Programmation Web Avancé

Chapitre 2 : Découverte d'un framework PHP Symfony

Guillaume-Olivier LAFFITTE
guillaume.laffitte@stratow.com

Pourquoi utiliser un framework ?

Un framework permet de réaliser des sites complexes rapidement, de façon structurée et avec un code clair et maintenable.

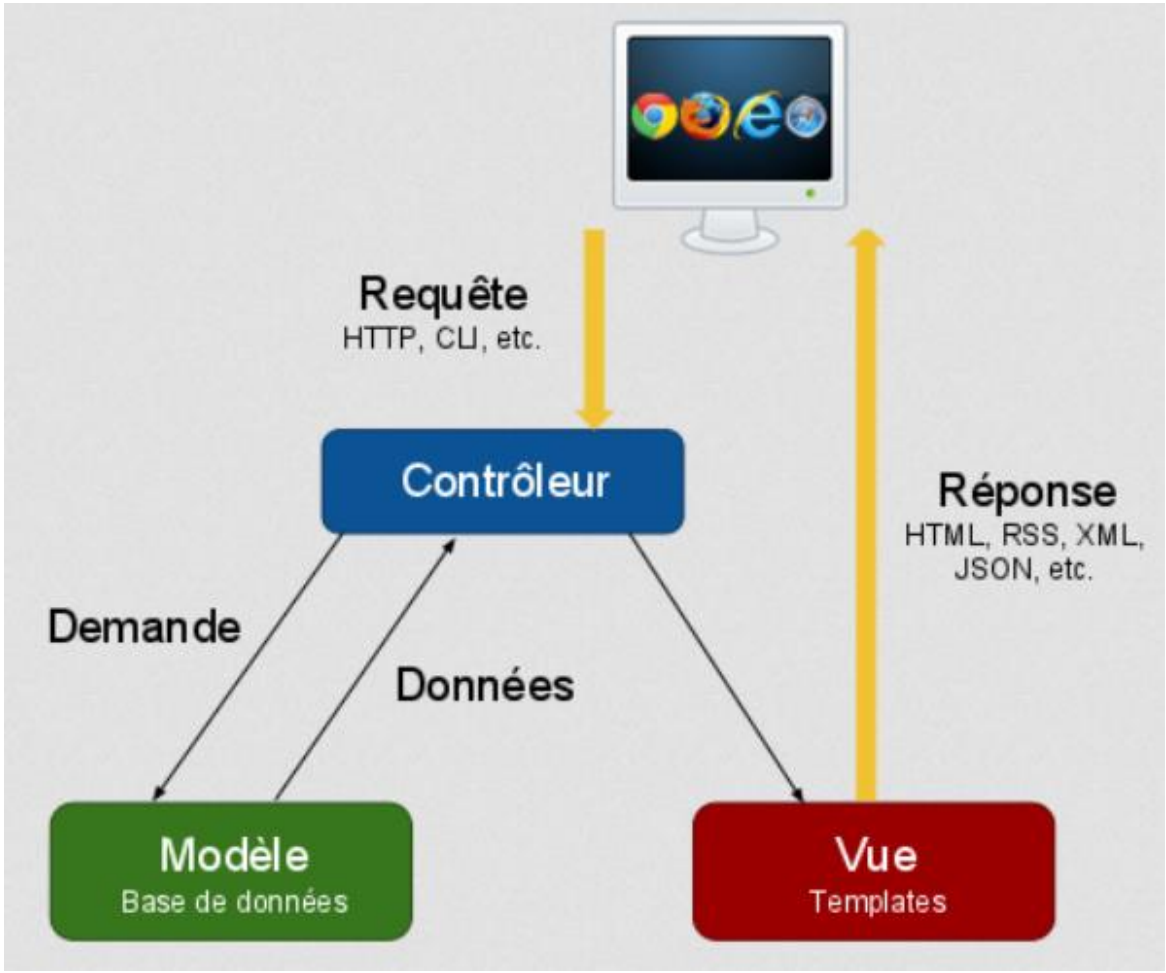
Les frameworks utilisent presque toujours l'architecture MVC, c'est une façon d'organiser le code qui sépare la partie PHP de la partie HTML.

Il améliore la façon de travailler avec d'autres développeurs / designers.

