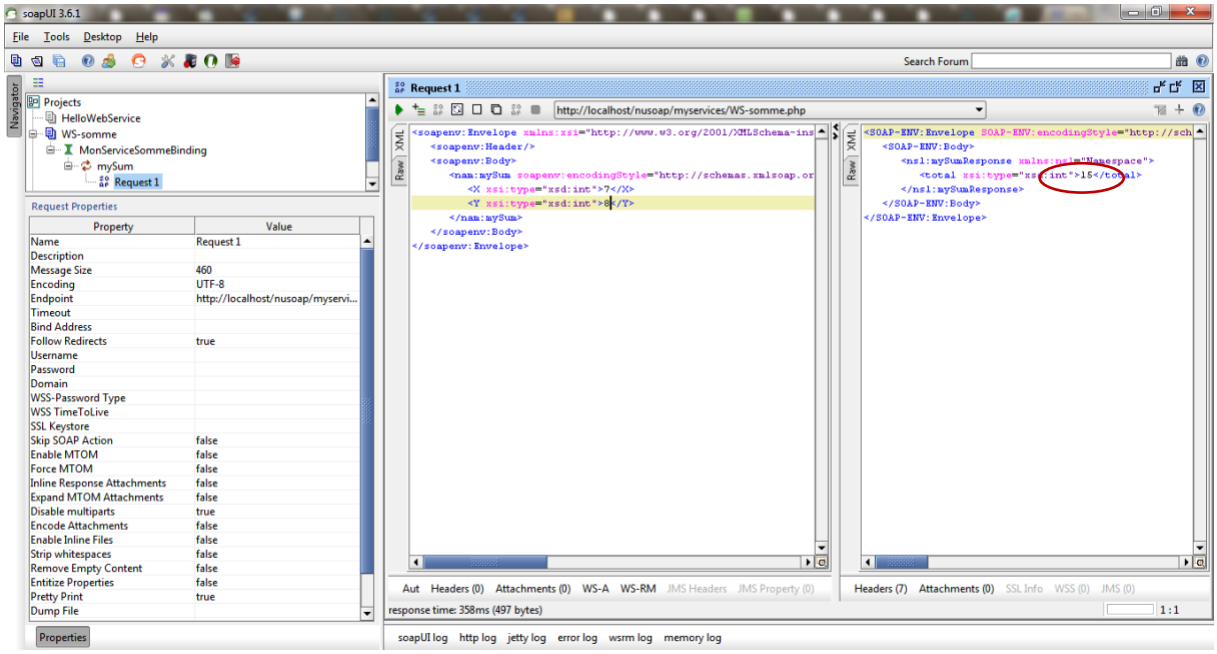
Écrire dans la zone « initial WSDL/WADL » le lien vers le fichier WSDL du web service WS-somme.php : http://localhost/nusoap/myservices/WS-somme.php?wsdl
Cocher la case "Create sample requests for all operations".
Nous allons tester notre web service: WS-somme :
Faites un double clique sur Request1 : le volet d'en face contient une requete SOAP et une réponse SOAP (vide) :remplisser le message SOAP par des valeurs de votre choix à la place des points d'interogration : par exemple 7 et 8 pour faire la somme de 7 et 8 :



une fois la valeur les paramètres renseignés, appuyez sur la flèche en vert pour invoquer le web service : WS-somme.php

[Exercice 4.]

l'exécution du Web service terminée, vous avez la réponse SOAP dans la fenêtre d'en face :



[Exercice 5.] **Tester le Service Web : écrire un client.php**

On va écrire un client Client-somme.php qui va appeler le Web service WS-somme.php :
Référez-vous au client.php suivant pour écrire le code client-somme.php :

```php
<?php require_once("../lib/nusoap.php"); // On importe la libraire NuSOAP
//créer une instance client
$client = new soapclient('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 aru une erreur, afficher l'erreur et arrêter 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>';?>
```

[Exercice 6.] **Test un Web service avec ligne de code de votre choix**

En vous référant au client **clientcountry.php** du cours, écrivez un client pour un Web services que vous choisirez dans la liste qui suit :

NB : certains d'entre eux ne sont peut-être plus disponibles !

Validation Web Services

http://ws.cdyne.com/emailverify/Emai...mail.asmx?wsdl - validate email
http://ws.cdyne.com/phoneverify/phoneverify.asmx?wsdl - validate phone number

Geographic Web Services

http://ws.cdyne.com/ip2geo/ip2geo.asmx?wsdl - find geo location on earth based on Zip code

Communication Web Services

http://ws.acrosscommunications.com/Fax.asmx?WSDL - send a FAX to someone
http://ws.strikeiron.com/ReversePhoneLookup?WSDL - reverse Phone number lookup/validator

Other web sites where you can find Web services

http://www.strikeiron.com/Products/OtherProducts.aspx

http://www.xignite.com/ Financial web services

https://www.programmableweb.com des ressources et des liens vers les web services.

http://www.cdyne.com/products/

http://www.fraudlabs.com/

http://www.pathfinder-xml.com/resources/index.html

[Exercice 7.] Générer .wsdl avec l'outil DIA + UML2PHP5

L'approche top-down exige qu'on a le fichier .WSDL du web service!

