Université IBM i 2018

16 et 17 mai

IBM Client Center Paris

S51 – Vos projets Web Services IBM i à l'aide de PHP

Gautier DUMAS
Notos
gdumas@notos.fr – 04 30 96 97 31

NoToS

- Expertise autour de l'IBM i
  - Regard moderne
  - Sécurité
  - Service
    - Formation, audit, développement…
- PHP sur IBM i avec Zend
  - Modernisation
  - Web Services…
- Développement de progiciels
  - Modernisation à valeur ajoutée des IBM i

Plan de la présentation

- Les Web Services (WS)
  - Généralités
  - SOAP vs REST
  - Les formats d'échanges XML et JSON
- PHP sur IBM i
  - Pourquoi le PHP pour les WS ?
  - Le Zend Server
  - Intégration des web services dans l'existant IBM i
- Exemples concrets de WS en PHP
  - Exposer un Web Service REST
  - Exposer un Web Service SOAP
  - Consommer un web service REST
  - Consommer un web service SOAP

Les Web Services

Généralités sur les WS

- Un web service est un outil utilisé pour standardiser les échanges de données entre deux interlocuteurs, potentiellement dans des environnements hétérogènes.
- Il permet de faire communiquer deux machines au travers de TCP/IP en milieu hétérogène, sans se soucier de l'OS hôte de la machine, ni du langage de programmation utilisé.
  Exemples :
  - Une application Windows développée en C# pourra interroger un web service sur IBM i développé en PHP.
  - Un script PHP sur IBM i pourra interroger un web service écrit en JAVA sur un serveur Linux.
  …
- Il existe deux grandes familles (deux standards donc) de Web Service :
  - SOAP
  - REST

Bien démarrer son projet WS

- Pour bien démarrer son projet Web Service, il faut se poser les questions suivantes :
  - Allons nous consommer ou exposer des web services ? (ou les deux ?)
  - Allons nous implémenter la norme REST ou la norme SOAP ?
  - Quelle technologie allons nous utiliser pour développer ces web services ?

➔ Ces points seront développés dans cette session au travers de 4 exemples en PHP :

- Exposer un web service REST
- Exposer un web service SOAP
- Consommer un web service REST
- Consommer un web service SOAP

SOAP vs REST

- SOAP (initialement acronyme de Simple Object Access Protocol)
- Protocole orienté objet bâti sur la norme XML
- Il est composé de deux parties :
  - Une enveloppe, contenant des information sur le message lui-même afin de permettre son acheminement et son traitement
  - Un modèle de données, définissant le format du message, c'est-à-dire les informations à transmettre
- Initialement défini par Microsoft et IBM, mais est devenu une référence depuis une recommandation du W3C, utilisée notamment le cadre d'architecture de type SOA (Service Oriented Architecture).
- S'appuie sur les fichiers WSDL, qui font office de contrats de services. Ils définissent précisément la nature des échanges entre les consommateurs et les producteurs de web services. C'est un atout indéniable par rapport aux services de type REST, plus simples à mettre en œuvre mais aussi moins formalisés, moins cadrés.

SOAP vs REST

- REST (acronyme de Representational State Transfer)
- N'est pas un protocole (tel que HTTP), ni un format d'échange
- Il définit une architecture qui utilise le protocole HTTP et ses méthodes associées pour implémenter les web services
- Les différents verbes HTTP seront utilisés pour différencier les méthodes à appeler :
  - GET (lecture), PUT (création), POST (modification), DELETE (suppression)
- Plus simple à mettre en œuvre (un simple script PHP derrière un serveur web)
- Plus performant que SOAP avec la possibilité de choisir son format d'échange
- Moins normé, et nécessitera donc une communication accrue entre le producteur et le consommateur (ou l'écriture de documentations associées)

SOAP vs REST

- La norme SOAP sera utilisée lorsque :
  - Besoin de règles strictes (cadrées)
  - La performance n'est pas un axe majeur
  - Le Web Service sera utilisé par un grand nombre de consommateurs différents (il suffira de leur fournir l'url du WSDL)
- La norme REST sera utilisée lorsque :
  - Le besoin de structuration est moins important
  - Le besoin de performance est majeur
  - Plus utilisé pour des constructions d'API autour d'objets

Les formats d'échanges

- Que l'on implémente un WS SOAP ou REST, la question du format d'échange doit se poser
  - En SOAP, le format XML est obligatoirement présent. Bien que nous puissions utiliser des chaînes JSON, elles seront intégrées dans un message global XML
  - En REST, le choix du format est libre. Nous pouvons même choisir de ne pas utiliser de format d'échanges…

- Le choix du format se fera en fonction :
  - des connaissances des développeurs
    Ont-ils déjà menés des projets XML ou JSON ?
  - des besoins de performances
    JSON pour le plus performant car moins lourd sur le réseau
  - Des compétences des interlocuteurs
    Maitrisent ils plutôt le XML ou le JSON ?

