

Chaker ALLAOUI
chaker.allaoui@gmail.com

WEBSERVICE API REST

SYMFONY 2

TABLE DES MATIERES

Contenu

Présentation	1
Technologies	2
Installation des bundles	3
Configuration des bundles	4
Configuration de Symfony	5
Implémentation de GET	6
Implémentation de POST	7
Implémentation de PUT	8
Configuration de la sérialisation	9
Le plan d'adressage	12
API REST Method GET	13
API REST Method POST	14
API REST Method PUT	15
Liens utiles	16

Présentation

Les WebServices sont un moyen rapide de distribution de l'information entre clients, fournisseurs, partenaires commerciaux et leurs différentes plates-formes. Les services Web sont basés sur le modèle SOA. Un Webservice est un composant logiciel identifié par une URI, dont les interfaces publiques sont définies et appelées en XML. Sa définition peut être découverte par d'autres systèmes logiciels. Les WebServices peuvent interagir entre eux d'une manière prescrite par leurs définitions, en utilisant des messages XML portés par les protocoles Internet.

Les WebServices fournissent un lien entre applications. Ainsi, des applications utilisant des technologies différentes peuvent envoyer et recevoir des données au travers de protocoles compréhensibles par tout le monde. Les WebServices sont normalisés car ils utilisent les standards XML et HTTP pour transférer des données et ils sont compatibles avec de nombreux autres environnements de développement. Ils sont donc indépendants des plates-formes.

C'est dans ce contexte qu'un intérêt très particulier a été attribué à la conception des WebServices puisqu'ils permettent aux entreprises d'offrir des applications accessibles à distance par d'autres entreprises. Cela s'explique par le fait que les WebServices n'imposent pas de modèles de programmation spécifiques. En d'autres termes, les WebServices ne sont pas concernés par la façon dont les messages sont produits ou consommés par des programmes. Cela permet aux vendeurs d'outils de développement d'offrir différentes méthodes et interfaces de programmation au-dessus de n'importe quel langage de programmation, sans être contraints par des standards. Ainsi, les fournisseurs d'outils de développement peuvent facilement différencier leurs produits avec ceux de leurs concurrents en offrant différents niveaux de sophistication.

Les WebServices représentent donc la façon la plus efficace de partager des méthodes et des fonctionnalités. De plus, ils réduisent le temps de réalisation en permettant de tirer directement parti de services existants.

Technologies

SYMFONY

Symfony est un framework MVC libre écrit en PHP 5 destiné majoritairement aux professionnels du développement. Il fournit des fonctionnalités modulables et adaptables qui permettent de faciliter et d'accélérer le développement d'un site web.

Version d'implémentation: 2.6.7

FOSRESTBUNDLE

FOSRestBundle est un bundle qui fournit plusieurs outils pour aider dans la construction d'une API REST, il permet : la manipulation de la couche View, génération automatique des routes pour une API REST, il supporte les Listener et ExceptionController.

Version d'implémentation : dev-master

JMSERIALIZERBUNDLE

ce bundle intègre la bibliothèque de serializer dans Symfony2 et permet d'adapter vos données dans un format comme JSON, XML, ou YAML.

Version d'implémentation: dev-master

RESTCLIENT

RESTClient est un module de Firefox qui supporte toutes les méthodes HTTP RFC2616 (HTTP/1.1).

Version d'implémentation : 2.0.3

CURL

cURL est un outil gratuit en ligne de commande qui permet de télécharger et d'envoyer des données sur divers protocoles: DICT, FILE, FTP, FTPS, Gopher, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, POP3, POP3S, RTMP, RTSP, SCP, SFTP, SMTP, SMTPS, Telnet et TFTP.

Version d'implémentation : 7.42.1

Installation des bundles

COMPOSER.JSON

Ajouter dans votre composer.json dans l'attribut require la déclaration de deux bundles, comme suit :

```
"friendsofsymfony/rest-bundle": "dev-master",  
"jms/serializer-bundle": "dev-master"
```

Et lancez la commande « **php composer.phar update** » pour mettre à jour vos vendors.

