

GUIDE

CRÉER UN WEB SERVICE (SOAP)

Contenu

1. Création du Web Service	1
2. Déploiement	6
3. Exploitation avec un client PHP	10

L'exemple que nous allons prendre est volontairement très simpliste et n'a du coup pas le moindre intérêt comme service web. Mais le but ici est juste de comprendre la démarche technique pour créer un Web Service.

L'exemple que nous allons créer permet juste d'offrir une fonction qui fait la somme de deux valeurs entières. Difficile de faire plus simple. Cependant, vous allez apprendre ainsi à créer ce service, le déployer et l'exploiter avec deux clients différents : un écrit en jsp et un écrit en PHP. Cela vous permettra de mieux comprendre qu'un service peut être exploité par différents clients.

1. Création du Web Service

Pour la création de ce Web Service, nous allons rester sur l'IDE NetBeans couplé au serveur GlassFish. Cependant, rappelez-vous qu'un web service peut être créé dans n'importe quel langage.

1A. Création du projet

Lancez NetBeans.

De la façon dont NetBeans a été configuré dans le guide précédent, il s'occupe d'initialiser le serveur GlassFish et de le démarrer lorsque cela est nécessaire.

Dans le menu "File", sélectionnez "New Project...". Dans la fenêtre, prenez "Java Web" dans "Java with Ant" et "Web Application". Faites Next. Dans la fenêtre suivante, donnez au projet le nom "Calcul" et choisissez l'emplacement que vous voulez. Cliquez sur Next puis Finish. Le projet va se créer.

Si le projet précédent (QuizzPays) est toujours ouvert, vous pouvez le fermer en faisant un clic droit sur le nom du projet et "close". Il va disparaître de l'IDE (sans bien sûr disparaître du disque). Normalement vous ne devriez maintenant voir que votre nouveau projet "Calcul".

1B. Création du Web Service

À gauche, faites un clic droit sur le nom du projet et sélectionnez "New", puis "Web Service...". Attention, ne vous trompez pas et ne prenez pas, comme dans le TP précédent "Web Service Client...". Vous obtenez cette fenêtre :

New Web Service

Steps

1. Choose File Type
2. **Name and Location**

Name and Location

Web Service Name:

Project:

Location:

Package:

☒ Create Web Service from Scratch

☐ Create Web Service from Existing Session Bean

Enterprise Bean:

☐ Implement Web Service as Stateless Session Bean

< Back Next > **Finish** Cancel Help

Comme vous pouvez le voir dans la capture, donnez le nom "CalculWS" au Web Service et donnez le nom "webservice" au package qui le contiendra. Cliquez sur "Finish".

Un code a été généré par défaut, le fichier portant le nom de "CalculWS.java" :

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package webservice;

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

/**
 *
 * @author Emds
 */
@WebService(serviceName = "CalculWS")
public class CalculWS {

    /**
     * This is a sample web service operation
     */
    @WebMethod(operationName = "hello")
    public String hello(@WebParam(name = "name") String txt) {
        return "Hello " + txt + " !";
    }
}

```

Vous remarquez que vous êtes bien dans le package webservice. Plusieurs imports nécessaires à la création d'un Web Service sont déjà insérés. La classe principale CalculWS est créée ainsi qu'une première méthode exemple qui attend un paramètre de type String et qui retourne un petit texte construit avec le contenu de ce paramètre. Analysons un peu le code. Vous remarquez avant tout les annotations :

```
@WebService(serviceName = "CalculWS")
```

Cette annotation précède la classe pour spécifier le nom du Web Service associé.

```
@WebMethod(operationName = "hello")
```

Cette annotation précède la méthode pour spécifier son nom qui sera visible de l'extérieur.