Les formats d'échanges

XML	JSON
eXtensible Markup Language	**JavaScript Object Notation**
Format d'échange utilisé depuis de nombreuses années (EDI, Echanges bancaires, SOAP etc…)	Format d'échange plus récent issu du développement web (très utilisé avec les web services REST)
Structure verbeuse	Structure légère
Stockage dans un stream file (*STMF) ou dans DB2 (zone XML ou CHAR)	Stockage dans un stream file (*STMF) ou dans DB2 (zone CHAR)
Lecture et écriture de chaines XML en RPG ou en PHP	Manipulation et écriture facilité en PHP à l'aide de fonctions intégrées (json_encode et json_decode)

Les formats d'échanges

- Exemple tableau : même jeu de données en XML et en JSON

```xml
<PO>
 <id>103</id>
 <orderDate>2014-06-20</orderDate>
 <customer>
   <cid>888</cid>
 </customer>
 <items>
  <item>
   <partNum>872-AA</partNum>
   <shipDate>2014-06-21</shipDate>
   <productName>Lawnmower</productName>
   <USPrice>749.99</USPrice>
   <quantity>1</quantity>
 </item>
 <item>
  <partNum>837-CM</partNum>
  <productName>Digital Camera</productName>
  <USPrice>199.99</USPrice>
  <quantity>2</quantity>
  <comment>2014-06-22</comment>
  </item>
 </items>
</PO>
```

```json
{
 "PO":{
 "id": 103,
 "orderDate": "2014-06-20",
 "customer": {"@cid": 888},
 "items": {
   "item": [
            { "partNum": "872-AA",
              "productName": "Lawnmower",
              "quantity": 1,
              "USPrice": 749.99,
              "shipDate": "2014-06-21"
            },
            { "partNum": "837-CM",
              "productName": "Digital Camera",
              "quantity": 2,
              "USPrice": 199.99,
              "comment": "2014-06-22"
            }
          ]
        }
    }
}
```

PHP sur IBM i
avec le Zend Server

Pourquoi le PHP pour les WS ?

- PHP permet d'unifier la technologie et les compétences utilisées pour tous les projets WS :
  - De type REST ou SOAP
  - Simples ou complexes (avec authentification, structure d'échanges complexes …)
  - Autant pour consommer qu'exposer des WS
  - Permet de manipuler aussi bien le XML que le JSON au travers de fonctions natives

  ➔ En natif IBM i, il faudra changer de technologies selon le projet ! (SQL, RPG, serveur de web services intégré)
- Il est facilement accessible aux développeurs IBM i (notamment en mettant de côté l'aspect interface graphique qui est inutile avec les WS)
- Il est complètement intégré depuis plusieurs années sur l'IBM i avec le Zend Server

