

Formation Symphony2

# Présentation

- Pourquoi symfony (1/2)
- Symfony1 vers Symfony2
- Les ressources disponibles

- Symfony un framework
  - Productivité
  - Offrir un cadre avec les fonctions de base
  - Forcer les bonnes pratiques
  - Meilleure relecture du code
  - Développement à plusieurs facilité

- Symfony1 vers Symfony2 ... pourquoi?
  - Php 5.3
  - Nouveau autoloader (namespaces)
  - Flexibilité du code (les bundles)
  - Optimisation des fichiers de configuration
  - Plus rapide et moins gourmand
  - Architecture MVC
  - Communauté (bundles)

- Ressources
  - **Symfony.com** pour la documentation
  - Les bundles déjà programmés par la communauté (Pagerfanta, Gedmo, ...)
  - Communauté active pour la correction de bug
  - Les sources internet dans les blogs (attention aux articles de symfony sur la version 2.0)

- Les fichiers de configuration Yaml
- Les templates (html,... ) écrit en langage Twig
- Les templates de fichiers écrit en PHP standard
- Les annotations
  - Utilise les commentaires prefixés d'un @
- La console
  - Assistant de création de bundles
  - Créer des commandes CRUD facilement
  - Possibilité de créer des commandes personnalisés
  - Gérer le modèle de votre base avec des commandes spécifiques à Doctrine2.

## – IDE

- Netbeans
- Eclipse

## – Editeur de fichiers simple

- UltraEdit
- Notepad++
- Dreamweaver
  - Css

## – Autres

- MySQLWorkbench
  - Créer des représentations visuels du model de la base de données.
  - Créer les jointures
  - La nature des champs
  - Exporter le model au format sql

- Les relations dans mysql:
  - <http://www.eliacom.com/mysql-gui-wp-errno-150.php>



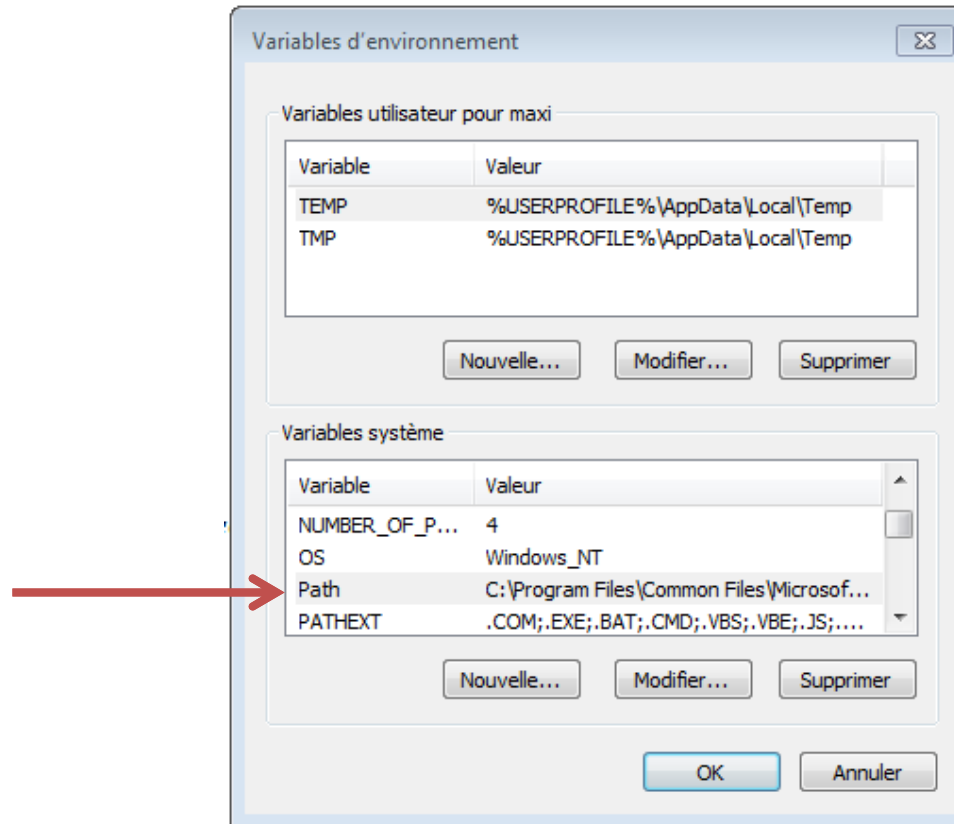
# Symfony - installations

- <http://symfony.com/download>
- Composer
- Github (voir le ppt)
- Regarder le fichier :
  - `/vendor/composer/installed.json`

