



Symfony : Présentation

Benoît Roche
@rocheb83



GÉNÉRALITÉS



GÉNÉRALITÉS



Les concepts :

Symfony est un framework de développement web en PHP.

Symfony est une Philosophie, et une communauté - le tout dans un ensemble harmonique.

Symfony a des concurrents : zend, cake, codeigniter,...



GÉNÉRALITÉS



Avertissement

Ce diaporama :

a pour objectif de présenter les concepts qui seront abordés dans l'apprentissage de symfony.
Il s'agit donc de l'aspect théorique.

Doit être étudié avec attention pour bien comprendre la suite. On ne peut faire l'économie de ne pas le lire

Est accompagné d'une série d'exercices permettant de faire un tour d'horizon de symfony



GÉNÉRALITÉS



Un Framework ?



Ça sert à ne pas réinventer la roue ...

Un framework est un ensemble de composants destinés à favoriser la productivité d'un développement logiciel.

Il est constitué de briques logicielles pouvant interagir entre elles.
modèle standardisé

les développeurs utilisent des fonctionnalités prêtes à l'emploi bien rodées

Investir en temps sur un framework c'est gagner en productivité dans son développement et sa maintenance

Benoît Roche / @rocheb83

5



GÉNÉRALITÉS



Symfony ?

Très utilisé dans l'industrie du développement web

Très forte communauté

Beaucoup de tutoriels et de documentations

Orienté objet, couplé à Doctrine pour la persistance des données

Respecte l'architecture MVC



Il est français ...

Benoît Roche / @rocheb83

6



GÉNÉRALITÉS



Symfony ?... Quelques références

Dailymotion
Yahoo
OpenSky
PhpBB (application web)
OpenClassroom
Drupal
Laravel
eZ Publish Community
...

Benoît Roche / @rocheb83

7



PRÉSENTATION DE SYMFONY

Benoît Roche / @rocheb83

8



PRÉSENTATION DE SYMFONY



Symfony intègre des briques logicielles déjà éprouvées et reconnues dans le développement web :

Twig : comme moteur de templates

Doctrine : pour la couche de persistance des données

Composer : pour la gestion des dépendances

Etc...

Symfony est également prévu pour être utilisé dans les principales IDE du marché :

Eclipse

Netbeans



PRÉSENTATION DE SYMFONY



MVC ? : une Architecture logicielle découpée en trois parties totalement indépendantes :


Model : représente les données et les règles métiers. C'est dans ce composant que s'effectuent les traitements fonctionnels. Les données peuvent être liées à une base de données, des services Web, ...

Le model restitue en général les données sous forme d'objets. La brique supplémentaire : ORM (Object Relational Mapping). **Doctrine** avec *symfony*


View : s'occupe des interactions avec l'utilisateur : présentation des informations, saisie et validation des données.

Dans une applications Web, il s'agit généralement d'une page HTML,

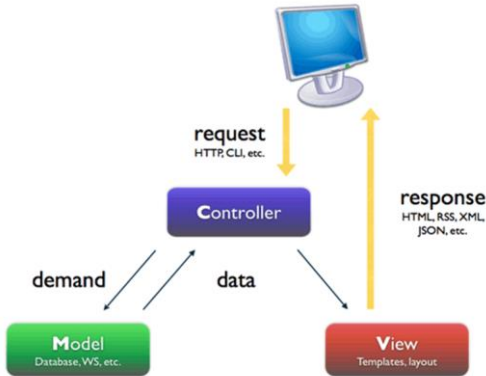
Controller : se charge d'intercepter les requêtes de l'utilisateur, d'appeler le modèle puis de rediriger vers la vue adéquate. Il ne doit faire aucun traitement. Il ne fait que de l'interception et de la redirection.



PRÉSENTATION DE SYMFONY



MVC ? Schéma général




The diagram illustrates the general MVC pattern. At the top, a computer icon represents the user interface. A yellow arrow labeled 'request' (with subtext 'HTTP, CLI, etc.') points from the user to the 'Controller' box. From the 'Controller', a yellow arrow labeled 'response' (with subtext 'HTML, RSS, XML, JSON, etc.') points back to the user. Below the Controller, two boxes represent the 'Model' (green, with subtext 'Database, WS, etc.') and the 'View' (red, with subtext 'Templates, layout'). A double-headed arrow labeled 'demand' and 'data' connects the Controller to the Model. Another double-headed arrow connects the Controller to the View.

