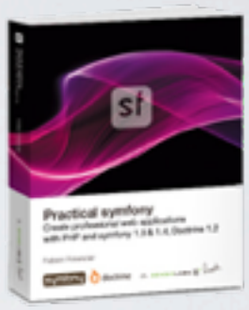
Exposer des services web SOAP et REST

RMLL – 6 juillet 2010

Qui suis-je ?

- **Hugo HAMON (@hhamon)**
- **Responsable des formations** chez **Sensio Labs**
- Coauteur et contributeur d'ouvrages
- Secrétaire Général de l'AFUP
- Webmaster du site Apprendre-PHP.com
- http://www.hugohamon.com



Introduction aux Web services

Les services web

- Programmes destinés à échanger des données entre systèmes hétérogènes
- Basés sur des standards (SOAP) ou des styles d'architecture (REST)
- Fonctionnent en mode Client / Serveur (requête / réponse)
- Echange de données préformattées (XML, JSON, YAML, HTML...)
- Faible couplage grâce aux standards ouverts comme XML
- Faciles à mettre en oeuvre

Architecture REST

REST c'est quoi ?

- REpresentational State Transfert
- Concept inventé par Roy Fielding en 2000
- REST est un style d'architecture !
- REST s'appuie sur les concepts suivants :
  - Le protocole HTTP (requête / réponse)
  - La notion d'URI qui identifie chaque ressource
  - Les méthodes HTTP (GET, POST, PUT, DELETE, HEAD)
  - Les codes de statut (200, 201, 403, 404, 500...)
  - Les formats ouverts (XML, JSON, HTML...)
- REST est simple à mettre en oeuvre

La notion d'URI dans REST

- Chaque URI identifie une ressource

`http://api.domain.com/users/hugo`

- Une URI ne contient pas de verbe !

- Une même URI peut être appelée avec différentes méthodes HTTP

`GET http://api.domain.com/users` — Récupère les utilisateurs

`POST http://api.domain.com/users` — Crée un nouvel utilisateur

Les méthodes HTTP

- GET, POST, PUT, DELETE et HEAD
- Identifient les actions réalisables sur la ressource

`GET http://api.domain.com/users` — Récupère les utilisateurs

`POST http://api.domain.com/users` — Crée un nouvel utilisateur

`PUT http://api.domain.com/users/hugo` — Modifie hugo

`DELETE http://api.domain.com/users/hugo` — Supprime hugo

- Opérations CRUD sur les ressources

Les codes HTTP de retour

- Indiquent l'état (succès ou échec) de l'opération effectuée

Méthodes	En cas de succès	En cas d'échec
POST	201 Created Redirection vers l'URI de l'objet	400 Bad Request
PUT	200 OK	400 Bad Request 404 Not Found 409 Conflict
DELETE	200 OK	400 Bad Request 404 Not Found 410 Gone
GET	200 OK	400 Bad Request 404 Not Found

