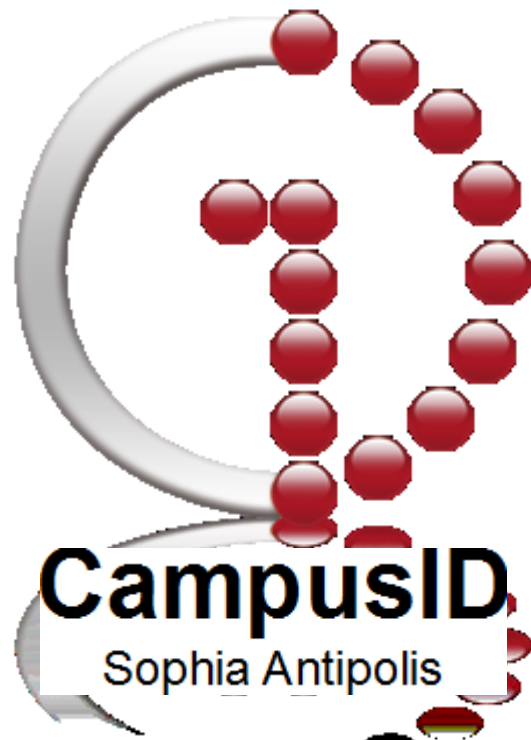


Software PHP

Framework Symfony 3

PARTIE 1 – Vue d'ensemble de Symfony

B3



Version 1.14

Crédits et références bibliographiques

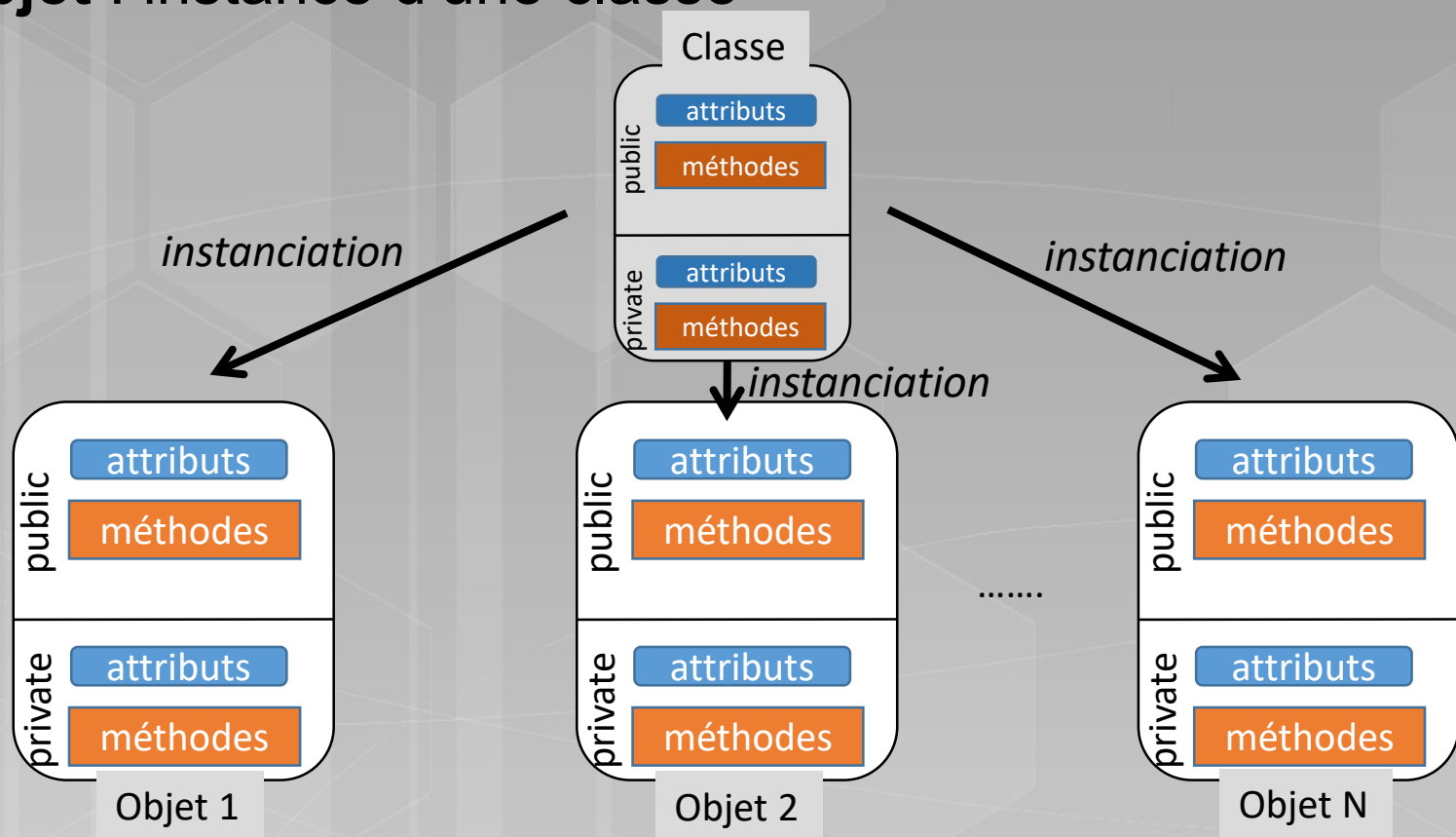
- [Site web de Symfony](#)
- [Site web de Sensiolabs](#)
- [Livres de référence : Développez votre site web avec le framework Symfony3](#) (Eyrolles/OC)

- Rappels sur la POO avec PHP
- PARTIE 1 – Vue d'ensemble de Symfony
 - Présentation et installation
 - Les bundles

Rappels sur la POO avec PHP

Classe et Objet

- **Classe** : Définition des objets avec des données (attributs ou propriétés) et des fonctions (méthodes) agissant sur ces données
- **Objet** : instance d'une classe



Déclarer une classe

- Pour déclarer une classe on utilise le **mot clé class**

```
class User  
{  
    // ....  
}
```

Visibilité des membres d'une classe

- Les mots clés **private** et **public** permettent de définir la visibilité des membres d'une classe
 - Les membres *public* pourront être accédés à l'extérieur de l'objet
 - Les membres *private* ne pourront être accédés qu'à l'intérieur d'un objet par l'intermédiaire de ses méthodes (principe d'encapsulation)

Attributs d'une classe

- Les **attributs d'une classe** sont parfois appelés « Variables membres », « Champs » ou bien encore « Propriétés »

```
class User
{
    private $_pseudo;
    public $foo = FALSE;
}
```



Tip!

- ❖ Un attribut peut être initialisé (à une expression constante)
- ❖ La convention d'écriture [PEAR](#), très utilisée en PHP, préconise un *underscore* (`_`) devant les attributs Private

Méthodes d'une classe

- Les **méthodes d'une classe** sont parfois appelés « Fonctions membres »
- Les méthodes permettent d'accéder aux attributs privés de la classe
- La variable spéciale **\$this** correspond à l'instance d'objet qui est utilisée
- L'accès aux attributs ou aux méthodes d'un objet se fait avec l'opérateur **->** (sans le faire suivre du signe \$)

```
class User
{
    private $_pseudo;
    public $foo = FALSE;

    public function getPseudo()
    {
        return $this->_pseudo;
    }
}
```

Accesseurs et mutateurs

- Conventionnellement les **méthodes accesseurs** et **mutateurs** commencent par *get* ou *set* (pour *getters* et *setters*)
- Elles ont pour rôle la lecture ou la modification d'attributs

```
public function getPseudo()  
{  
    return $this->_pseudo;  
}  
  
public function setPseudo($nouveauPseudo)  
{  
    if (!empty($nouveauPseudo) AND strlen($nouveauPseudo) < 30)  
    {  
        $this->_pseudo = $nouveauPseudo;  
    }  
}
```

Création et utilisation d'un objet

- On crée un **instance d'objet** avec l'opérateur **new**
- On utilise les méthodes avec l'opérateur **->**

```
<?php
    include_once('User.class.php');

    $user = new User; // ou new User();
    $user->foo = TRUE;
    $user->setPseudo('Jo10');

    echo $user->getPseudo();
    echo "<br>foo vaut : ".$user->foo;

?>
```

Héritage

- L'héritage consiste à créer une classe qui hérite **des attributs et méthodes d'une autre classe**
 - La classe qui hérite est dite **classe fille**
 - La classe dont on hérite les attributs et méthodes est dite **classe mère**
- Les instances de la **classe fille** peuvent utiliser **uniquement les méthodes publiques de la classe mère**
- Utilisation du mot-clé **extends**

```
class Ville {  
    // ...  
}  
  
class Capitale extends Ville {  
    // ...  
}
```

Rappel : Importation de classes depuis un espace de nom

- Il est possible en PHP d'importer une classe d'un certain espace de nom dans l'espace de nom courant
 - Soit le fichier F.ns.php où est définie une classe :
 - Pour importer la classe, il suffit de taper
use espace_de_nom\MaClasse

```
<?php
namespace A\B\C\D\E\F;

class MaClasse
{
    public function hello()
    {
        echo 'Hello world !';
    }
}
?>
```

fichier F.ns.php

```
<?php
require 'F.ns.php';

use A\B\C\D\E\F\MaClasse;

$a = new MaClasse; // Se transforme en $a = new A\B\C\D\E\F\MaClasse.
$a->hello(); // Affichera "Hello World !"
?>
```

PARTIE 1 – Vue d'ensemble de Symfony

Présentation et installation

Qu'est-ce qu'un framework ?