```
public String hello(@WebParam(name = "name") String txt) {
```

Pour finir avec les annotations : il y en a aussi au niveau des paramètres,. Donc là, on voit que le paramètre txt de type String, portera le nom "name" à l'extérieur du Web Service, quand la méthode hello sera évoquée de l'extérieur. Ces annotations vont permettre à l'IDE de générer automatiquement le fichier WSDL.

Ceci dit, en réalité, les annotations peuvent être simplifiées si on ne souhaite pas donner des noms précis au web service, aux méthodes et paramètres. Sans ces précisions, l'IDE donne des noms par défaut : le web service porte le nom de la classe suivi de "Service", la méthode garde le même nom, les paramètres accessibles de l'extérieur portent les nom arg0, arg1 etc... Ceci sera appliqué si vous mettez le code suivant :

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package webservice;

import javax.ws.WebService;
import javax.ws.WebMethod;
import javax.ws.WebParam;

/**
 *
 * @author emds
 */
@WebService
public class CalculWS {

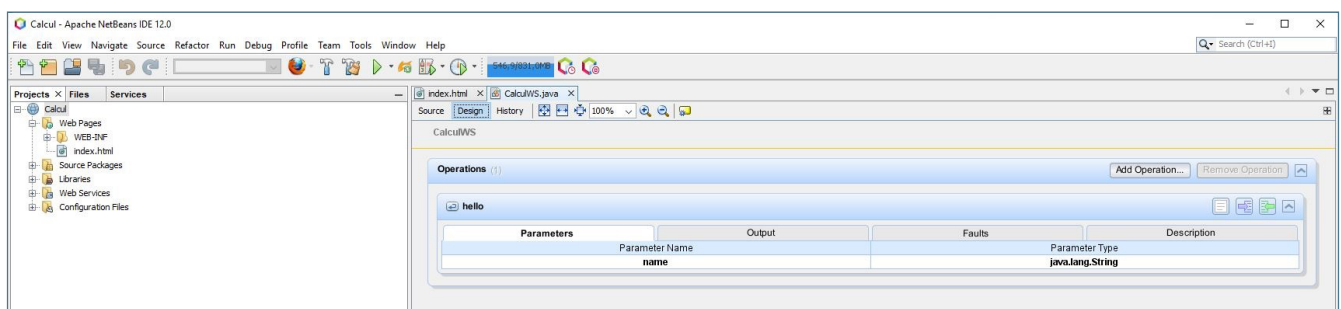
    /**
     * This is a sample web service operation
     */
    @WebMethod
    public String hello(String txt) {
        return "Hello " + txt + " !";
    }
}

```

1C. Création d'une méthode

La méthode exemple est bien pratique pour voir la syntaxe à respecter, mais bien sûr elle ne va pas nous servir. Cependant, avant de la supprimer, nous allons créer notre méthode d'addition. Il est évidemment possible d'écrire le code directement. Cependant, pour poursuivre la découverte de l'IDE NetBeans, voici une autre technique plus visuelle :

Juste au-dessus du code du fichier "CalculWS.java", et en dessous du nom du fichier, remarquez les boutons "Source" et "Design". Pour le moment c'est "Source" qui est sélectionné. Cliquez sur "Design". Vous obtenez ceci :



Vous retrouvez, en visuel, la méthode "hello", avec le nom et le type du paramètre, ainsi que le type retourné (si vous allez sur l'onglet Output).

Pour ajouter une nouvelle méthode, cliquez sur "Add operation..." qui se trouve en haut à droite. Vous obtenez cette fenêtre :

Add Operation...

Name:

Return Type:

Parameters Exceptions

Name	Type	Final
------	------	-------

Vous comprenez le principe : il suffit de donner le nom de la méthode (appelez-la "somme"), de préciser le type retourné (tapez "int" dans la partie "Return Type") et d'ajouter les paramètres nécessaires en utilisant le bouton "Add". Ajoutez les paramètres val1 et val2 de type int (en tapant le nom du paramètre et en sélectionnant à chaque fois le type dans la liste). Au final, vous devriez obtenir la fenêtre suivante :

Add Operation...

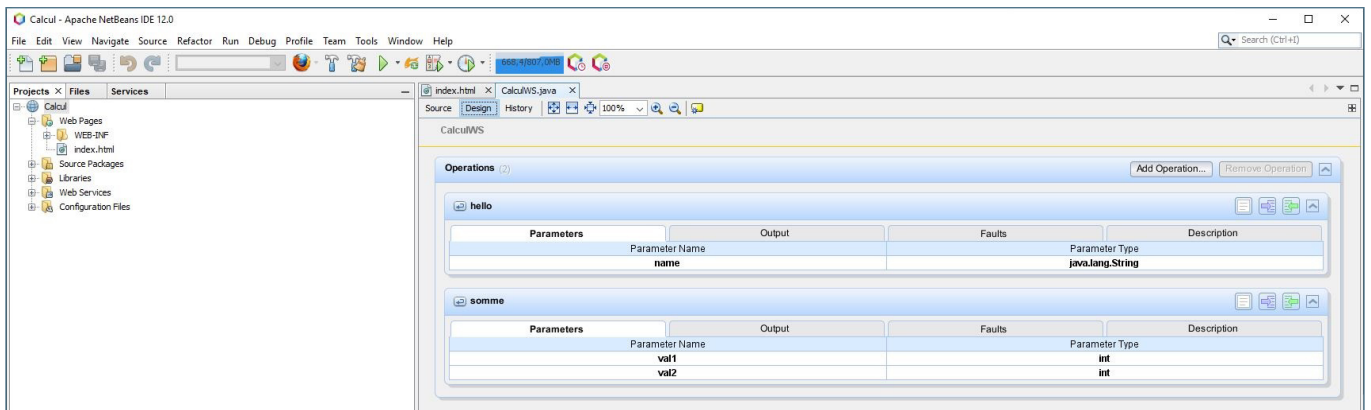
Name:

Return Type:

Parameters Exceptions

Name	Type	Final
val1	int	<input type="checkbox"/>
val2	int	<input type="checkbox"/>

Cliquez sur OK. Cette fois, toujours dans la partie Design, les deux méthodes sont affichées :



La méthode "hello" ne va pas nous servir : vous pouvez la supprimer aussi dans ce mode Design, en faisant un clic droit sur le nom "hello" et en choisissant "Remove Operation" puis en cliquant sur Yes.

Revenez le source du code (en cliquant sur "Source"). Vous remarquerez que la méthode somme a été générée. Les annotations nécessaires et son entête sont créées, avec les deux paramètres. Il ne reste plus qu'à écrire le contenu de la méthode. Vu le but de cette méthode, ça ne va pas être très difficile : il faut retourner la somme des deux paramètres. Donc remplacez le "return 0" par :

```
return val1 + val2 ;
```

Tout ceci a été fait pour vous montrer les possibilités de l'IDE et l'utilisation poussée des annotations. Mais rappelez-vous qu'en réalité, si vous voulez garder les mêmes noms visibles en externe, vous n'avez pas besoin d'autant de détail et votre méthode peut très bien s'écrire juste ainsi :

```
@WebMethod
public int somme(int val1, int val2) {
    return val1+val2;
}
```

Enregistrez.

2. Déploiement

2A. Génération

Pour compiler le projet, allez dans le menu "Run" et choisissez "Clean and Build Project". Dans la fenêtre du bas, vous pouvez suivre les étapes de génération des fichiers nécessaires pour la gestion du projet. Si tout s'est bien passé, vous obtenez au final un message en vert :

```
BUILD SUCCESSFUL
```

Si vous ne voyez pas la fenêtre d'output du bas, affichez-là avec le menu "Window > Output".