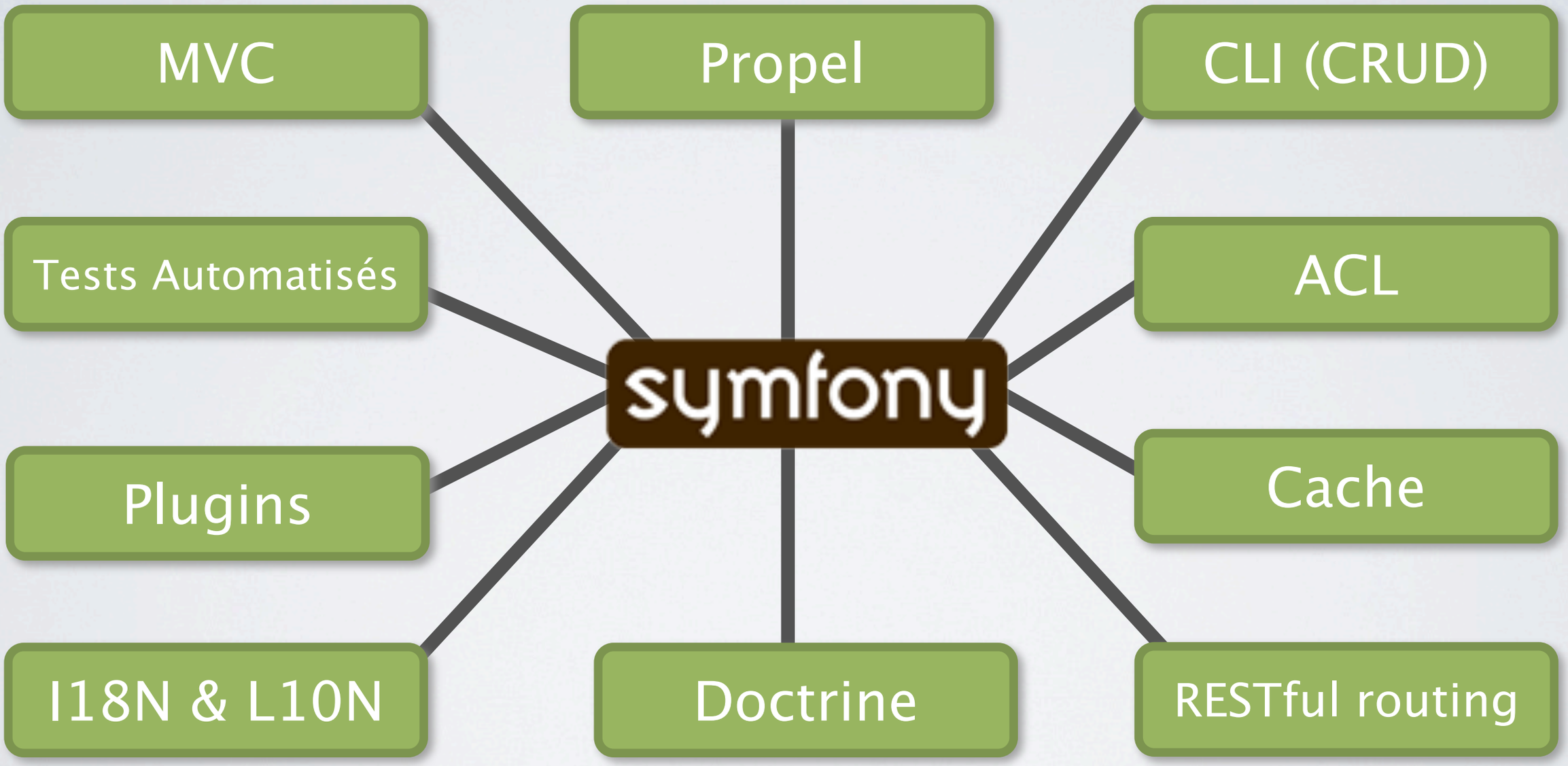
Symfony
un framework RESTfull

Introduction

Présentation de symfony

- Framework PHP 5 MVC Open Source « full stack »
- Rapid Application Development
- Publié en 2005 sous licence MIT
- Bâti pour répondre à des besoins complexes
- « best practices » (design patterns, mvc, tests unitaires...)
- Maintenu par Sensio Labs jusqu'à fin 2012
- Documentation riche et Open Source (4 livres officiels)
- Communauté très active

Outillage de symfony



Un routing RESTFull

Routage de symfony

- Symfony découple les URLs du code (implémentation technique)

/blog.php?section=symfony&article_id=18475

/blog/2008-01-30/symfony-happy-new-year

- Optimisation pour les moteurs de recherche
- URIs faciles à copier / coller, mémoriser ou bookmarker
- Masque l'implémentation technique (php, python, ruby ?)

Contraindre les routes aux méthodes HTTP

- sfRequestRoute permet de contraindre les URIs aux méthodes HTTP
- Une même URI peut être atteinte avec des méthodes HTTP différentes

```
user_edit:
  class: sfRequestRoute
  url:   /api/users/:username
  param: { module: user, action: update }
  requirements: { sf_method: put }
```

Contraindre les routes aux méthodes HTTP

```
$ php symfony app:routes api
```

```
Hugo:RestServer Hugo$ ./symfony app:routes api
>> app          Current routes for application "api"
Name          Method Pattern
users         GET    /api/users
user_create   POST   /api/users
user_show     GET    /api/users/:username
user_edit     PUT    /api/users/:username
user_delete   DELETE /api/users/:username
homepage      ANY    /
Hugo:RestServer Hugo$
```

Contraindre les routes aux méthodes HTTP

- Symfony simule les requêtes PUT et DELETE depuis un navigateur
- Les formulaires embarquent un champ caché « sf_method »

```
<form action="/api_dev.php/api/user/hugo" method="post">
  <input type="hidden" value="put" name="sf_method"/>
  <input type="hidden" id="rmll_user_id" value="2" name="rmll_user[id]"/>
  <input type="hidden" id="rmll_user__csrf_token"
value="94c132e7c771411590124aea52bb9d4f" name="rmll_user[_csrf_token]"/>
  <!-- ... -->
</form>
```

Contraindre les routes aux méthodes HTTP

```
$params = array(
  'method' => 'delete',
  'confirm' => 'Etes-vous sûr de vouloir supprimer cet objet ?'
);

echo link_to('supprimer', 'user_delete', $object, $params);
```