- Un **framework** signifie littéralement un **cadre de travail** fournissant des composants qui servent à créer les fondations, l'architecture et les grandes lignes d'un logiciel
- En PHP il en existe plusieurs : Symfony, CakePHP, Laravel ..
- Un framework diffère d'un CMS (ex: Drupal ou WordPress) dans le fait qu'il n'est pas utilisable tel quel par l'utilisateur final.
- Un framework aide à écrire du code robuste et réutilisable
- Un framework facilite le travail en équipe
- Un framework bénéficie d'une large communauté
- Un framework demande une période d'apprentissage

Le framework Symfony

- **Symfony** est un **framework PHP (objets)** créé en 2005 par Fabien Potencier et édité par la société

<https://sensiolabs.com/>

- Actuellement en version 3 support long-terme : 3.4.15 (Nov 2017) ou en nouvelle version 4 (4.1.4 de août 2018)

<https://symfony.com/roadmap>

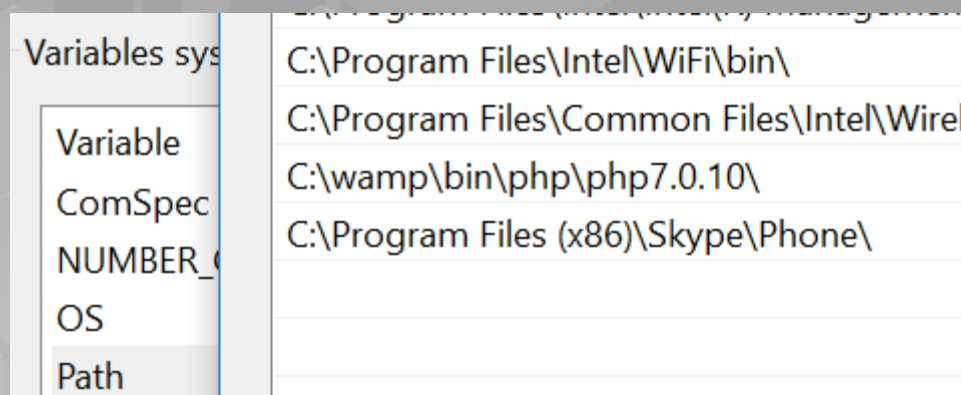
- **Populaire et puissant** (exemple de site : Dailymotion, exemple de CMS : Drupal 8)

- Les **développeurs** Symfony sont **recherchés par les entreprises**

- Plusieurs **conférences annuelles**

Prérequis pour une installation locale

- Installer WAMP (ou équivalent : EasyPHP, XAMP, MAMP..)
- Ajouter le PATH de **PHP 7** aux variables systèmes
 - Sous Windows :



Test de PHP pour une installation locale

- Tester avec la commande `php -v`

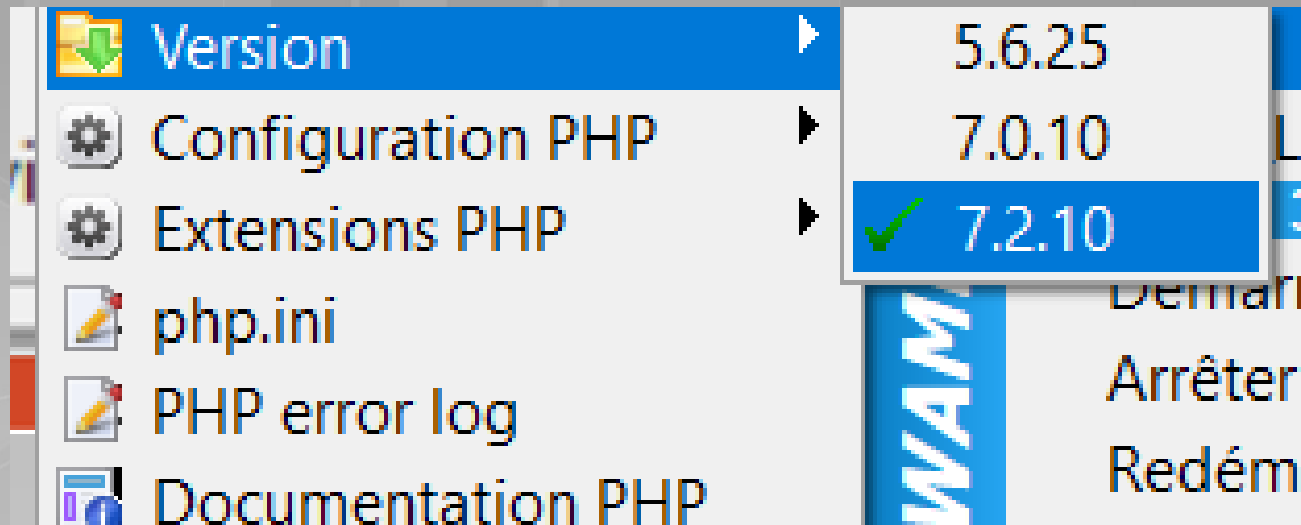
```
C:\Program Files\VCG\MeshLab>php -v
PHP 7.2.10 (cli) (built: Sep 13 2018 00:48:27) ( ZTS MSVC15 (Visual C++ 2017) x64 )
Copyright (c) 1997-2018 The PHP Group
Zend Engine v3.2.0, Copyright (c) 1998-2018 Zend Technologies
```

- Tester également la commande `php --ini`

```
C:\Program Files\VCG\MeshLab>php --ini
Configuration File (php.ini) Path: C:\WINDOWS
Loaded Configuration File:      C:\wamp64\bin\php\php7.2.10\php.ini
Scan for additional .ini files in: (none)
Additional .ini files parsed:    (none)
```

Activer PHP 7

- S'assurer que vous utilisez bien PHP7 dans WAMP



Installer Symfony 3

- Télécharger le fichier :

<https://curl.haxx.se/ca/cacert.pem> puis le copier dans votre répertoire PHP

- Et ajouter la ligne suivante dans le php.ini se trouvant dans votre répertoire PHP (ex: "C:\wamp\bin\php\php7.2.10\php.ini")

```
;opcache.opt_debug_level=0
[curl]
; A default value for the CURLOPT_CAINFO option. This is re
; absolute path.
curl.cainfo = C:\wamp64\bin\php\php7.2.10\cacert.pem

[openssl]
; The location of a Certificate Authority (CA) file on the
```

Installer GIT

- Il faut avoir **GIT** (logiciel de gestion de versions) installé
- Sous Windows, il faut utiliser **msysgit** : <https://git-for-windows.github.io/>
- Puis ajouter au Path Windows les chemins des binaires :

```
C:\Program Files\Git\mingw64\bin
```

```
C:\Program Files\Git\bin
```

- Sous Linux : `sudo apt-get install git-core`

Créer un projet Symfony

- Installer Composer : <https://getcomposer.org/download/>

- Installer la dernière version 3 de Symfony :

```
composer create-project symfony/framework-standard-edition Symfony "3.*"
```

```
C:\wamp64\www>composer create-project symfony/framework-standard-edition Symfony "3.*"
Installing symfony/framework-standard-edition (v3.4.17)
 - Installing symfony/framework-standard-edition (v3.4.17): Loading from cache
Created project in Symfony
Loading composer repositories with package information
Installing dependencies (including require-dev) from lock file
Package operations: 39 installs, 0 updates, 0 removals
 - Installing doctrine/lexer (v1.0.1): Loading from cache
 - Installing doctrine/annotations (v1.4.0): Loading from cache
 - Installing symfony/polyfill-ctype (v1.9.0): Loading from cache
 - Installing twig/twig (v1.35.4): Loading from cache
 - Installing paragonie/random_compat (v2.0.17): Loading from cache
 - Installing symfony/polyfill-php70 (v1.9.0): Loading from cache
 - Installing symfony/polyfill-util (v1.9.0): Loading from cache
 - Installing symfony/polyfill-php56 (v1.9.0): Loading from cache
 - Installing symfony/polyfill-mbstring (v1.9.0): Loading from cache
```