APPKERNEL.PHP

Déclarer vos bundles dans le fichier AppKernel.php de votre dossier app, comme suit :

```
public function registerBundles()  
{  
    $bundles = array(  
        // ...  
        new FOS\RestBundle\FOSRestBundle(),  
        new JMS\SerializerBundle\JMSSerializerBundle(),  
        // ...  
    );  
}
```

Configuration des bundles

FOSRESTBUNDLE

Dans le fichier « app/config/config.yml » vous devez configurer ce bundle comme suit :

```
#app/config/config.yml
fos_rest:
  param_fetcher_listener: true
  body_listener: true
  format_listener: true
  view:
    view_response_listener: 'force'
  formats:
    xml: true
    json : true
  templating_formats:
    html: true
  force_redirects:
    html: true
  failed_validation: HTTP_BAD_REQUEST
  default_engine: twig
  routing_loader:
  default_format: json
```

JMSERIALIZERBUNDLE

Il est possible de configurer les propriétés de la sérialisation de chaque « Modèle » suivant le mapping suivant :

- @ExclusionPolicy("all") : Chaque propriété de votre entité sera ignoré lors de la sérialisation.

@Expose : Cette propriété sera sérialisée.

@VirtualProperty : Cette méthode sera appelée et sérialisée comme propriété virtuelle.

Configuration de Symfony

CONFIGURATION DU FICHIER « ROUTING.YML »

```
#app/config/routing.yml
webservice_api:
  resource: "@YourBundle/Resources/config/routing.yml"
  prefix: /
  rest :
  type : rest
  resource : "routing_rest.yml"
  prefix : /api
```

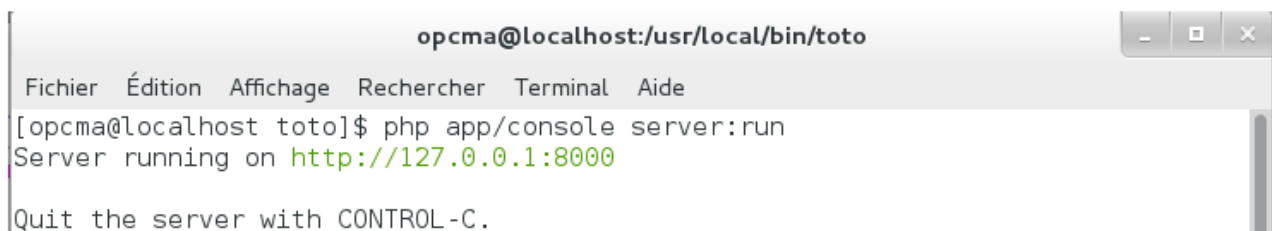
CREATION-CONFIGURATION DU FICHIER « ROUTING_REST.YML »

```
#app/config/routing_rest.yml
users :
  type: rest
  resource: "YourBundle:UserRest"
  name_prefix: api_
```

CREATION-CONFIGURATION DU FICHIER « ROUTING_REST.YML » DANS LE DOSSIER « RESOURCES » DE VOTRE BUNDLE

```
#src/YourBundle/Resources/config/routing_rest.yml
Rest_User :
  type : rest
  resource: "@YourBundle/Resources/config/routing_rest.yml"
```

VERIFICATION DE L'ETAT DU SERVEUR



```
opcma@localhost:usr/local/bin/toto
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
[opcma@localhost toto]$ php app/console server:run
Server running on http://127.0.0.1:8000

Quit the server with CONTROL-C.
```

Implémentation de GET

Après avoir créé votre bundle de Webservice, vous devez déclarer vos méthodes pour gérer votre API REST, la première est GET, celle qui permet de récupérer un utilisateur à partir de votre base de données, donc sous le dossier Controller et dans la classe UserRestController vous la définissez comme suivant, ainsi que sa route qui sera automatiquement interprétée par FOSRestBundle :

```
/**
 * @Route("/getUser/{username}")
 */
public function getUserAction($username) {
    $user = $this->getDoctrine()->getRepository('YourBundle:Users')
->findOneByUsername($username);
    if(!is_object($user)) {
        throw $this->createNotFoundException();
    }
    return New JsonResponse(array('Hello ' => $user->getUsername()
, 'Your email is ' => $user->getEmail(), 'Your phonenumber is ' => $user
->getPhonenumber()
, 'You have registred since ' => $user->getRegistredOn()
));
}
```

Information : FOSRestBundle interprète la méthode « **getUserAction** » pour définir le type de la requête HTTP, **GET** dans le présent cas.