Contraindre les routes aux méthodes HTTP

```php
$params = array(
  'method' => 'delete',
  'confirm' => 'Etes-vous sûr de vouloir supprimer cet objet ?'
);

echo link_to('supprimer', 'user_delete', $object, $params);
```

```
<li class="sf_admin_action_delete"><a href="/api_dev.php/api/user/hugo"
onclick="if (confirm('Are you sure?')) { var f = document.createElement
('form'); f.style.display = 'none'; this.parentNode.appendChild(f);
f.method = 'post'; f.action = this.href;var m = document.createElement
('input'); m.setAttribute('type', 'hidden'); m.setAttribute('name',
'sf_method'); m.setAttribute('value', 'delete'); f.appendChild(m);var m
= document.createElement('input'); m.setAttribute('type', 'hidden');
m.setAttribute('name', '_csrf_token'); m.setAttribute('value',
'f0271a52cca50e62df7deb1568f80489'); f.appendChild(m);f.submit
(); };return false;">Supprimer</a></li>
```

Les routes d'objets


```
user_edit:
  class: sfDoctrineRoute
  url:   /api/users/:username
  param: { module: user, action: update }
  options: { model: RmllUser, column: username, type: object }
  requirements: { sf_method: put }
```

Les routes d'objets

```
public function executeUpdate(sfWebRequest $request)
{
  $user = $this->getRoute()->getObject();

  $form = new RmllUserForm($user, array(), false);
  $form->bind($request->getParameter('rmll_user'));

  $status = 400;
  if ($form->isValid())
  {
    $form->save();
    $status = 200;
  }

  $this->getResponse()->setStatusCode($status);
  return sfView::NONE;
}
```

Les collections de routes d'objets

- sfDoctrineRouteCollection ou sfPropelRouteCollection
- Actions CRUD sur un objet de modèle

Les collections de routes d'objets

```
user:
  class: sfDoctrineRouteCollection
  options:
    model:                RmllUser
    module:               user
    prefix_path:          /api/users
    column:               username
    with_wildcard_routes: true
    actions: [list, create, update, delete, show]
```

Les collections de routes d'objets

Default

```
Hugo:RestServer Hugo$ ./symfony app:routes api
>> app        Current routes for application "api"
Name             Method Pattern
user             GET    /api/users.:sf_format
user_create      POST   /api/users.:sf_format
user_update      PUT    /api/users/:username.:sf_format
user_delete      DELETE /api/users/:username.:sf_format
user_show        GET    /api/users/:username.:sf_format
user_object      GET    /api/users/:username/:action.:sf_format
user_collection  POST   /api/users/:action/action.:sf_format
homepage         ANY    /
Hugo:RestServer Hugo$
```