Dans certains cas on a juste l'accès à application php mais pas de fichier WSDL !

On va utiliser l'outil DIA avec l'extension UML2PHP5 pour générer le WSDL d'un web service comme celui que vous avez créé à l'exercice 3 (WS-somme) :

Commencer par accédez à l'adresse : http://uml2php5.zpmag.com

Télécharger l'outil DIA et l'extension UML2PHP5

Pour UML2PHP5, décompressez complètement l'archive téléchargée dans le répertoire xslt de Dia. Attention, il faut aussi copier le contenu du sous-répertoire TOOLS présent dans l'archive.

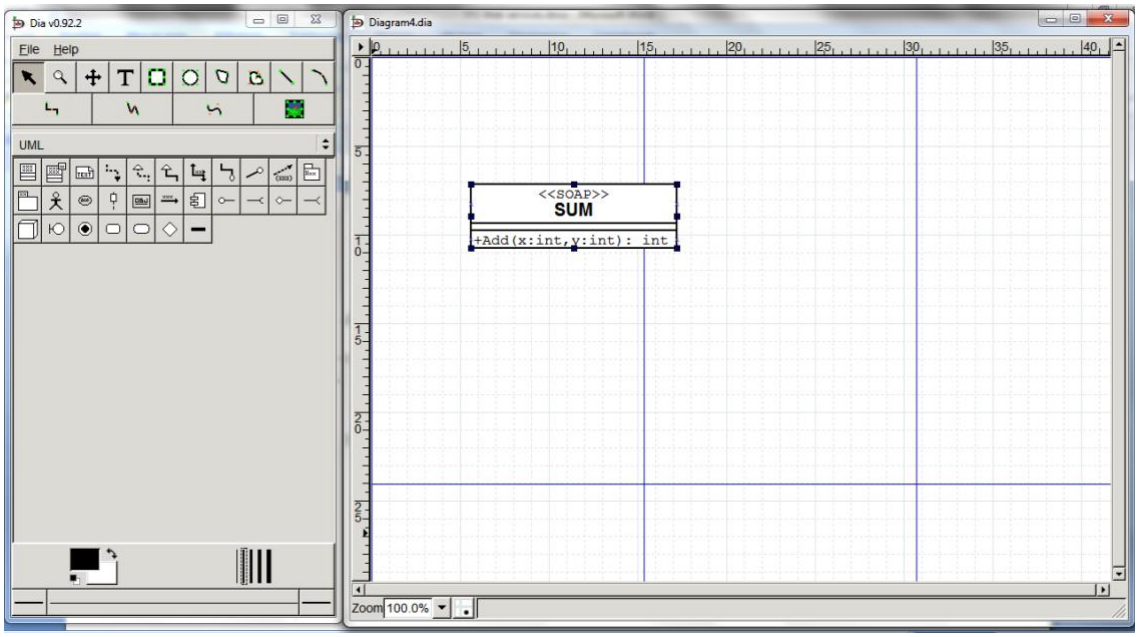
Ensuite lancez Dia, faites votre schéma UML :