Implémentation de POST

```
/**
 * @Route("/addUser")
 */
public function postAddUserAction(Request $request)
{
    if (0 === strpos($request->headers->get('Content-Type'), 'application/json'))
    {
        $res = $request->getContent();
        $data = json_decode($res, true);
        $profil = new Users();
        $profil->setUsername($data['username']);
        $profil->setPassword($data['password']);
        $profil->setEmail($data['email']);
        $profil->setRegistredOn(new \Datetime());
        $profil->setPhonenumber($data['phonenumber']);
        $em = $this->getDoctrine()->getManager();
        $em->persist($profil);
        $em->flush();
        $exist = $this->getDoctrine()->getRepository('YourBundle:Users')
        ->findOneByUsername($data['username']);
        if($exist !=null){
            return New JsonResponse(array('Results'=>$data));
        }
        else{
            return New JsonResponse(array('Errors' => 'erreur'));
        }
    }
    else
    {
        return New JsonResponse(array("error" => "JSON syntax require"));
    }
}
```

Information : FOSRestBundle interpète la méthode « **postAddUserAction** » pour définir le type de la requête HTTP, **POST** dans le présent cas.

Implémentation de PUT

```
/**
 * @Route("/updateUser/{username}")
 */
public function putUpdateUserAction(Request $request){
    if (0 === strpos($request->headers->get('Content-Type'),'application/json'))
    {
        $res = $request->getContent();
        $data = json_decode($res, true);
        $user = $this->getDoctrine()->getRepository('YourBundle:Users')->findOneByUsername($data['username']);
        $user->setPassword($data['password']);
        $user->setEmail($data['email']);
        $user->setRegistredOn(new \Datetime());
        $user->setPhonenumber($data['phonenumber']);
        $em = $this->getDoctrine()->getManager();
        $em->persist($user);
        $em->flush();
        $exist = $this->getDoctrine()->getRepository('YourBundle:Users')->findOneByUsername($data['username']);
        if($exist !=null){
            return New JsonResponse(array('Results'=>$data));
        }
        else{
            return New JsonResponse(array('Errors' => 'erreur'));
        }
    }
    else
    {
        return New JsonResponse(array("error" => "JSON syntax require"));
    }
}
```

Information : FOSRestBundle interprète la méthode « **putUpdateUserAction** » pour définir le type de la requête HTTP, **PUT** dans le présent cas.

Configuration de la sérialisation

JMSerializerBundle permet d'exposer nos méthodes à la sérialisation via des différents formats, ainsi, vous devez exposer vos méthodes qui retourne une valeur à la sérialisation dans votre « Modèle » et seulement ces méthodes, la démarche est la suivante :

```
use FOS\UserBundle\Entity\User as BaseUser;
use Doctrine\ORM\Mapping as ORMU;
use JMS\Serializer\Annotation\ExclusionPolicy;
use JMS\Serializer\Annotation\Expose;
use JMS\Serializer\Annotation\Groups;
use JMS\Serializer\Annotation\VirtualProperty;
/**
 * Users
 *
 * @ORMU\Table()
 *
 * @ORMU\Entity(repositoryClass="path_to\YourBundle\Entity\UsersRepository")
 *
 * @ExclusionPolicy("all")
 */
class Users
{
    /**
     * @var integer
     *
     * @ORMU\Column(name="id", type="integer")
     * @ORMU\Id
     * @ORMU\GeneratedValue(strategy="AUTO")
     * @Expose
     */
    private $id;
    /**
     * @var string
     *
     * @ORMU\Column(name="username", type="string", length=255)
     * @Expose
     */
    private $username;
    /**
     * @var string
     *
     * @ORMU\Column(name="password", type="string", length=255)
     * @Expose
     */
}
```

```

*/
private $password;
/**
 * @var string
 *
 * @ORMU\Column(name="email", type="string", length=255)
 * @Expose
 */
private $email;
/**
 * @var \DateTime
 *
 * @ORMU\Column(name="registred_on", type="datetime")
 * @Expose
 */
private $registredOn;
/**
 * @var string
 *
 * @ORMU\Column(name="phonenumber", type="string", length=255)
 * @Expose
 */
private $phonenumber;
//...
/**
 * Get username
 *
 * @return string
 * @VirtualProperty
 */
public function getUsername()
{
    return $this->username;
}
//...
/**
 * Get password
 *
 * @return string
 * @VirtualProperty
 */
public function getPassword()
{
    return $this->password;
}
//...
/**

