



Services Web SOAP & REST en PHP

© 2019-2020 Madani Boukebeche

madani.boukebeche@ynou.com

Plan

1/3

- **Services Web SOAP**
- **Sans Framework PHP**
- **Services Web REST**
 - **Sans Framework PHP**
 - **Avec Framework PHP**



© 2019-2020 Madani Boukebeche

madani.boukebeche@ynou.com

3

SERVICES WEB SOAP

© 2019-2020 Madani Boukebeche

madani.boukebeche@dynov.com

3

3

SERVICES WEB SOAP

- Permet à deux applications de communiquer en mode client / serveur à travers des protocoles standards HTTP, FTP, SMTP, ...
 - Basé sur le protocole standard SOAP 1.1 et 1.2
 - L'invocation des traitements distants se fait par appel de méthodes comme appels locaux
 - La description des web services est faite au moyen d'un fichier XML WSDL (Web Service Description Language) que le fournisseur du service web met à disposition de ses clients

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://webservices.dilogic.com" xmlns
* <!--WSDL created by Apache Axis version: 1.4
Built on Apr 22, 2006 (06:55:48 PDT)-->
* <wsdl:types><
*   <wsdl:message name="setGreetingRequest"><
*   <wsdl:message name="sayHelloRequest"><
*   <wsdl:message name="sayHelloResponse"><
*   <wsdl:message name="setGreetingResponse"><
*   <wsdl:portType name="HelloWorld"><
*   <wsdl:binding name="HelloWorldSoapBinding" type="impl:HelloWorld"><
*   <wsdl:service name="HelloWorldService"><
</wsdl:definitions>
```

© 2019-2020 Madani Boukebeche

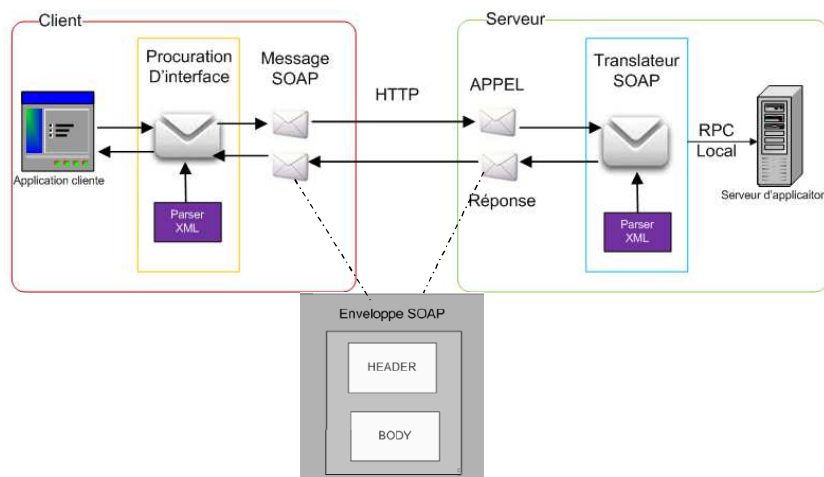
madani.boukebeche@dynov.com

4

4

SERVICES WEB SOAP

- Le client et le serveur échangent des messages SOAP encapsulés dans des enveloppes SOAP



© 2019-2020 Madani Boukebeche

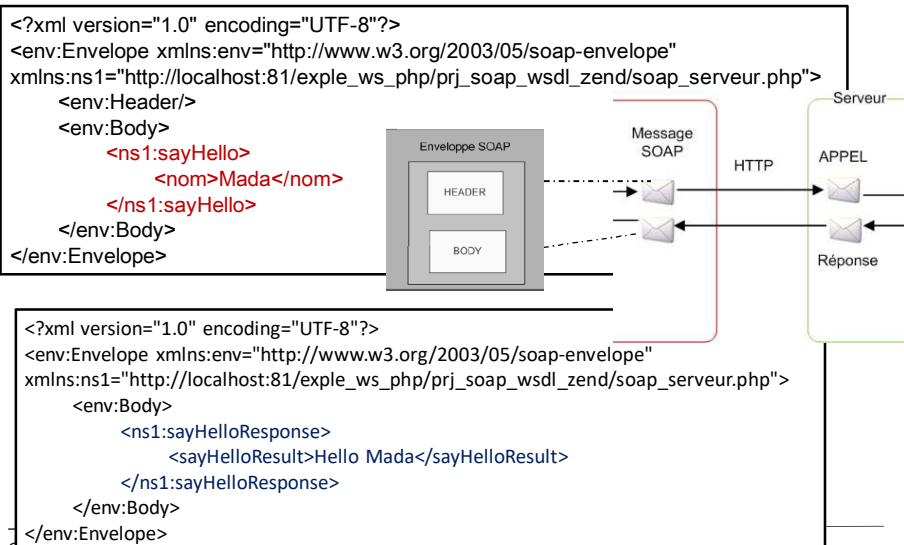
madani.boukebeche@ynov.com

5

5

SERVICES WEB SOAP

- Les enveloppes SOAP échangées entre le client et le serveur sont des documents XML



SERVICES WEB SOAP

- Un Service Web est une classe (dans un langage POO) qui expose des méthodes (fonctions) invocable à distance (comme un RPC ou un RMI)
- D'une manière générale, la mise en œuvre des Services Web SOAP peut se faire selon deux approches :
 - Code first :
 - On écrit la classe Service Web qui sera exposée à travers son contrat WSDL (généralisé la plupart des cas d'une manière automatique)
 - Contract first
 - On écrit le contrat WSDL puis on crée la classe Service Web (généralement d'une manière automatique depuis WSDL)
- En PHP natif, deux classes sont proposées :
 - SoapServer
 - SoapClient
 - Qui fonctionnent avec ou sans WSDL
 - Sans possibilité de génération automatique (sauf à travers certains frameworks comme Zend, Symfony, Laravel, ...)

© 2019-2020 Madani Boukebeche

madani.boukebeche@ynou.com

7

7

SOAP SANS FRAMEWORK

• Classe SoapServer