Créer un nouveau fichier, dans la boite à outil UML et sélectionner classe

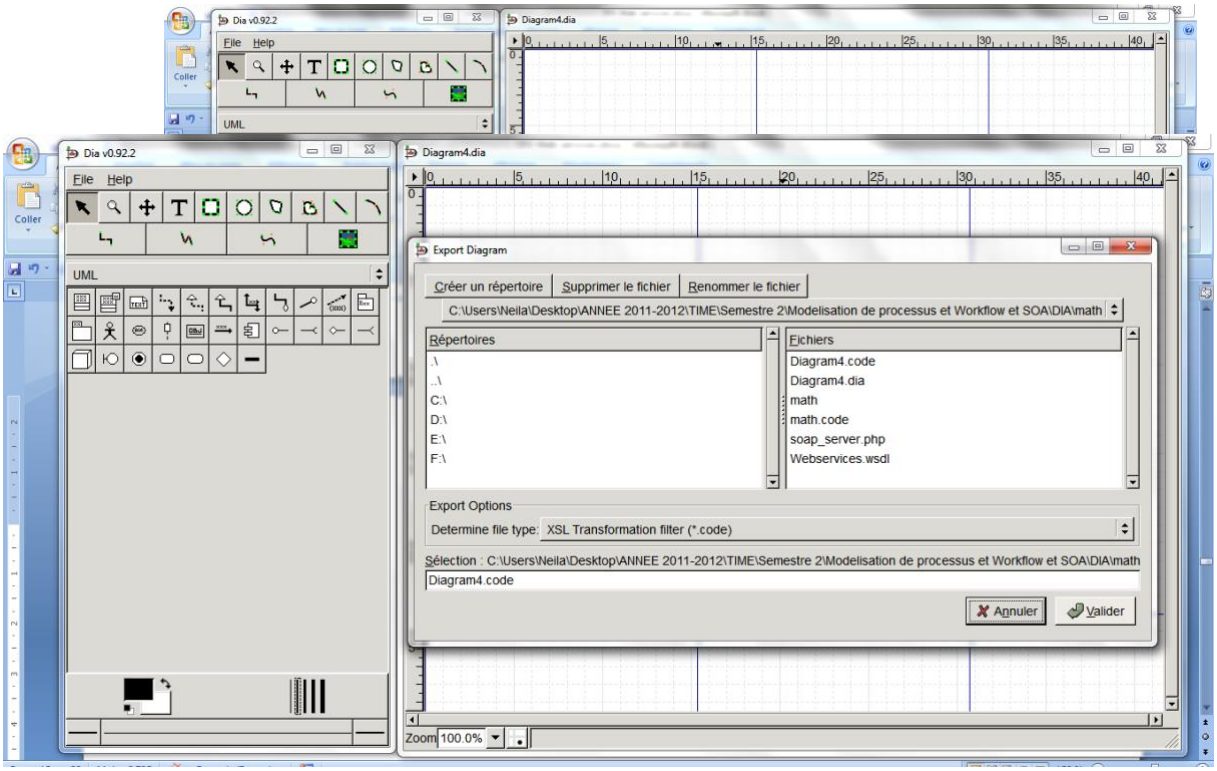
Cliquer sur la classe et configurer là

Avec le name : **sum**, et sans oublier de prototyper votre classe avec SOAP (en Majuscule).

Définir une opération qui s'appelle **add** qui prendra deux paramètres **x** et **y** de type int. La valeur de retour est aussi de type int



Votre classe est prête, on va générer le fichier WSDL d'un web service qui contiendra cette fonction add : enregistrer votre fichier puis aller au menu file export



Cliquer valider après avoir choisi l'emplacement de votre fichier (dans wamp/www/nusoap)

Une petite fenêtre apparaitra :

Sélectionner UML-CLASSES-EXTENDED

Et

PHP5/WSDL/SOAP WEBservices

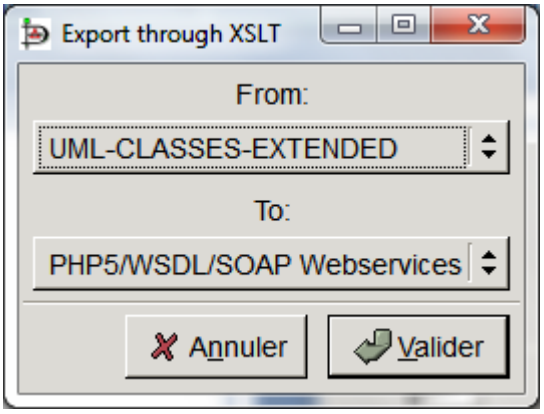
Puis valider.

Les fichiers suivants seront générés pour vous :

- Webservices.wsdl
- soap_server.php

Renommer le fichier **Webservices.wsdl** sous le nom somme.wsdl et éditer-le avec notepad++ ou encore editex pour voir son contenu :

Vérifier qu'il correspond à ce que vous avez défini : messages, part, operation, etc.



[Exercice 8.] Écrire un serveur et un client pour le .wsdl avec l'outil DIA + UML2PHP5

Voici le contenu du fichier **Webservices.wsdl générée par DIA :**

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
This file has been generated by UML2PHP5
UML2PHP5 is free and released under GPL
Copyright KDO kdo@zpmag.com
UML2PHP5 : uml2php5.zpmag.com
--><definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="urn:webservice"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" name="WebServices"
targetNamespace="urn:webservice">
<message name="AddRequest">
<part name="x" type="xsd:int"/>
```

```
<part name="y" type="xsd:int"/>
</message>
<message name="AddResponse">
<part name="return" type="xsd:int"/>
</message>
<portType name="SUMPortType">
<operation name="Add">
<input message="tns:AddRequest"/>
<output message="tns:AddResponse"/>
</operation>
</portType>
<binding name="SUMBinding" type="tns:SUMPortType">
<soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
<operation name="Add">
<soap:operation soapAction="urn:SUMAction"/>
<input>
<soap:body use="encoded" namespace="urn:xmethods"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
</input>
<output>
<soap:body use="encoded" namespace="urn:xmethods"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
</output>
</operation>
</binding>
<service name="Webservice">
<port name="SUMPort" binding="tns:SUMBinding">
<soap:address location="http://URL/soap_server.php?action=SUM"/>
</port>
</service>
</definitions>
```

Pour pouvoir se référer à ce .wsdl, il faut modifier l'adresse du service web pour y mettre à la place l'adresse du script serveur. L'URL à remplacer se trouve tout à la fin dans la balise <service> 'en gras).

Si votre script serveur SOAP tourne sur le poste local (127.0.0.1), dans le `fi`chier serveur.php on obtient la balise XML suivante :

```
<soap:address location="http://127.0.0.1/serveur.php"/>
```

Commencer par créer un serveur SOAP pour le script somme.php :

```
<?php
function Add($x, $y)
{
    return $x+$y;
}
?>
```

Nommer le serveur : **sommeServeur.php** et enregistrer-le dans le dossier NUSOAP

Ainsi la balise service pour vous aura comme contenu :