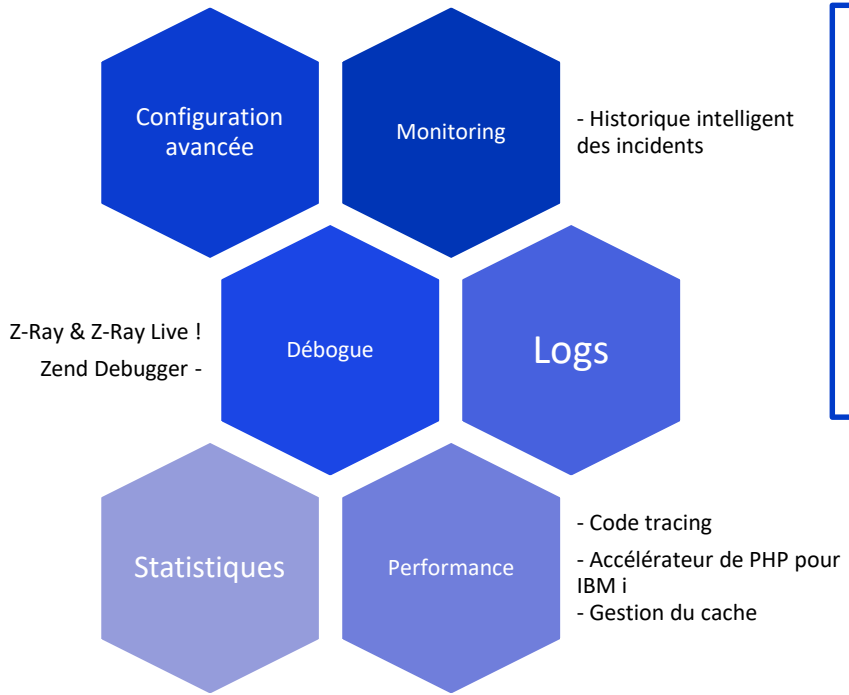
PHP sur IBM i avec le Zend Server

- Ecosystème PHP complètement intégré à l'IBM i

- Zend Server actuellement en version 9.1 (PHP 7.1)
  - Un environnement conçu et testé pour la production avec :
    - Les dernières versions stables de PHP
    - Mise à disposition des mises à jour
    - Un support dédié à la plateforme IBM i

- Zend Studio actuellement en version 13.6
  - S'appuie sur noyau Eclipse (comme Rdi)
    - Nombreux plugIns disponibles
    - Interface moderne de développement

Le Zend Server



- Intégration complète
  - Interface graphique
  - Interface 5250
- Professionnalisation des applications PHP
- Accès

PHP sur IBM i : exposer des web services

- Lorsque le besoin sera d'exposer des web services, en PHP il sera possible :
  - D'interroger la base de données DB2 directement en SQL (connecteur ibm_db2 en natif en PHP)
  - De manipuler des fichiers de l'IFS (fabrications et envois de fichiers PDF, Excel, Images etc…)
  - Mais aussi d'appeler des programmes existants (RPG, CL, COBOL…) au travers de procédures stockées ou du PHP Toolkit for IBM i

PHP sur IBM i : consommer des web services

- Lorsque le besoin sera de consommer un web service, le script PHP pourra être appelé :
  - Bien évidemment dans les applications web (lors du chargement d'une page, au click sur un bouton, à la validation d'un formulaire…)
  - Mais aussi directement dans les logiques métiers existantes (*PGM RPG, CL…) à l'aide de l'API QP2SHELL qui permet d'appeler des scripts PHP en ligne de commandes
  - Cette technique est très utilisée pour intégrer des flux web services dans les applications 5250
  - Elle est également utilisée pour planifier l'exécution de clients web services au travers du WRKJOBSCDE

Exemples concrets en PHP : Exposer un web service

Exposer un web service

- **Objectif :** rendre disponible une information ou une fonction pour une autre application au sein de l'entreprise ou à l'extérieur (Intranet, Site Web, un client, un fournisseur etc…)
- Se poser les bonnes questions :
  - Qui va consommer le web service ?
  - Combien de méthodes sont à exposer ?
  - A quelle fréquence ?
  - Quels seront les volumes de données échangés ?

➔ Les réponses à ces questions permettront de faire un choix approprié entre SOAP et REST

Exposer un web service (REST) 1

- Une fois l'environnement web PHP installé (PHP/APACHE avec le Zend Server), aucun autre prérequis n'est nécessaire pour développer son web service REST
- En PHP, il faudra commencer par récupérer le verbe HTTP envoyé pour connaitre la méthode à appeler :

```
// Bloc de code pour identifier le verbe HTTP utilisé par le client
$method = $_SERVER['REQUEST_METHOD'];
if ($method == 'POST') {
   if (array_key_exists('HTTP_X_HTTP_METHOD', $_SERVER)) {
      if ($_SERVER['HTTP_X_HTTP_METHOD'] == 'DELETE') {
         $method = 'DELETE';
      } else
         if ($_SERVER['HTTP_X_HTTP_METHOD'] == 'PUT') {
            $method = 'PUT';
         } else {
            throw new Exception("Unexpected Header");
         }
   } else {
      $method = 'POST' ;
   }
}
```

Exposer un web service (REST) 2

- Puis en fonction de la méthode, nous appellerons des fonctions différentes

```
switch ($method) {
   case 'PUT':{
      echo 'update => ';
      parse_str(file_get_contents('php://input'),$datas);
      print_r($datas);
      break ;
   }
   case 'DELETE':{
      echo 'delete => ';
      parse_str(file_get_contents('php://input'),$datas);
      print_r($datas);
      break ;
   }
   case 'POST':{
      echo 'insert => ';
      print_r($_POST);
      break ;
   }
```