```
class SoapServer {
    /* Méthodes */
    public void addFunction ( mixed $functions )
    public void addSoapHeader ( SoapHeader $object )
    public __construct ( mixed $wsdl [, array $options ] )
    public void fault ( string $code , string $string
                      [, string $actor [, string $details [, string $name ]]] )
    public array getFunctions ( void )
    public void handle ( [ string $soap_request ] )
    public void setClass ( string $class_name
                        [, mixed $args [, mixed $... ] ] )
    public void setObject ( object $object )
    public void setPersistence ( int $mode )
    public SoapServer ( mixed $wsdl [, array $options ] )
}
```

SoapServer

```
addFunction
addSoapHeader
__construct
fault
getFunctions
handle
setClass
setObject
setPersistence
```

<https://www.php.net/manual/fr/class.soapserver.php>

© 2019-2020 Madani Boukebeche

madani.boukebeche@ynou.com

8

8

SOAP SANS FRAMEWORK

• Classe SoapClient

```
class SoapClient {
    /* Méthodes */
    public mixed __call ( string $function_name , string $arguments )
    public SoapClient ( mixed $wsdl [, array $options ] )
    public string __doRequest ( string $request , string $location ,
        string $action , int $version [, int $one_way = 0 ] )
    public array __getFunctions ( void )
    public string __getLastRequest ( void )
    public string __getLastRequestHeaders ( void )
    public string __getLastResponse ( void )
    public string __getLastResponseHeaders ( void )
    public array __getTypes ( void )
    public void __setCookie ( string $name [, string $value ] )
    public string __setLocation ( [ string $new_location ] )
    public bool __setSoapHeaders ( [ mixed $soapheaders ] )
    public mixed __soapCall ( string $function_name , array $arguments
        [, array $options [, mixed $input_headers [, array &$output_headers ]]] )
    public SoapClient ( mixed $wsdl [, array $options ] )
}
```

SoapClient

```
__call
__construct
__doRequest
__getCookies
__getFunctions
__getLastRequest
__getLastRequestHeaders
__getLastResponse
__getLastResponseHeaders
__getTypes
__setCookie
__setLocation
__setSoapHeaders
__soapCall
```

<https://www.php.net/manual/fr/class.soapclient.php>

SOAP SANS FRAMEWORK

• Exemple de serveur SOAP **sans WSDL** et **sans classe**

```
<?php // ws_hello_server

function sayHello($name){
    return "Hello, $name";
}

$server = new SoapServer(NULL, array('uri' => "urn:wsphp"));
$server->AddFunction("sayHello");
$server->handle();
```

ws_hello_server.php

• Exemple de client SOAP **sans WSDL**

```
<?php // ws_hello_client
try{
    $opts = array('location'
        => 'http://localhost:81/exple_ws_php/ws_hello_server.php',
        'uri' => "urn:wsphp",
        'style' => SOAP_RPC,
        'use' => SOAP_ENCODED
    );
    $client = new SoapClient(NULL, $opts);
    $response = $client->sayHello("Mada");
    echo $response;
} catch (SoapFault $e) { var_dump($e); }
```

ws_hello_client.php

localhost:81/exple_ws_php/ws_hello_client.php
Hello, Mada

SOAP SANS FRAMEWORK

- Exemple de serveur SOAP **sans WSDL** et **avec classe**

```
<?php // ws_hello_classe_server
class HelloWS{
    public function sayHello($nom)
    {
        return 'Hello ' . $nom;
    }
}
$server = new SoapServer(NULL, array('uri' => "urn:wsphp"));
$server->setClass("HelloWS");
$server->handle();
```

ws_hello_classe_server.php

- Exemple de client SOAP **sans WSDL**

```
<?php // ws_hello_classe_client
try{
    $opts = array('location'
        =>'http://localhost:81/exple_ws_php/ws_hello_classe_server.php',
        'uri'=> "urn:wsphp", 'style'=> SOAP_RPC, 'use'=> SOAP_ENCODED);
    $client = new SoapClient(NULL, $opts);
    echo $client->sayHello("Mada");
} catch(SoapFault $e){ var_dump($e); }
```

ws_hello_classe_client.php

L'appel est idem que
l'exemple précédent

© 2019-2020 Madani Boukebeche

madani.boukebeche@ynov.com

11

11

SERVICES WEB SOAP

- Exemple 2 avec objet JSON sans WSDL : Le serveur

```
<?php // ws_users_server

require_once("dao_user.php");

function getUser($idUser=null){
    $resultat = ($idUser != null)? DaoUser::find($idUser):DaoUser::findAll();
    return json_encode($resultat);
}
$server = new SoapServer(NULL, array('uri' => "urn:wsphp"));
$server->AddFunction("getUser");
$server->handle();
```

© 2019-2020 Madani Boukebeche

madani.boukebeche@ynov.com

12

12

SERVICES WEB SOAP

• Exemple 2 avec objet JSON sans WSDL : la DAO