```
<soap:address location="http://localhost/nusoap/ sommeServeur.php"/>
```

Référez-vous à :

- HelloYouWS.php
- Application cliente: HelloyouClient.php

Du cours pour écrire le serveur et le client de la fonction somme.php

[Exercice 9.] Tester Générer .wsdl avec Eclipse

Dans cet exercice, nous allons apprendre à faire notre premier web-service SOAP à partir de son fichier WSDL.

Pour suivre, il est indispensable d'utiliser Eclipse IDE for Java EE Developers :

http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/heliossr2

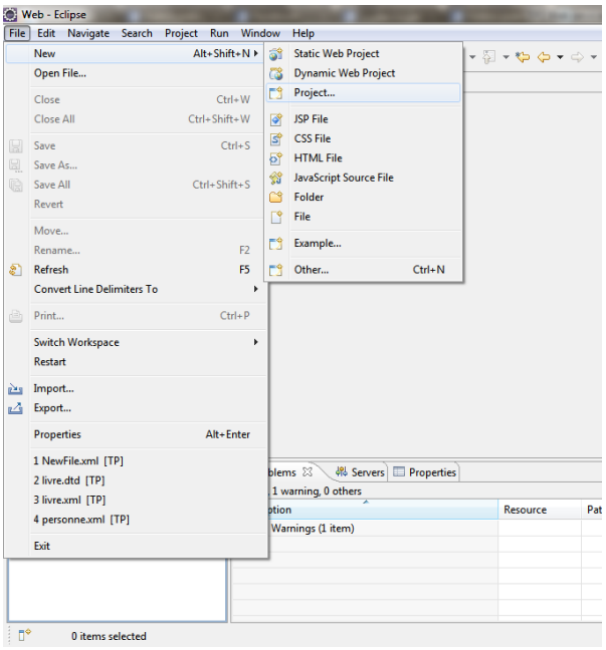
car celui-ci possède un mode graphique pour la création de web-service qui va nous faciliter la vie.