Créer un projet Symfony

- Entrer les paramètres de MySQL (database) selon votre configuration

```
> Incenteev\ParameterHandler\ScriptHandler::buildParameters
Creating the "app/config/parameters.yml" file
Some parameters are missing. Please provide them.
database_host (127.0.0.1):
database_port (null):
database_name (symfony):
database_user (root):
database_password (null):
mailer_transport (smtp):
mailer_host (127.0.0.1):
mailer_user (null):
mailer_password (null):
secret (ThisTokenIsNotSoSecretChangeIt):
> Sensio\Bundle\DistributionBundle\Composer\ScriptHandler::buildBootstrap
> Sensio\Bundle\DistributionBundle\Composer\ScriptHandler::clearCache

// Clearing the cache for the dev environment with debug true

[OK] Cache for the "dev" environment (debug=true) was successfully cleared.

> Sensio\Bundle\DistributionBundle\Composer\ScriptHandler::installAssets

Trying to install assets as relative symbolic links.

[OK] No assets were provided by any bundle.

> Sensio\Bundle\DistributionBundle\Composer\ScriptHandler::installRequirementsFile
> Sensio\Bundle\DistributionBundle\Composer\ScriptHandler::prepareDeploymentTarget
```


Créer un projet Symfony

- Tester l'URL :

http://localhost/Symfony/web/app_dev.php/

dans votre navigateur pour vérifier si tout est correct

Welcome to Symfony 3.4.17



Your application is now ready. You can start working on it at:

C : \wamp64\www\Symfony\

What's next?



Read the documentation to learn

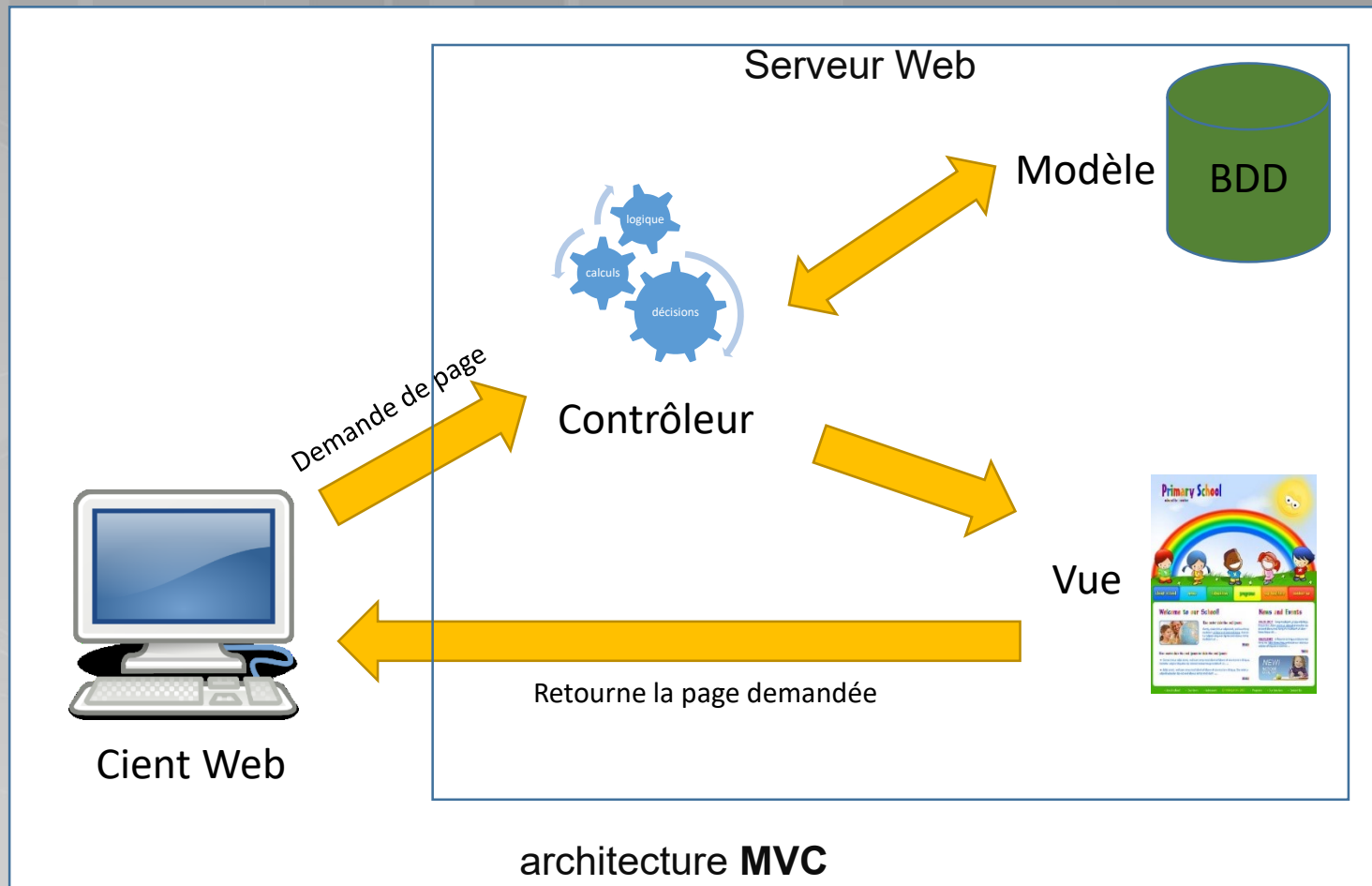
[How to create your first page in Symfony.](#)

L'architecture conceptuelle

- Symfony est architecturé MVC :
 - **Contrôleur** : son rôle est de générer la réponse à la requête HTTP demandée par un visiteur. Prend des décisions en analysant les données qu'il aura demandées puis reçues du modèle avant de les envoyer à la vue. Gère également les droits d'accès aux données.
 - **Modèle** : a pour rôle la récupération, l'organisation et l'assemblage des données stockées dans une base de données ou des fichiers. Permet au contrôleur de manipuler les articles, mais sans savoir comment les articles sont stockés, gérés..
 - **Vue** : a pour rôle d'afficher le contenu qu'elle reçoit sans avoir connaissance de la signification des données (exemple: messages d'un forum, articles en vente..)

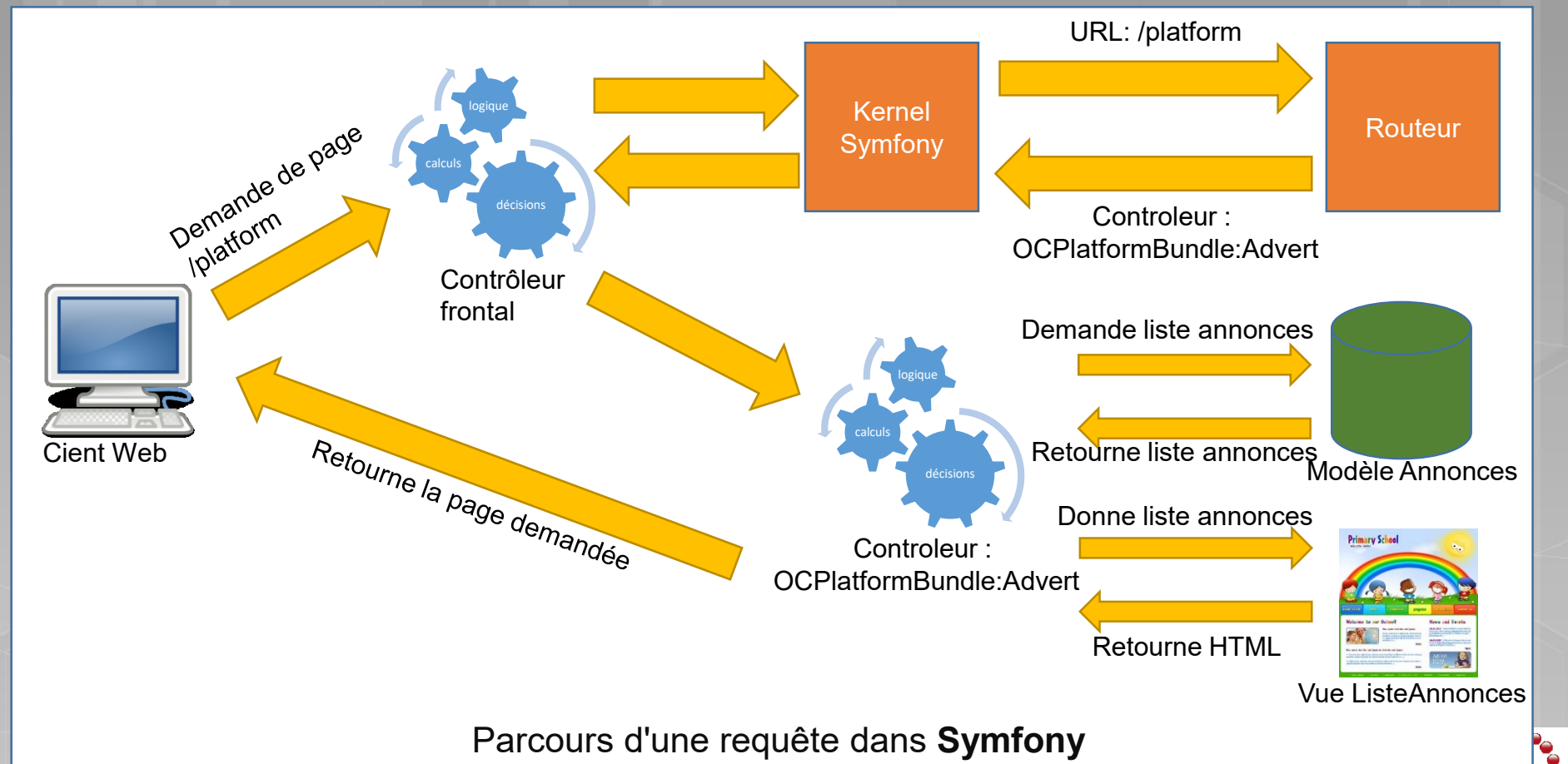
Demande d'une page Web

- Le client Web demandera la page au contrôleur et c'est la vue qui lui sera retournée