```
// dao_user.php
<?php
class DaoUser {
    const TABLE_NAME = "user";
    public static function connect() {
        $hote = 'localhost';
        $nom_bdd = 'formation';
        $utilisateur = 'root';
        $mot_de_passe = '';
        $pdo = new PDO('mysql:host='.$hote.';dbname='.$nom_bdd, $utilisateur, $mot_de_passe);
        $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        return $pdo;
    }
    public static function find($idUser) {
        if ($pdo = self::connect()) {
            $requete = $pdo->prepare("SELECT * FROM ".self::TABLE_NAME ." WHERE `id_user` = :id");
            $requete->bindParam(':id', $idUser);
            $requete->execute();
            return $requete->fetchAll(PDO::FETCH_ASSOC);
        }
        return null;
    }
    public static function findAll() {
        if ($pdo = self::connect()) {
            $requete = $pdo->prepare("SELECT * FROM ".self::TABLE_NAME );
            $requete->execute();
            return $requete->fetchAll(PDO::FETCH_ASSOC);
        }
        return null;
    }
}
```

13

SERVICES WEB SOAP

• Exemple 2 avec objet JSON sans WSDL : Le client

```
// ws_users_client.php
<?php
try{
    $opts = array('location'=>'http://localhost:81/exple_ws_php/ws_users_server.php',
                  'uri'=> "urn:wsphp",
                  'style'=> SOAP_RPC,
                  'use'=> SOAP_ENCODED
                );
    $sClient = new SoapClient(NULL, $opts);
    echo "<br/> ** Get all users : <br/>";
    $response = $sClient->getUser();
    $users = json_decode($response);
    // print_r($users);
    foreach($users as $user){
        echo $user->id_user . " " . $user->nom . " " . $user->prenom . "<br/>";
    }

    echo " <br/> ** Get user with id = 2 : <br/>";
    $responseJson = $sClient->getUser(2);
    $response = json_decode($responseJson);
    if (count($response) > 0) {
        $user2 = $response[0];
        print_r($user2);
    }
} catch(SoapFault $e){ var_dump($e);}
```

© 2019-2020 Madani Boukebeche

madani.boukebeche@ynou.com

14

14

SOAP SANS FRAMEWORK

- Exemple de serveur SOAP **avec WSDL** et **avec classe**

```

helloWorld.wsdl
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="HelloWS" targetNamespace="urn:dilogic.com" xmlns:tns="urn:dilogic.com" ...>
  <message name="sayHelloRequest"> <part name="params" type="xsd:string"/> </message>
  <message name="sayHelloResponse"> <part name="result" type="xsd:string"/> </message>
  <message name="helloWorld"/>
  <message name="helloWorldResponse"> <part name="result" type="xsd:string"/> </message>
  <binding name="HelloWSBinding" type="tns:HelloWSPort">
    <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="sayHello">
      <soap:operation soapAction="urn:sayHelloAction"/>
      <input> <soap:body use="encoded" namespace="urn:dilogic.com"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
      </input>
      <output> <soap:body use="encoded" namespace="urn:dilogic.com"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
      </output>
    </operation>
    <operation name="helloWorld">
      ...
    </operation>
  </binding>
  <service name="HelloWS">
    <port name="HelloWSPort" binding="tns:HelloWSBinding">
      <soap:address location="http://localhost:81/exple_ws_php/ws_wsdl_hello_server.php"/>
    </port>
  </service>
</definitions>

```

SOAP SANS FRAMEWORK

- Exemple de serveur SOAP **avec WSDL** et **avec classe**

```

ws_wsdl_hello_server.php
<?php // ws_wsdl_hello_server
class HelloWS{
    public function helloWorld(){ return 'Hi World'; }
    public function sayHello($nom) {return "Hello, $nom";}
}

$server = new SoapServer('helloWorld.wsdl');
$server->setClass("HelloWS");
$server->handle();

```

- Exemple de client SOAP **avec WSDL**

```

ws_wsdl_hello_client.php
<?php // ws_wsdl_hello_client
ini_set("soap.wsdl_cache_enabled", 0);
try{
    $client = new SoapClient('http://localhost:81/exple_ws_php/helloWorld.wsdl');
    var_dump($client->__getFunctions()); // Affiche les web method trouvées
    $response = $client->sayHello("Mada");
    var_dump( $response);
} catch(SoapFault $e){var_dump($e);}

```


SOAP AVEC LE FRAMEWORK ZEND

- Exemple de serveur SOAP **avec génération de WSDL**

```
<?php
require_once __DIR__ . '/vendor/autoload.php';
class HelloWS{
    /**
     * greeeting World
     *
     * @return string
     */
    public function helloWorld() { return 'Hi World'; }
    /**
     * say Hello
     *
     * @param string $nom
     * @return string
     */
    public function sayHello($nom) { return 'Hello ' . $nom; }
}
...
```

ws_wsdll_hello_server.php

Les commentaires sont obligatoire pour indiquer les paramètres et le type de retour @

SOAP AVEC LE FRAMEWORK ZEND

- Exemple de serveur SOAP **avec génération de WSDL**