En MVC2 : un seul Contrôleur qui intercepte toutes les requêtes utilisateurs


Le contrôleur ne contient donc que peu de code, car il se contente d'utiliser des modèles et des vues en leur attribuant des tâches précises. Il s'agit de la tour de contrôle de l'application.

Benoît Roche / @rocheb83

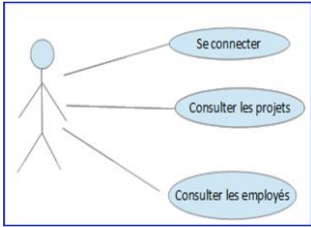
11




PRÉSENTATION DE SYMFONY



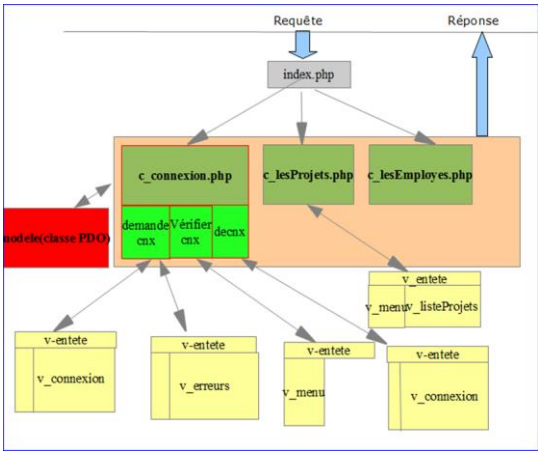
MVC ? : Un exemple



A stick figure represents a user with three actions: 'Se connecter', 'Consulter les projets', and 'Consulter les employés'.




Exercice: Repérer les couches




The diagram shows a detailed MVC example. At the top, 'Requête' (Request) and 'Réponse' (Response) are indicated. The flow starts at 'index.php', which points to three controllers: 'c_connexion.php', 'c_lesProjets.php', and 'c_lesEmployes.php'. 'c_connexion.php' is further divided into 'demande', 'Vérifier', and 'decnx' sub-components. A red box labeled 'code(class PDO)' points to the 'demande' sub-component. Arrows from the controllers point to various view components: 'v-entete', 'v_connexion', 'v_erreurs', 'v_menu', and 'v_connexion' (repeated). Some views like 'v_erreur' and 'v_connexion' are shown as stacked boxes. The flow ends with an arrow pointing to 'Réponse'.

Benoît Roche / @rocheb83

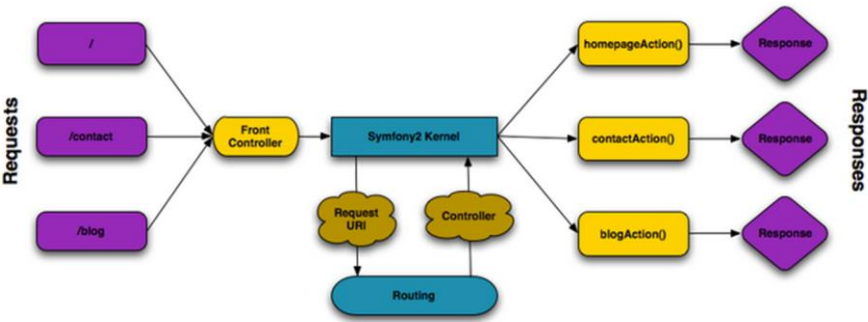
12




PRÉSENTATION DE SYMFONY




MVC ? : Application à Symfony



Les requêtes entrantes sont interprétées par le routing et passées aux fonctions des contrôleurs qui retournent des objets Response.



PRÉSENTATION DE SYMFONY

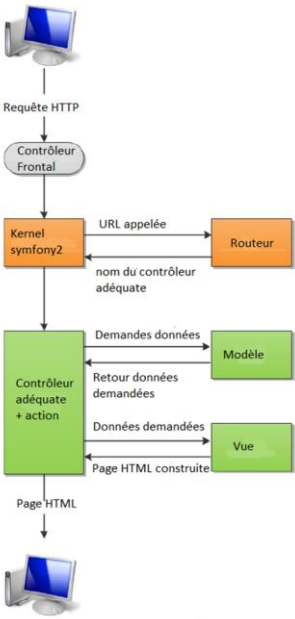



MVC ? : Application à Symfony

Le contrôleur frontal intercepte toutes les requêtes


Il charge le noyau qui demande au routeur de lui fournir le contrôleur à charger et l'action à exécuter avec les paramètres fournis par l'url

Le kernel charge le contrôleur et exécute la méthode (action) demandée avec les paramètres