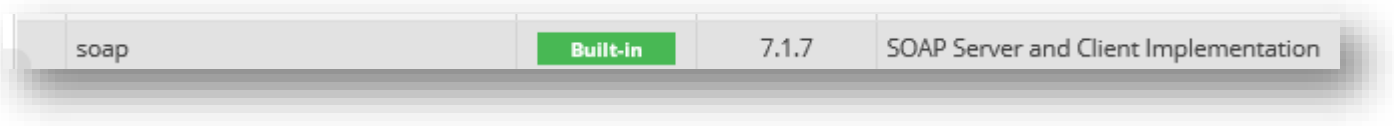
```
case 'GET':{
      echo 'select => ';
      print_r($_GET);
      break;
   }
   default:
      die('Invalid Method '. 405);
      break;
}
```

Exposer un web service (REST) 3

```
case 'GET':{
        //Méthode implémentée pour la démonstration
        include './config.inc.php';
        $cnx = db2_connect(HOST, USR, PWD);

        //Initialisation des paramètres
        $codeArticle= $_GET['codeArticle'];//'1002';
        $QTTE  = 0;

        $cpro = db2_prepare($cnx,"CALL IBM2016.STOCKP( ? , ? ) ");

        db2_bind_param($cpro, 1, 'codeArticle', DB2_PARAM_IN);
        db2_bind_param($cpro, 2, 'QTTE', DB2_PARAM_OUT);

        if (db2_execute($cpro)) {
            $tabRetour = array(
                "CODEART"=>$codeArticle,
                "QTE"=>$QTTE
            );

            echo json_encode($tabRetour);
        }else {
            echo db2_stmt_errormsg();
        }
        break;
    }
```

Exposer un web service (SOAP)

- Nécessite l'utilisation de l'extension PHP soap disponible en standard dans le Zend Server sans installation supplémentaire



- La première étape pour développer un WS SOAP est la construction du fichier WSDL qui va décrire les méthodes à appeler pour le consommateur
- Cette étape très importante, peut se faire de plusieurs façons :
  - A la main avec un bon éditeur (Zend Studio) : nécessite la connaissance complète de la syntaxe WSDL
  - A l'aide de la classe PHP nusoap qui permet en quelques lignes de code PHP, de nous générer le WSDL

Génération du WSDL avec nusoap

```php
<?php
// Inclusion de la classe
require './nusoap/lib/nusoap.php';

// Création d'un nouveau serveur SOAP
$serv = new nusoap_server();

// Configuration du WSDL
$serv->configureWSDL("IBM2018", "urn:IBM2018/",

"http://IP_IBMi/IBM2018/WSServerSOAP.php");

// Définition du namespace
$serv->wsdl->schemaTargetNamespace = "http://IP_IBMi/IBM2018/";
```

Génération du wsdl avec nusoap

```
// Enregistrement dans le serveur SOAP de la fonction PHP GetQtyInStock
//  et définition de ses paramètres E/S
$serv->register("GetQtyInStock",
    array(
      "Code_Article" => "xsd:string"
    ),
    array(
      "QtyInStock" => "xsd:int"
    ),
    'urn:GetQtyInStock2',
    'urn:IBM2018/GetQtyInStock',
    '', '', 'Retourne la qte en stock pour un article'
);
// Récupère la requête SOAP entrante
$POST_DATA = file_get_contents("php://input");

// Analyse le XML, exécute la (ou les) fonction(s), et renvoie la réponse au format XML
$serv->service($POST_DATA);
```

Exposer un web service (SOAP)

- Une fois le WSDL créé, nous pouvons commencer à développer les méthodes qui iront, en fonction du besoin, lire/écrire dans DB2.
- Ces méthodes devront respecter les paramètres d'Entrée/Sortie qui auront été déclaré dans le WSDL (bien respecter le nombre et le type des paramètres échangés).