```
$serverUrl = "http://localhost:81/exple_ws_php/prj_soap_wsdll zend/soap_serveur.php";
$options = [
    'uri' => $serverUrl,
    'cache_wsdll' => WSDLL_CACHE_NONE
];
$server = new Zend\Soap\Server(null, $options);
if (isset($_GET['wsdl'])) {
    $soapAutoDiscover = new \Zend\Soap\AutoDiscover(
        new \Zend\Soap\Wsdll\ComplexTypeStrategy\ArrayOfTypeSequence());
    $soapAutoDiscover->setBindingStyle(array('style' => 'document'));
    $soapAutoDiscover->setOperationBodyStyle(array('use' => 'literal'));
    $soapAutoDiscover->setClass('HelloWS');
    $soapAutoDiscover->setUri($serverUrl);

    header("Content-Type: application/xml");
    echo $soapAutoDiscover->generate()->toXml();
} else {
    $soap = new \Zend\Soap\Server($serverUrl . '?wsdl', $options);
    $soap->setObject(new \Zend\Soap\Server\DocumentLiteralWrapper(new HelloWS()));
    $soap->handle();
}
```

ws_wsdll_hello_server.php

SOAP AVEC LE FRAMEWORK ZEND

• Exemple : Génération du WSDL

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<definitions
  targetNamespace="http://localhost:81/exple_ws_php/prj_soap_wsdl_zend/soap_serveur.php"
  name="HelloWS" xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/" xmlns:soap-
  enc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns="http://localhost:81/exple_ws_php/prj_soap_wsdl_zend/soap_serveur.php"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns="http://schemas.xmlsoap.org/wsdl/">
  <types>
    <xsd:schema
      targetNamespace="http://localhost:81/exple_ws_php/prj_soap_wsdl_zend/soap_serveur.php"
      <xsd:element name="helloWorld">
        <xsd:complexType/>
      </xsd:element>
    <xsd:element name="helloWorldResponse">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="helloWorldResult" type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="sayHello">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="nom" type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
  </types>
  <message name="sayHelloRequest" part="sayHello"/>
  <message name="sayHelloResponse" part="sayHelloResponse"/>
  <portType name="HelloWS" base="tns:HelloWS">
    <operation name="sayHello" input="sayHelloRequest" output="sayHelloResponse"/>
  </portType>
  <binding name="HelloWS" type="tns:HelloWS">
    <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="sayHello" input="sayHelloRequest" output="sayHelloResponse" soap:operation-
    soap:use="literal"/>
  </binding>
  <service name="HelloWS" base="tns:HelloWS">
    <port name="HelloWS" binding="tns:HelloWS" location="http://localhost:81/exple_ws_php/prj_soap_wsdl_zend/soap_serveur.php"
    </port>
  </service>
</definitions>
```

SOAP AVEC LE FRAMEWORK ZEND

• Exemple de client SOAP avec WSDL

ws_wsdl_hello_client.php

```
<?php
require_once __DIR__ . '/vendor/autoload.php';
$serverUrl = "http://localhost:81/exple_ws_php/prj_soap_wsdl_zend/soap_serveur.php?wsdl";

try{
    $client = new Zend\Soap\Client($serverUrl, array('cache_wsdl' => WSDL_CACHE_NONE));

    $result = $client->helloWorld();
    var_dump($result);
    echo $result->helloWorldResult;
    echo"<br/>";

    $result = $client->sayHello(['nom' => 'Mada']);
    echo $result->sayHelloResult;
    echo"<br/>";

}
catch (Exception $e){
    echo $e;
}
```

```
D:\workspace\applis\wamp64\www\exple_ws_php\prj_soap_wsdl_zend\soap_client.php:12:
object(stdClass)[4]
  public 'helloWorldResult' => string 'Hi World' (length=8)

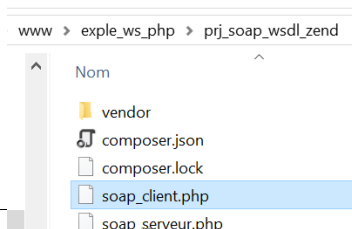
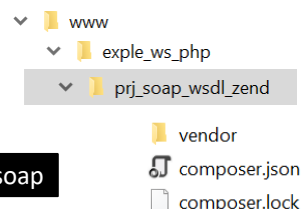
Hi World
Hello Mada
```

SOAP AVEC LE FRAMEWORK ZEND

- Création du projet avec le framework Zend
 - Créer un dossier sous le dossier de travail du serveur Apache-Php
 - Ex. `www\exple_ws_php\prj_soap_wsdl zend`
- Installer zend-soap library

`composer require zendframework/zend-soap`

 - Pour l'installation de composer voir <https://openclassrooms.com/fr/courses/2260876-premiers-pas-avec-le-framework-php-silex>
- Créer les fichiers :
 - `soap_serveur.php`
 - `soap_client.php`



© 2019-2020 Madani Boukebeche

madani.boukebeche

21

SERVICES WEB REST

© 2019-2020 Madani Boukebeche

madani.boukebeche@ynov.com

22

22

SERVICES WEB REST

- Php n'offre pas une API spécifique Rest.
- Toutefois, il est facile de mettre en œuvre une tel API avec le php natif.
- Des Frameworks Php comme Symfony/Silex/Flex, Laravel, CakePhp, ... offrent des API Rest

REST SANS FRAMEWORK

⇒ Une 1^{ère} version : Le serveur REST

```
<?php // rest_user_server_v0
require_once("dao_user.php");
header('Content-Type: application/json');
function getUser($idUser=null){
    $resultat = ($idUser != null)?DaoUser::find($idUser):DaoUser::findAll();
    $retour["success"] = true;
    $retour["results"]["nb"] = count($resultat);
    $retour["results"]["users"] = $resultat;
    return json_encode($retour);
}
$id = (empty($_GET['id']))?null:$_GET['id'];
echo getUser($id) ;
```

```
{
  "success":true,
  "results":{
    "nb":1,
    "users":[
      {
        "id_user":"4",
        "prenom":"luc",
        "nom":"barry",
        "email":"luc@hotmail.fr",
        ...
      }
    ]
  }
}
```