PRÉSENTATION DE SYMFONY



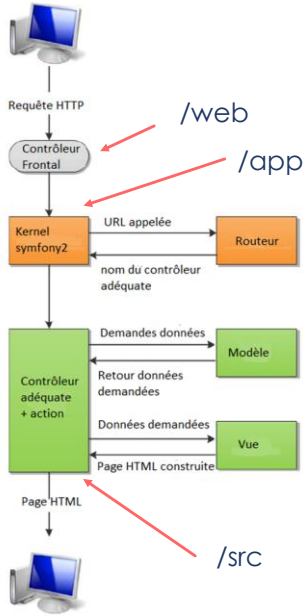
MVC ? : Application à Symfony

le contrôleur :

- appelle le modèle pour récupérer les données
- charge la vue et lui transmet les données récupérées du modèle

La vue les met en forme et retourne la page HTML ainsi construite


Le contrôleur la renvoie à l'utilisateur qui en a fait la demande




The diagram illustrates the MVC flow in Symfony. It starts with a 'Requête HTTP' (HTTP Request) from a browser to the 'Contrôleur Frontal' (Front Controller). The Front Controller then passes the request to the 'Kernel symfony2'. The 'Kernel' interacts with the 'Routeur' (Router) to determine the 'URL appelée' (called URL) and the 'nom du contrôleur adéquate' (name of the appropriate controller). The 'Kernel' then calls the 'Contrôleur adéquate + action' (Appropriate controller + action). This controller interacts with the 'Modèle' (Model) to get 'Demandes données' (data requests) and return 'Retour données demandées' (requested data). It also interacts with the 'Vue' (View) to get 'Données demandées' (requested data) and return a 'Page HTML construite' (constructed HTML page). Finally, the 'Page HTML' is sent back to the browser. Red arrows indicate the file locations: '/web' for the Front Controller, '/app' for the Kernel, and '/src' for the Controller and View.

Benoît Roche / @rocheb83

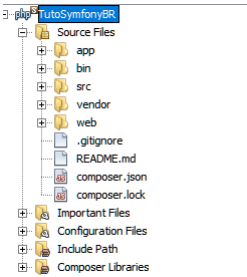
15



PRÉSENTATION DE SYMFONY




Structure d'un projet



The tree shows the project structure: Source Files (app, bin, src, vendor, web), .gitignore, README.md, composer.json, composer.lock, Important Files, Configuration Files, Include Path, and Composer Libraries.


/app	Contient l'environnement du projet. Tous les fichiers communs à l'application tels les fichiers de configuration, le cache, les fichiers logs et les fichiers de routage
/src	Contient le code source de l'application web. Il contiendra l'ensemble des contrôleurs, les vues et les modèles. Il sera contiendra le bundle par défaut AppBundle plus éventuellement d'autres Bundles.




On verra plus loin l'organisation en Bundles

Benoît Roche / @rocheb83

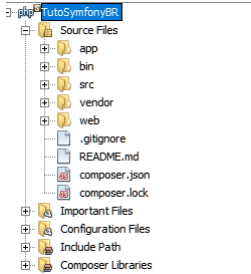
16




PRÉSENTATION DE SYMFONY




Structure d'un projet




/vendor	Contient tous les bundles externes à l'application fournis par Symfony, y compris un Bundle Symfony. Entre autres, on remarque les Bundles Doctrine, Twig,...
/web	Contient tous les fichiers qui doivent être visibles de l'extérieur : images, fichiers css, JavaScript. Il contient aussi et surtout le contrôleur frontal de l'application : fichier app.php (en production), fichier app_dev.php (en développement)

 On verra plus loin l'organisation en Bundles

Benoît Roche / @rocheb8317



PRÉSENTATION DE SYMFONY



Bundle ? Vous avez dit Bundle ?

un répertoire qui a une structure bien définie


Il peut héberger des classes , des contrôleurs, des ressources web

C'est aussi un espace de noms PHP auquel on a ajouté une classe Bundle


On peut en avoir plusieurs par projet

On peut en télécharger ex : FOSUser pour la gestion des utilisateurs

Benoît Roche / @rocheb8318



PRÉSENTATION DE SYMFONY



Bundle ? Vous avez dit Bundle ?