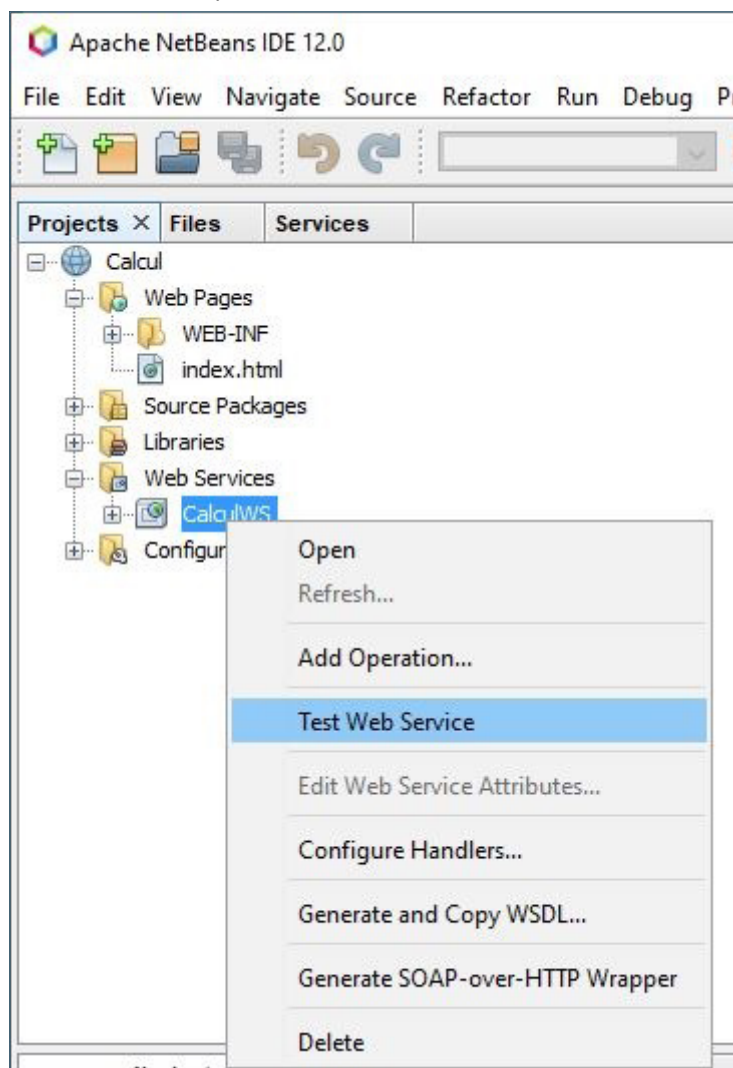
2B. Déploiement

Une fois le projet correctement généré, il faut le déployer pour qu'il soit mis à disposition. Attention, ce déploiement va se faire ici, en local, sur le serveur GlassFish qui est installé sur votre ordinateur. Donc pour le moment, seules les applications locales peuvent y accéder. Vous constaterez que l'adresse du

fichier wsdl est locale. Pour une mise en disposition sur internet, il faudrait gérer un déploiement sur un serveur avec une adresse publique. Pour réaliser le déploiement, faites un clic droit sur le nom du projet "Calcul" (à gauche) et demandez "Deploy". Attendez la fin du déploiement (barre de défilement en bas). Le serveur GlassFish se lance et est prêt à recevoir les requêtes pour faire appel au Web Service.

2C. Test

Sous NetBeans, il est possible de tester le Web Service, avant même de l'exploiter avec un client. Déployez le projet, à gauche, et, dans la partie "Web Services", faites un clic droit sur "CalculWS".



Sélectionnez "Test Web Service". Le navigateur s'ouvre et vous obtenez la fenêtre de test suivante :

CalculWS Web Service Tester

This form will allow you to test your web service implementation ([WSDL File](#))

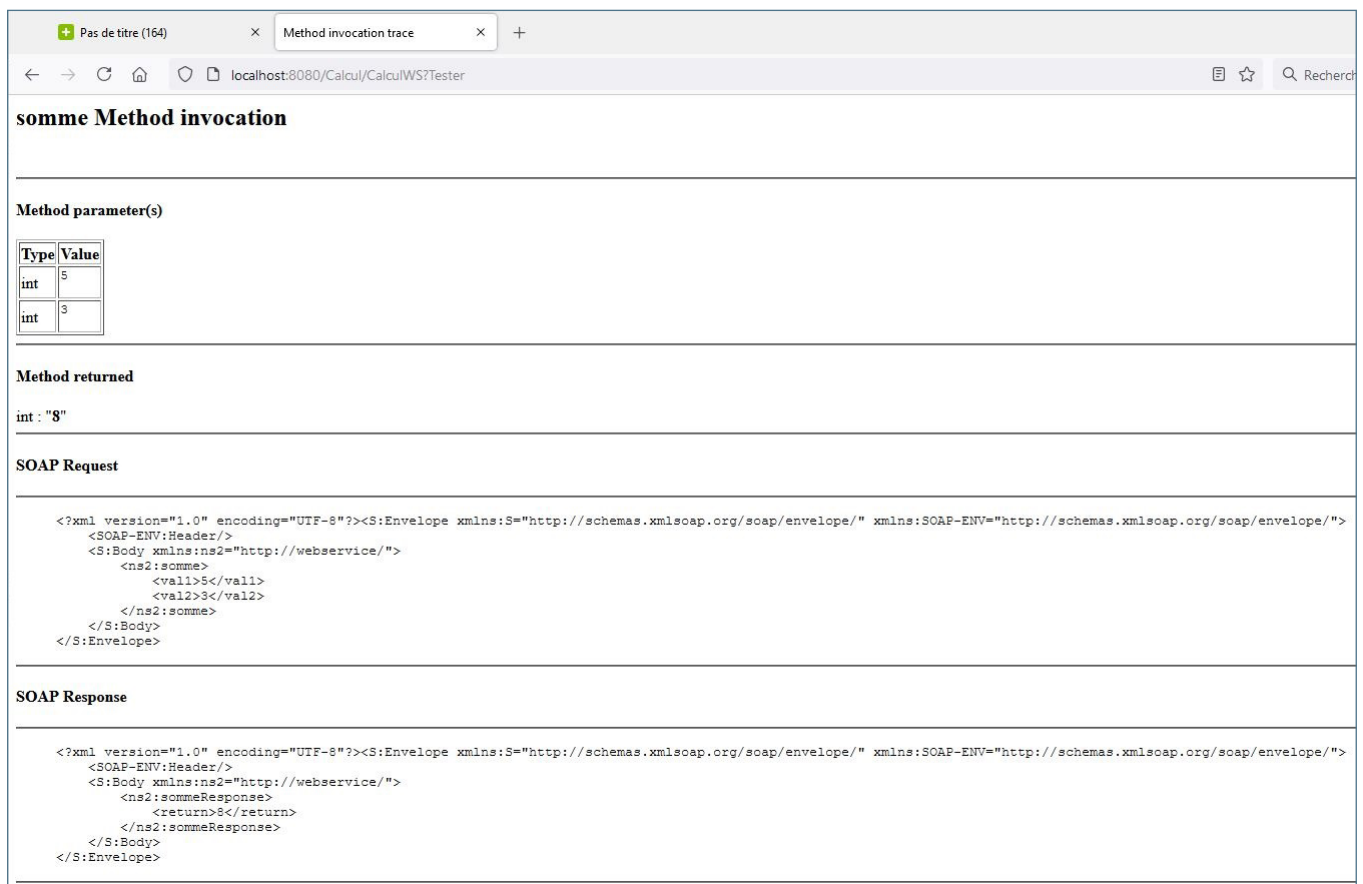
To invoke an operation, fill the method parameter(s) input boxes and click on the button labeled with the method name.