# Installation de symfony sous Windows

Installation sous windows des lignes de commande de php avec **wamp**

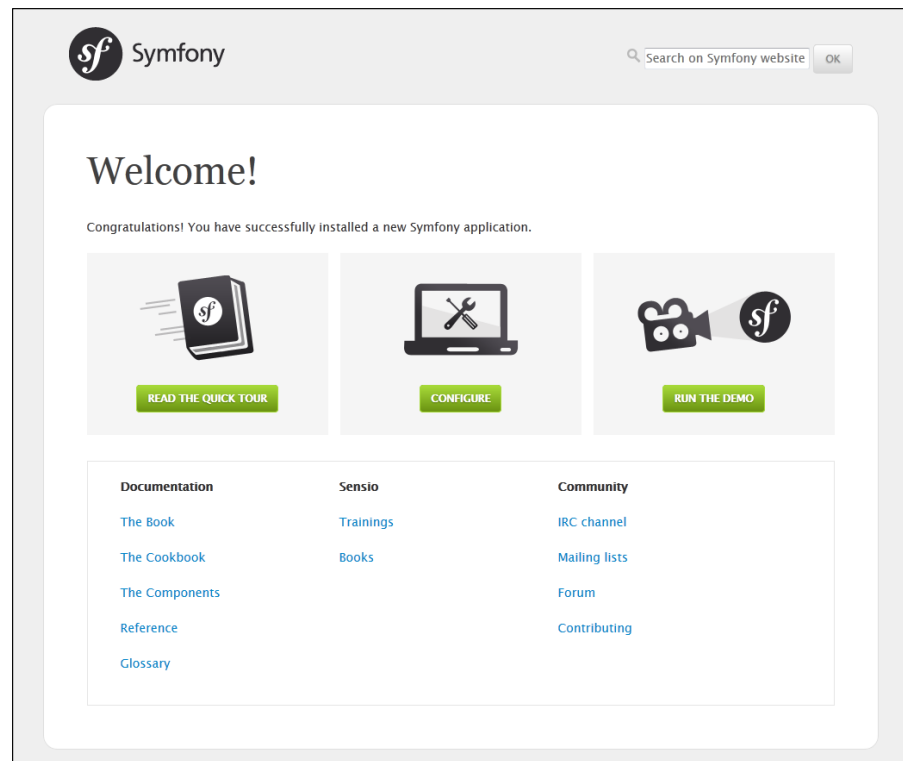
- Aller dans « propriétés système »
- Cliquez sur le bouton « Variables d'environnements »
- Dans l'onglet « Variables système », modifier la ligne **path** et ajouter à la fin   
`;C:\wamp\bin\php\php5.3.x`



# Le bundle Acme

Le bundle **acme** propose des exemples simple afin de commencer à utiliser le modèle MVC :

- Une page pour démarrer et vous guider.



- Des exemples de contrôleurs et de vues.
- Un exemple de formulaire d'identification.

