

Niveau : STS SIO 2^{ème} année☐ Cours ☐ TD ☒ TP

1/18

Module : SLAM5 – Solutions applicatives



Intitulé : Découverte de Symfony 2

Durée : 2 heures

- Objectifs :
- ✓ Réaliser une application simple basée sur le modèle MVC
 - ✓ Installer et configurer Symfony en environnement Windows
 - ✓ Découverte des principes de base du Framework Symfony2
 - ✓ Programmer au sein d'un Framework

Contenu

Objectifs de ce premier TP	1
Installation et configuration.....	3
Création du Bundle	5
Configurer correctement notre éditeur	11
Gérer nos routes	12
Définir le contrôleur	13
Créer nos vues.....	14
Un peu de ménage	18

Objectifs de ce premier TP

Nous allons faire très simple dans ce premier TP : afficher une page qui contient « Salut, toi ! » ou « Bonjour » ou « Bonjour Tartempion » si l'utilisateur a précisé son nom (Tartempion) dans l'URL

L'URL / affichera « Salut, toi ! »

L'URL /bonjour/ affichera « Bonjour ! »

L'URL /bonjour/Jeanot affichera « Bonjour Jeanot ! »

Les URL données sont des url relatives par rapport à l'url de base de votre application !

Ce premier TP a pour objectif de vous faire découvrir :

- comment installer Symfony2 sous Windows
- comment créer un Bundle et définir ses routes
- comment créer des templates (au niveau application et au niveau Bundle avec un héritage de template)
- comment coder le contrôleur pour lui faire faire le travail !

Installation et configuration

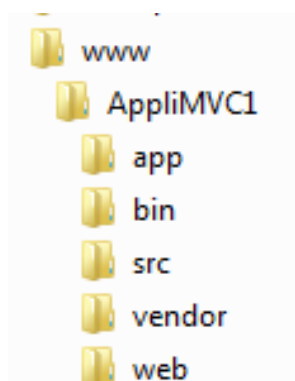
Il existe plusieurs façons d'obtenir et d'installer Symfony2. Nous allons utiliser la plus simple et pour cela installer la distribution standard zippée.

- Chez vous : rendez-vous sur le [site de téléchargement de Symfony](#) et télécharger la version .zip (voir page suivante).

The screenshot shows the Symfony website's download page. The main heading is 'Using the Symfony full-stack framework'. It offers two options: 'Use Symfony 2.5 for the latest features' (with an 'end of support July 2015' note) and 'Use Symfony 2.3 for long-term support' (with an 'end of support May 2016' note). A terminal command is displayed: `$ composer create-project symfony/framework-standard-edition path/ "2.5.*"`, followed by a 'COPY' button. Below the command is a link to 'What is Composer and how to install it'. At the bottom, a red box highlights the 'Evaluating Symfony?' section, which includes the text 'Download the packaged version for a quick test-drive of Symfony2 features.' and a link to 'Download Symfony-2.5.2.zip'.

- Au lycée : récupérez l'archive sur le réseau (Symfony_Standard_Vendors_2.5.2.zip).
- Créez, dans votre répertoire www de OneDrive un dossier AppliMVC1 et décompressez dans ce dossier l'archive Symfony.

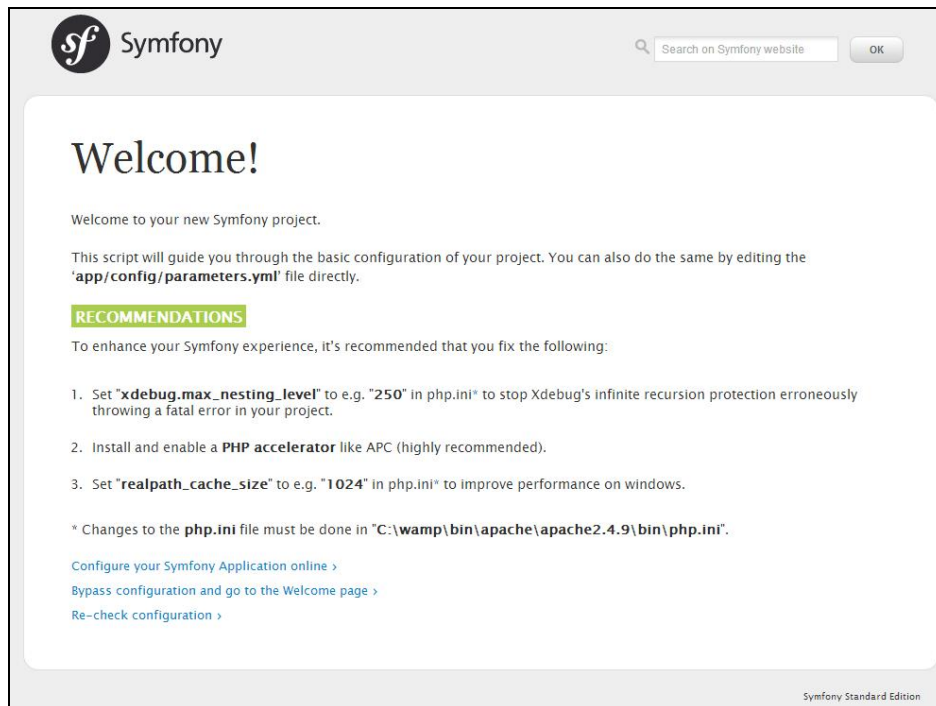
Vous obtenez :