Parcours d'une requête dans Symfony

- Le visiteur demande la page /platform :



Parcours d'une requête dans Symfony

1. Le visiteur demande la page `/platform`
2. Le contrôleur frontal reçoit la requête, charge le Kernel et la lui transmet
3. Le **Kernel** demande au Routeur quel contrôleur exécuter pour l'URL `/platform`. Le Routeur est un composant Symfony qui fait la correspondance entre URL et contrôleurs. Le Routeur signale au Kernel qu'il faut exécuter le contrôleur `OCPlatformBundle:Advert`
4. Le Kernel exécute donc ce contrôleur. Le contrôleur demande au modèle `Annonce` la liste des annonces, puis la donne à la vue `ListeAnnonces` pour qu'elle construise la page HTML et la lui retourne. Une fois cela fini, le contrôleur envoie au visiteur la page HTML complète.

Note : la vue n'est pas limitée à retourner du HTML, elle peut par exemple renvoyer du JSON ou du XML

Organisation des fichiers

- Liste des répertoires :
 - **/app** : ressources communes au projet (vues, fichiers de traduction), le fichier PHP Kernel
 - **/bin** : scripts utiles durant le développement
 - **/src** : code source du site (*bundles*)
 - **/tests** : tests de regression (indépendant de Symfony)
 - **/var** : fichiers générés par Symfony (logs, cache..)
 - **/vendor** : libs externe au site (Doctrine, Twig..)
 - **/web** : contrôleur frontal (point d'accès au site), fichier robots.txt et .htaccess

Le contrôleur frontal


- Point d'entrée de l'application (i.e du site Web)
- Dans Symfony, le contrôleur frontal se situe dans le répertoire **/web**, il s'agit de `app.php` ou `app_dev.php`
 - `app.php` : contrôleur côté visiteurs (correspond à l'environnement de production) – Les erreurs sont écrites dans un fichier de log (`/var/logs/prod.log`)
 - `app_dev.php` : contrôleur côté développement (affiche des informations de debug pour les développeurs) - Les erreurs sont affichées à l'écran
- Testez les différences d'affichage :

<http://localhost/Symfony/web/app.php/pagequinexistepas>

http://localhost/Symfony/web/app_dev.php/pagequinexistepas

Le contrôleur frontal

<http://localhost/Symfony/web/app.php/pagequinexistepas>

 prod.log - Bloc-notes

Fichier Edition Format Affichage ?

```
[2018-10-29 09:22:36] request.ERROR: Uncaught PHP Exception Symfony\Component\HttpKernel\Exception\NotFoundHttpException: "Page 'pagequinexistepas' not found" at C:\wamp64\www\Symfony\vendor\symfony\symfony\src\Symfony\Component\HttpKernel\Exception\NotFoundHttpException.php:57  
{"exception": "[object] (Symfony\\Component\\HttpKernel\\Exception\\NotFoundHttpException(code: 0): Message: Page 'pagequinexistepas' not found at C:\\wamp64\\www\\Symfony\\vendor\\symfony\\symfony\\src\\Symfony\\Component\\HttpKernel\\EventListeners\\Routing\\Exception\\ResourceNotFoundException(code: 0): Message: Page 'pagequinexistepas' not found at C:\\wamp64\\www\\Symfony\\var\\cache\\p
```


Le contrôleur frontal

http://localhost/Symfony/web/app_dev.php/pagequinexistepas

Symfony Exception

Symfony Docs

Symfony Support

ResourceNotFoundException > NotFoundHttpException

HTTP 404 Not Found

No route found for "GET /pagequinexistepas"

Exception!

Exceptions 2Logs 1Stack Traces 2

Symfony\Component\HttpFoundation\Exception\NotFoundHttpException

in vendor\symfony\symfony\src\Symfony\Component\HttpFoundation\Event\RouterListener.php (line 139)

```
134.
135.         if ($referer = $request->headers->get('referer')) {
136.             $message .= sprintf(' (from "%s")', $referer);
137.         }
138.
139.         throw new NotFoundHttpException($message, $e);
140.     } catch (MethodNotAllowedException $e) {
141.         $message = sprintf('No route found for "%s %s": Method Not Allowed (Allow: %s)', $request->getMethod(), $request->getPat
142.
143.         throw new MethodNotAllowedHttpException($e->getAllowedMethods(), $message, $e);
144.     }
```

404723 ms2.0 MB111 in 19.79 msn/a433 ms

Symfony 3.4.17

Le contrôleur frontal

- D'un point de vue rôle, le contrôleur frontal appelle le noyau (*kernel*) de Symfony et lui passe la requête reçue
- Le noyau va alors traiter la requête

app.php

```
<?php

use Symfony\Component\HttpFoundation\Request;

/**
 * @var Composer\Autoload\ClassLoader
 */
$loader = require __DIR__.'../../app/autoload.php';
include_once __DIR__.'../../var/bootstrap.php.cache';

$kernel = new AppKernel('prod', false);
$kernel->loadClassCache();
//$kernel = new AppCache($kernel);

// When using the HttpCache, you need to call the method in your front c
//Request::enableHttpMethodParameterOverride();
$request = Request::createFromGlobals();
$response = $kernel->handle($request);
$response->send();
$kernel->terminate($request, $response);
```

- **Symfony** utilise l'architecture **MVC** pour bien organiser les différentes parties du code source.
- Symfony est organisé en **six répertoires** : `app`, `bin` , `src`, `var` , `vendor` **et** `web`.
- Il existe **deux environnements de travail** :
 - L'environnement « **prod** » destiné aux visiteurs : rapide à exécuter et ne divulgue pas les messages d'erreur.
 - L'environnement « **dev** » destiné aux développeurs: plus lent mais offre des d'informations utiles au développement.