Methods :

```
public abstract int webservice.CalculWS.somme(int,int)
```

somme (,)

Vous retrouvez la liste des méthodes du Web Service (ici, une seule méthode : somme) avec la possibilité de saisir des valeurs en paramètre de de tester le retour en cliquant sur le bouton correspondant au nom de la méthode. Faites un test. Vous obtenez une nouvelle page avec le résultat mais beaucoup d'autres informations :



somme Method invocation

Method parameter(s)

Type	Value
int	5
int	3

Method returned

int : "8"

SOAP Request

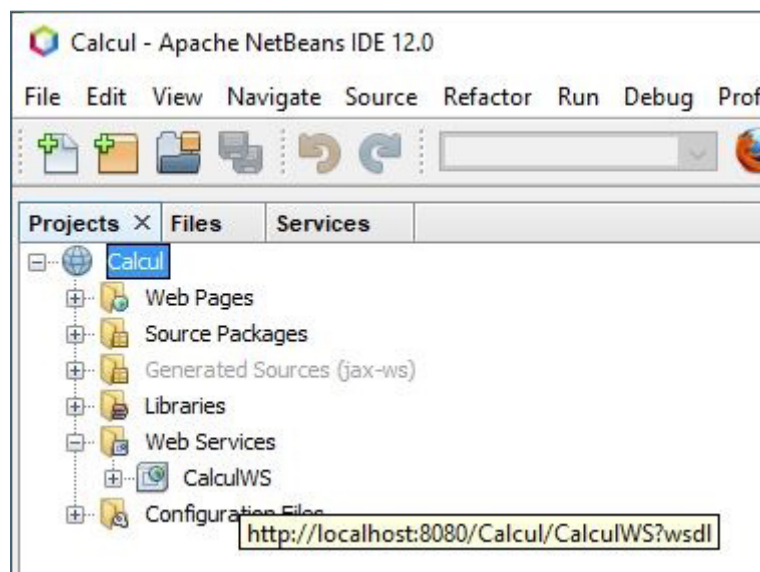
```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <S:Body xmlns:ns2="http://webservice/">
    <ns2:somme>
      <val1>5</val1>
      <val2>3</val2>
    </ns2:somme>
  </S:Body>
</S:Envelope>
```

SOAP Response

```
<?xml version="1.0" encoding="UTF-8"?><S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header/>
  <S:Body xmlns:ns2="http://webservice/">
    <ns2:sommeResponse>
      <return>8</return>
    </ns2:sommeResponse>
  </S:Body>
</S:Envelope>
```

2D. Fichier XSDL

Suite à la génération du Web Service, un fichier XSDL a été créé (en particulier grâce aux annotations dans le code). C'est lui qui est public et qui donne toutes les informations nécessaires à l'utilisation du Web Service. NetBeans s'occupe de générer automatiquement le fichier WSDL. Pour connaître l'adresse de publication du fichier wsdl, placez la souris sur le nom du web service et attendez l'affichage du chemin.