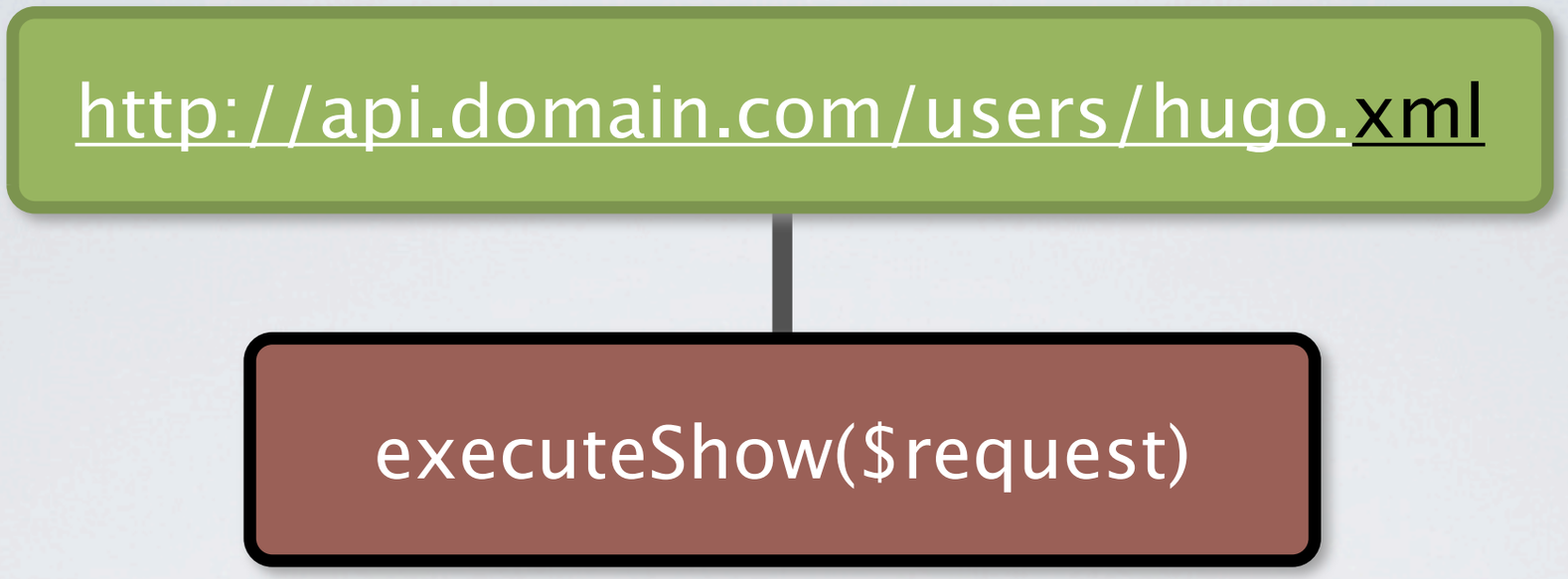
Support des formats de sortie

Support des formats de sortie

- Par défaut, symfony supporte le format HTML
- Le paramètre « sf_format » définit le format de sortie
- Symfony se charge d'envoyer les bons en-têtes en fonction de l'extension

```
user:
  class: sfDoctrineRouteCollection
  options:
    model:                RmllUser
    module:               user
    prefix_path:          /api/users
    column:               username
    with_wildcard_routes: true
    actions: [list, create, update, delete, show]
  requirements: { sf_format: (?:html|xml|json) }
```

http://api.domain.com/users/hugo.xml


http://api.domain.com/users/hugo.xml
executeShow($request)

http://api.domain.com/users/hugo.xml

executeShow($request)

showSuccess.php

showSuccess.xml.php

showSuccess.json.php

http://api.domain.com/users/hugo.xml

executeShow($request)

showSuccess.php

showSuccess.xml.php

showSuccess.json.php

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<user>
  <username>hhamon</username>
  <firstName>Hugo</firstName>
  <lastName>hamon</lastName>
  <birthdate>1987-07-02</birthdate>
</user>
```

showSuccess.php

```php
<h1><?php echo $user ?></h1>
<table>
  <tbody>
    <tr>
      <td>Username</td>
      <td><?php echo $user->getUsername() ?></td>
    </tr>
    <tr>
      <td>Password</td>
      <td><?php echo $user->getPassword() ?></td>
    </tr>
    <tr>
      <td>First name</td>
      <td><?php echo $user->getFirstName() ?></td>
    </tr>
    <tr>
      <td>Last name</td>
      <td><?php echo $user->getLastName() ?></td>
    </tr>
    <tr>
      <td>Birthdate</td>
      <td><?php echo $user->getBirthdate() ?></td>
    </tr>
  </tbody>
</table>
```

```php
<?php echo '<?xml version="1.0" encoding="UTF-8" ?>'."\n" ?>
<user>
  <username><?php echo $user->getUsername() ?></username>
  <firstName><?php echo $user->getFirstName() ?></firstName>
  <lastName><?php echo $user->getLastName() ?></lastName>
  <birthdate><?php echo $user->getBirthdate() ?></birthdate>
</user>
```

showSuccess.xml.php

showSuccess.json.php

```php
{
  "username":  <?php echo json_encode($user->getUsername()) ?>,
  "firstName": <?php echo json_encode($user->getFirstName(ESC_RAW)) ?>,
  "lastName":  <?php echo json_encode($user->getLastName(ESC_RAW)) ?>,
  "birthdate": <?php echo json_encode($user->getBirthdate()) ."\n" ?>
}
```

Hugo hamon

Username hhamon
Password e5e9fa1ba31ecd1ae84f75caaa474f3a663f05f4
First name Hugo
Last name hamon
Birthdate 1987-07-02

Sortie HTML

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<user>
  <username>hhamon</username>
  <firstName>Hugo</firstName>
  <lastName>hamon</lastName>
  <birthdate>1987-07-02</birthdate>
</user>
```

Sortie XML

```json
{
  "username":  "hhamon",
  "firstName": "Hugo",
  "lastName":  "hamon",
  "birthdate": "1987-07-02"
}
```

Sortie JSON

Cache des résultats

Cache des résultats

- Symfony dispose nativement d'un cache HTML des pages
  - Chaque sortie générée peut être mise en cache sur le serveur
  - La durée de vie du cache de chaque page est configurable
  - Le cache est par défaut stocké sur le système de fichier (sfFileCache)