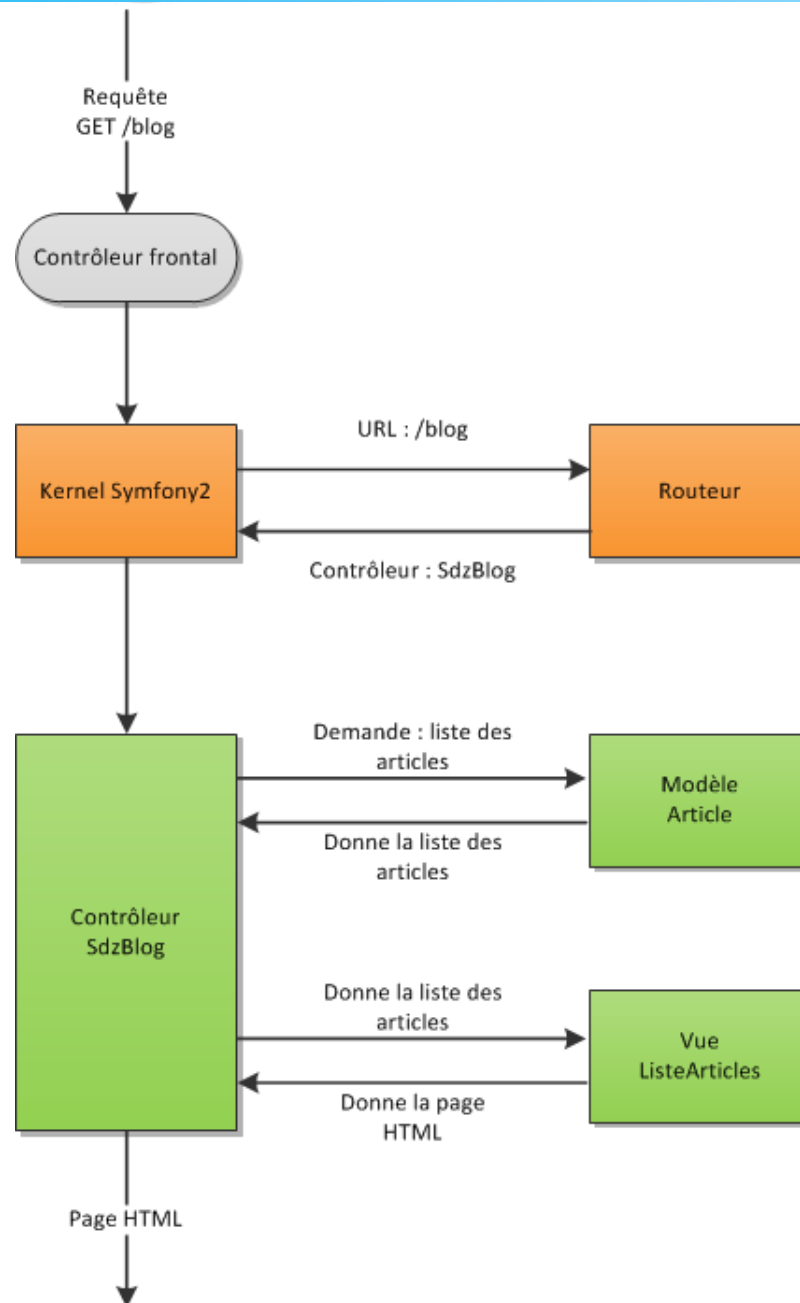
# Le bundle Acme

Photo de la page de configuration

# Présentation

1. Architecture MVC
2. Architecture de Symfony
3. Architecture d'un bundle
4. Le contrôleurs et les vues
5. Les environnements
6. Architecture des fichiers
7. Le bundle Acme
8. Du contrôleur à la vue.
9. Namespace et annotations
10. La déclaration d'un bundle
11. Exemple d'annotation
12. Du contrôleur à la vue.
13. La déclaration d'un bundle
14. Exemple d'annotation
15. Le contrôleurs
16. Les services de base

# Architecture MVC



# Architecture des fichiers

**/app** <- class de configuration

**/config** <- Yaml de configuration , le cache, les fichiers logs

**/src** <- votre code source sous forme de « bundles »

**/vendor** <- librairies: Symfony2, Doctrine, Twig, SwiftMailer

**/web** <- images, CSS, Js (dossier visible par les utilisateurs)

le .htaccess interdit l'accès vers les répertoires supérieurs

# Architecture d'un bundle

....Bundle/Controller <- les contrôleurs

....Bundle/DependencyInjection <- chargeur de services

....Bundle/Entity <- vos modèles Doctrine

....Bundle/Repository<- vos requêtes stockées Doctrine

....Bundle/Resources<- votre

    /config <- fichiers de configuration

    /views <- les vues (template twig ou php...)



# Du contrôleur à la vue

symfony2\_default

Source Files

app

bin

src

Acme

DemoBundle

Controller

DemoController.php

SecuredController.php

WelcomeController.php

DependencyInjection

EventListener

Form

Resources

config

public

css

images

views

Demo

Secured

Welcome

index.html.twig

layout.html.twig

Tests

Twig

AcmeDemoBundle.php

Les contrôleurs

Un contrôleur possède un dossier qui porte le même nom dans le dossier views.

Les vues (twig, php, ...)