Un Bundle est un ensemble structuré de répertoires contenant les fichiers nécessaires à la gestion d'une application ou d'une partie d'une application


Un Bundle peut héberger des classes , des contrôleurs, des ressources web

C'est aussi un espace de noms PHP auquel on a ajouté une classe Bundle


Depuis la version 2.5, Symfony recommande l'utilisation du bundle AppBundle créée lors de la création d'un projet symfony

Benoît Roche / @rocheb83

19



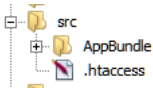
PRÉSENTATION DE SYMFONY



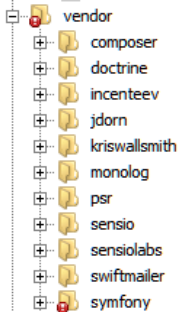
Bundle ? Vous avez dit Bundle ?

On peut en avoir plusieurs par projet :

- ✓ Les bundles propres à l'application en plus de l'AppBundle (répertoire /src)



- ✓ Les bundles téléchargés (répertoire /vendor)



Benoît Roche / @rocheb83

20



PRÉSENTATION DE SYMFONY



Le contrôleur frontal

C'est le point d'entrée de l'application

Toutes les demandes de page passent par ce fichier

Situé dans le dossier /web

Sa mission sera de charger le kernel (noyau) qui se chargera de trouver la bonne route pour renvoyer la bonne page

En fait, il y en a deux !!!

- Celui de production (app.php)
- Celui de développement (app_dev.php)

Benoît Roche / @rocheb83

21



PRÉSENTATION DE SYMFONY



Le contrôleur frontal

Pourquoi 2 contrôleurs frontaux ????

Fonctionnellement ils font la même chose

Il n'y a pas redondance de code

En développement, on utilisera **TOUJOURS** l'environnement dev


En production, on utilisera **TOUJOURS** l'environnement prod

Mais pourquoi ?


Parce que l'environnement de développement affiche en bas de la page une toolbar fournissant au développeur des outils de débogage

Benoît Roche / @rocheb83

22



PRÉSENTATION DE SYMFONY



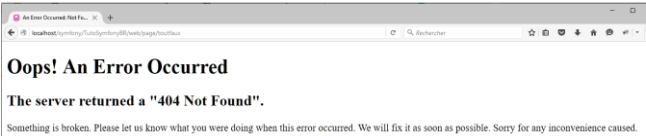
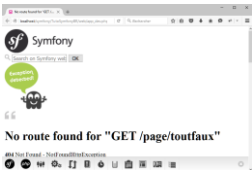
Le contrôleur frontal

Pourquoi 2 contrôleurs frontaux ???

Petit test


Avec une page qui existe : http://symfony.br/TutoSymfonyBR/web/app_dev.php/hello/le_dévelop
http://symfony.br/TutoSymfonyBR/web/app_dev.php/hello/la_production

Avec une page qui n'existe pas: http://localhost/symfony/TutoSymfonyBR/web/app_dev.php/page/toutfaux
<http://localhost/symfony/TutoSymfonyBR/web/app.php/page/toutfaux>




Benoît Roche / @rocheb83

23



PRÉSENTATION DE SYMFONY



Les journaux de log

En production, on pourra récupérer les erreurs dans le fichier app/logs/prod.log



```
prod.log
1 [2016-02-20 10:30:15] request.ERROR: Uncaught PHP Exception Symfony\Component\HttpKernel\Exce
2 [2016-02-20 10:30:15] request.ERROR: Uncaught PHP Exception Symfony\Component\HttpKernel\Exce
3 [2016-02-20 10:30:15] request.ERROR: Uncaught PHP Exception Symfony\Component\HttpKernel\Exce
4 [2016-02-20 10:33:06] request.ERROR: Uncaught PHP Exception Symfony\Component\HttpKernel\Exce
```

En développement, on pourra récupérer les erreurs dans le fichier app/logs/dev.log

```
dev.log
1 [2016-02-20 10:30:13] request.ERROR: Uncaught PHP Excepti
2 [2016-02-20 10:30:13] event.DEBUG: Notified event "kernel
3 [2016-02-20 10:30:13] event.DEBUG: Notified event "kernel
4 [2016-02-20 10:30:13] event.DEBUG: Notified event "kernel
5 [2016-02-20 10:30:13] event.DEBUG: Notified event "kernel
```

Benoît Roche / @rocheb83



24



EXERCICES

Benoît Roche / @rocheb83

25



EXERCICES

Petit Quizz

De combien de dossiers se compose un projet Symfony

- ☐ 2
- ☐ 3
- ☐ 4

Mes vues et mes contrôleurs seront dans le dossier

- ☐ /app
- ☐ /src
- ☐ /vendor
- ☐ /web

Benoît Roche / @rocheb83

26

**EXERCICES****Petit Quizz**

Le noyau (kernel) du framework est dans le dossier

- ☐ /app
- ☐ /vendor
- ☐ /web

Le contrôleur frontal est dans le fichier

- ☐ /app
- ☐ /src
- ☐ /web

Benoît Roche / @rocheb83

27

**EXERCICES****Installation**

Après toute cette théorie, un peu de pratique.