```

```
* Get email
*
* @return string
* @VirtualProperty
*/
public function getEmail()
{
return $this->email;
}
//...
/**
* Get registredOn
*
* @return \DateTime
* @VirtualProperty
*/
public function getRegistredOn()
{
return $this->registredOn;
}
//...
/**
* Get registerOn
*
* @return \DateTime
* @VirtualProperty
*/
public function getRegisterOn()
{
return $this->registerOn;
}
//...
/**
* Get phonenumber
*
* @return string
* @VirtualProperty
*/
public function getPhonenumber()
{
return $this->phonenumber;
}
```

Le plan d'adressage

PRESENTATION

Les routes correspondantes à nos méthodes GET, POST et PUT sont interpréter automatiquement par FOSRestBundle, ainsi nos requêtes sont acheminées via des entêtes correspondantes à chaque HTTP Headers.

COMMANDE DE DEBUG DES ROUTES

php app/console router:debug | grep api_

```
opcma@localhost:/usr/local/bin/toto
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
[opcma@localhost toto]$ php app/console router:debug | grep api_
api_get_user      GET      ANY      ANY      /api/getUser/{username}.{_format}
api_post_add_user POST     ANY      ANY      /api/addUser.{_format}
api_put_update_user PUT      ANY      ANY      /api/updateUser/{username}.{_format}
[opcma@localhost toto]$
```

API REST Method GET

PRESENTATION

La méthode GET retourne une représentation au format JSON de notre utilisateur présent dans votre base de données, dans le cas contraire la méthode retourne un message d'erreur.

COMMANDE LINUX

```
curl -i http://localhost:8000/api/getUser/{username}
```



```
opcma@localhost:usr/local/bin/toto
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
[opcma@localhost toto]$ curl -i http://localhost:8000/api/getUser/chaker
HTTP/1.1 200 OK
Host: localhost:8000
Connection: close
X-Powered-By: PHP/5.4.16
Cache-Control: no-cache
Date: Fri, 15 May 2015 14:06:55 GMT
Content-Type: application/json
X-Debug-Token: 2f7d18
X-Debug-Token-Link: /_profiler/2f7d18

{"Hello ":"chaker","Your email is ":"chaker.allaoui@gmail.com","Your phonenumber is ":"71001001","Your have registred since ":{ "date":"2015-05-14 16:50:40","timezone_type":3,"timezone":"Europe\\Paris"}}[opcma@localhost toto]$
```

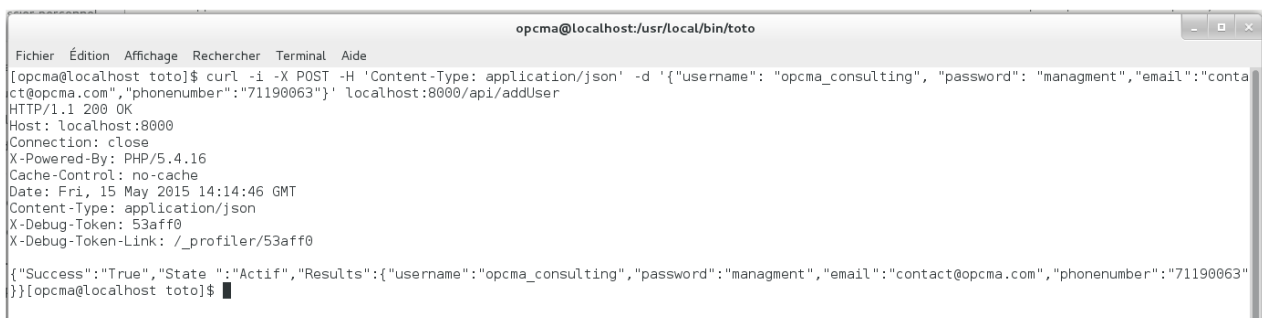
API REST Method POST

PRESENTATION

La méthode POST permet d'ajouter un utilisateur dans votre base de données, dans le cas contraire cette méthode retourne une erreur.