Les bundles

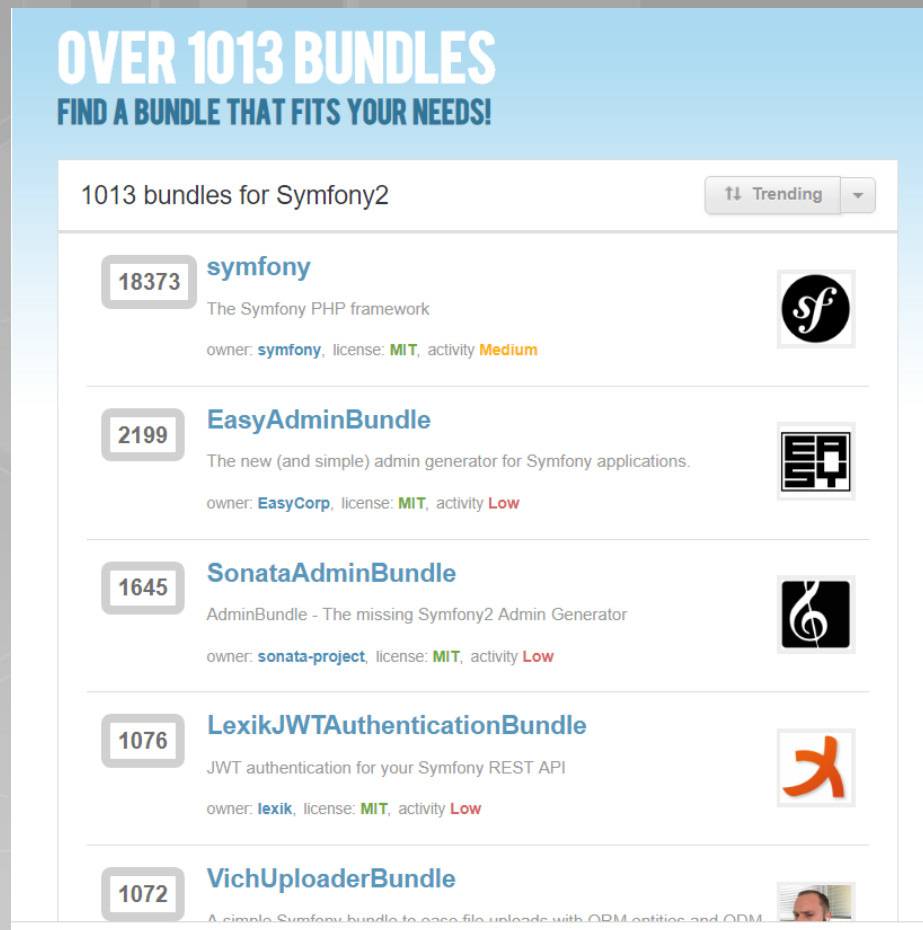
Les bundles

- un ***bundle*** est un package regroupant en un même endroit tout ce qui concerne une même **fonctionnalité**
 - Exemple : un *bundle* "blog" regroupant les contrôleurs, les modèles, les vues, les fichiers CSS et JavaScript, ...
- Un bundle peut utiliser d'autres bundles
 - Exemple : Le bundle "blog" pourrait utiliser le bundle "utilisateur" pour faire un lien vers les profils des auteurs des articles et des commentaires
- Intérêt : Partage et emploi de code
- Inconvénient : Peut engendrer un surcoût en terme de temps de développement






Les bundles de la communauté

- Plus de 1000 *bundles* sont disponibles ici :

<http://knpbundles.com/>



The screenshot displays the homepage of knpbundles.com, which features a light blue header with the text "OVER 1013 BUNDLES" and "FIND A BUNDLE THAT FITS YOUR NEEDS!". Below the header, a white box contains the text "1013 bundles for Symfony2" and a "Trending" dropdown menu. The main content area lists five bundles, each with a download count in a grey box, the bundle name, a description, owner, license, activity, and a logo.

Download Count	Bundle Name	Description	Owner	License	Activity	Logo
18373	symfony	The Symfony PHP framework	symfony	MIT	Medium	
2199	EasyAdminBundle	The new (and simple) admin generator for Symfony applications.	EasyCorp	MIT	Low	
1645	SonataAdminBundle	AdminBundle - The missing Symfony2 Admin Generator	sonata-project	MIT	Low	
1076	LexikJWTAuthenticationBundle	JWT authentication for your Symfony REST API	lexik	MIT	Low	
1072	VichUploaderBundle	A simple Symfony bundle to ease file uploads with ORM entities and ORM				

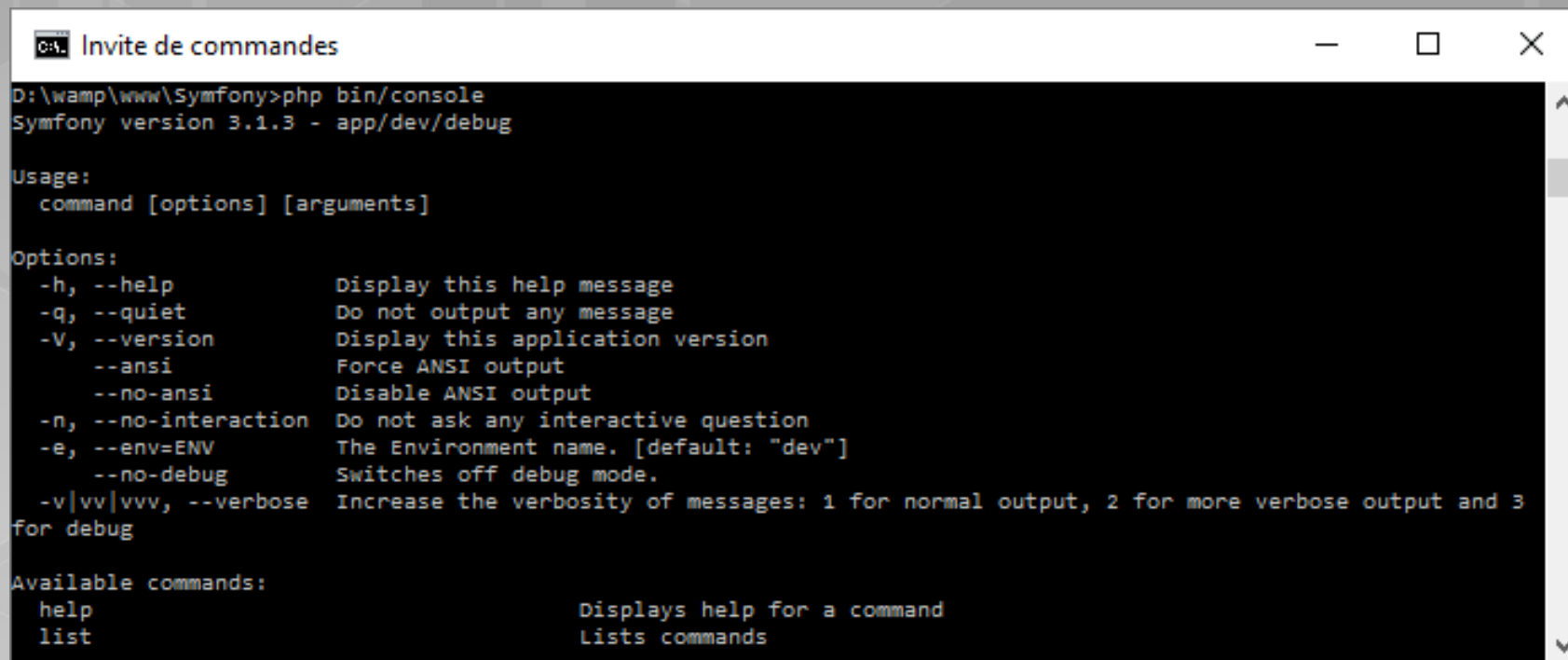
Structure d'un bundle

- La structure est la suivante :
 - `/src/controller` : Contient les contrôleurs
 - `/src/dependencyInjection` : Contient des informations sur le bundle (chargement automatique de la configuration par exemple)
 - `/src/entity` : Contient les modèles
 - `/src/form` : Contient les éventuels formulaires
 - `/src/resources`
 - `/config` : Contient les fichiers de configuration du bundle (les routes, par exemple)
 - `/public` : Contient les fichiers publics du bundle : fichiers CSS et JavaScript, images, etc.
 - `/views` : Contient les vues du bundle, les templates Twig

La console

- Symfony intègre des commandes disponibles via une console dédiée :

`php bin/console`



```
C:\> Invite de commandes

D:\wamp\www\Symfony>php bin/console
Symfony version 3.1.3 - app/dev/debug

Usage:
  command [options] [arguments]

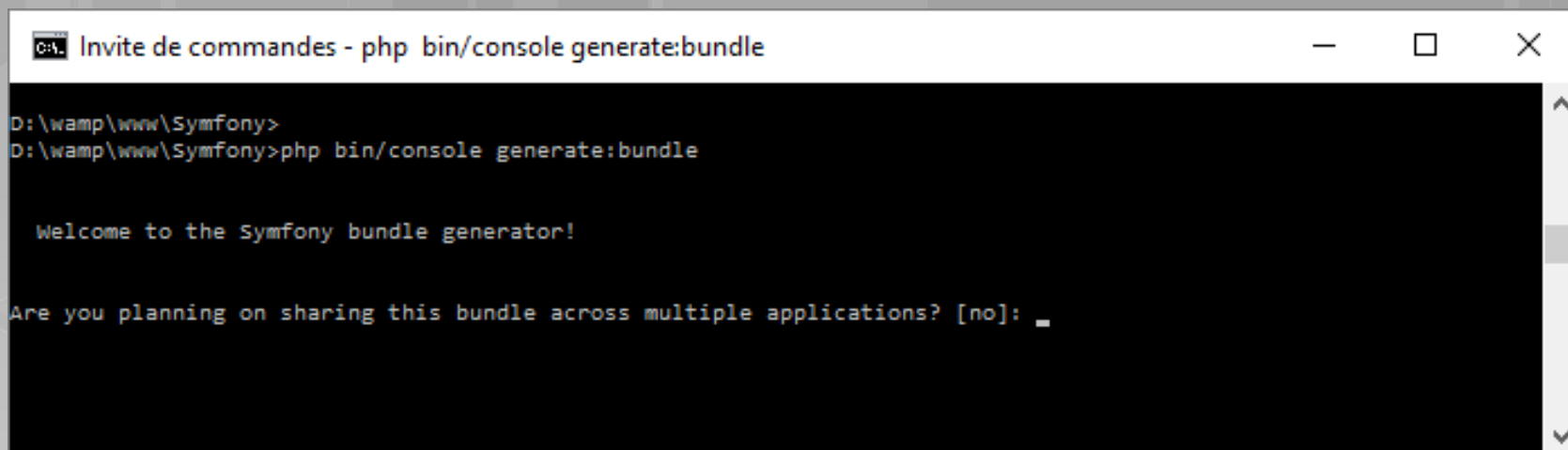
Options:
  -h, --help                Display this help message
  -q, --quiet               Do not output any message
  -V, --version             Display this application version
      --ansi                Force ANSI output
      --no-ansi             Disable ANSI output
  -n, --no-interaction      Do not ask any interactive question
  -e, --env=ENV             The Environment name. [default: "dev"]
      --no-debug            Switches off debug mode.
  -v|vv|vvv, --verbose     Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and 3 for debug

Available commands:
  help                Displays help for a command
  list                Lists commands
```