Le but de l'exercice est d'installer Netbeans, de le configurer et de créer un projet

Voir : [installation et configuration](#)

Benoît Roche / @rocheb83

28







RÈGLES DE NOMMAGE

Benoît Roche / @rocheb83

29






RÈGLES DE NOMMAGE

Avant de commencer : règles de nommage

Le développement d'une application doit respecter un certain nombre de règles de nommage.

Ces règles ont pour objectif :

- De faciliter la maintenance et l'évolutivité du code:
- lisibilité et cohérence du code,
- compréhension du code par tous




De faciliter le travail en équipe


Sous Symfony, ces règles doivent IMPÉRATIVEMENT être respectées pour rester cohérent avec la communauté.

Benoît Roche / @rocheb83

30



RÈGLES DE NOMMAGE



Avant de commencer : règles de nommage

Rappels :
2 grandes notations sont acceptées :


camelCase : première lettre en minuscule, autres initiales en majuscule, autres lettres en minuscule

Upper CamelCase Toutes les initiales en majuscule, autres lettres en minuscule


Une de ces deux écritures devra être adoptée en fonction de ce que l'on code.

Sous Symfony, ces règles doivent IMPÉRATIVEMENT être respectées pour rester cohérent avec la communauté.

Benoît Roche / @rocheb8331



RÈGLES DE NOMMAGE



Avant de commencer : règles de nommage

Upper Camel Case	une classe	ClsExemple
	une interface	IntExemple
	Un espace de noms	NsApplication
lowerCamelCase	Une variable	maVariable
	Une fonction	maFonction
	Un attribut	unAttributPrivé
	Une méthode	uneMethode
MAJUSCULE	Une constante	MACONSTANTE

Sous Symfony, ces règles doivent IMPÉRATIVEMENT être respectées pour rester cohérent avec la communauté.

Benoît Roche / @rocheb8332



RÈGLES DE NOMMAGE



Avant de commencer : règles de nommage

Codage des fichiers : UTF-8 (sans BOM)

Un fichier par classe. Le nom du fichier est le nom de la classe plus l'extension

Ajouter un espace après chaque virgule séparatrice

Ajouter un espace autour de chaque opérateur (&& , ==, ...)

Ajouter une ligne vide avant l'expression de retour return, à moins que le retour soit seul à l'intérieur du groupe parent (comme une expression if);

Utilisez les accolades pour indiquer les structures de contrôle quel que soit le nombre d'expressions qu'elles entourent.

Sous Symfony, ces règles doivent IMPÉRATIVEMENT être respectées pour rester cohérent avec la communauté.

Benoît Roche / @rocheb83

33



RÈGLES DE NOMMAGE



Avant de commencer : règles de nommage

Nommage des espaces de noms et bundles :

Utiliser uniquement des caractères alphanumériques et des underscore

Utiliser un nom en notation UpperCamelCase » ;

Utiliser un nom court et descriptif (pas plus de 2 mots) ;


Préfixer le nom avec la concaténation de la racine des bundles

Suffixer le nom avec Bundle.


Sous Symfony, ces règles doivent IMPÉRATIVEMENT être respectées pour rester cohérent avec la communauté.

Benoît Roche / @rocheb83

34



RÈGLES DE NOMMAGE



Avant de commencer : règles de nommage

Espace de nom d'un bundle

L'espace de nom devrait commencer avec un nom « commercial » comme le nom de l'entreprise, le nom du projet, suivi par un ou plusieurs sous-espace(s) de nom facultatifs, et devrait être terminé par le nom du bundle lui-même (suffixé par *Bundle*) :

Espace de noms	Nom du Bundle (classe)
AppBundle	AppBundle
Doctrine\Bundle\DoctrineBundle	DoctrineBundle (voir)
Symfony\Bundle\TwigBundle	TwigBundle (voir)
Bdls\Bundle\TutorielBundle	BdlsTutorielBundle

Benoît Roche / @rocheb83

35



EXERCICE

Benoît Roche / @rocheb83

36

**EXERCICE**

Création d'un Bundle: Le but est de créer un bundle et de commencer à comprendre ce qui se passe

Préparation du travail :

Répertoire racine : Bdl (il s'agit du nom du namespace)

Nom du Bundle : on le crée pour ce tutoriel Tutoriel donc on l'appellera TutorielBundle
Bundle étant le suffixe obligatoire