REST SANS FRAMEWORK

➡ Une 1^{ère} version : Client en PHP

```
<?php // rest_users_client_v0
// Get user by id :
$data = file_get_contents('http://localhost:81/exple_ws_php/rest_users_server_v0.php?id=3');
var_dump($data); //display json object
$obj_data = json_decode($data);
$users = $obj_data->results->users;
if (count($users) > 0){
    $user = $users[0];
    var_dump($user); //display user object
    echo $user->id_user . " : " // display some user attributes
        . $user->nom . " - " . $user->prenom . " - " . $user->email;
}

<?php // rest_users_client_v0
// Get all users
$data = file_get_contents('http://localhost:81/exple_ws_php/rest_users_server_v0.php');
$obj_data = json_decode($data);
$array_users = $obj_data->results->users;
for($i = 0; $i < count($array_users); $i++){
    echo $array_users[$i]->id_user . " : "
        . $array_users[$i]->nom . " - " . $array_users[$i]->prenom . " - "
        . $array_users[$i]->email . "<br/>";
}
- }
```

© 2019-2020 Madani Boukebeche madani.boukebeche@univ-boumerdes.dz 25

REST SANS FRAMEWORK

➡ Une 1^{ère} version : Client en JavaScript avec appel Ajax

```
<!DOCTYPE html> <!-- // rest_users_client_v0 -->
<html>
<head> <meta charset="utf-8"/>
<style> span{background-color: Lightgray;}</style>
</head>
<body>
<p>Message : <span id="message"></span></p>
<p>Nombre de users : <span id="nb"></span></p>
<p>Tous les users : <span id="users"></span></p>
<script>
    var xmlhttp = new XMLHttpRequest();
    xmlhttp.onreadystatechange = function() {
        if (xmlhttp.readyState == 4 && xmlhttp.status == 200){
            var retour = JSON.parse(this.responseText);
            document.getElementById("message").innerHTML = retour.success;
            document.getElementById("nb").innerHTML = retour.results.nb;
            document.getElementById("users").innerHTML =
                JSON.stringify(retour.results.users);
        }
    }
    xmlhttp.open("GET", "rest_users_server_v0.php?id=1");
    xmlhttp.send();
</script>
</body> </html>
```

© 2020 26

REST SANS FRAMEWORK

⇒ Une 1^{ère} version : Client en JavaScript avec appel Ajax

Message : true

Nombre de users : 1

Tous les users :

```
[{"id_user": "1", "login": "dp", "nom": "Dupond", "password": "abc", "prenom": "Paul", "email": "dp@gmail.com", "profil": "1"}]
```

Message : true

Nombre de users : 9

Tous les users :

```
[{"id_user": "1", "login": "dp", "nom": "Dupond", "password": "abc", "prenom": "Paul", "email": "dp@gmail.com", "profil": "1"}, {"id_user": "2", "login": "pl", "nom": "Un user", "password": "azerty", "prenom": "david", "email": "david@gmail.com", "profil": "2"}, {"id_user": "3", "login": "legrand", "nom": "Le Grand", "password": "unMotdePass", "prenom": "Pierre", "email": "legrand@free.fr", "profil": "3"}, {"id_user": "4", "login": "diaz", "nom": "Diaz", "password": "dzt", "prenom": "Laura", "email": "laura@gmail.com", "profil": "4"}, {"id_user": "5", "login": "chris", "nom": "Dessard", "password": "lemdp", "prenom": "Chris", "email": "chris@gmail.com", "profil": "5"}, {"id_user": "6", "login": "pif", "nom": "Le Petit", "password": "fip", "prenom": "alexandra", "email": "alexandra@gmail.com", "profil": "6"}, {"id_user": "7", "login": "utilis", "nom": "utilisateur", "password": "abc", "prenom": "Un nouvel", "email": "utilis@gmail.com", "profil": "0"}, {"id_user": "9", "login": "zj", "nom": "Zadi", "password": "jo", "prenom": "Jo", "email": "jozad@gmail.com", "profil": "9"}, {"id_user": "10", "login": "Lenouveau", "nom": "LE NOUVEAU", "password": "mdp", "prenom": "Patrick", "email": "lenouveau@gmail.com", "profil": "10"}]
```

© 2019-2020 Madani Boukebeche

REST SANS FRAMEWORK

⇒ Une 2^{ème} version : avec réécriture d'url selon REST

⇒ URL de la forme : ressource/{id}

localhost:81/exple_ws_php/rest/user/

localhost:81/exple_ws_php/rest/user/7

JSON Données brutes En-têtes

Enregistrer Copier Tout réduire Tout développer Filtre le JSON

```
{
  "success": true,
  "results": 1,
  "users": [
    {
      "id_user": "7",
      "login": "utilis",
      "nom": "utilisateur",
      "password": "abc",
      "prenom": "Un",
      "email": "utilis@gmail.com",
      "profil": "0"
    }
  ]
}
```

Redirection au niveau de .htaccess

```
# Redirect incoming REST URLs to rest_users_server.php
<IfModule mod_rewrite.c>
  Options -MultiViews
  RewriteEngine On
  RewriteCond %{REQUEST_FILENAME} !-f
  RewriteRule ^rest/(.*)$ rest_users_server.php [QSA,L]
</IfModule>
```