COMMANDE LINUX

```
curl -i -X POST -H 'Content-Type:application/json' -d  
'{"username":"user","password":"pwd","email":"email@example.com","phonenumber":"11111111"}'  
http://localhost:8000/api/addUser/
```



```
opcma@localhost:usr/local/bin/toto  
Fichier Édition Affichage Rechercher Terminal Aide  
[opcma@localhost toto]$ curl -i -X POST -H 'Content-Type: application/json' -d '{"username": "opcma_consulting", "password": "managment", "email": "contact@opcma.com", "phonenumber": "71190063"}' localhost:8000/api/addUser  
HTTP/1.1 200 OK  
Host: localhost:8000  
Connection: close  
X-Powered-By: PHP/5.4.16  
Cache-Control: no-cache  
Date: Fri, 15 May 2015 14:14:46 GMT  
Content-Type: application/json  
X-Debug-Token: 53aff0  
X-Debug-Token-Link: /_profiler/53aff0  
{ "Success": "True", "State": "Actif", "Results": { "username": "opcma_consulting", "password": "managment", "email": "contact@opcma.com", "phonenumber": "71190063" } }  
[opcma@localhost toto]$
```

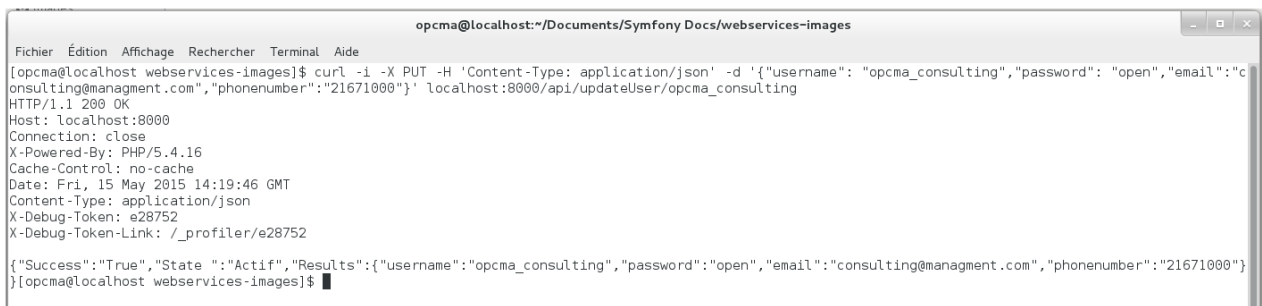

API REST Method PUT

PRESENTATION

La méthode PUT permet de modifier les paramètres et coordonnées d'utilisateur présent dans votre base de données, dans le cas contraire cette méthode retourne une erreur.

COMMANDE LINUX

```
curl -i -X PUT -H 'Content-Type:application/json' -d '{"username":"user","password":"new_pwd","email":"new_email@example.com","phoneNumber":"22222222"}' http://localhost:8000/api/updateUser/{username}
```



```
opcma@localhost:~/Documents/Symfony Docs/webservices-images
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
[opcma@localhost webservices-images]$ curl -i -X PUT -H 'Content-Type: application/json' -d '{"username": "opcma_consulting","password": "open","email": "consulting@managment.com","phoneNumber": "21671000"}' localhost:8000/api/updateUser/opcma_consulting
HTTP/1.1 200 OK
Host: localhost:8000
Connection: close
X-Powered-By: PHP/5.4.16
Cache-Control: no-cache
Date: Fri, 15 May 2015 14:19:46 GMT
Content-Type: application/json
X-Debug-Token: e28752
X-Debug-Token-Link: /_profiler/e28752
{"Success": "True", "State": "Actif", "Results": {"username": "opcma_consulting", "password": "open", "email": "consulting@managment.com", "phoneNumber": "21671000"}}
[opcma@localhost webservices-images]$
```



Liens utiles

[SYMFONY](#)

<https://symfony.com/>

[FOSRESTBUNDLE](#)

<https://github.com/FriendsOfSymfony/FOSRestBundle>

[JMSSERIALIZERBUNDLE](#)

<https://github.com/schmittjoh/JMSSerializerBundle>

[RESTCLIENT](#)

<https://addons.mozilla.org/fr/firefox/addon/restclient/>

[CURL](#)

http://linux.about.com/od/commands/l/blcmdl1_curl.htm