- Symfony offre la possibilité de choisir où stocker le cache :
  - Fichiers (sfFileCache)
  - SQLite (sfSQLiteCache)
  - XCache (sfXCacheCache)
  - MemCache (sfMemCacheCache)
  - APC... (sfAPCCache)

Cache des résultats

- apps/api/modules/user/config/cache.yml

```
show:
  enabled:        true
  with_layout:    true
  lifetime:       3600
```

Avant la mise en cache

Après la mise en cache

Tests Fonctionnels de l'API

Simuler des requêtes GET

```
include(dirname(__FILE__).'/../../bootstrap/functional.php');

$browser = new RmllTestFunctional(new sfBrowser());
$browser->loadFixtures();

$browser->
  get('/api/users/hhamon.xml')->

  with('request')->begin()->
    isFormat('xml')->
    isParameter('module', 'user')->
    isParameter('action', 'show')->
  end()->

  with('response')->begin()->
    isStatusCode(200)->
    isValid()->
  end();
```

Simuler des requêtes GET

```
include(dirname(__FILE__).'/../../bootstrap/functional.php');

$browser = new RmllTestFunctional(new sfBrowser());
$browser->loadFixtures();

$browser->
  get('/api/users/hhamon.xml')->

  with('request')->begin()->
    isFormat('xml')->
    isParameter('module', 'user')->
    isParameter('action', 'show')->
  end()->

  with('response')->begin()->
    isStatusCode(200)->
    isValid()->
  end();
```

GET sur le fichier XML

Simuler des requêtes GET

```
include(dirname(__FILE__).'/../../bootstrap/functional.php');

$browser = new RmllTestFunctional(new sfBrowser());
$browser->loadFixtures();

$browser->
  get('/api/users/hhamon.xml')->

  with('request')->begin()->
    isFormat('xml')->
    isParameter('module', 'user')->
    isParameter('action', 'show')->
  end()->

  with('response')->begin()->
    isStatusCode(200)->
    isValid()->
  end();
```

GET sur le fichier XML

Vérification de la requête

Simuler des requêtes GET

```
include(dirname(__FILE__).'/../../bootstrap/functional.php');

$browser = new RmllTestFunctional(new sfBrowser());
$browser->loadFixtures();

$browser->
  get('/api/users/hhamon.xml')->

  with('request')->begin()->
    isFormat('xml')->
    isParameter('module', 'user')->
    isParameter('action', 'show')->
  end()->

  with('response')->begin()->
    isStatusCode(200)->
    isValid()->
  end();
```

GET sur le fichier XML

Vérification de la requête

Vérification de la réponse

Simuler des requêtes GET

Default

```
Hugo:RestServer Hugo$ php test/functional/api/userActionsTest.php
# get /api/users/hhamon.xml
ok 1 - request format is xml
ok 2 - request parameter module is user
ok 3 - request parameter action is show
ok 4 - status code is 200
ok 5 - response is well-formed xml
1..5
# Looks like everything went fine.
Hugo:RestServer Hugo$
```

```php
$user = array(
  'username' => 'pmartin',
  'password' => 'something',
  'first_name' => 'Paul',
  'last_name' => 'Martin'
);

$user = getUser(array('birthdate' => ''));
$browser->
  post('/api/users', array('rmll_user' => $user))->
  with('response')->isStatusCode(400);

$browser->
  post('/api/users', array('rmll_user' => getUser()))->
  with('response')->begin()->
    isStatusCode(201)->
    isRedirected()->
  end()->
  followRedirect()
```

```php
$user = array(
  'username' => 'pmartin',
  'password' => 'something',
  'first_name' => 'Paul',
  'last_name' => 'Martin'
);
```

```php
$user = getUser(array('birthdate' => ''));
$browser->
  post('/api/users', array('rmll_user' => $user))->
  with('response')->isStatusCode(400);
```

```php
$browser->
  post('/api/users', array('rmll_user' => getUser()))->
  with('response')->begin()->
    isStatusCode(201)->
    isRedirected()->
  end()->
  followRedirect()
```

```php
$user = array(
  'username' => 'pmartin',
  'password' => 'something',
  'first_name' => 'Paul',
  'last_name' => 'Martin'
);
```

Test de la création avec des paramètres manquants

```php
$user = getUser(array('birthdate' => ''));
$browser->
  post('/api/users', array('rmll_user' => $user))->
  with('response')->isStatusCode(400);
```

```php
$browser->
  post('/api/users', array('rmll_user' => getUser()))->
  with('response')->begin()->
    isStatusCode(201)->
    isRedirected()->
  end()->
  followRedirect()
```

```php
$user = array(
  'username' => 'pmartin',
  'password' => 'something',
  'first_name' => 'Paul',
  'last_name' => 'Martin'
);
```