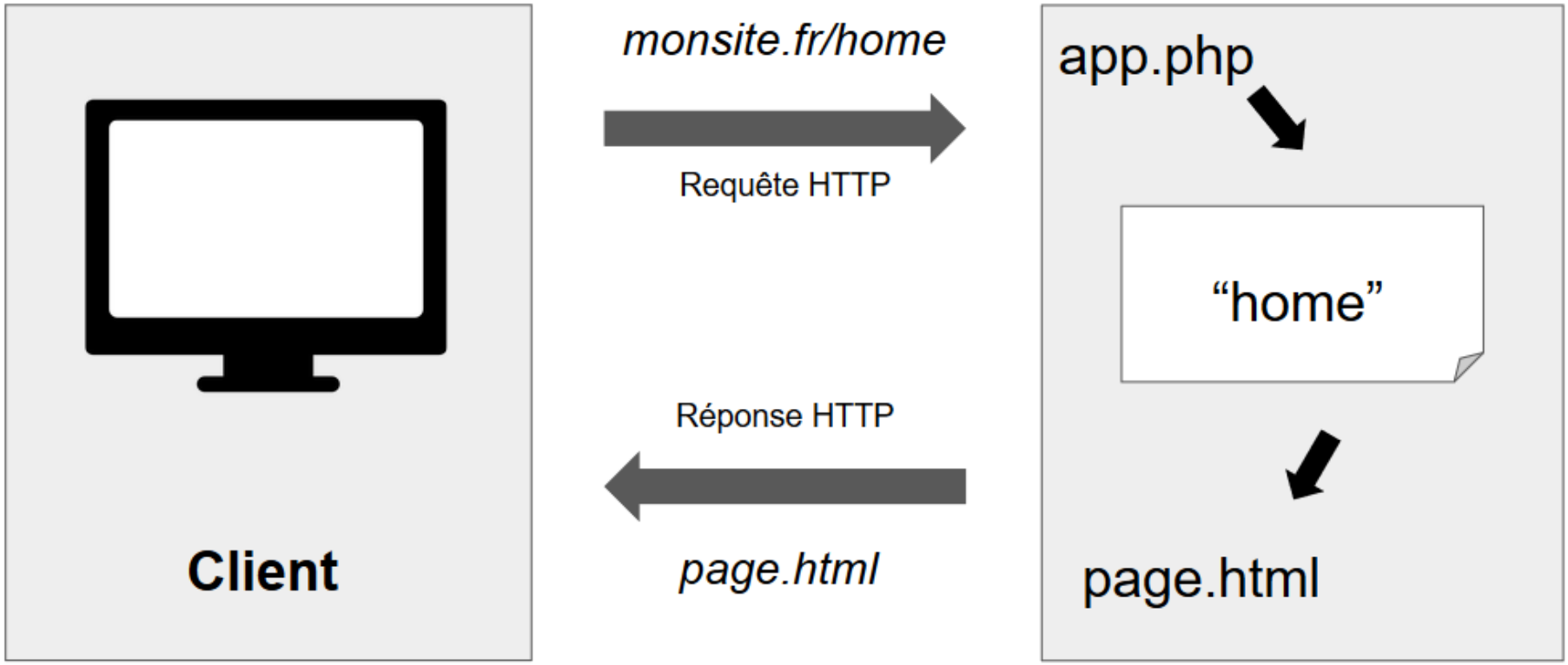
Symfony

- Symfony est donc un framework PHP flexible et puissant
- La première version est sortie en 2005
- Open source et français

L'architecture MVC



Le routing


monsite.fr/home
Requête HTTP
app.php
"home"
page.html
Réponse HTTP
page.html
Client