© 2019-2020 Madani Boukebeche

REST SANS FRAMEWORK

⇒ Une 2^{ème} version : avec réécriture d'url selon REST

```

<?php // rest_user_server
require_once("dao_user.php");
define ("REST_URI_BASE", "/exple_ws_php/rest/");
header('Content-Type: application/json');
function getUser($idUser=null){
    $resultat = ($idUser != null)? DaoUser::find($idUser):DaoUser::findAll();
    $retour["success"] = true;
    $retour["results"]["nb"] = count($resultat);
    $retour["results"]["users"] = $resultat;
    return json_encode($retour);
}
// Obtenir la query string (sans serveur) : "/exple_ws_php/rest/users/..."
$uri = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
// retirer la partie du chemin définie dans "REST_URI_BASE"
$uriRessource = str_replace(REST_URI_BASE, "", $uri);
if (!empty($uriRessource)){
    $operation = $_SERVER['REQUEST_METHOD'];
    // Scinder le reste de l'uri users/....
    $params = explode("/", $uriRessource);
    if (isset($params[0])){
        if ($params[0] == "users"){// Entité Users
            $id = (isset($params[1]))?$params[1]:null;
            if ($operation=="GET"){echo getUser($id);}
        }
    }
    elseif ($ressource == "autre"){/*echo getAutre ...;*/
    }
}

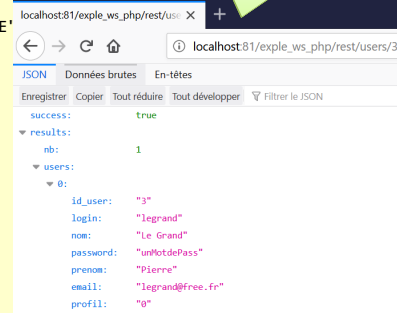
```

Chemin relatif pour les scripts REST

localhost81/exple_ws_php/rest/users

localhost81/exple_ws_php/rest/users/3

Tester avec un navigateur



```

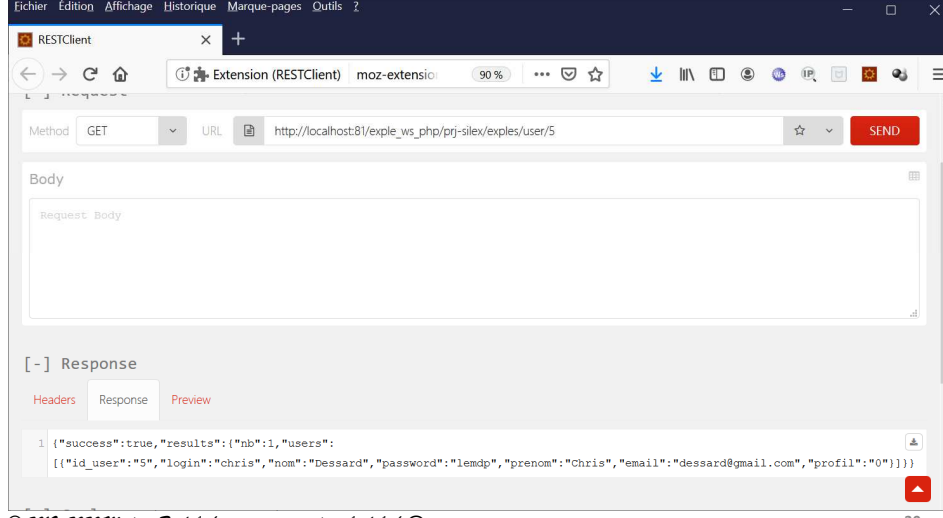
{
  "success": true,
  "results": {
    "nb": 1,
    "users": [
      {
        "id_user": "3",
        "login": "Le Grand",
        "nom": "Le Grand",
        "password": "unMotdePass",
        "prenom": "Pierre",
        "email": "legrand@free.fr",
        "profil": "0"
      }
    ]
  }
}

```

REST SANS FRAMEWORK

⇒ Une 2^{ème} version : avec réécriture d'url selon REST

⇒ Tester depuis le navigateur en utilisant le plugin RestClient



Method: GET URL: http://localhost81/exple_ws_php/prj-silex/exples/user/5

Body: Request Body

[-] Response

Headers Response Preview

```

1 {
  "success": true,
  "results": {
    "nb": 1,
    "users": [
      {
        "id_user": "5",
        "login": "chris",
        "nom": "Dessard",
        "password": "lemdp",
        "prenom": "Chris",
        "email": "dessard@gmail.com",
        "profil": "0"
      }
    ]
  }
}

```

© 2019-2020 Madani Boukebeche madani.boukebeche@ynov.com 30

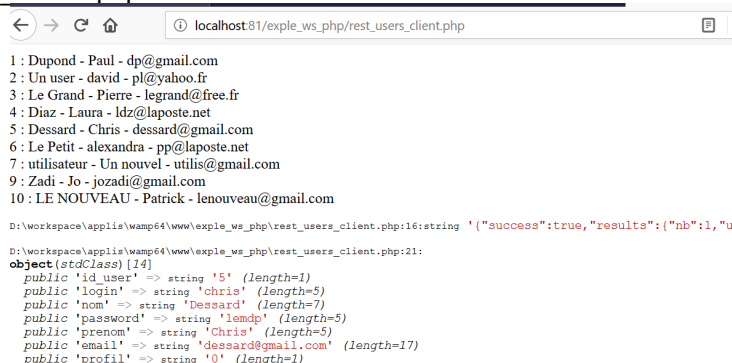
REST SANS FRAMEWORK

⇒ Une 2^{ème} version : avec réécriture d'url selon REST

⇒ Exercice 1 :

⇒ Faire une copie de rest_users_client_v0.php (renommer la copie en rest_users_client.php)