Création du wsdl

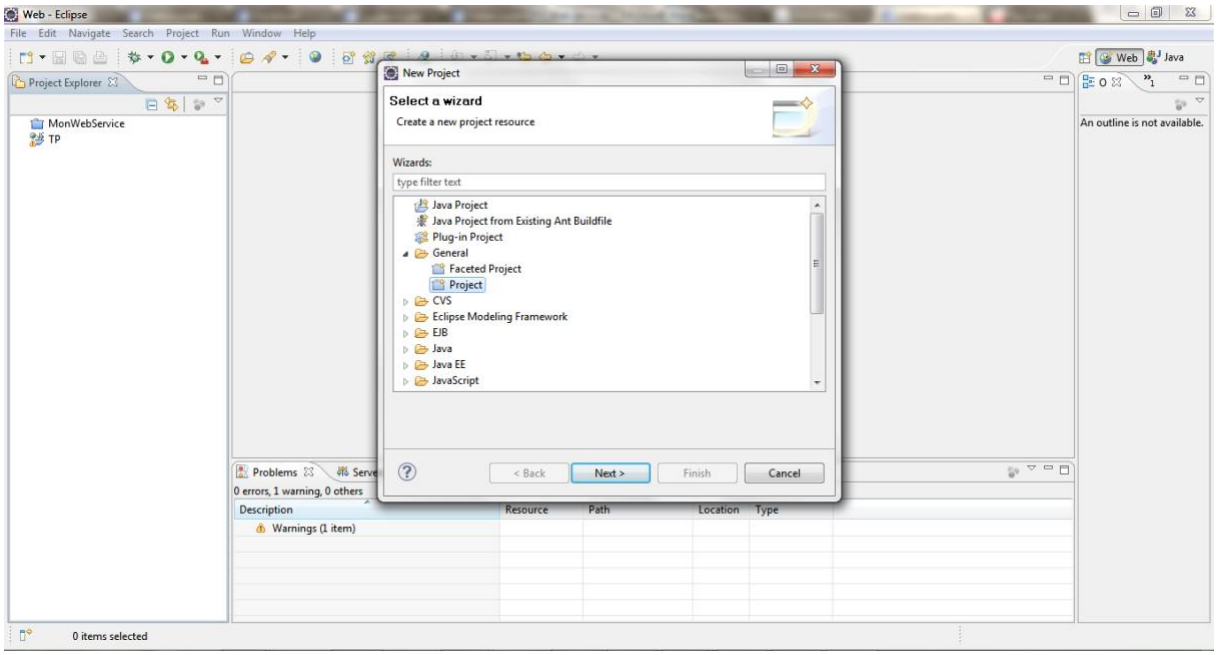
Nous allons créer un nouveau projet que nous appellerons MonWebService :

Ouvrez Eclipse

Choisissez : Menu : File-New-Project



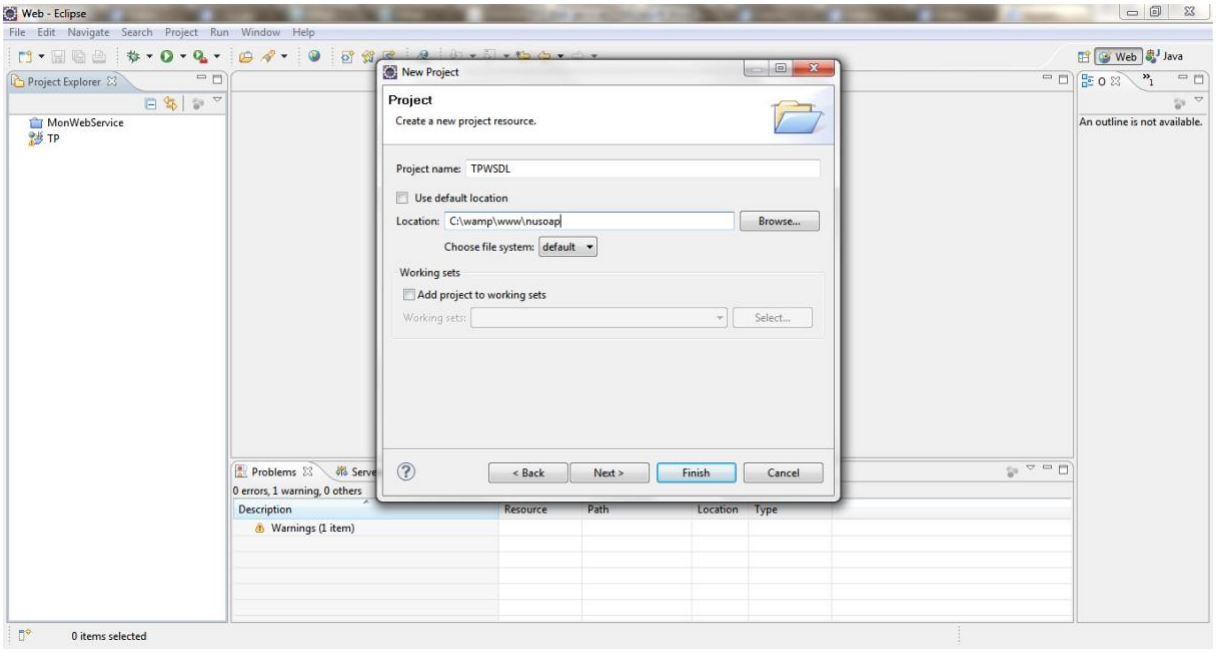
Choisissez : General-Project et cliquez sur Next



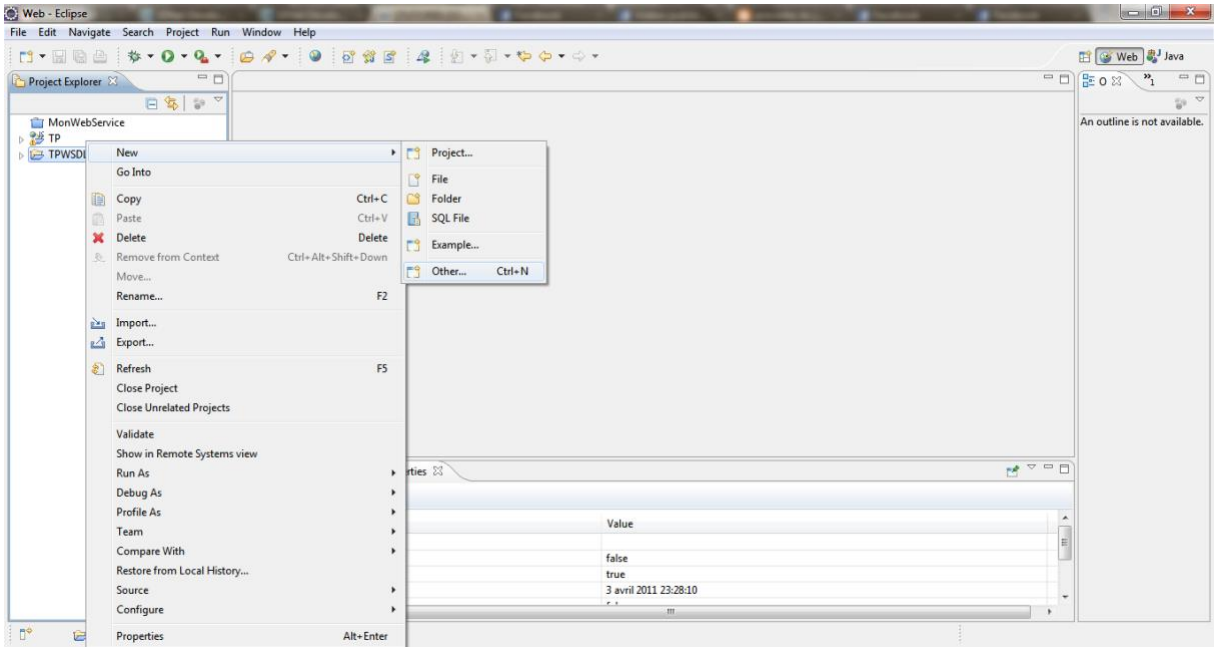
Donnez un nom avec projet : Project Name : TPWS

Décochez la case : Use Default Location

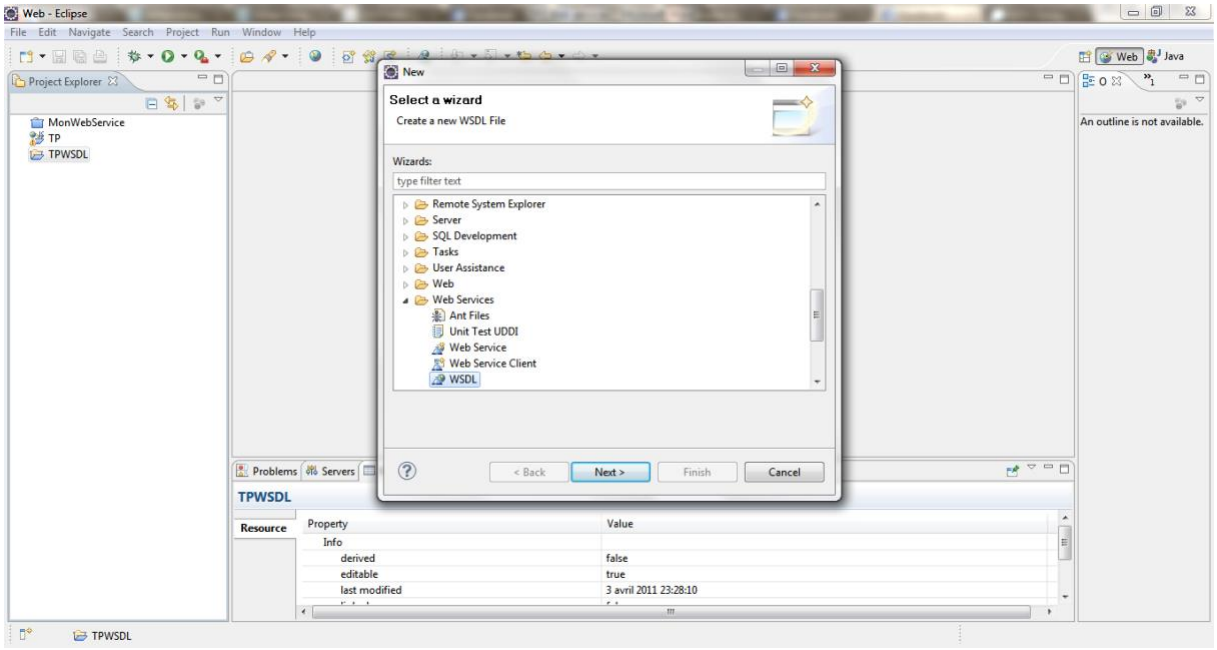
Enregistrez le projet dans le dossier nusoap : C:\wamp\www\nusoap et ensuite cliquez sur Finish. Vérifiez qu'un projet à été créé (il apparait dans le volet Project explorer de votre fenêtre Eclipse).