Test de la création avec des paramètres manquants

```php
$user = getUser(array('birthdate' => ''));
$browser->
  post('/api/users', array('rmll_user' => $user))->
  with('response')->isStatusCode(400);
```

```php
$browser->
  post('/api/users', array('rmll_user' => getUser()))->
  with('response')->begin()->
    isStatusCode(201)->
    isRedirected()->
  end()->
  followRedirect()
```

```php
$user = array(
    'username' => 'pmartin',
    'password' => 'something',
    'first_name' => 'Paul',
    'last_name' => 'Martin'
);
```

Test de la création avec des paramètres manquants

```php
$user = getUser(array('birthdate' => ''));
$browser->
    post('/api/users', array('rmll_user' => $user))->
    with('response')->isStatusCode(400);
```

```php
$browser->
    post('/api/users', array('rmll_user' => getUser()))->
    with('response')->begin()->
        isStatusCode(201)->
        isRedirected()->
    end()->
    followRedirect()
```

Test de la création avec des paramètres valides

Simuler des requêtes POST

Default

```
Hugo:RestServer Hugo$ php test/functional/api/userActionsTest.php
# get /api/users/hhamon.xml
ok 1 - request format is xml
ok 2 - request parameter module is user
ok 3 - request parameter action is show
ok 4 - status code is 200
ok 5 - response is well-formed xml
# post /api/users
ok 6 - status code is 400
# post /api/users
ok 7 - status code is 201
ok 8 - page redirected to /index.php/api/users/pmartin
1..8
# Looks like everything went fine.
Hugo:RestServer Hugo$
```

Simuler des requêtes PUT

```
$user = getUser(array('first_name' => 'Toto'));
$browser->
  call('/api/users/pmartin', 'put', array('rmll_user' => $user))->
  with('response')->isStatusCode(200)
;
```

Simuler des requêtes PUT

```
$user = getUser(array('first_name' => 'Toto'));
$browser->
  call('/api/users/pmartin', 'put', array('rmll_user' => $user))->
  with('response')->isStatusCode(200)
;
```

```
Hugo:RestServer Hugo$ php test/functional/api/userActionsTest.php
# get /api/users/hhamon.xml
ok 1 - request format is xml
ok 2 - request parameter module is user
ok 3 - request parameter action is show
ok 4 - status code is 200
ok 5 - response is well-formed xml
# post /api/users
ok 6 - status code is 400
# post /api/users
ok 7 - status code is 201
ok 8 - page redirected to /index.php/api/users/pmartin
# put /api/users/pmartin
ok 9 - status code is 200
1..9
# Looks like everything went fine.
```

Simuler des requêtes DELETE

```
$browser->
  call('/api/users/pmartin', 'delete')->
  with('response')->isStatusCode(200);
```

Simuler des requêtes DELETE

```
$browser->
  call('/api/users/pmartin', 'delete')->
  with('response')->isStatusCode(200);
```

```
# put /api/users/pmartin
ok 9 - status code is 200
# delete /api/users/pmartin
ok 10 - status code is 200
1..10
# Looks like everything went fine.
Hugo:RestServer Hugo$
```

Architecture SOAP

Introduction à SOAP

Qu'est-ce que SOAP ?

- « Simple Object Access Protocol »
- SOAP est un protocole standard (http://www.w3.org/TR/soap/)
- Appel de procédure à distance reposant sur un standard XML
- SOAP permet d'invoquer des méthodes sur un objet distant
- La réponse est servie au client SOAP sous forme d'une réponse SOAP XML
- Transport des requêtes et réponses SOAP avec le protocole HTTP
- Les requêtes et réponses SOAP sont des chaînes XML
- Les données sont traitées sur le serveur puis converties en XML

Avantages et Inconvénients

❖ **Avantages** :

- ✤ Standard éprouvé
- ✤ Indépendant de la plateforme
- ✤ Indépendant du langage
- ✤ Extensible

❖ **Inconvénients** :

- ✤ Lourd et lent
- ✤ Verbosité du XML
- ✤ Difficile à tester et à cacher

Fonctionnement de SOAP


```
$service = new FooService();
$response = $s->calledMethod($param1, $param2);
```