- Rendez-vous à l'adresse suivante :

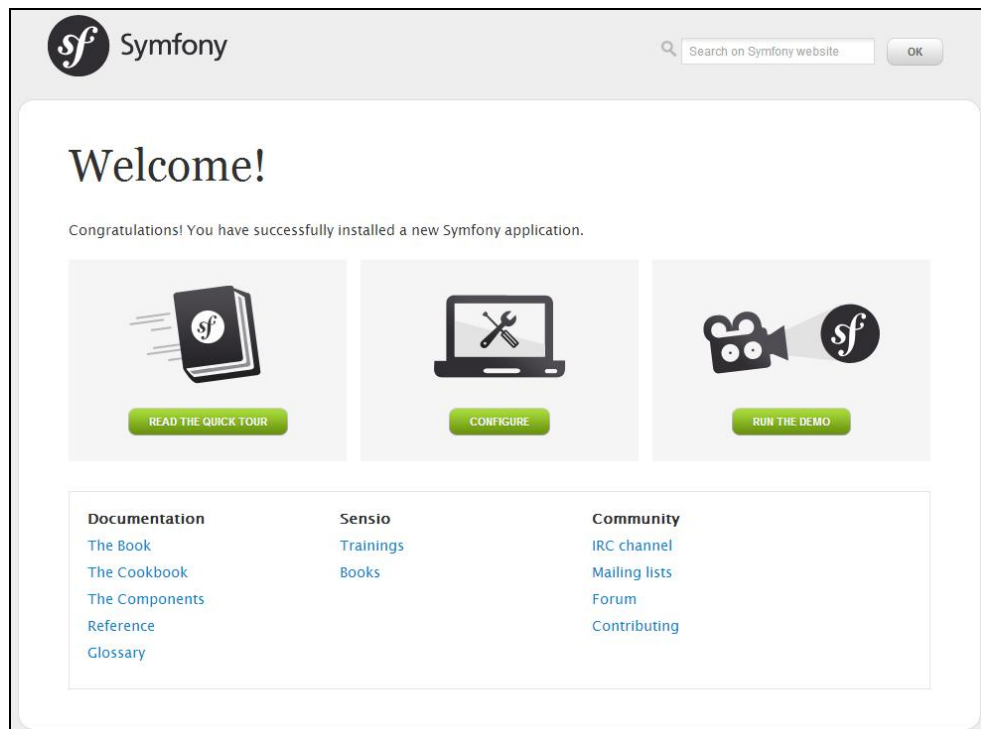
<http://localhost/votreAlias/AppliMVC1/web/config.php>

Si tout fonctionne correctement, vous obtenez la page suivante :



En cas d'incompatibilité (version de PHP notamment), Symfony2 vous demande de régler les problèmes avant de continuer. S'il ne vous propose que des recommandations, vous pouvez continuer sans problème

- Vous pouvez maintenant exécuter Symfony2 en vous rendant sur la page suivante : http://localhost/votreAlias/AppliMVC1/web/app_dev.php



- Il vous reste juste à vérifier le fonctionnement de PHP en mode console, car vous en aurez besoin
Vérifiez tout d'abord votre variable d'environnement système path : elle doit contenir « ;C:\wamp\bin\php\php5.5.12 » (puisque nous avons la version 5.5.12 de PHP installée au lycée. Chez vous, il faut bien entendu adapter !). Si besoin, modifiez la variable path.
Ouvrez une invite de commande et tapez : `php -v`
Vous devez obtenir la version de PHP

Création du Bundle

Vous allez maintenant créer votre premier bundle. Et c'est là que toute la puissance du Framework intervient !