Création d'un bundle en utilisant la console

- La commande est la suivante :

```
php bin/console generate:bundle
```



```
Invite de commandes - php bin/console generate:bundle

D:\wamp\www\Symfony>
D:\wamp\www\Symfony>php bin/console generate:bundle

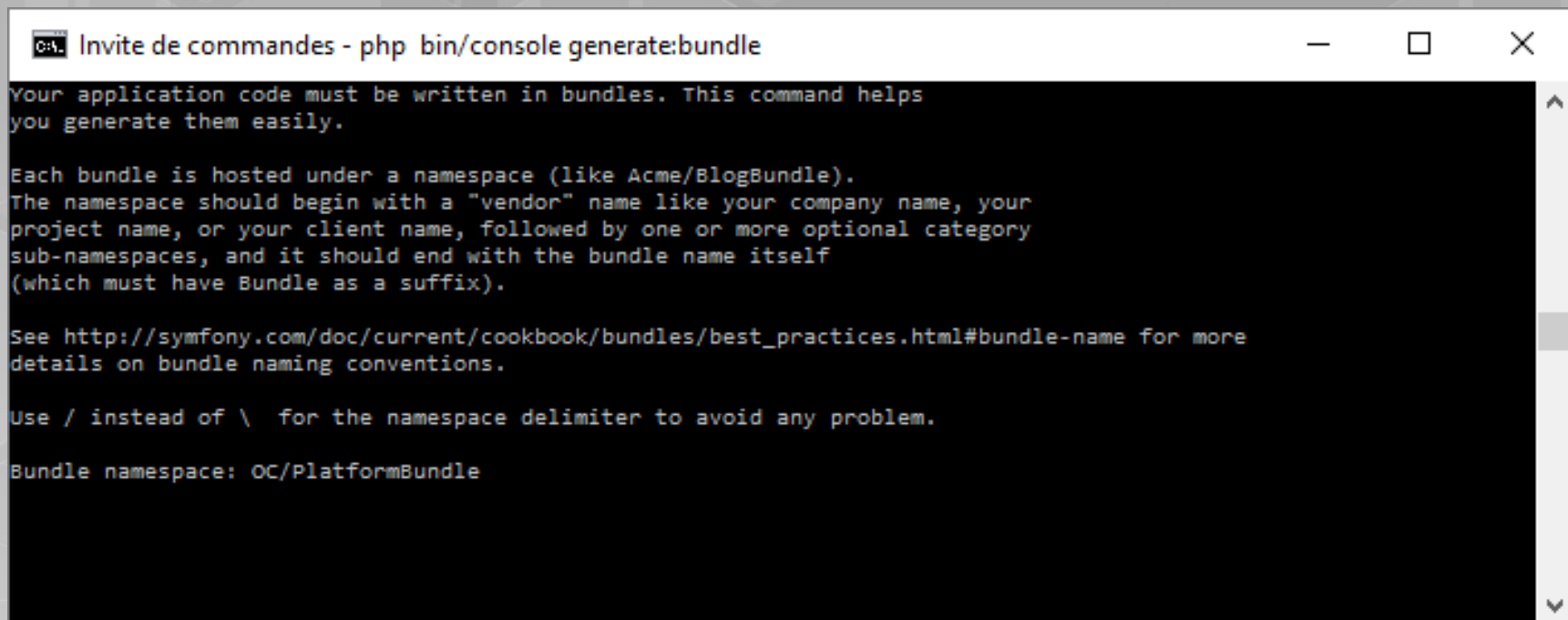
Welcome to the Symfony bundle generator!

Are you planning on sharing this bundle across multiple applications? [no]: _
```

- Répondez par `yes`

Choix du namespace

- Il faut que le nom du bundle se termine par le suffixe `Bundle`
- Il faut qu'il soit préfixer par un nom de *Vendor*
- Nous allons nommer notre namespace : `OC/PlatformBundle`



```
0% Invite de commandes - php bin/console generate:bundle

Your application code must be written in bundles. This command helps
you generate them easily.

Each bundle is hosted under a namespace (like Acme/BlogBundle).
The namespace should begin with a "vendor" name like your company name, your
project name, or your client name, followed by one or more optional category
sub-namespaces, and it should end with the bundle name itself
(which must have Bundle as a suffix).

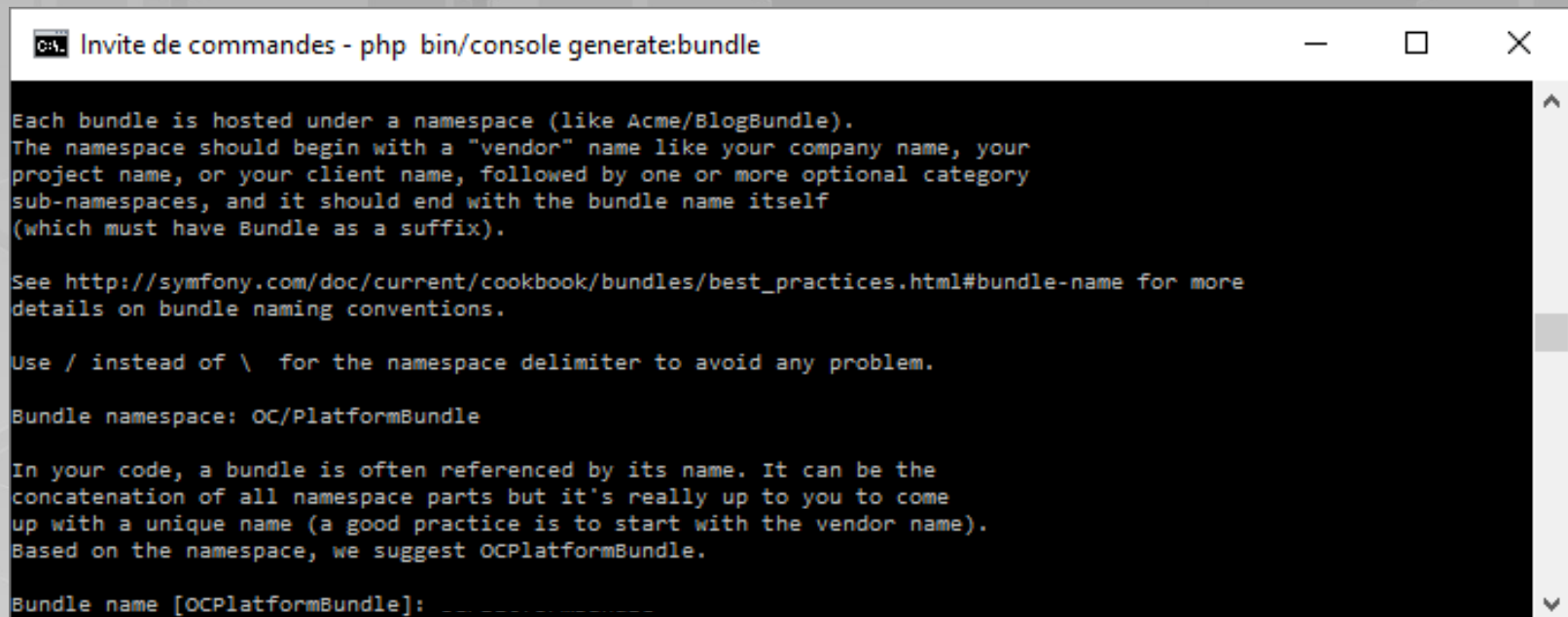
See http://symfony.com/doc/current/cookbook/bundles/best\_practices.html#bundle-name for more
details on bundle naming conventions.

Use / instead of \ for the namespace delimiter to avoid any problem.

Bundle namespace: OC/PlatformBundle
```