Attention, si vous avez simplifié les annotations, vous avez peut-être une adresse différente car normalement le mot "Service" est automatiquement ajouté à la suite du nom du web service.

Pour voir son contenu, ouvrez un navigateur et tapez l'adresse du fichier wsdl.

Vous remarquez que l'adresse est bien en local. Cependant, c'est avec cette adresse que vous allez travailler pour faire vos tests en créant des clients qui font appel à ce Web Service.

2E. Documentation

Il est possible de générer automatiquement une documentation à partir du code écrit. Cette documentation se construit à l'aide des commentaires et annotations contenus dans le code.

Dans NetBeans, sélectionnez votre projet "Calcul" à gauche. Dans le menu "Run", cliquez sur "Generate Javadoc". Le navigateur va s'ouvrir présentant la javadoc créée. Dans notre exemple, elle ne contient qu'un package qui lui-même ne contient qu'une classe. Si vous cliquez sur le nom de la classe, vous obtenez le détail de son contenu (uniquement la partie visible, bien sûr) :

CalculWS

file:///Go:/test/testWebService/Calcul/dist/javadoc/index.html

Rechercher

All Classes

CalcuWS

PACKAGE CLASS USE TREE DEPRECATED INDEX HELP

PREV CLASS NEXT CLASS FRAMES NO FRAMES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

webservice

Class CalculWS

java.lang.Object
webservice.CalculWS

public class **CalculWS**
extends java.lang.Object

Constructor Summary

Constructors

Constructor and Description

CalculWS()

Method Summary

All Methods **Instance Methods** **Concrete Methods**

Modifier and Type	Method and Description
int	somme (int val1, int val2) Web service operation

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

CalculWS

public CalculWS()

Method Detail

2F. Exploitation avec un client JSP

Maintenant que le Web Service est créé, il va être possible de l'exploiter, exactement comme vous l'avez fait dans le guide précédent, avec un client en JSP.

Petit calcul :

12 + 15 = 27

+ =

Tout a été vu dans le guide précédent. Donc, en utilisant vos connaissances, créez une application JSP sous NetBeans pour utiliser ce nouveau Web Service. Inutile de faire une interface compliquée : faites deux zones de saisie et en cliquant sur le bouton de soumission (submit) affichez le calcul avec le résultat.

Normalement vous ne devriez avoir aucune difficulté à réaliser cette petite application. Si toutefois vous êtes bloqué, reprenez les étapes de création d'une application JSP faisant appel à un Web Service, décrites dans le guide précédent.

3. Exploitation avec un client PHP

Pour vous convaincre que n'importe quel client peut exploiter ce Web Service, nous allons faire un test en PHP.

3A. Serveur

Lancez votre serveur wamp. Si wamp n'est pas installé sur votre ordinateur, téléchargez sur internet wampserver (<http://www.wampserver.com/>) et installez-le. Vous n'êtes cependant pas obligé d'utiliser wamp : si vous êtes habitué à un autre serveur web pour interpréter vos pages PHP, ce n'est pas un problème. En revanche, je pars du principe pour la suite de ce guide, que vous avez des connaissances de base en PHP et wamp (ou équivalent).

Lorsque vous allez lancer le serveur wamp (ou un autre), il est possible que vous ayez un souci et que le serveur refuse de se lancer correctement. Si c'est le cas, cela signifie qu'il y a un conflit au niveau de l'utilisation des ports et qu'un autre serveur utilise le même port que le serveur wamp. Ce problème est généralement facile à résoudre : il suffit de dire à wamp d'écouter sur un autre port : cliquez sur l'icône de wamp, allez dans "apache" et ouvrez le fichier "httpd.conf". Faites une recherche de la ligne "Listen". Attention, pas la ligne qui commence par un # car c'est un commentaire, mais la ligne qui commence directement avec "Listen". Le mot "Listen" est suivi du numéro de port d'écoute. Il suffit de changer ce port, d'enregistrer le fichier et de relancer wamp. En revanche, si le port n'est plus 80 (port par défaut), lors de vos tests dans le navigateur, il faudra préciser le nouveau port de cette façon (en remplaçant 80 par le nouveau numéro) :

`localhost:80`

Il suffit donc de préciser le numéro de port à la suite du localhost. Normalement GlassFish se met sur le port 8080, donc a priori si wamp est sur le port 80, cela ne devrait pas poser de problème. Mais il est toujours bon de connaître la manipulation.

