Module 3

Développement de Services Web SOAP avec PHP

Contenu du module

- Principes généraux de développement de Services Web en PHP.
  - Organisation du code et considération sur les fonctions/méthodes.
- Présentation des différentes bibliothèques de développement de Services Web PHP.
  - L'extension SOAP de PHP
  - Les bibliothèques alternatives telles que NuSOAP
- L'extension SOAP de PHP
  - Activation de l'extension SOAP de PHP : options du fichier php.ini
  - Présentation des principales classes (SoapServer, SoapClient, SoapFault)
- Bonnes pratiques d'écriture d'un service et de son client
  - La notion de Facade
  - Règles d'implémentation des classes

Concepts généraux

- Le développement de services Web en PHP permet d'exposer des fonctions ou des méthodes de classes sous forme de services SOAP
- Plusieurs librairies sont utilisables pour ce faire :
  - L'extension SOAP de PHP
    - Pas de génération de WSDL
    - Utilisable en mode RPC/Encoded uniquement
  - NuSOAP
    - Bibliothèque développée par NuSphere
    - Une référence !
    - Génération de WSDL
    - Adaptatif pour ce qui du style et de l'encodage !
  - Tout framework PHP permettant d'utiliser les technologies SOAP et WSDL
    - Symfony
    - Laravel
    - ...

L'extension SOAP de PHP

- L'extension SOAP de PHP est disponible dans le langage depuis PHP 5
  - Elle doit être activée via le fichier php.ini
- Cette extension se base sur l'usage de 3 classes essentielles :
  - SoapServer
    - Pour créer et déclarer un service.
  - SoapClient
    - Pour créer et déclarer un client.
  - SoapFault
    - Pour la représentation des erreurs.
- WSDL ?
  - PHP interprète de façon transparente les fichiers WSDL lors de la déclaration d'un client à un Service WEB, et gère lui-même le formatage SOAP des messages envoyés au serveur : formatage XML, respect des règles du WSDL...
  - Le serveur SOAP de PHP propose les mêmes facilités que pour le client, hormis pour la description WSDL qui doit être réalisée par un autre biais.
    - Dans ce cas, le serveur « sert » le Service WEB selon les spécificités décrites dans le WSDL.

Création d'un service

- L'instanciation de la classe `SoapServer` va permettre de créer le service.
  - Elle prend en premier paramètre le nom et l'emplacement du fichier WSDL à utiliser.
  - Ou `null` si aucun WSDL n'est disponible !
- On ajoute ensuite les méthodes qui doivent faire partie du service avec `addFunction()`
- Si les fonctionnalités à exposer sont définies dans une classe, on pourra exposer cette classe avec la méthode `setClass()`.

```php
<?php

// Désactiver le cache lors de la phase de test
ini_set("soap.wsdl_cache_enabled", "0");

// On indique le fichier WSDL
$serveurSOAP = new SoapServer('Hello.wsdl');

// ajouter la fonction getHello au serveur
$serveurSOAP->addFunction('getHello');

// lancer le serveur
if ($_SERVER['REQUEST_METHOD'] == 'POST')

{
    $serveurSOAP->handle();
}
else
{
    echo 'Erreur de requête !';
}

function getHello($prenom, $nom)
{
    return 'Hello ' . $prenom . ' ' . $nom;
}

?>
```

Création d'un client

- L'instanciation de la classe `SoapClient` permet la création d'un client de service Web.
  - Le constructeur prend en paramètre le nom et l'emplacement du fichier WSDL.
  - Ou `null` s'il n'y en a pas !
- Il suffit ensuite d'invoquer les méthodes en transmettant les paramètres appropriés.

```php
<?php

// 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('Hello.wsdl');

// Executer la methode getHello
echo $clientSOAP->getHello('Marc','DUPUIS');

?>
```

Conclusion sur l'extension SOAP de PHP

- Cette extension comprend plusieurs lacunes qui empêchent de l'utiliser concrètement aujourd'hui !
  - Pas de génération de WDSL
  - Limité à RPC/Encoded
- Cependant, pour des fonctionnalités d'intégration inter-applications PHP, elle permet une mise en œuvre rapide.
  - On développe le service et le client.
  - Pas besoin de WSDL.
  - L'interopérabilité n'est pas un problème.

La bibliothèque NuSoap

- Développée par NuSphere et Dietrich Ayala.
- Sous licence GPL

- Elle possède de nombreux avantages :
  - Écrit en PHP (pas de nouveaux modules à installer ou à configurer).
  - Interface simple orientée objet.
  - Peut fonctionner avec ou sans fichiers WSDL.
  - Peut générer automatiquement un fichier WSDL pour le service.

Installation de NuSoap

- Télécharger la librairie à partir de http://sourceforge.net/projects/nusoap/
- Décompressez le fichier et copiez le répertoire `lib/` dans votre répertoire de Projet PHP
- Importez les classes nécessaires (client et serveur) :
  - `require_once ('lib/nusoap.php');`

- NuSoap est également disponible sur **Packagist**, elle est donc installable via **Composer**.
  - `composer require econea/nusoap`

Un serveur (service !) NuSoap minimal

```php
<?php

require_once('lib/nusoap.php') ;

// Instancier le serveur
$server = new nusoap_server();
// Ajouter une fonction au service
$server->register('myFunction') ;

function myFunction($parameters) {
    . . .
    return $result;
}

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

- Dans ce cas, on utilise pas de document WSDL.

Un client NuSoap minimal

```php
<?php

require_once('lib/nusoap.php');

// Créer l'objet client
$client = new nusoap_client('http://localhost/ws.php', false); // false: Pas de WSDL !