Exposer des services SOAP avec symfony et Zend

```php
class ProjectConfiguration extends sfProjectConfiguration
{
  static protected $zendLoaded = false;

  public function setup()
  {
    $this->enablePlugins('sfDoctrinePlugin');
    self::registerZend();
  }

  static public function registerZend()
  {
    if (self::$zendLoaded) {
      return;
    }

    set_include_path(
      sfConfig::get('sf_lib_dir').'/vendor'.PATH_SEPARATOR.get_include_path()
    );
    require_once sfConfig::get('sf_lib_dir').'/vendor/Zend/Loader/Autoloader.php';
    Zend_Loader_Autoloader::getInstance();
    self::$zendLoaded = true;
  }
}
```

```php
class ProjectConfiguration extends sfProjectConfiguration
{
  static protected $zendLoaded = false;

  public function setup()
  {
    $this->enablePlugins('sfDoctrinePlugin');
    self::registerZend();
  }

  static public function registerZend()
  {
    if (self::$zendLoaded) {
      return;
    }

    set_include_path(
      sfConfig::get('sf_lib_dir').'/vendor'.PATH_SEPARATOR.get_include_path()
    );
    require_once sfConfig::get('sf_lib_dir').'/vendor/Zend/Loader/Autoloader.php';
    Zend_Loader_Autoloader::getInstance();
    self::$zendLoaded = true;
  }
}
```

Framework Zend dans lib/vendor/Zend

```php
class ProjectConfiguration extends sfProjectConfiguration
{
  static protected $zendLoaded = false;

  public function setup()
  {
    $this->enablePlugins('sfDoctrinePlugin');
    self::registerZend();
  }

  static public function registerZend()
  {
    if (self::$zendLoaded) {
      return;
    }

    set_include_path(
      sfConfig::get('sf_lib_dir').'/vendor'.PATH_SEPARATOR.get_include_path()
    );
    require_once sfConfig::get('sf_lib_dir').'/vendor/Zend/Loader/Autoloader.php';
    Zend_Loader_Autoloader::getInstance();
    self::$zendLoaded = true;
  }
}
```

Framework Zend dans lib/vendor/Zend

Initialisation de l'autoloader de Zend

Création d'un module et d'une route

- Création d'un nouveau module converter

```
$ php symfony generate:module api converter
```

- Création d'une nouvelle route converter_service dans routing.yml

```
converter_service:
  url:          /converter/service
  param:        { module: converter, action: index }
```

```php
class TemperatureConverterService
{
  /**
   * Converts a Celsius temperature into a Fahrenheit temperature.
   *
   * @param float $value The value to convert
   * @return float
   */
  public function convertCelsiusToFahrenheit($value)
  {
    return 1.8 * $value + 32;
  }

  /**
   * Converts a Fahrenheit temperature into a Celsius temperature.
   *
   * @param float $value The value to convert
   * @return float
   */
  public function convertFahrenheitToCelsius($value)
  {
    return (5/9) * $value - 160/9;
  }
}
```

PHPDoc importante

Dévoiler le WSDL

```
public function executeWsdl(sfWebRequest $request)
{
  try
  {
    $wsdl = new Zend_Soap_AutoDiscover();
    $wsdl->setClass('TemperatureConverterService');
    $wsdl->handle();
  }
  catch (Exception $e)
  {
    $this->logMessage($e->getMessage());
  }

  return sfView::NONE;
}
```

Dévoiler le WSDL

```php
public function executeWsdl(sfWebRequest $request)
{
  try
  {
    $wsdl = new Zend_Soap_AutoDiscover();
    $wsdl->setClass('TemperatureConverterService');
    $wsdl->handle();
  }
  catch (Exception $e)
  {
    $this->logMessage($e->getMessage());
  }

  return sfView::NONE;
}
```

Zend génère le WSDL par introspection de la PHPDoc de la classe du service

Mozilla Firefox

http://weather-server.local/converter/service?wsdl

Désactiver Cookies CSS Form. Images Information Divers Redimensionner Outils Voir Source Options

http://weather-s...ter/service?wsdl

Aucune information de style ne semble associée à ce fichier XML.

http://api.domain.com/converter/service?wsdl

```
- <definitions name="TemperatureConverterService" targetNamespace="http://weather-server.local/converter/service">
  - <types>
      <xsd:schema targetNamespace="http://weather-server.local/converter/service"/>
    </types>
  - <portType name="TemperatureConverterServicePort">
    - <operation name="convertCelsiusToFahrenheit">
      - <documentation>
          Converts a Celsius temperature into a Fahrenheit temperature.
        </documentation>
        <input message="tns:convertCelsiusToFahrenheitIn"/>
        <output message="tns:convertCelsiusToFahrenheitOut"/>
      </operation>
  </portType>
- <binding name="TemperatureConverterServiceBinding" type="tns:TemperatureConverterServicePort">
    <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
  - <operation name="convertCelsiusToFahrenheit">
      <soap:operation soapAction="http://weather-server.local/converter/service#convertCelsiusToFahrenheit"/>
    - <input>
        <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://weather-server.local/converter/service"/>
      </input>
    - <output>
        <soap:body use="encoded" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" namespace="http://weather-server.local/converter/service"/>
      </output>
    </operation>
  </binding>
- <service name="TemperatureConverterServiceService">
  - <port name="TemperatureConverterServicePort" binding="tns:TemperatureConverterServiceBinding">
      <soap:address location="http://weather-server.local/converter/service"/>
    </port>
```