Vous n'allez pas vous amuser à créer l'arborescence à la main et à ajouter les différents fichiers vous-mêmes. Symfony le fait pour vous.

Seule contrainte : l'utilisation de la console. Mais ce n'est vraiment pas bien contraignant.

Ouvrez une console directement sur votre dossier AppliMVC1 (dans l'explorateur, faites Shift + clic droit puis « Ouvrir une fenêtre de commande ici »).

Exécutez la commande : `php app/console`

En fait vous exécutez le script console qui se trouve dans le répertoire app. C'est ce script qui va vous permettre d'exécuter toutes les commandes dont vous avez besoin.

Lorsque vous l'exécutez sans lui passer de commande en paramètre, vous obtenez la liste des commandes disponibles. C'est bien pratique lorsque l'on a trou de mémoire :

```
Symfony version 2.5.2 - app/dev/debug

Usage:
  [options] command [arguments]

Options:
  --help           -h Display this help message.
  --quiet          -q Do not output any message.
  --verbose        -v|vv|vvv Increase the verbosity of messages: 1 for normal output, 2 for more details, 3 for debugging.
  --version        -V Display this application version.
  --ansi           Force ANSI output.
  --no-ansi        Disable ANSI output.
  --no-interaction -n Do not ask any interactive question.
  --shell          -s Launch the shell.
  --process-isolation Launch commands from shell as a separate process.
  --env            -e The Environment name.
  --no-debug       Switches off debug mode.

Available commands:
  help                Displays help for a command
  list                Lists commands
  acme
  acme:hello          Hello World example command
  assetic
  assetic:dump         Dumps all assets to the filesystem
  assets
  assets:install       Installs bundles web assets under a public web directory
  cache
  cache:clear          Clears the cache
  cache:warmup         Warms up an empty cache
  config
  config:debug         Dumps the current configuration for an extension
  config:dump-reference Dumps the default configuration for an extension
```

La commande pour créer un bundle est `generate:bundle`.

Vous allez donc exécuter : `php app/console generate:bundle`

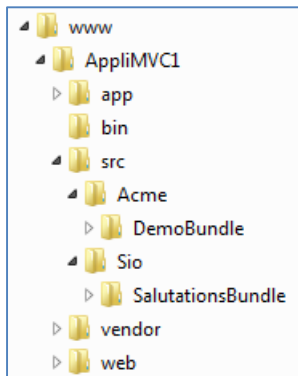
Il n'y a plus qu'à répondre aux différentes questions :

- Bundle namespace: Sio/SalutationsBundle
- Bundle name [SioSalutationsBundle]:
- Target directory [C:/wamp/www/catalogue/src]:
- Configuration format (yml, xml, php, or annotation): yml
- Do you want to generate the whole directory structure [no]? yes
- Do you confirm generation [yes]?
- Confirm automatic update of your Kernel [yes]?
- Confirm automatic update of the Routing [yes]?

Partout où il n'y a pas de réponse, c'est que l'on choisit la valeur par défaut proposée.

Comme vous pouvez le remarquer, nous l'avons laissé générer tout ce qui était possible et notamment les fichiers pour le routage ainsi que le noyau.