Choix du nom

- Par convention, on nomme le bundle de la même manière que le namespace, sans les slashes. Validez le choix par défaut (OCPlatformBundle) par Entrée.



```
Invite de commandes - php bin/console generate:bundle

Each bundle is hosted under a namespace (like Acme/BlogBundle).
The namespace should begin with a "vendor" name like your company name, your
project name, or your client name, followed by one or more optional category
sub-namespaces, and it should end with the bundle name itself
(which must have Bundle as a suffix).

See http://symfony.com/doc/current/cookbook/bundles/best_practices.html#bundle-name for more
details on bundle naming conventions.

Use / instead of \ for the namespace delimiter to avoid any problem.

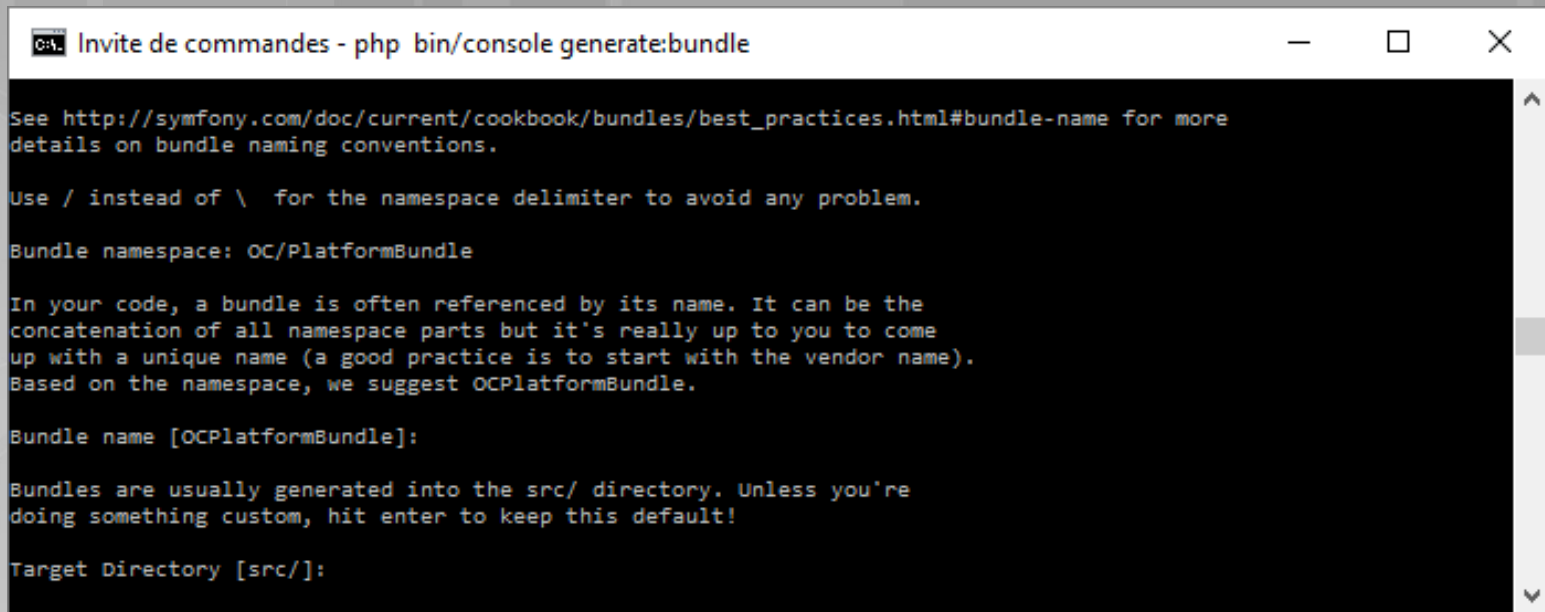
Bundle namespace: OC/PlatformBundle

In your code, a bundle is often referenced by its name. It can be the
concatenation of all namespace parts but it's really up to you to come
up with a unique name (a good practice is to start with the vendor name).
Based on the namespace, we suggest OCPlatformBundle.

Bundle name [OCPlatformBundle]:
```

Choix de la destination

- C'est l'endroit où se trouvera le bundle – Validez le choix par défaut (`/src`) par Entrée



```
Invite de commandes - php bin/console generate:bundle

See http://symfony.com/doc/current/cookbook/bundles/best\_practices.html#bundle-name for more
details on bundle naming conventions.

Use / instead of \ for the namespace delimiter to avoid any problem.

Bundle namespace: OC/PlatformBundle

In your code, a bundle is often referenced by its name. It can be the
concatenation of all namespace parts but it's really up to you to come
up with a unique name (a good practice is to start with the vendor name).
Based on the namespace, we suggest OCPlatformBundle.

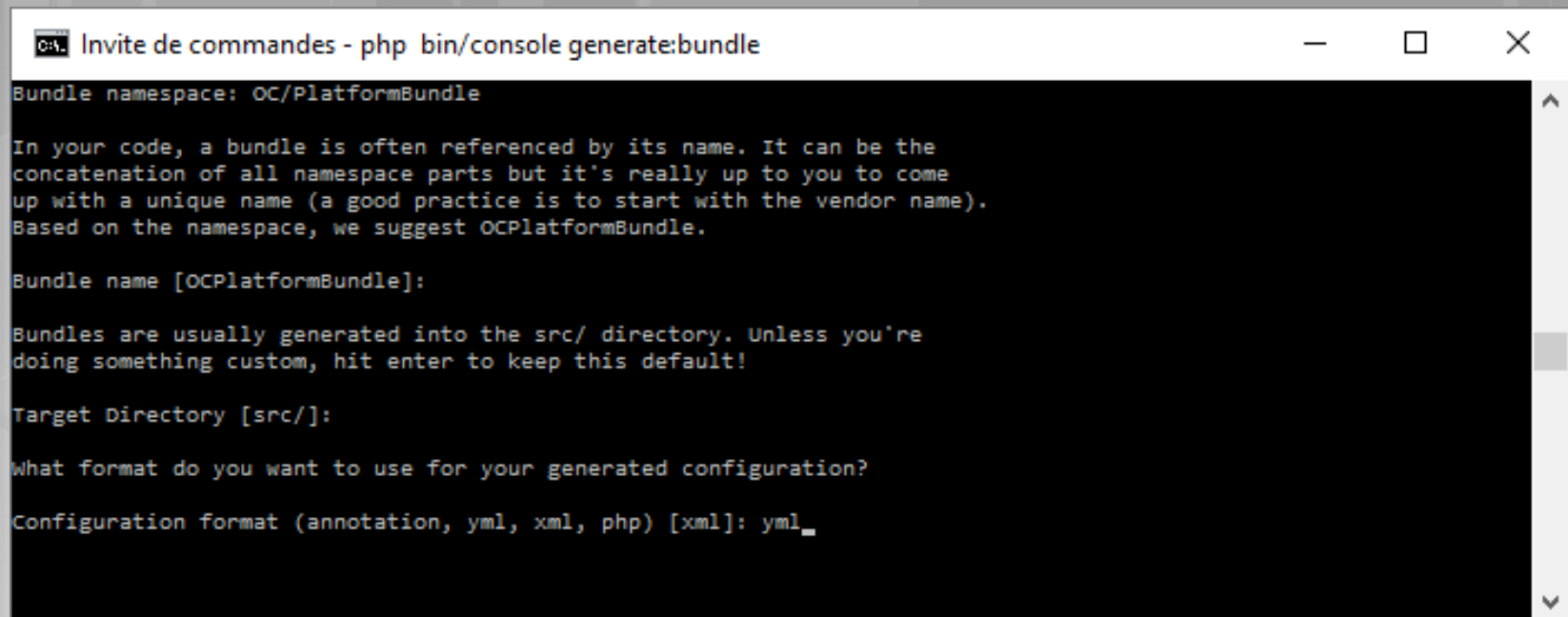
Bundle name [OCPlatformBundle]:

Bundles are usually generated into the src/ directory. Unless you're
doing something custom, hit enter to keep this default!

Target Directory [src/]:
```

Choix du format de configuration

- Il existe plusieurs formats de configuration : Choisissons le YAML (yml)



```
CA\ Invite de commandes - php bin/console generate:bundle

Bundle namespace: OC/PlatformBundle

In your code, a bundle is often referenced by its name. It can be the
concatenation of all namespace parts but it's really up to you to come
up with a unique name (a good practice is to start with the vendor name).
Based on the namespace, we suggest OCPlatformBundle.

Bundle name [OCPlatformBundle]:


Bundles are usually generated into the src/ directory. Unless you're
doing something custom, hit enter to keep this default!

Target Directory [src/]:

What format do you want to use for your generated configuration?

Configuration format (annotation, yml, xml, php) [xml]: yml_
```

- Vous devriez avoir cette erreur :



```
> Generating a sample bundle skeleton into C:\wamp64\www\Symfony\app\..\src\OC\PlatformBundle
created .\app\..\src\OC\PlatformBundle/
created .\app\..\src\OC\PlatformBundle\OCPlatformBundle.php
created .\app\..\src\OC\PlatformBundle\DependencyInjection/
created .\app\..\src\OC\PlatformBundle\DependencyInjection\OCPlatformExtension.php
created .\app\..\src\OC\PlatformBundle\DependencyInjection\Configuration.php
created .\app\..\src\OC\PlatformBundle\Controller/
created .\app\..\src\OC\PlatformBundle\Controller\DefaultController.php
created .\app\..\src\OC\PlatformBundle\Tests\Controller/
created .\app\..\src\OC\PlatformBundle\Tests\Controller\DefaultControllerTest.php
created .\app\..\src\OC\PlatformBundle\Resources\views\Default/
created .\app\..\src\OC\PlatformBundle\Resources\views\Default\index.html.twig
created .\app\..\src\OC\PlatformBundle\Resources\config/
created .\app\..\src\OC\PlatformBundle\Resources\config\services.yml
created .\app\..\src\OC\PlatformBundle\Resources\config\routing.yml
> Checking that the bundle is autoloaded
FAILED
> Enabling the bundle inside C:\wamp64\www\Symfony\app\AppKernel.php
updated .\app\AppKernel.php
OK
> Importing the bundle's routes from the C:\wamp64\www\Symfony\app\config\routing.yml file
updated .\app\config\routing.yml
OK
```

The command was not able to configure everything automatically.
You'll need to make the following changes manually.