// Appeler l'opération SOAP
$result = $client->call(
    'myFunction',
    array('param' => 'abc')
);

?>
```

- Instanciation de la classe `nusoap_client`
  - Le premier paramètre est l'URL du point de terminaison du service Web ou l'URL du document WSDL.
  - Le second paramètre est un booléen qui indique si on utilise WSDL ou non.
- Appel d'opération SOAP
  - Le premier paramètre est le nom de l'opération (fonction) appelée.
  - Le second paramètre est un tableau avec la liste des entrées SOAP.
    - nom de paramètre => valeur de paramètre

NuSoap et WSDL

- WSDL doit décrire le service Web, et en particulier le nom et le type de paramètres d'entrée et de sortie.
  - Dans certains cas, un fichier de description WSDL existe déjà.
- Dans d'autres cas, le service peut fournir automatiquement un fichier WSDL décrivant ses services.
- Génération WSDL avec un service NuSoap
  - `$server->configureWSDL('demows', 'http://example.org/demo');`

- Explication :
  - `configureWSDL()` : indique au serveur NuSOAP qu'il faut prendre en charge la génération du WSDL.
  - Paramètres : Le nom du service, L'URL de l'espace de noms.

Déclaration des types de données dans le WSDL

- Pour chaque fonction, il est nécessaire déclarer les types des paramètres d'entrée et de sortie.
  - nom => tableaux de types
  - Utilise la syntaxe XSD
- Exemple :

```
$server->register(
    'myFunction',
    array('param' => 'xsd:string'),      // inputs
    array('result' => 'xsd:string'),     // outputs
    'http://example.org/demo'            // Namespace URI
);
```

Utiliser Document/Literal

- Pour utiliser la combinaison `document/literal`, il faut ajouter des informations supplémentaire au moment de l'enregistrement des méthodes/fonctions dans le service.

```
$server->register(
    'myFunction',                                   // method name
    array('param' => 'xsd:string'),                 // input parameters
    array('return' => 'xsd:string'),                // output parameters
    'http://example.org/demo',                      // Namespace URI
    'http://example.org/demo#myFunction',           // Soap Action
    'document',                                     // Style
    'literal',                                      // Use
    'My Soap Operations'                            // Documentation
);
```

Utiliser Document/Literal (suite)

- Il faut également spécifier le style dans la génération du WSDL.

```
$service->configureWSDL(
    "my_service",
    "http://www.formation.fr/pol/soap",
    false,
    "document"
);
```

Sérialisation des résultats Document/Literal

- Les fonctions du service doivent sérialiser les résultats dans un objet `soapval`.

```
function myFunction($param) {
    return new soapval('return', 'xsd:string', 'Hello ' . $param);
}
```

- 'return'
  - Le nom du paramètre de retour SOAP (toujours 'return')
- Le type de données XMLSchema
- La valeur à sérialiser

Appel du service en document/literal

- D'un point de vue du client, il faut également spécifier le style et l'encodage `document/literal` au moment de l'appel des opérations du service.

```
$result = $client->call(
    'myFunction',                                   // Method name
    array('parameters' => array('param' => 'abc')),  // Parameters (wrapped in 'parameters' !)
    'http://example.org/demo',                      // NS URI
    'http://example.org/demo#myFunction',           // SOAP Action
    false,                                          // SOAP Headers
    null,                                           // RPC parameters
    'document',                                     // Style
    'literal'                                       // Use
);
```

Gestion des types de données complexes

- NuSOAP ne sait travailler qu'avec des tableaux associatifs !
  - Il faut donc prévoir de convertir les objets métiers en tableaux associatifs et vice-versa…
- Il est nécessaire de spécifier ensuite les types complexes d'un point de vue SOAP / XMLSchema.
- Exemple pour un objet :

```
$service->wsdl->addComplexType(
    "Auteur",         // Nom donné au type
    "complexType",    // Type de base XMLSchema (complexType ou simpleType)
    "struct",         // Type PHP ('struct' pour les objets, 'array' pour les tableaux d'objets
    "sequence",       // Structure XMLSchema
    "",               // Type a restreindre
    array(            // Description du type
            "identifiant" => array("name" => "identifiant", "type" => "xsd:string"),
            "motDePasse" => array("name" => "motDePasse", "type" => "xsd:string"),
            "nom" => array("name" => "nom", "type" => "xsd:string"),
            "prenom" => array("name" => "prenom", "type" => "xsd:string")
    )
);
```

Gestion des types de données complexes (suite)

- Exemple pour un tableau d'objets :

```php
$service->wsdl->addComplexType(
    "ArrayOfAuteur",     // Nom donné au type
    "complexType",       // Type de base XMLSchema (complexType ou simpleType)
    "array",             // Type PHP ('struct' pour les objets, 'array' pour les tableaux d'objets
    "",                  // Pas de structure XMLSchema, on travaille par restriction
    "SOAP-ENC:Array",    // Type de base a restreindre : un tableau SOAP
    array(),             // Pas de description, on restreint un type existant
    array(               // Description de la restriction
        array(
            'ref' => 'SOAP-ENC:arrayType',
            'wsdl:arrayType' => 'tns:Auteur[]'
        )
    ),
    "tns:Auteur"         // Type de chaque élément du tableau
);
```