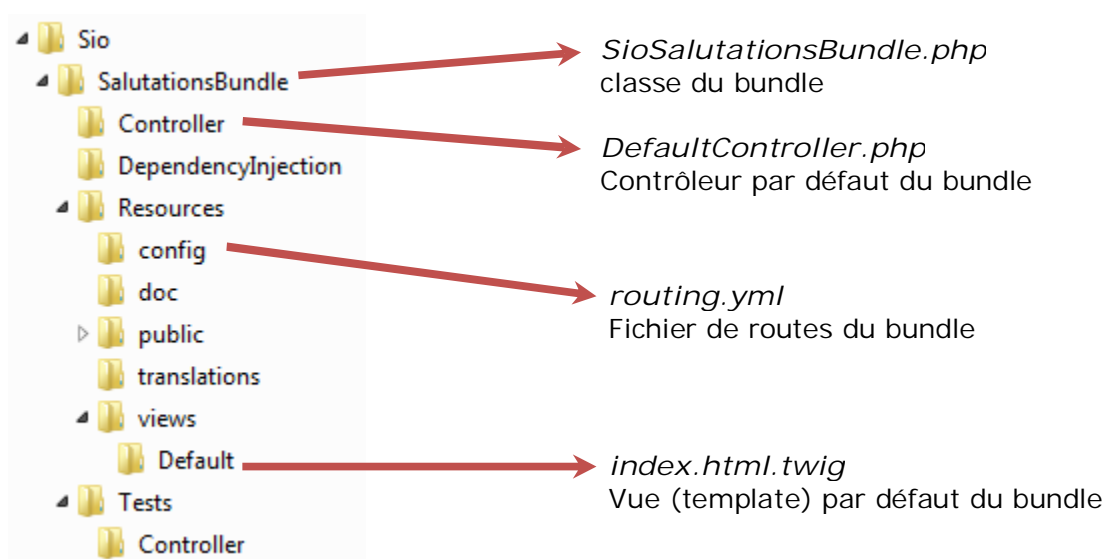
Allons voir d'un peu plus près.



Vous pouvez constater que le répertoire `src` contient 2 bundles : `AcmeDemoBundle` et `SioSalutationsBundle`.

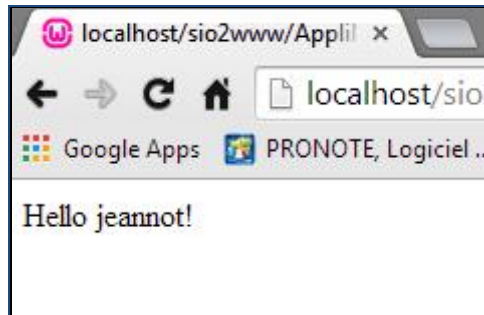
- `AcmeDemoBundle` est un bundle de démo fourni avec Symfony et qui peut vous aider lors du développement de vos propres bundles. Nous le supprimerons un peu plus tard.
- `SioSalutationsBundle` est notre bundle, généré par Symfony.

Comme vous pouvez le voir, toute l'arborescence de notre bundle a été générée et les fichiers indispensables ont été créés.