# Namespace

La déclaration des nouveaux bundles se fait dans le fichier  
app/AppKernel.php


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

class AppKernel extends Kernel {

    public function registerBundles()
    {
        $bundles = array(
            new Symfony\Bundle\FrameworkBundle\FrameworkBundle(),
            new Symfony\Bundle\SecurityBundle\SecurityBundle(),
            new Symfony\Bundle\TwigBundle\TwigBundle(),
            new Symfony\Bundle\MonologBundle\MonologBundle(),
            new Symfony\Bundle\SwiftmailerBundle\SwiftmailerBundle(),
            new Symfony\Bundle\AsseticBundle\AsseticBundle(),
            new Doctrine\Bundle\DoctrineBundle\DoctrineBundle(),
            new Sensio\Bundle\FrameworkExtraBundle\SensioFrameworkExtraBundle(),
            new JMS\AopBundle\JMSAopBundle(),
            new JMS\DiExtraBundle\JMSDiExtraBundle($this),
            new JMS\SecurityExtraBundle\JMSSecurityExtraBundle(),
        );

        if(in_array($this->getEnvironment(), array('dev', 'test')) {
            $bundles[] = new Acme\DemoBundle\AcmeDemoBundle();
            $bundles[] = new Symfony\Bundle\WebProfilerBundle\WebProfilerBundle();
            $bundles[] = new Sensio\Bundle\DistributionBundle\SensioDistributionBundle();
            $bundles[] = new Sensio\Bundle\GeneratorBundle\SensioGeneratorBundle();
        }

        return $bundles;
    }
}
```



# Les environnements

## Aperçu du fichier de `web/app_dev.php`

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

$kernel = new AppKernel('dev', true);
$kernel->loadClassCache();
$request = Request::createFromGlobals();
$response = $kernel->handle($request);
$response->send();
$kernel->terminate($request, $response);
```

La class **AppKernel** définit le nom de l'environnement et l'activation ou non de la barre débogage.

## Aperçu du fichier de `config.yml`

```
imports:
  - { resource: parameters.yml }
  - { resource: security.yml }

framework:
  #esi: ~
  #translator: { fallback: %locale% }
  secret: %secret%
  router:
    resource: "%kernel.root_dir%/config/routing.yml"
    strict_requirements: %kernel.debug%
  form: true
  csrf_protection: true
  validation: { enable_annotations: true }
  templating: { engines: ['twig', 'php'] } #assets_version
  default_locale: %locale%
  trust_proxy_headers: false # Whether or not the Request
```

Importez d'autres fichiers de configurations

Paramètres de symfony2:  
**Templating** permet définir les langages de template que l'on pourra utiliser dans les vues (ici: twig et php)

# Namespace

```
<?php

namespace Acme\DemoBundle\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\Controller;
use Symfony\Component\HttpFoundation\RedirectResponse;
use Acme\DemoBundle\Form\ContactType;

// these import the "@Route" and "@Template" annotations
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Template;

class DemoController extends Controller
{
    /**
     * @Route("/", name="_demo")
     * @Template()
     */
    public function indexAction()
    {
        return array();
    }

    /**
     * @Route("/hello/{name}", name="_demo_hello")
     * @Template()
     */
    public function helloAction($name)
    {
        return array('name' => $name);
    }
}
```

} Namespace de la class (php 5.3)

} Indique les emplacements des class

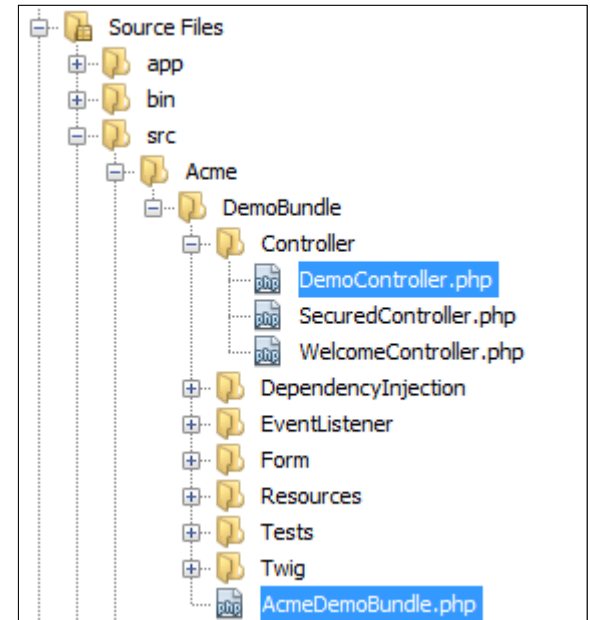
Le namespace doit être indiqué en début de fichier.

Si vous indiquez un namespace, vous devez utiliser la fonction « **use** » afin de déclarer vos class utilisées.