Ecriture de la méthode

```
function GetQtyInStock($codeArticle){
    include 'config.inc.php';

    $cnx = db2_connect(HOST, USR, PWD);

    //Initialisation des paramètres
    $codeArticle= $codeArticle;//'1002';
    $QTTE  = 0;

    $cpro = db2_prepare($cnx,"CALL IBM2016.STOCKP( ? , ? ) ");


    db2_bind_param($cpro, 1, 'codeArticle', DB2_PARAM_IN);
    db2_bind_param($cpro, 2, 'QTTE', DB2_PARAM_OUT);

    if (db2_execute($cpro)) {
        return $QTTE;
    }else {
        return db2_stmt_errormsg();
    }

}
```

Sécurisez vos web services

- Le besoin de sécurisation peut varier en fonction de la nature des données échangées, et de la localisation du consommateur (sur internet ? Sur le réseau local ?)
- Sécurisation possible à différents niveaux :
  - Au niveau de l'instance web APACHE par SSL afin :
    - de crypter les informations qui vont transiter entre le client et le serveur WS
    - d'assurer votre identité au consommateur
  - Au niveau du script PHP en demandant une authentification (par exemple par utilisateur / mot de passe)
    Au choix et en fonction du projet, la vérification pourra se faire par rapport à un USRPRF de l'IBM i, un compte du domaine (par connexion LDAP) ou tout autre solution (base de données utilisateurs, fichier .txt, autre sgbd etc…)
  - Au niveau réseau avec la configuration du firewall de l'entreprise
    Exemple : Autorisations par IP
  - …

Exemples concrets en PHP : Consommer un web service

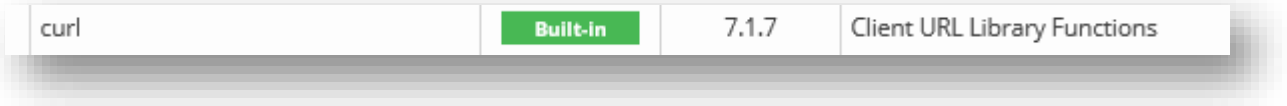
Consommer un web service (REST)

- Pour consommer un web service REST simple, nous pouvons utiliser la fonction PHP file_get_contents

```
/*
* METHODE 1 : avec file_get_contents
*/

$reponse =
file_get_contents("http://192.168.1.3/IBM/Universite2018/WSServerREST
.php?codeArticle=1002");

echo "<h1>Résultat JSON avec file_get_contents pour l'article
1002</h1>";
echo $reponse."</br>";
```

Consommer un web service (REST)

- Pour consommer des WS REST plus complexes (authentification, passage de paramètres autre que GET, …), il est préférable d'utiliser CURL. CURL est livrée en standard avec le Zend Server sous forme d'extension.



- Les fonctionnalités de CURL peuvent donc être appelées directement en PHP sans installation supplémentaire

- Plus d'infos sur CURL en PHP sur :

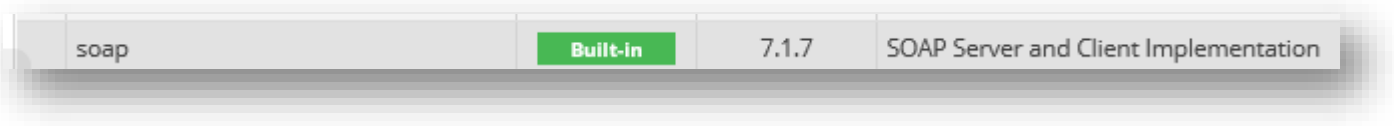
http://php.net/manual/fr/ref.curl.php

Consommer un web service (REST)

```
/*
 * METHODE 2 : avec CURL
 */

$url = "http://192.168.1.3/IBM/Universite2018/WSServerREST.php?codeArticle=1003";
$client = curl_init($url);
curl_setopt($client, CURLOPT_RETURNTRANSFER, 1);
$reponse = curl_exec($client);
curl_close($client);

echo "<h1>Résultat JSON avec CURL pour l'article 1003</h1>";
echo $reponse."</br>";
```

Consommer un web service (SOAP)

- Nécessite l'utilisation de l'extension PHP soap disponible en standard dans le Zend Server sans installation supplémentaire



Consommer un web service (SOAP)

```php
//Connexion au WSDL
$client = new
SoapClient("http://192.168.1.3/IBM/Universite2018/WSServerSOAP.php?wsdl");

//Affichage des fonctions publiées
echo "<h1>Liste des fonctions publiées dans le WSDL</h1>";
var_dump($client->__getFunctions());

//Appel de la méthode SOAP GetQtyInStock
$reponse = $client-
>__soapCall("GetQtyInStock",array("Code_Article"=>"1002"));

//Affichage du résultat de l'appel
echo "<h1>Résultat avec SoapClient pour l'article 1002</h1>";
echo $reponse."</br>";
```

Merci de votre attention

Pour plus d’informations
gdumas@notos.fr