⇒ Adapter le client rest_users_client.php pour appeler rest_users_server.php



```

1 : Dupond - Paul - dp@gmail.com
2 : Un user - david - pl@yahoo.fr
3 : Le Grand - Pierre - legrand@free.fr
4 : Diaz - Laura - ldz@laposte.net
5 : Dessard - Chris - dessard@gmail.com
6 : Le Petit - alexandra - pp@laposte.net
7 : utilisateur - Un nouvel - utilis@gmail.com
9 : Zadi - Jo - jozadi@gmail.com
10 : LE NOUVEAU - Patrick - lenouveau@gmail.com

D:\workspace\apllis\wamp64\www\exple_ws_php\rest_users_client.php:16:string '{"success":true,"results":{"nb":1,"u:
D:\workspace\apllis\wamp64\www\exple_ws_php\rest_users_client.php:21:
object(stdClass) [14]
  public 'id_user' => string '5' (length=1)
  public 'login' => string 'chris' (length=5)
  public 'nom' => string 'Dessard' (length=7)
  public 'password' => string 'lemdp' (length=5)
  public 'prenom' => string 'Chris' (length=5)
  public 'email' => string 'dessard@gmail.com' (length=17)
  public 'profil' => string '0' (length=1)
  
```

© 2019-2020 Madani Boukebeche Dessard - Chris - dessard@gmail.com

31

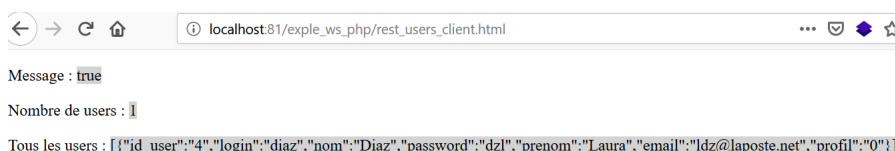
REST SANS FRAMEWORK

⇒ Une 2^{ème} version : avec réécriture d'url selon REST

⇒ Exercice 2 :

⇒ Faire une copie de rest_users_client_v0.html (renommer la copie en rest_users_client.html)

⇒ Adapter le client rest_users_client.html pour appeler rest_users_server.php



```

Message : true
Nombre de users : 1
Tous les users : [{"id_user":"4","login":"diaz","nom":"Diaz","password":"dzl","prenom":"Laura","email":"ldz@laposte.net","profil":"0"}]
  
```

© 2019-2020 Madani Boukebeche

madani.boukebeche@ynov.com

32

32

REST SANS FRAMEWORK

⇒ Une 2^{ème} version : avec réécriture d'url selon REST

⇒ **Exercice 3** : Développez l'exemple précédent, en lui rajoutant la possibilité de modifier, d'ajouter et de supprimer un utilisateur.

⇒ Indication 1 : Déterminer la commande HTTP (méthode d'envoi de requête)

GET, POST, PUT, DELETE : `$operation = $_SERVER['REQUEST_METHOD'];`

⇒ Indication 2: Récupérer un paramètre HTTP JSON

```
if ($params[0] == "users"){
    switch ($operation){
        case "GET": echo getUser($id); break;
        case "POST":
            // Takes raw data from the request
            $json = file_get_contents('php://input');
            // Converts it into a PHP object
            $data = json_decode($json);
            if (isset($data->id_user) && !empty($data->id_user)){
                echo getError("unexpected id " . $data->id_user);
                exit;
            }
            $data = get_object_vars($data);
            echo addUser($data);
            break;
    }
}
```

⇒ Indication 3 : Voir dao_user.php

REST AVEC LE FRAMEWORK SILEX/SYMFONY

⇒ Une 3^{ème} version

⇒ Pour installer Silex, suivez le tutoriel :

⇒ <https://openclassrooms.com/fr/courses/2260876-premiers-nas-avec-le-framework-php-silex>

⇒ Après installation, créer un dossier « prj-silex »

⇒ Initialiser un projet silex dedans

⇒ Puis créer un dossier exples dans lequel on déposera les scripts REST et le fichier .htaccess

www > exple_ws_php > prj-silex > exples

Nom

- .htaccess
- index.php
- user.php

www > exple_ws_php > prj-silex

Nom

- exples
- vendor
- composer.json
- composer.lock

```
# Redirect incoming URLs to appropriate URI
<IfModule mod_rewrite.c>
    Options -MultiViews
    RewriteEngine On

    RewriteCond %{REQUEST_FILENAME} !-f
    RewriteRule ^user rest_user_server_v2.php [QSA,L]

    RewriteRule ^$ index.php [QSA,L]
    RewriteRule ^hello index.php [QSA,L]
</IfModule>
```

REST AVEC LE FRAMEWORK SILEX/SYMFONY

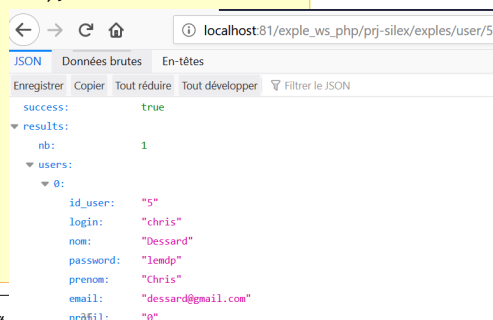
⇒ Une 3ème version

```
<?php // rest_users_server_v2.php
require_once __DIR__.'../vendor/autoload.php';
require_once __DIR__.'../dao_user.php';

$app = new Silex\Application();
function toJson($resultat, $statusCode=200){
    global $app;
    $retour["success"] = true;
    $retour["results"]["nb"] = count($resultat);
    $retour["results"]["users"] = $resultat;
    return $app->json($retour, $statusCode);
}

$app->get('/users', function () {
    $resultat = DaoUser::findAll();
    return toJson($resultat);
});

$app->get('/users/{id}', function ($id) {
    $resultat = DaoUser::find($id);
    return toJson($resultat);
});
$app->run();
```



© 2019-2020 Madani Boukebeche

madani.boukebeche@ynov.

