

SERVICES WEB REST

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, Lavarel, 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",
        ...
      }
    ]
  }
}
```

© 2019-2020 Madani Boukebeche

madani.boukebeche@ynov.com

24

24

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@ynov.com

25

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>
```

© 20

26

REST SANS FRAMEWORK

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

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": "1"}, {"id_user": "3", "login": "legrand", "nom": "Le Grand", "password": "unMotdePass", "prenom": "Pierre", "email": "legrand@free.fr", "profil": "1"}, {"id_user": "4", "login": "diaz", "nom": "Diaz", "password": "dzl", "prenom": "Laura", "email": "laura@gmail.com", "profil": "1"}, {"id_user": "5", "login": "chris", "nom": "Dessard", "password": "lemdp", "prenom": "Chris", "email": "chris@gmail.com", "profil": "1"}, {"id_user": "6", "login": "pif", "nom": "Le Petit", "password": "fip", "prenom": "alexandra", "email": "alexandra@gmail.com", "profil": "1"}, {"id_user": "7", "login": "utilis", "nom": "utilisateur", "password": "abc", "prenom": "Un nouvel", "email": "utilis@gmail.com", "profil": "0"}, {"id_user": "8", "login": "zj", "nom": "Zadi", "password": "jo", "prenom": "Jo", "email": "jozad@gmail.com", "profil": "1"}, {"id_user": "9", "login": "Lenouveau", "nom": "LE NOUVEAU", "password": "mdp", "prenom": "Patrick", "email": "lenouveau@gmail.com", "profil": "1"}, {"id_user": "10", "login": "Lenouveau", "nom": "LE NOUVEAU", "password": "mdp", "prenom": "Patrick", "email": "lenouveau@gmail.com", "profil": "1"}]
```

© 2019-2020 Madani Boukebeche

madi

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/

The screenshot shows a web browser at `localhost:81/exple_ws_php/rest/user/7`. The JSON response is:

```

{
  "success": true,
  "results": {
    "nb": 1,
    "users": [
      {
        "id_user": "7",
        "login": "utilis",
        "nom": "utilisateur",
        "password": "abc",
        "prenom": "ut",
        "email": "ut",
        "profil": "0"
      }
    ]
  }
}

```

Below the JSON, a snippet of the `.htaccess` file is shown:

```

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

```

A green callout bubble points to the `.htaccess` file with the text: "Redirection au niveau de .htaccess".

© 2019-2020 Madani Bouba

REST SANS FRAMEWORK

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

The screenshot shows PHP code for a REST API server:

```

<?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/..."
$url = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
// retirer la partie du chemin définie dans "REST_URI_BASE"
$urlRessource = str_replace(REST_URI_BASE, "", $url);
if (!empty($urlRessource)){
    $operation = $_SERVER['REQUEST_METHOD'];
    // Scinder le reste de l'uri users/....
    $params = explode("/", $urlRessource);
    if (isset($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 ...;*/
    }
}
}

```

A blue callout bubble points to the `REST_URI_BASE` definition with the text: "Chemin relatif pour les scripts REST".

The browser shows the response for `localhost:81/exple_ws_php/rest/users/3`:

```

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

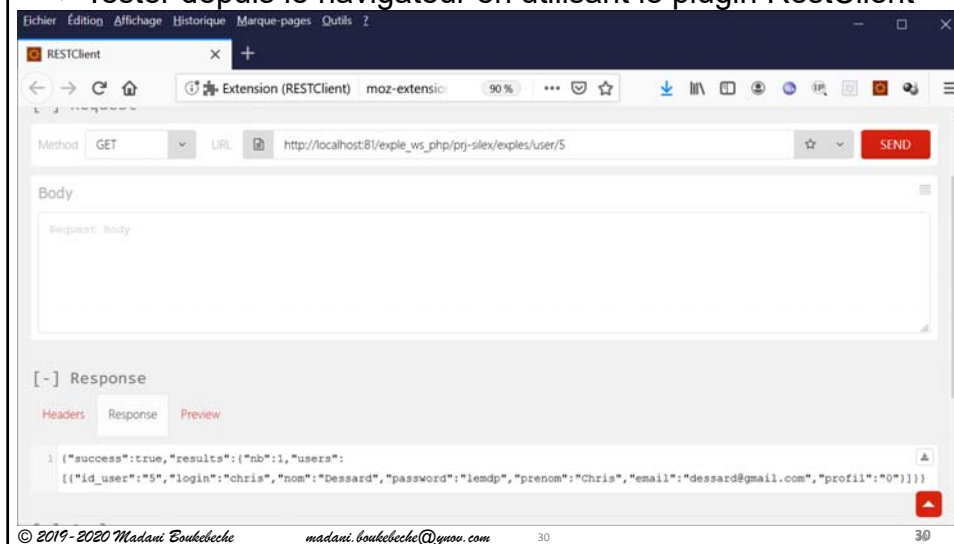
```

A green callout bubble points to the browser with the text: "Tester avec un navigateur".

29

REST SANS FRAMEWORK

- ⇒ Une 2^{ème} version : avec réécriture d'url selon REST
- ⇒ Tester depuis le navigateur en utilisant le plugin RestClient



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

```

localhost/exple_ws_php/rest_users_client.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\appls\wamp64\www\exple_ws_php\rest_users_client.php:14:string '{"success":true,"results":{"nb":1,"u
D:\workspace\appls\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)

```

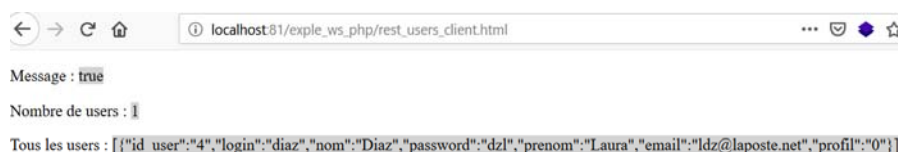
© 2019-2020 Madani Boukebeche Dessard - Chris - dessard@gmail.com

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



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-pas-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>
```

© 2019-2020 Madani Boukebeche

madani.boukebeche@ynov.com

34

34

REST AVEC LE FRAMEWORK SILEX/SYMFONY

⇒ Une 3^{ème} 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, $httpCode=200){
    global $app;
    $retour["success"] = true;
    $retour["results"]["nb"] = count($resultat);
    $retour["results"]["users"] = $resultat;
    return $app->json($retour, $httpCode);
}

$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();
```

localhost:81/exple_ws_php/prj-silex/exples/user/5

JSON Données brutes En-têtes

Enregistrer Copier Tout réduire Tout développer Filtre le JSON

```
success: true
results:
  nb: 1
  users:
    0:
      id_user: "5"
      login: "chris"
      nom: "Dessard"
      password: "lmdp"
      prenom: "Chris"
      email: "dessard@gmail.com"
      pre5il: "q"
```

© 2019-2020 Madani Boukebeche

madani.boukebeche@ynov.com

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);
    }
});
```

REST AVEC LE FRAMEWORK SILEX/SYMFONY

⇒ Les autres opérations CUD

⇒ Insertion

⇒ Tester depuis le navigateur avec RestClient

The screenshot shows the RestClient interface with the following details:

- Method:** POST
- URL:** http://localhost:81/exple_ws_php/prj-silex/exples/user
- Headers:** Content-Type: application/json
- Body:** {"login": "unLogin3", "nom": "unNom", "password": "dz1", "prenom": "Laure", "email": "ldz5@laposte.net"}
- Response:**
 - Status Code:** 201 Created
 - cache-control:** no-cache
- [-] Response:** {"success": true, "results": [{"nb": 1, "users": [{"id_user": "21", "login": "unLogin3", "nom": "unNom", "password": "dz1", "prenom": "Laure", "email": "ldz5@laposte.net", "profil": "1"}]}

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 permettra 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>
```