3B. Configuration SOAP

Vous devez aussi ajouter une extension au PHP pour qu'il puisse utiliser des fonctions spécifiques à SOAP pour l'accès au Web Service. Cliquez sur l'icône de wamp, allez dans "PHP" et dans "PHP extensions". Si ce n'est pas déjà le cas, sélectionnez "soap" et relancez wamp.

3C. Création d'une page PHP

Vous allez maintenant pouvoir créer une simple page qui va exploiter le Web Service. Soit vous allez dans le dossier www de wamp et vous créez un dossier pour votre petit projet, soit vous créez un alias pour accéder à un dossier qui se situe ailleurs : faites comme vous avez l'habitude.

Dans le dossier du projet, créez un fichier index.php.

Le but est d'obtenir la même présentation que précédemment avec le client JSP. Donc mettez le titre et un petit formulaire en post pour saisir 2 valeurs (zones de saisie portant respectivement les noms txtVal1 et txtVal2), sans oublier le bouton submit. Le formulaire doit rappeler la même page.

Entre le titre et le formulaire, insérez les balises PHP. Faites un test sur l'existence de la variable post du nom "txtVal1". Si la variable existe, c'est que le bouton submit a été utilisé donc il faut calculer. Voici les étapes :

Commencez par accéder au Web Service et plus précisément à faire appel au fichier wsdl (pensez à mettre la bonne adresse dans les parenthèses) :

```
$service = new SoapClient("http://localhost:8080/Calcul/CalculWS?wsdl");
```

Le but ensuite, est de récupérer les 2 variables envoyées en post, car elles vont servir de paramètres à l'appel de la fonction somme du Web Service. Mais vu comment est constitué le fichier wsdl, un seul paramètre est attendu, combinant les 2 paramètres. Du coup, il faut créer un tableau associatif pour enregistrer les 2 paramètres, en utilisant bien les noms des paramètres (soit val1 et val2 si vous les avez renommé, soit arg0 et arg1).. Sachant en plus qu'ils sont numériques, voici la syntaxe pour récupérer les variables post et correctement initialiser les paramètres :

```
// récupération des valeurs post
$val1 = intval($_POST["txtVal1"]) ;
$val2 = intval($_POST["txtVal2"]) ;
// création du tableau de paramètres
$params = array(
    'arg0' => $val1,
    'arg1' => $val2
);
```

La fonction intval permet juste la conversion d'une chaîne en entier. Maintenant il est possible d'appeler la fonction somme, en utilisant l'objet \$service qui a été créé, et en lui envoyant le tableau \$params en paramètre :

```
// appel de la méthode du web service
$result = $service->somme($params) ;
```

Il ne reste plus qu'à gérer l'affichage, sachant que le résultat sera obtenu dans la propriété return de l'objet \$result.

```
// affichage du résultat
echo "$val1 + $val2 = ".$result->return
```

Vous pouvez tester votre page : normalement elle doit fonctionner de la même façon que la page créée en JSP.

En réalité, si le fichier wsdl avait été généré un peu différemment, en séparant les paramètres lors de la description de la méthode, l'appel de la fonction serait plus direct, sans passer par un tableau. Vous aurez l'occasion de voir cette possibilité quand vous allez vous-même créer un fichier wsdl.



Les cours du CNED sont strictement réservés à l'usage privé de leurs destinataires et ne sont pas destinés à une utilisation collective. Les personnes qui s'en serviraient pour d'autres usages, qui en feraient une reproduction intégrale ou partielle, une traduction sans le consentement du CNED, s'exposeraient à des poursuites judiciaires et aux sanctions pénales prévues par le Code de la propriété intellectuelle. Les reproductions par reprographie de livres et de périodiques protégés contenues dans cet ouvrage sont effectuées par le CNED avec l'autorisation du Centre français d'exploitation du droit de copie (20, rue des Grands-Augustins, 75006 Paris).

CNED, BP 60200, 86980 Futuroscope Chasseneuil Cedex, France

© CNED 2021

87D22TDWB1F21