Le format du fichier de configuration sera yml

On va utiliser la console symfony2 intégrée à Netbeans

[Voir : Créer Bundle](#)

Benoît Roche / @rocheb83

37

**FONCTIONNEMENT DE SYMFONY
PAR L'EXEMPLE**

Benoît Roche / @rocheb83

38



FONCTIONNEMENT DE SYMFONY PAR L'EXEMPLE



Objectif de l'exemple : créer une page qui dit Welcome To Benoît

url de la page :

http://localhost/symfony/tutoSymfonyBR/web/app_dev.php/welcome/Benoît

Les différents éléments entrant en jeu :

- ✓ Les routes,
- ✓ Les contrôleurs,
- ✓ Les actions,
- ✓ Les templates

Benoît Roche / @rocheb83

39



FONCTIONNEMENT DE SYMFONY PAR L'EXEMPLE



Les différents éléments entrant en jeu

Schéma de principe : Toutes les demandes de page passent par le contrôleur frontal qui va charger le kernel

http://localhost/TutoSymfony/web/
...



Benoît Roche / @rocheb83

k?php

```

use Symfony\Component\ClassLoader\ApcClassLoader;
use Symfony\Component\HttpFoundation\Request;


$loader = require_once __DIR__.'../app/bootstrap.php.cache';

// Use APC for autoloading to improve performance.
// Change 'sf2' to a unique prefix in order to prevent cache key conflicts
// with other applications also using APC.
/*
$loader = new ApcClassLoader('sf2', $loader);
$loader->register(true);
*/


require_once __DIR__.'../app/AppKernel.php';
//require_once __DIR__.'../app/AppCache.php';

$kernel = new AppKernel('prod', false);
$kernel->loadClassCache();
//$kernel = new AppCache($kernel);
$request = Request::createFromGlobals();
$response = $kernel->handle($request);
$response->send();
$kernel->terminate($request, $response);
  
```

40

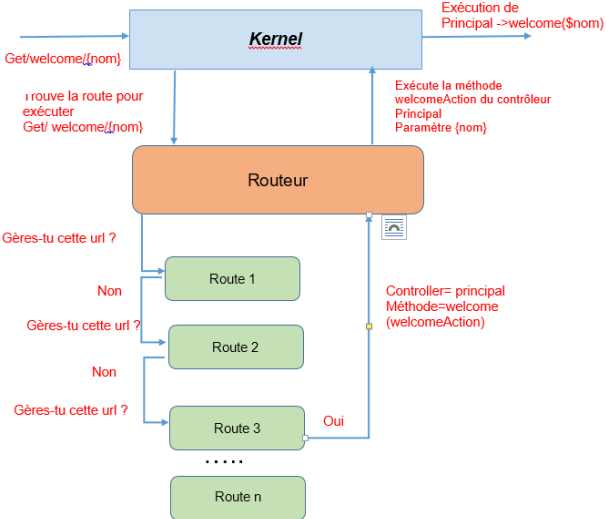


FONCTIONNEMENT DE SYMFONY PAR L'EXEMPLE




Les différents éléments entrant en jeu

Schéma de principe




Benoît Roche / @rocheb83

41



FONCTIONNEMENT DE SYMFONY PAR L'EXEMPLE




Les différents éléments entrant en jeu


Le contrôleur frontal est vu comme un fichier de l'application(il est dans notre répertoire /web),
le Kernel est vu comme un composant Symfony2 ou encore comme une boite noire (il est dans le répertoire /vendor).
On voit que le contrôleur a délégué la gestion de la requête au Kernel.
Le Kernel aura besoin de nos informations pour savoir quoi exécuter comme code,
Le kernel gère en plus plusieurs choses que nous avons vues : la gestion des erreurs, l'ajout de la toolbar en bas de l'écran (mode dev), etc.
On n'a encore rien fait, et pourtant on a déjà gagné du temps !