- Edit the `composer.json` file and register the bundle namespace in the "autoload" section:

Bug autoload bundle

- Les versions 3.2 à 3.4 contiennent le bug :
<https://github.com/sensiolabs/SensioGeneratorBundle/issues/555>

Conséquence du bug autoload bundle

- Les versions 3.2 à 3.4 contiennent le bug :

<https://github.com/sensiolabs/SensioGeneratorBundle/issues/555>

- Le bundle n'est pas chargé et ne fonctionne donc pas :

http://localhost/Symfony/web/app_dev.php/

Whoops, looks like something went wrong.

1/1 **ClassNotFoundException** in AppKernel.php line 19:


Attempted to load class "OCPlatformBundle" from namespace "OC\PlatformBundle".
Did you forget a "use" statement for another namespace?

1. in AppKernel.php line 19
2. at AppKernel->registerBundles() in Kernel.php line 406
3. at Kernel->initializeBundles() in Kernel.php line 113
4. at Kernel->boot() in Kernel.php line 165
5. at Kernel->handle(object(Request)) in app_dev.php line 27

Workaround pour le bug autoload bundle

- Néanmoins on peut le corriger en éditant le fichier composer.json de la manière suivante :

```
"license": "proprietary",
"type": "project",
"autoload": {
    "psr-4": {
        "AppBundle\\": "src/AppBundle",
        "OC\\PlatformBundle\\": "src/OC/PlatformBundle"
    },
    "classmap": [
        "app/AppKernel.php",
        "app/AppCache.php"
    ]
},
"autoload-dev": {
    "psr-4": {
```



Workaround pour le bug autoload bundle

- Puis exécutez la commande suivante :

```
php -r "eval('?'>'.file_get_contents('http://getcomposer.org/installer'));"
```

```
C:\wamp64\www\Symfony>php -r "eval('?'>'.file_get_contents('http://getcomposer.org/installer'));"  
All settings correct for using Composer  
Downloading...  
  
Composer (version 1.5.2) successfully installed to: C:\wamp64\www\Symfony\composer.phar  
Use it: php composer.phar
```

- Et :

```
php composer.phar dump-autoload
```

```
C:\wamp64\www\Symfony>php composer.phar dump-autoload  
Generating autoload files
```

Arborescence créée

wamp > www > Symfony > src >

	Nom	Modifié le	Type
	AppBundle	02/09/2016 17:22	Dossier de fichiers
	OC	05/09/2016 17:29	Dossier de fichiers
	.htaccess	02/09/2016 17:22	Fichier HTACCESS

wamp > www > Symfony > src > OC > PlatformBundle >

	Nom	Modifié le	Type
	Controller	05/09/2016 17:29	Dossier de fichiers
	DependencyInjection	05/09/2016 17:29	Dossier de fichiers
	Resources	05/09/2016 17:29	Dossier de fichiers
	Tests	05/09/2016 17:29	Dossier de fichiers
	OCPlatformBundle.php	05/09/2016 17:29	Fichier PHP

Note : le seul fichier obligatoire est la classe `OCPlatformBundle.php` à la racine du répertoire

Bundle et kernel

- Le bundle a été enregistré dans le kernel (fichier `app/AppKernel.php`)

```
<?php

use Symfony\Component\HttpKernel\Kernel;
use Symfony\Component\Config\Loader\LoaderInterface;

class AppKernel extends Kernel
{
    public function registerBundles()
    {
        $bundles = [
            new Symfony\Bundle\FrameworkBundle\FrameworkBundle(),
            new Symfony\Bundle\SecurityBundle\SecurityBundle(),
            new Symfony\Bundle\TwigBundle\TwigBundle(),
            new Symfony\Bundle\MonologBundle\MonologBundle(),
            new Symfony\Bundle\SwiftmailerBundle\SwiftmailerBundle(),
            new Doctrine\Bundle\DoctrineBundle\DoctrineBundle(),
            new Sensio\Bundle\FrameworkExtraBundle\SensioFrameworkExtraBundle(),
            new AppBundle\AppBundle(),
            new OC\PlatformBundle\OCPlatformBundle(),
        ];
    }
}
```

- Cette classe permet uniquement de définir quels bundles charger pour l'application

Bundle et kernel

- La classe `OCPlatformBundle` se trouve dans le fichier :

`\src\OC\PlatformBundle\OCPlatformBundle.php`

```
<?php

namespace OC\PlatformBundle;

use Symfony\Component\HttpKernel\Bundle\Bundle;

class OCPlatformBundle extends Bundle
{
}
```

- On notera la définition de l'espace de nom `OC\PlatformBundle` déclaré au moment de la création du bundle

Bundle et environnement de développement

- On trouve également d'autres bundles chargés uniquement pour l'environnement de développement

```
];  
  
if (in_array($this->getEnvironment(), ['dev', 'test'], true)) {  
    $bundles[] = new Symfony\Bundle\DebugBundle\DebugBundle();  
    $bundles[] = new Symfony\Bundle\WebProfilerBundle\WebProfilerBundle();  
    $bundles[] = new Sensio\Bundle\DistributionBundle\SensioDistributionBundle();  
    $bundles[] = new Sensio\Bundle\GeneratorBundle\SensioGeneratorBundle();  
}  
  
return $bundles;  
}
```

- Symfony a également enregistré la route, c'est-à-dire le contrôleur à exécuter en fonction de l'URL appelée
`src/OC/PlatformBundle/Resources/config/routing.yml`

Le bundle ne dispose par défaut qu'une seule route

```
oc_platform_homepage:
    path:      /
    defaults: { _controller: OCPlatformBundle:Default:index }
```

- Il faut aussi indiquer au routeur où aller chercher la route à utiliser dans le fichier : `app/config/routing.yml`

```
oc_platform:
    resource: "@OCPlatformBundle/Resources/config/routing.yml"
    prefix:   /

app:
    resource: "@AppBundle/Controller/"
    type:     annotation
```

Contrôleur par défaut du bundle

- Symfony a créé un contrôleur par défaut :

`\src\OC\PlatformBundle\Controller\DefaultController.php`

```
<?php

namespace OC\PlatformBundle\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\Controller;

class DefaultController extends Controller{
    public function indexAction()    {
        return $this->render('OCPlatformBundle:Default:index.html.twig');
    }
}
```

Note : avec Symfony 3.4 ajoutez cette ligne dans le fichier : `\app\config\config.yml`

```
framework:
    templating:
        engines: ['twig']
```


Test depuis le navigateur

- Le bundle est donc déjà opérationnel et affiche "Hello World" :

http://localhost/Symfony/web/app_dev.php/

- Remarque : la toolbar a disparue car le html ne contient aucune balise. Modifiez le fichier `src/OC/PlatformBundle/Resources/views/Default/index.html.twig` comme suit :

```
{# src/OC/PlatformBundle/Resources/views/Default/index.html.twig #}  
<html>  
  <body>  
    Hello World!  
  </body>  
</html>
```

Note : encodage du fichier en « UTF-8 sans BOM »

- La toolbar s'affiche

200

@ oc_platform_homepage

412 ms

17.0 MB



anon.



4 ms

- Un **bundle** est une **brique de votre application** : il contient tout ce qui concerne une fonctionnalité donnée. Cela permet de bien organiser les différentes parties du site.
- Pour qu'un bundle soit opérationnel, il faut :
 - Son code source, situé dans `src/Application/Bundle`, et dont le seul fichier obligatoire est la classe à la racine `OCPlatformBundle.php`
 - Enregistrer le bundle dans le noyau pour qu'il soit chargé, en modifiant le fichier `app/AppKernel.php`
 - Enregistrer les routes (si le bundle en contient) dans le Routeur pour qu'elles soient chargées, en modifiant le fichier `app/config/routing.yml`
 - Ces trois points sont effectués automatiquement lorsqu'on utilise le générateur