Implémenter la passerelle du service

```php
public function executeIndex(sfWebRequest $request)
{
  $this->forwardIf(null !== $request->getParameter('wsdl'), 'converter', 'wsdl');

  try
  {
    $wsdl   = $this->generateUrl('converter_service', array(), true).'?wsdl';
    $server = new Zend_Soap_Server($wsdl);
    $server->setClass('TemperatureConverterService');
    $server->setReturnResponse(true);
    $this->renderText($server->handle());
  }
  catch (Exception $e)
  {
    $this->logMessage($e->getMessage());
  }

  return sfView::NONE;
}
```

Consommer le service SOAP

Consommer le service SOAP

```php
try
{
  $client = new Zend_Soap_Client(
    'http://api.domain.com/converter/service?wsdl'
  );

  $celsiusToFahrenheit = $client->convertCelsiusToFahrenheit(30.50);
  $fahrenheitToCelsius = $client->convertFahrenheitToCelsius(72.30);
}
catch (SoapFault $e)
{
  exit($e->getMessage());
}

echo $celsiusToFahrenheit;
echo $fahrenheitToCelsius;
```

Tester le service SOAP

Tester le service SOAP

- Tester un service SOAP avec PHP est assez compliqué !
  - La requête et la réponse doivent être testées
  - Les réponses doivent être simulées (mock)

- Zend_Soap a l'avantage d'être déjà testé par Zend
- Le fonctionnement de symfony est déjà testé également
- A tester unitairement : la classe TemperatureConverterService

Tests unitaires du service

```php
// test/unit/service/TemperatureConverterServiceTest.php
require dirname(__FILE__).'/../../bootstrap/unit.php';

$t = new lime_test(3);
$service = new TemperatureConverterService();

$t->diag('->convertCelsiusToFahrenheit()');

$result = $service->convertCelsiusToFahrenheit(0);
$t->is($result, 32, '->convertCelsiusToFahrenheit() returns 32');

$result = $service->convertCelsiusToFahrenheit(1);
$t->is($result, 33.8, '->convertCelsiusToFahrenheit() returns 33.8');

$result = $service->convertCelsiusToFahrenheit(10);
$t->is($result, 50, '->convertCelsiusToFahrenheit() returns 50');
```

Tests unitaires du service

```
$ php symfony test:unit TemperatureConverterService
```

```
# ->convertCelsiusToFahrenheit()
ok 1 - ->convertCelsiusToFahrenheit() returns 32
ok 2 - ->convertCelsiusToFahrenheit() returns 33.8
ok 3 - ->convertCelsiusToFahrenheit() returns 50
# Looks like everything went fine.
Hugo:RestServer Hugo$
```

Tests unitaires du service

```php
// test/unit/service/TemperatureConverterServiceTest.php
require dirname(__FILE__).'/../../bootstrap/unit.php';

$t = new lime_test(6);
$service = new TemperatureConverterService();

// ...

$t->diag('->convertFahrenheitToCelsius()');

$result = $service->convertFahrenheitToCelsius(0);
$t->is($result, -160/9, '->convertFahrenheitToCelsius() returns -160/9');

$result = $service->convertFahrenheitToCelsius(33.8);
$t->is($result, 1, '->convertFahrenheitToCelsius() returns 1');

$result = $service->convertFahrenheitToCelsius(50);
$t->is($result, 10, '->convertFahrenheitToCelsius() returns 10');
```

Tests unitaires du service

```
$ php symfony test:unit TemperatureConverterService
```

```
# ->convertCelsiusToFahrenheit()
ok 1 - ->convertCelsiusToFahrenheit() returns 32
ok 2 - ->convertCelsiusToFahrenheit() returns 33.8
ok 3 - ->convertCelsiusToFahrenheit() returns 50
# ->convertFahrenheitToCelsius()
ok 4 - ->convertFahrenheitToCelsius() returns -160/9
ok 5 - ->convertFahrenheitToCelsius() returns 1
ok 6 - ->convertFahrenheitToCelsius() returns 10
# Looks like everything went fine.
Hugo:RestServer Hugo$
```

Questions ?

Sensio Labs recrute !

hugo.hamon@sensio.com