Maintenant que le projet est crée, on va commencer par faire le fichier WSDL de notre web service qu'on nommera HellYou.wsdl. Pour nous aider, Eclipse nous construit automatiquement un format wsdl par défaut, nous allons le reprendre et le paramétrer. Pour ce faire sélectionnez le projet TPWSDL du volet project explorer, et sélectionnez du menu contextuel (bouton droit de la souris) New-Other :

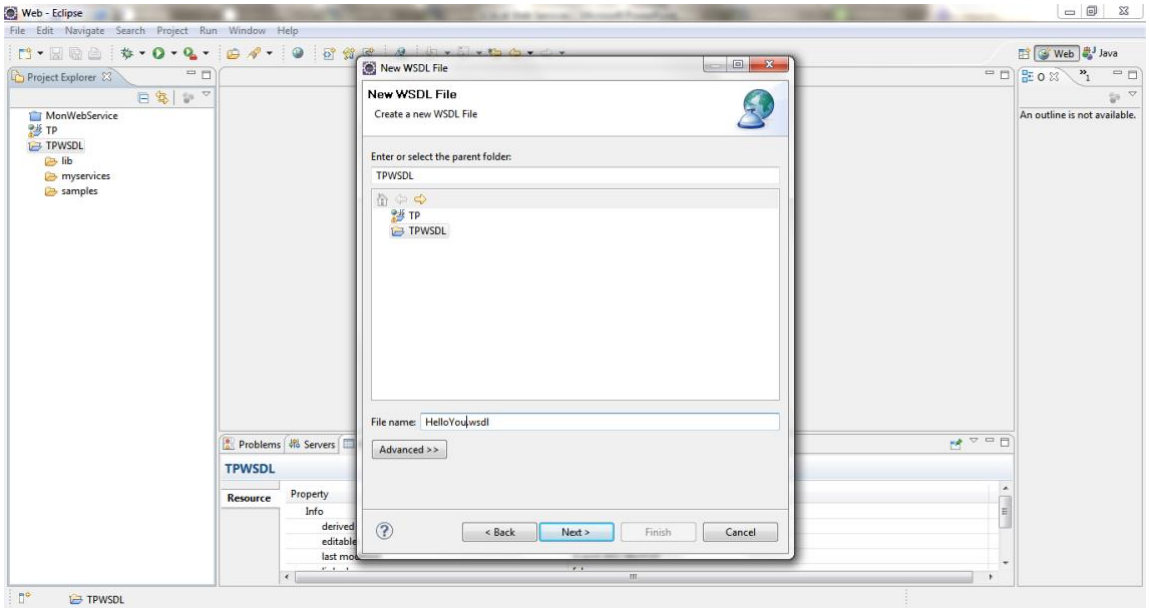


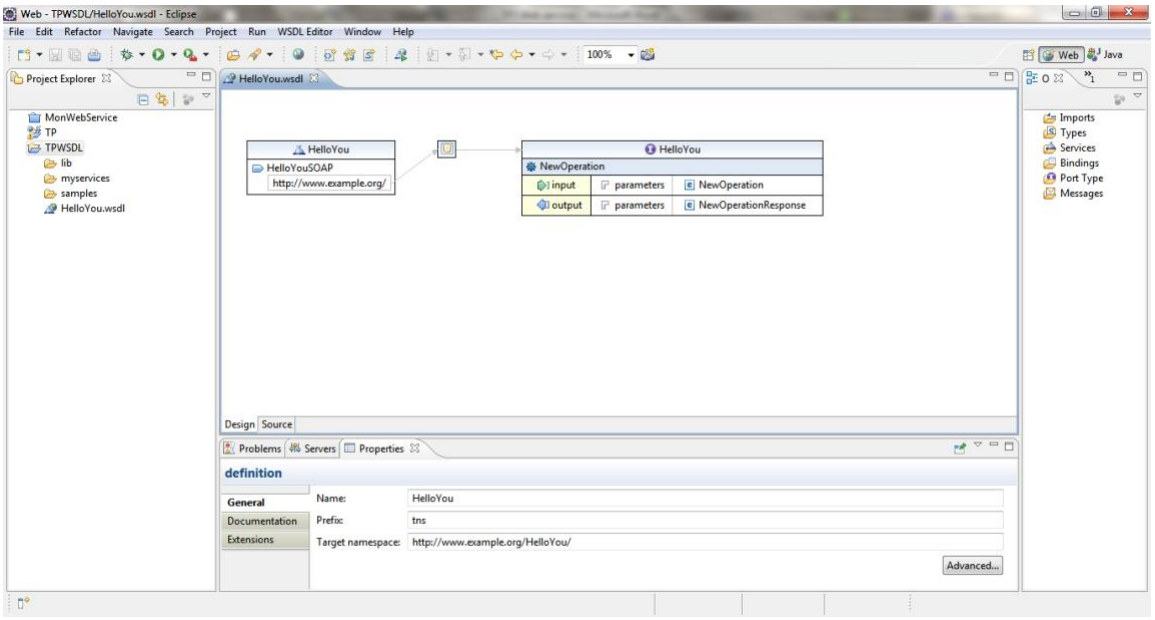
Un Wizard apparait, sélectionnez Web service→WSDL puis cliquez Next.



Donnez un nom à votre fichier WSDL : HelloYou.wsdl puis cliquez Next.