# Namespace

Regardez la correspondance entre le namespace et l'arborescence (respect de la casse):

```
namespace Acme\DemoBundle\Controller;  
  
use Symfony\Bundle\FrameworkBundle\Controller\Controller;  
use Symfony\Component\HttpFoundation\RedirectResponse;  
use Acme\DemoBundle\Form\ContactType;
```



# Namespace

Lorsque vous utilisez un namespace, les class php standards doivent être déclarées avec **use** ou précédées d'un slash.

```
use Datetime;

class DemoController extends Controller
{
    /**
     * @Route("/", name="_demo")
     * @Template()
     */
    public function indexAction()
    {
        new Datetime; //déclaré avec use
        new \Datetime; //non déclaré
        return array();
    }
}
```

# Annotations

**Les annotations:** Les **classes** utilisent le système des annotations comme des commentaires.

Une annotation exécute du code comme une **classe** ou une **fonction** déclarer dans un script php standard.

```
namespace Acme\DemoBundle\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\Controller;
use Symfony\Component\HttpFoundation\RedirectResponse;
use Acme\DemoBundle\Form\ContactType;

// these import the "@Route" and "@Template" annotations
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Template;

class DemoController extends Controller
{
    /**
     * @Route("/", name="_demo")
     * @Template()
     */
    public function indexAction()
    {
        return array();
    }

    /**
     * @Route("/hello/{name}", name="_demo_hello")
     * @Template()
     */
    public function helloAction($name)
    {
        return array('name' => $name);
    }
}
```

# Annotations

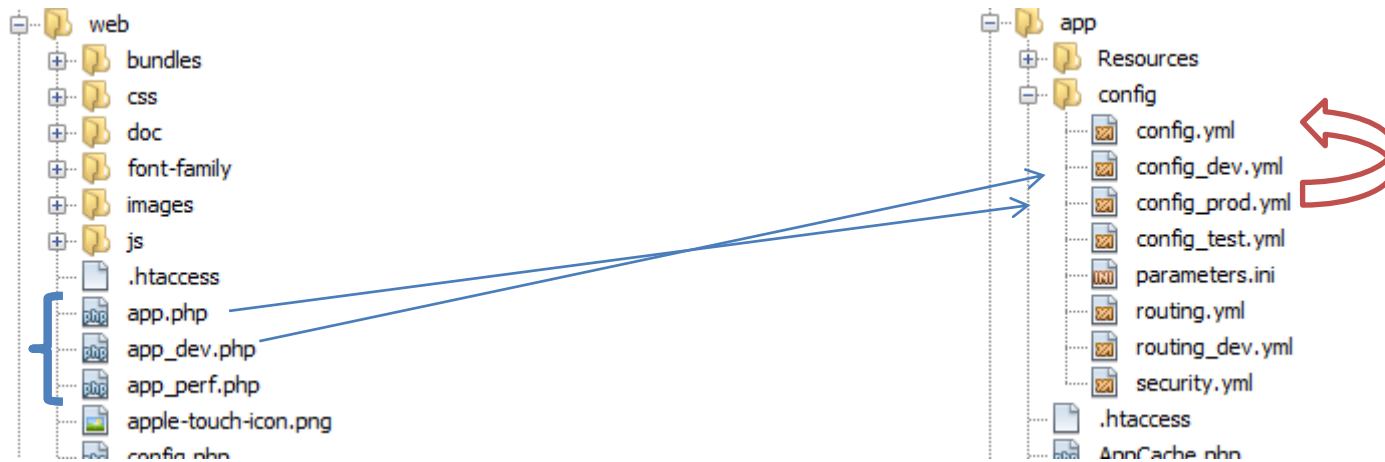
Exemple d'annotation dans un contrôleur :

```
/**
 * @Route("/{id}")
 * @Method("GET")
 * @ParamConverter("post", class="SensioBlogBundle:Post")
 * @Template("SensioBlogBundle:Annot:post.html.twig", vars={"post"})
 * @Template(engine="php")
 * @Cache(smaxage="15")
 */
```



# Les environnements

Symfony2 propose plusieurs environnements de travail.



Lorsque vous accédez à partir de l'url .../**app.php**, le système lit le fichier de configuration **config.yml**

## Config.yml stocke

- les paramètres de la bdd
- Le service d'e-mail (swift)
- Les différents types de template utilisés dans les contrôleurs (twig (par défaut), php, ...)
- Les nouveaux services
- La configuration de bundles

# Annotations

## Exemple d'une action dans un contrôleur:

```
public function aideRubriqueAction()
{
    $request = $this->getRequest();
    $chapitres = $request->get('chapitres');

    $count = 1200 * $chapitres;
    $now = new \DateTime('now');

    $message_html = '<b style="color:#009900">Aide rubrique:</b>';

    return array('count' => $count, 'now' => $now, 'message_html' => $message_html);
}
```

Les variables sont envoyées dans le template sous forme de tableaux.