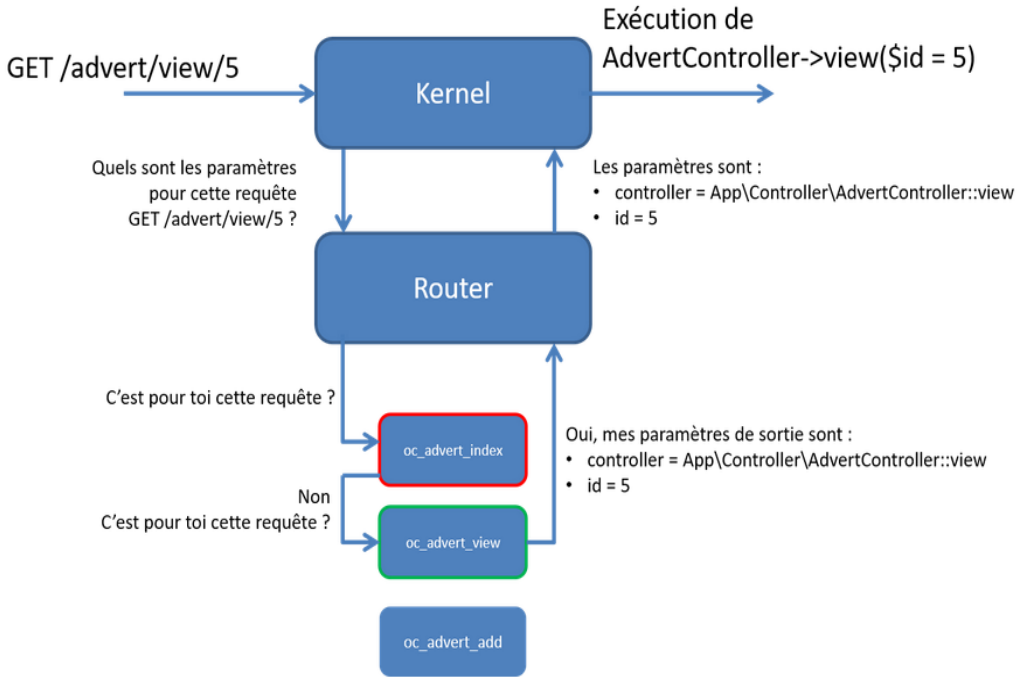
Le routing

```
# config/routes.yaml

oc_advert_index:
    path:       /advert
    controller: App\Controller\AdvertController::index

oc_advert_view:
    path:       /advert/view/{id}
    controller: App\Controller\AdvertController::view

oc_advert_add:
    path:       /advert/add
    controller: App\Controller\AdvertController::add
```



Les contrôleurs

```php
// src/AppBundle/Controller/LuckyController.php
namespace AppBundle\Controller;

use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

class LuckyController
{
    /**
     * @Route("/lucky/number")
     */
    public function numberAction()
    {
        $number = random_int(0, 100);

        return new Response(
            '<html><body>Lucky number: '.$number.'</body></html>'
        );
    }
}
```

Les contrôleurs - Lever une 404

```php
public function indexAction()
{
    // retrieve the object from database
    $product = ...;
    if (!$product) {
        throw $this->createNotFoundException('The product does not exist');
    }

    return $this->render(...);
}
```

Les contrôleurs - $request

Type de paramètres	Méthode Symfony	Méthode traditionnelle	Exemple
Variables d'URL	`$request->query`	$_GET	`$request->query->get('tag')`
Variables de formulaire	`$request->request`	$_POST	`$request->request->get('tag')`
Variables de cookie	`$request->cookies`	$_COOKIE	`$request->cookies->get('tag')`
Variables de serveur	`$request->server`	$_SERVER	`$request->server->get('REQUEST_URI')`
Variables d'entête	`$request->headers`	$_SERVER['HTTP_*']	`$request->headers->get('USER_AGENT')`
Paramètres de route	`$request->attributes`	n/a	On utilise `$id` dans les arguments de la méthode, mais vous pourriez également écrire `$request->attributes->get('id')`

Les contrôleurs – La session

```php
use Symfony\Component\HttpFoundation\Session\SessionInterface;

public function indexAction(SessionInterface $session)
{
    // stores an attribute for reuse during a later user request
    $session->set('foo', 'bar');

    // gets the attribute set by another controller in another request
    $foobar = $session->get('foobar');

    // uses a default value if the attribute doesn't exist
    $filters = $session->get('filters', []);
}
```

Les templates Twig

- Le modale MVC sépare le code PHP du code HTML
- Mais pour faire du HTML de présentation, on a toujours besoin d'un peu de code dynamique : boucle, condition, variables
- Pour faciliter ce code dynamique dans les templates, le moteur de templates Twig offre son pseudo-langage

Les templates Twig - Syntaxe

Description	Exemple Twig	Équivalent PHP
Afficher une variable	`Pseudo : {{ pseudo }}`	`Pseudo : <?php echo $pseudo; ?>`
Afficher l'index d'un tableau	`Identifiant : {{ user['id'] }}`	`Identifiant : <?php echo $user['id']; ?>`
Afficher l'attribut d'un objet, dont le getter respecte la convention `$objet->getAttribut()`	`Identifiant : {{ user.id }}`	`Identifiant : <?php echo $user->getId(); ?>`