Observez l'interface qui suit du Wizard : il défini pour vous un espace de nommage par défaut, un préfixe, le protocole de transport, et le type. Cliquez sur Finish. Le fichier HelloYou.WSDL apparait dans le projet TPWSDL.

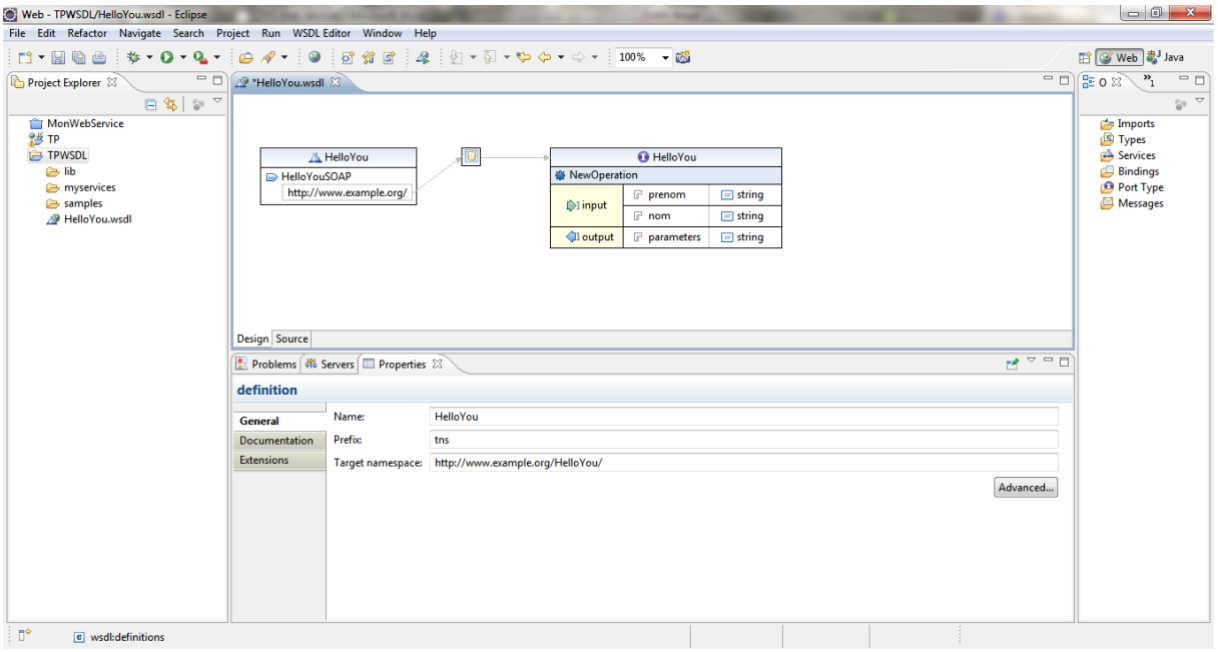




Modifiez le fichier HelloYou.wsdl : pour avoir 2 input : prenom et nom et un seul output. Tous de type String :

Cliquez sur input avec le bouton droit de la souris : sélectionnez **add part** pour ajouter un nouveau paramètre.

Pour définir le type, cliquez **set Type →existing Type →string** pour définir le type du paramètre.

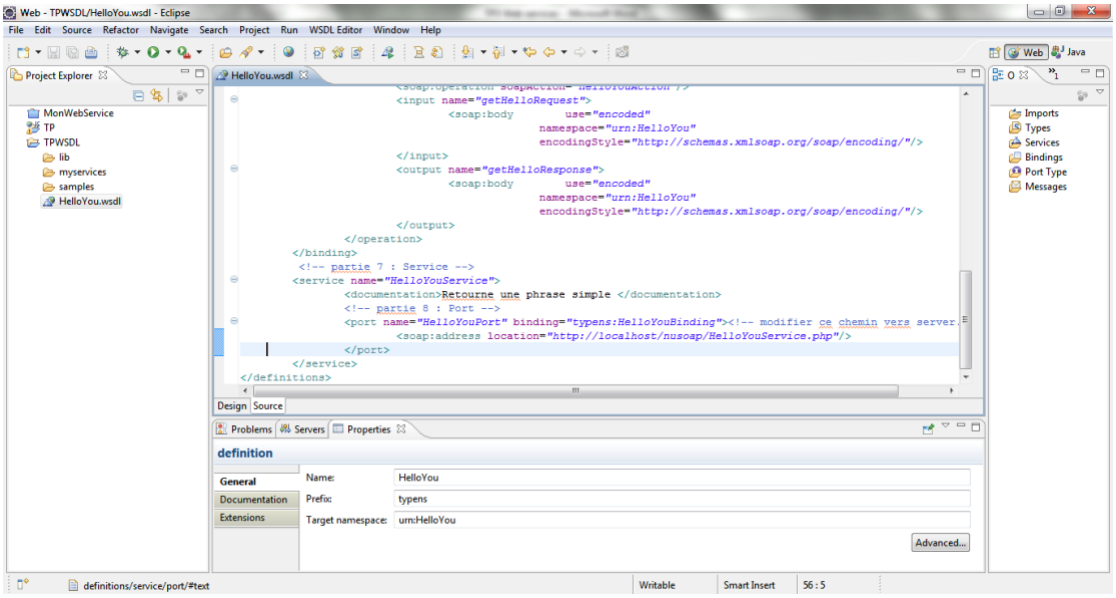


Voici le fichier HelloYou.WSDL complet fourni pour vous :