Benoît Roche / @rocheb83

42



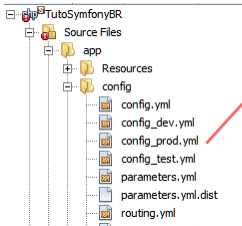
FONCTIONNEMENT DE SYMFONY PAR L'EXEMPLE



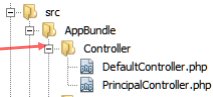
Les différents éléments entrant en jeu

Schéma de principe URL :
http://symfony.br/TutoSymfonyBR/web/app_dev.php/welcome/benoît

Etape 1 : trouver le la méthode (action) à exécuter




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




```
class PrincipalController extends Controller {  
  
    /**  
     * @Route("/welcome/{nom}")  
     */  
    public function welcomeAction($nom) {  
        return $this->render('AppBundle:Principal:welcome.html.twig', array(  
            "nom" => $nom  
        ));  
    }  
}
```

Benoît Roche / @rocheb83

43



FONCTIONNEMENT DE SYMFONY PAR L'EXEMPLE


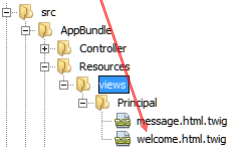
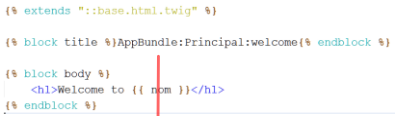


Les différents éléments entrant en jeu

Schéma de principe
http://symfony.br/TutoSymfonyBR/web/app_dev.php/welcome/benoît

Etape 2 : Exécuter la méthode (action), ici appel à la vue à afficher

```
class PrincipalController extends Controller {  
  
    /**  
     * @Route("/welcome/{nom}")  
     */  
    public function welcomeAction($nom) {  
        return $this->render('AppBundle:Principal:welcome.html.twig', array(  
            "nom" => $nom  
        ));  
    }  
}
```



Benoît Roche / @rocheb83

44



FONCTIONNEMENT DE SYMFONY PAR L'EXEMPLE



Cas des applications simples ne devant pas être diffusées:

Depuis peu, Symfony recommande, uniquement dans les applications simples dont aucun module ne sera mis dans la communauté

Un seul Bundle appelé AppBundle.

Routing par annotation dans ce bundle.

Ce bundle peut contenir autant de contrôleurs que l'on veut. ON va en créer un seul : PrincipalController

Benoît Roche / @rocheb83

45



FONCTIONNEMENT DE SYMFONY PAR L'EXEMPLE



Les différents éléments entrant en jeu :

Notion de template

Pour réaliser l'affichage, on devra passer par un moteur de templates

Un moteur de Template est une technique de programmation permettant de séparer distinctement :

l'interface graphique (HTML/CSS/javascript)

du reste de l'application (Contrôleur)

Comment ?

Avec un langage spécifique au moteur de template

Benoît Roche / @rocheb83

46



FONCTIONNEMENT DE SYMFONY PAR L'EXEMPLE



Les différents éléments entrant en jeu

Un exemple avec le template choisi : twig

```
<!DOCTYPE html>
<html>
<head>
<title>Bienvenue dans Symfony2 !</title>
</head>
<body>
<h1><?php echo $titre_page; ?></h1>
<ul id="navigation">
<?php foreach ($navigation as $item) { ?>
<li>
<a href="<?php echo $item->getHref(); ?>">
<?php echo $item->getTitre(); ?></a>
</li>
<?php } ?>
</ul>
</body>
</html>
```

Code HTML

```
<!DOCTYPE html>
<html>
<head>
<title>Bienvenue dans Symfony2 !</title>
</head>
<body>
<h1>{{ titre_page }}</h1>
<ul id="navigation">
{% for item in navigation %}
<li><a href="{{ item.href }}">{{ item.titre }}</a></li>
{% endfor %}
</ul>
</body>
</html>
```

Code Twig

Benoît Roche / @rocheb83

47



FONCTIONNEMENT DE SYMFONY PAR L'EXEMPLE



Les différents éléments entrant en jeu

Rudiments de twig

Twig est facile à lire

Il facilite le travail du designer qui n'a pas à connaître le php



Exemple

Twig : {{ mavariable }}

Équivalent php : <?php echo \$mavariable; ?>

Twig : {{ unTitre | upper }}

Équivalent php : <?php echo strtoupper(\$unTitre); ?>

Benoît Roche / @rocheb83

48

**FONCTIONNEMENT DE SYMFONY PAR L'EXEMPLE****Les différents éléments entrant en jeu**

Règles de nommage

Le dossier des vues (templates) se trouve dans le dossier `\Resources\views\` du bundle.