# Annotations

Voir les classes de services disponibles dans un contrôleur:

`Symfony\Component\HttpFoundation\Request`

`Symfony\Component\HttpFoundation\Session`

# Les services de symfony

## Le service **request**:

Récupérer l'objet request:

```
$request = $this->getRequest();  
$request->get('id');
```

\$\_GET parameter

```
$request->query->get('page');
```

\$\_POST parameter

```
$request->request->get('page');
```

Parametres cookie

```
$request->cookies->get('page');
```

get REQUEST\_URI

```
$request->getPathInfo()
```

Si le header est un Ajax (retourne un boolean)

```
if ($request->isXmlHttpRequest()) { ... }
```

Récupérer la locale courante

```
$this->getRequest()->getLocale();
```

Quel est le langage préféré ?

```
->getPreferredLanguage(array('en', 'fr'));
```

Récupérer le "baseUrl (+ ou - complet)" de l'appli

```
$this->getRequest()->getBasePath();
```

```
$this->getRequest()->getBaseUrl();
```

# Les variables du contrôleur

Liste des autres services les plus utilisés:

Service **EntityManager** de Doctrine

```
$em = $this->getDoctrine()->getManager();
```

Service **kernel**:

Récupérer le chemin physique du Root App (.../Symfony/app):

```
$dir_web = $this->get('kernel')->getRootDir() . ' ../web ';
```

Environnement (ex: prod ou dev):

```
$this->get('kernel')->getEnvironment();
```

Service **session**:

Une variable flash s'efface juste après avoir été affichée:

```
$this->getRequest()->getSession();
```

```
$this->get('session')->getFlashBag()->set('notice', 'votre message');
```

Récupérer un paramètre de l'application défini par exemple dans **parameters.yml**

```
$param = $this->container->getParameter('my_parameter');
```

# Les variables du contrôleur

Récupérer le service template

```
$templating = $this->get('templating')
```

Service de routage:

```
$router = $this->get('router');
```

Service de mail

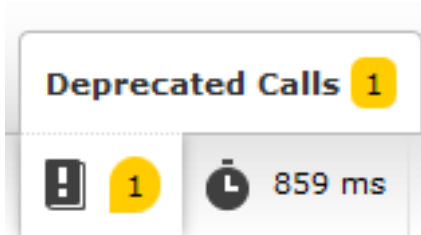
```
$mailer = $this->get('mailer');
```

Récupérer un service

```
$service = $this->get("myService");
```

# Les variables du contrôleur

Les fonctions dépréciées sont indiqués en jaune dans la barre de débogage.



DEPRECATION - getEntityManager is deprecated since Symfony 2.1. Use getManager instead [+](#)

# Retour d'un contrôleur

Renvoie un template twig

```
return $this->render( 'MonBundle:Produit:voir.html.twig', array( 'id' => $id) );
```

Transfère à un autre controlleur (sans changer de requête)

```
return $this->forward('MonBundle:Produit:voir', array( 'id' => $id) );
```

Fait une redirection 302 vers une page

```
return $this->redirect("/blog/mon_article");
```

Fait une redirection vers une page définie par le routeur

```
return $this->redirect( $this->generateUrl("produit_edit", array("id"=>$)) );
```

Renvoie un texte simple vers le navigateur

```
return new Response('My text response');
```

Renvoie un json

```
return new JsonResponse(array());
```



# Namespace

La déclaration des nouveaux namespaces se fait dans le fichier `app/autoload.php`

```
use Doctrine\Common\Annotations\AnnotationRegistry;

$loader = require __DIR__ . '/../vendor/autoload.php';
$loader->add('WsSource', __DIR__ . '/../vendor/ws-source/lib');
$loader->add('WsGene', __DIR__ . '/../vendor/ws-generator');
$loader->add('Pagerfanta', __DIR__ . '/../vendor/pagerfanta/src');

// intl
if(!function_exists('intl_get_error_code')) {
    require_once __DIR__ . '/../vendor/symfony/symfony/src/Symfony/
    $loader->add('', __DIR__ . '/../vendor/symfony/symfony/src/Symfony/
}

AnnotationRegistry::registerLoader(array($loader, 'loadClass'));

return $loader;
```