REST AVEC LE FRAMEWORK SILEX/SYMFONY

⇒ Les autres opérations CUD

⇒ Insertion

⇒ Ajout du traitement du POST

```
use Symfony\Component\HttpFoundation\Request;

$app->post('/users', function (Request $request) {
    if (0 === strpos($request->headers->get('Content-Type'), 'application/json')) {
        $data = json_decode($request->getContent(), true);
        $request->request->replace(is_array($data) ? $data : array());
        $newUser = DaoUser::add($data);
        return toJson( $newUser, 201);
    }
});
```

© 2019-2020 Madani Boukebeche

madani.boukebeche@ynov.com

36

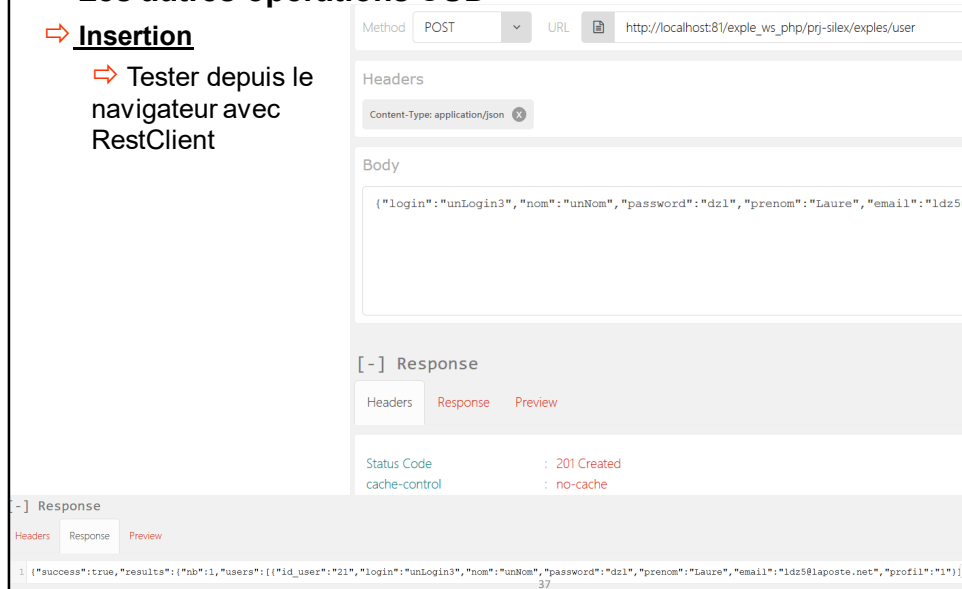
36

REST AVEC LE FRAMEWORK SILEX/SYMFONY

⇒ Les autres opérations CUD

⇒ Insertion

- ⇒ Tester depuis le navigateur avec RestClient



REST AVEC LE FRAMEWORK SILEX/SYMFONY

⇒ Les autres opérations CUD

⇒ Exercices

- ⇒ Mettre en œuvre les 2 opérations restantes :
 - ⇒ Update (PUT)
 - ⇒ Delete

PROJET API REST PHP

- ⇒ Réaliser une application web permettant de gérer les produits :
 - ⇒ Une API Rest basée sur PHP gérant les CRUD des produits
 - ⇒ /produits
 - ⇒ /produits/{id}
 - ⇒ Un front basé sur PHP et Javascript se servant de l'API Rest au moyen d'Ajax
 - ⇒ La page front affichera la liste des produits : *produits.html*
 - ⇒ Un lien hypertexte sur le « nom » et la « référence » de chaque produit permettra d'accéder aux détails d'un produit sur la même page *produits.html* (via Ajax)
 - a) Les détails d'un produit sont affichées dans un formulaire permettant la modification : bouton « Modifier » et bouton « Annuler ».
 - ⇒ Le bouton « Modifier » envoie une requête Ajax pour la création d'un nouveau produit, masquera le formulaire et actualisera la liste des produits.
 - ⇒ Le bouton « Annuler » masquera le formulaire
 - ⇒ Un bouton « supprimer » pour chaque produit avec une demande de confirmation de suppression. La suppression se fera aussi au moyen de Ajax et actualisera la liste des produits
 - ⇒ Un bouton « ajouter » permettra d'afficher un formulaire (le même que celui utilisé pour les détails de produit en a), avec un bouton « Créer » et un bouton « Annuler ».
 - ⇒ Le bouton « Créer » permettra de créer un nouveau produit, masquer le formulaire et actualiser la liste des produits
 - ⇒ Le bouton « Annuler » permettra de masquer le formulaire

Bonus :

Client en JavaScript avec Angular (Ng-Client)

```
<!DOCTYPE html>
<html>
<meta charset="UTF-8">
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js">
</script>
<body ng-app="myApp">
  <div ng-controller="getUsersCtrl">
    Filter : <input type="text" ng-model="myFilter">
    <table border=1>
    <tr> <th>Id</th> <th>Nom</th> <th>Prénom</th></tr>
    <tr ng-repeat="usr in users | filter : myFilter">
      <td>{{ usr.id_user }}</td> <td>{{ usr.nom }}</td> <td>{{ usr.prenom }}</td>
    </tr>
    </table>
  </div>
<script>
  angular.module('myApp', [])
    .controller('getUsersCtrl',
      function($scope, $http) {
        $http.get("rest/users").then(function (response) {
          $scope.users = response.data.results.users;
          //console.log($scope.users);
        });
      });
</script>
</body>
</html>
```