Les templates Twig - Syntaxe

Description	Exemple Twig	Équivalent PHP
Afficher une variable en lui appliquant un filtre. Ici, « upper » met tout en majuscules :	`Pseudo en majuscules : {{ pseudo|upper }}`	`Pseudo en lettre majuscules : <?php echo strtoupper($pseudo); ?>`
Afficher une variable en combinant les filtres. « striptags » supprime les balises HTML. « title » met la première lettre de chaque mot en majuscule. Notez l'ordre d'application des filtres, ici `striptags` est appliqué, puis `title`.	`Message : {{ news.texte|striptags|title }}`	`Message : <?php echo ucwords(strip_tags($news->getTexte())); ?>`
Utiliser un filtre avec des arguments. Attention, il faut que `date` soit un objet de type `Datetime` ici.	`Date : {{ date|date('d/m/Y') }}`	`Date : <?php echo $date->format('d/m/Y'); ?>`

Les templates Twig - Condition

```
1 {% if membre.age < 12 %}
2   Il faut avoir au moins 12 ans pour ce film.
3 {% elseif membre.age < 18 %}
4   OK bon film.
5 {% else %}
6   Un peu vieux pour voir ce film non ?
7 {% endif %}
```

Les templates Twig - Boucle

```
1 <select>
2   {% for valeur, option in liste_options %}
3     <option value="{{ valeur }}">{{ option }}</option>
4   {% endfor %}
5 </select>
```

Les templates Twig - Boucle

Variable	Description
`{{ loop.index }}`	Le numéro de l'itération courante (en commençant par 1).
`{{ loop.index0 }}`	Le numéro de l'itération courante (en commençant par 0).
`{{ loop.revindex }}`	Le nombre d'itérations restantes avant la fin de la boucle (en finissant par 1).
`{{ loop.revindex0 }}`	Le nombre d'itérations restantes avant la fin de la boucle (en finissant par 0).
`{{ loop.first }}`	`true` si c'est la première itération, `false` sinon.
`{{ loop.last }}`	`true` si c'est la dernière itération, `false` sinon.
`{{ loop.length }}`	Le nombre total d'itérations dans la boucle.

Les templates Twig - Block

```twig
{# templates/layout.html.twig #}

<!DOCTYPE HTML>
<html>
<head>
  <meta charset="utf-8">
  <title>{% block title %}Plateforme d'annonces{% endblock %}</title>
</head>
<body>
  {% block body %}
  {% endblock %}
</body>
</html>
```

Les templates Twig - Block

```
1 {# templates/Advert/index.html.twig #}
2
3 {% extends "layout.html.twig" %}
4
5 {% block title %}{{ parent() }} - Index{% endblock %}
6
7 {% block body %}
8   Notre plateforme est un peu vide pour le moment, mais cela viendra !
9 {% endblock %}
```

Les services

- Un service est un objet PHP qui remplit une fonction
  - Envoyer des e-mails
  - gérer une base de données
  - générer un formulaire
  - logger une erreur

- Les services ne sont instanciés qu'une fois et sont accessible par le conteneur de services

Les services

```php
// src/AppBundle/Controller/ProductController.php
namespace AppBundle\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\Controller;
use Symfony\Component\Routing\Annotation\Route;

class ProductController extends Controller
{
    /**
     * @Route("/products")
     */
    public function listAction()
    {
        $logger = $this->container->get('logger');
        $logger->info('Look! I just used a service');

        // ...
    }
}
```

Les services

```php
<?php
// src/OC/PlatformBundle/Antispam/OCAntispam.php

namespace OC\PlatformBundle\Antispam;

class OCAntispam
{
  /**
   * Vérifie si le texte est un spam ou non
   *
   * @param string $text
   * @return bool
   */
  public function isSpam($text)
  {
    return strlen($text) < 50;
  }
}
```

Les services

```yaml
# src/OC/PlatformBundle/Resources/config/services.yml

services:
    oc_platform.antispam:
        class: OC\PlatformBundle\Antispam\OCAntispam
```

Les services

```php
<?php

// src/OC/PlatformBundle/Controller/AdvertController.php

namespace OC\PlatformBundle\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\Controller;
use Symfony\Component\HttpFoundation\Request;

class AdvertController extends Controller
{
  public function addAction(Request $request)
  {
    // On récupère le service
    $antispam = $this->container->get('oc_platform.antispam');

    // Je pars du principe que $text contient le texte d'un message quelconque
    $text = '...';
    if ($antispam->isSpam($text)) {
      throw new \Exception('Votre message a été détecté comme spam !');
    }

    // Ici le message n'est pas un spam
  }
}
```