On crée dans ce dossier un dossier du même nom que le contrôleur pour mieux se repérer

Le nom de la vue (du template) porte le même nom que la méthode (action) qui l'a appelé

L'extension du fichier de la vue est toujours `.html.twig`

Benoît Roche / @rocheb83

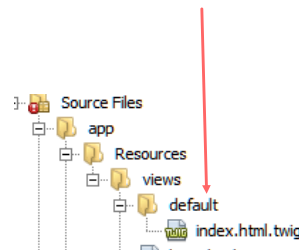
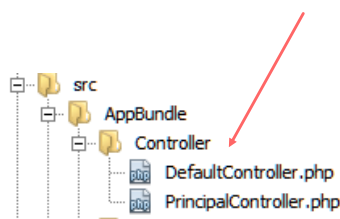
49

**FONCTIONNEMENT DE SYMFONY PAR L'EXEMPLE****Les différents éléments entrant en jeu**

Un peu de nettoyage

Pour chaque bundle créé, on supprime :

- Le contrôleur `Contrôleur/DéfautController.php`
- Les vues du dossier `Resources\views\Default`



Benoît Roche / @rocheb83

50







LE CACHE

Benoît Roche / @rocheb83


51






LE CACHE

Afin d'améliorer ses performances, Symfony met beaucoup de choses dans le cache....
Et ne le vide jamais.



Il se peut donc qu'après certaines modifications, notamment sur les fichiers de configuration, le cache ne soit plus à jour.


Exemple : les fichiers Yaml convertis en php sont placés dans le cache




Solution : il faut vider le cache pour qu'il se reconstruise !!!

Benoît Roche / @rocheb83

52



LE CACHE



Vider le cache : commande **Cache:clear**

En mode console ou en mode console symfony

Mode production (prod) :

php app/console cache:clear --env=prod

```
E:\Wampsites\TutoSymfonyBR>C:\wamp\bin\php\php5.4.16\php app/console cache:clear --env=prod
Clearing the cache for the prod environment with debug false
E:\Wampsites\TutoSymfonyBR>
```

Mode développement (dev) :


php app/console cache:clear

(mode dev par défaut)


```
E:\Wampsites\TutoSymfonyBR>C:\wamp\bin\php\php5.4.16\php app/console cache:clear
Clearing the cache for the dev environment with debug true
E:\Wampsites\TutoSymfonyBR>
```

Benoît Roche / @rocheb83

53

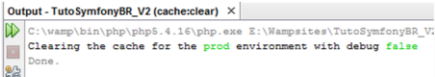
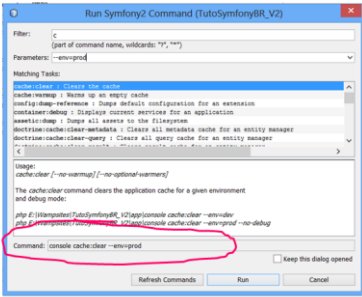



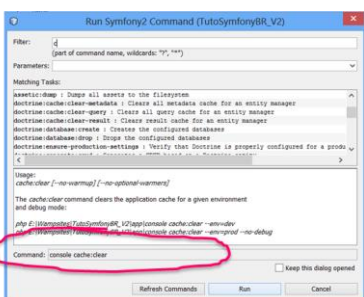
LE CACHE



Vider le cache : commande **Cache:clear**

Vider le cache : mode console symfony





Benoît Roche / @rocheb83

54



LE CACHE



Vider le cache : commande `Cache:clear`

Vider le cache : commande

Si la commande `cache:clear` génère des erreurs lors de son exécution :

il faut essayer de la relancer.

Si ça ne marche pas encore,

il faut supprimer le cache à la main

en supprimant simplement le répertoire `app/cache/dev` ou `app/cache/prod` suivant l'environnement.

Benoît Roche / @rocheb83

55



EXERCICE

Benoît Roche / @rocheb83

56

**EXERCICE**

Vider le cache : commande **Cache:clear**



Exercice:

Il est temps de mettre en pratique tout ceci...

Voir : [création de la première page en utilisant le bundle AppBundle](#)

Benoît Roche / @rocheb83

57

**SOURCES**

Benoît Roche / @rocheb83

58

**SOURCES**

Installation Netbeans et symfony :

<http://www.dascoub.com/configurer-php-dans-netbeans/>

<http://www.dascoub.com/configurer-netbeans-pour-symfony2/>

Tutoriels symfony :

<http://symfony.com/>

[https://openclassrooms.com/courses/developpez-votre-site-web-avec-le-framework-symfony2\(s'inscrire pour télécharger le tutoriel\)](https://openclassrooms.com/courses/developpez-votre-site-web-avec-le-framework-symfony2(s'inscrire pour télécharger le tutoriel))

http://symfony.com/legacy/doc/jobeeet/1_4/fr/01?orm=Doctrine

Benoît Roche / @rocheb83

59



Fin du cours, merci

Questions/Réponses

Benoît Roche
@rocheb83

Benoît Roche / @rocheb83

60