```
<?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>
        <!-- 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>
```

Cliquez sur l'onglet : source et coller le code source du fichier HelloYou.wsdl :



Ouvrez notepad++ et créez un script HelloYou.php :

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

?>
```

Créer le web service : HelloYouService.php :

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

// Inclusion du script contenant l'implémentation des fonctions du Service Web
include('HelloYou.php');
 // on indique au serveur à quel fichier de description il est lié
$serveurSOAP = new SoapServer('HelloYou.wsdl');
 // ajouter la fonction getHello au serveur
$serveurSOAP->addFunction('getHello');
 // lancer le serveur
// Appel du Service Web (requête POST uniquement autorisée)
if ($_SERVER['REQUEST_METHOD'] == 'POST')
 {
// Prise en charge de la requête si le service à été appelé par un client
	$serveurSOAP->handle();
}
// Sinon, on affiche un lien vers le WSDL
else {
echo 'Web Service HelloYou.<br />';
echo '<a href="http://localhost/nusoap/HelloYou.wsdl">WSDL</a><br />';
}
?>
```

Créer le web service : HelloYouclient.php :

```
<?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 methode 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 />Requete SOAP : '.htmlspecialchars($clientSOAP->__getLastRequest()).'<br />';
```

```
echo '<br />Reponse SOAP : '.htmlspecialchars($clientSOAP->__getLastResponse()).'<br />';
 ?>
```

Enregistrer HelloYou.php, HelloYouService.php et HelloYouClient.php dans Wamp/WWW/nusoap/ . Vérifiez que le fichier HelloYou.wsdl y est aussi.

Accédez à votre Web service via: http://localhost/nusoap/HelloYouService.php

Invoquez le Web service via : http://localhost/nusoap/HelloYouClient.php