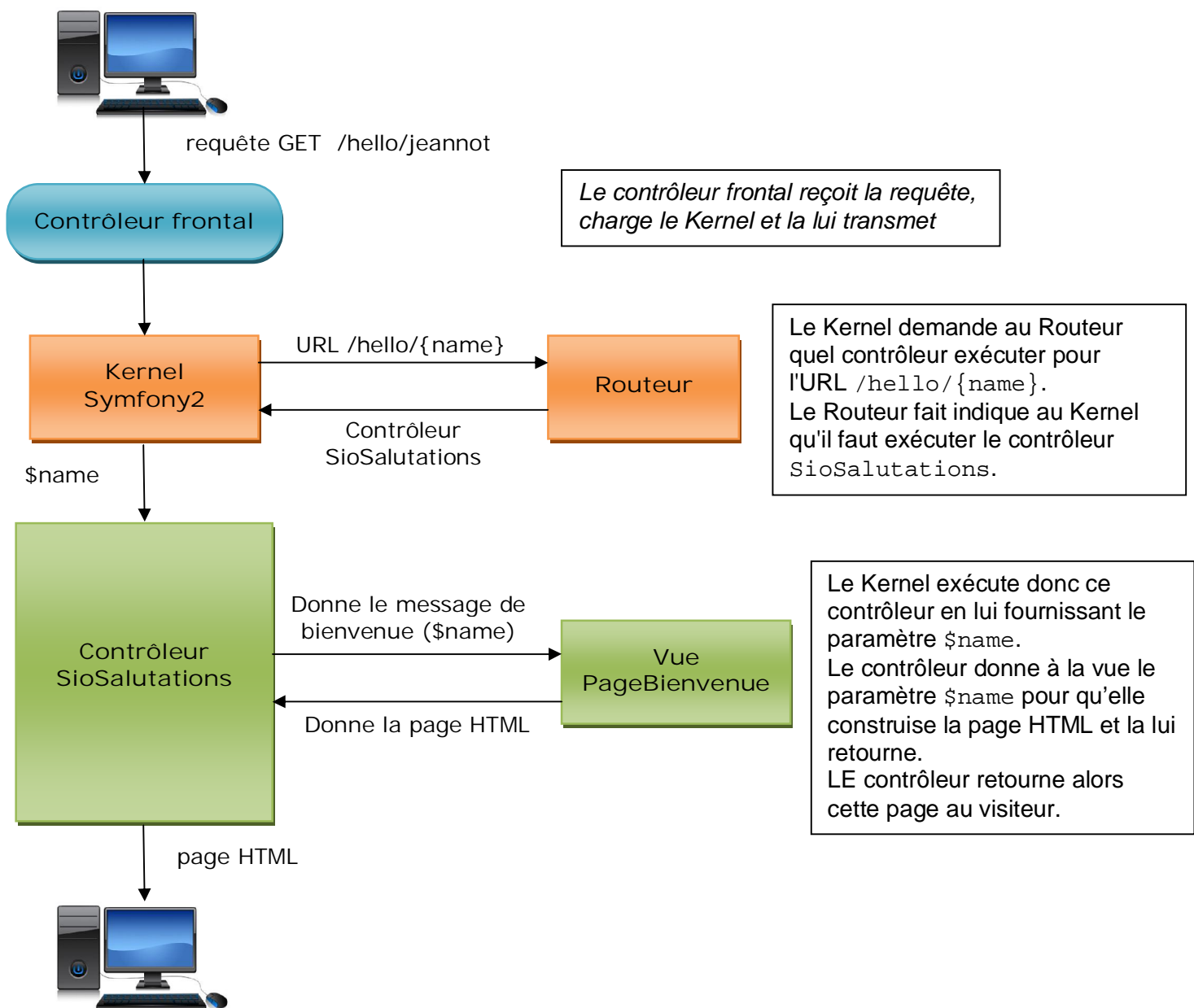


Vous pouvez tester votre bundle. Ouvrez la page suivante :
http://localhost/votreAlias/AppliMVC1/web/app_dev.php/hello/jeannot

Vous obtenez :



Essayons de reprendre tout ce que nous avons vu sur la partie théorique.



Voyons un peu ce qui a été généré dans les différents fichiers lors de la création du bundle.

- Symfony a enregistré le bundle auprès du Kernel

Fichier `/app/AppKernel.php`

```

1  <?php
2
3  use Symfony\Component\HttpKernel\Kernel;
4  use Symfony\Component\Config\Loader\LoaderInterface;
5
6  class AppKernel extends Kernel
7  {
8      public function registerBundles()
9      {
10         $bundles = array(
11             new Symfony\Bundle\FrameworkBundle\FrameworkBundle(),
12             new Symfony\Bundle\SecurityBundle\SecurityBundle(),
13             new Symfony\Bundle\TwigBundle\TwigBundle(),
14             new Symfony\Bundle\MonologBundle\MonologBundle(),
15             new Symfony\Bundle\SwiftmailerBundle\SwiftmailerBundle(),
16             new Symfony\Bundle\AsseticBundle\AsseticBundle(),
17             new Doctrine\Bundle\DoctrineBundle\DoctrineBundle(),
18             new Sensio\Bundle\FrameworkExtraBundle\SensioFrameworkExtraBundle(),
19             new Sio\SalutationsBundle\SioSalutationsBundle(),
20         );
21
22         if (in_array($this->getEnvironment(), array('dev', 'test'))) {
23             $bundles[] = new Acme\DemoBundle\AcmeDemoBundle();
24             $bundles[] = new Symfony\Bundle\WebProfilerBundle\WebProfilerBundle();
25             $bundles[] = new Sensio\Bundle\DistributionBundle\SensioDistributionBundle();
26             $bundles[] = new Sensio\Bundle\GeneratorBundle\SensioGeneratorBundle();
27         }
28
29         return $bundles;
30     }
31
32     public function registerContainerConfiguration(LoaderInterface $loader)
33     {
34         $loader->load(__DIR__.'/config/config_'.$this->getEnvironment().'.yml');
35     }
36 }

```

- Symfony a généré le fichier de routes du bundle

Fichier `/src/Sio/SalutationsBundle/Resources/config/routing.yml`

```

1  sio_salutations_homepage:
2      path:      /hello/{name}
3      defaults: { _controller: SioSalutationsBundle:Default:index }

```

Bundle
Contrôleur
Action

- Symfony2 a enregistré ces routes auprès du Routeur

Fichier `/app/config/routing.yml`

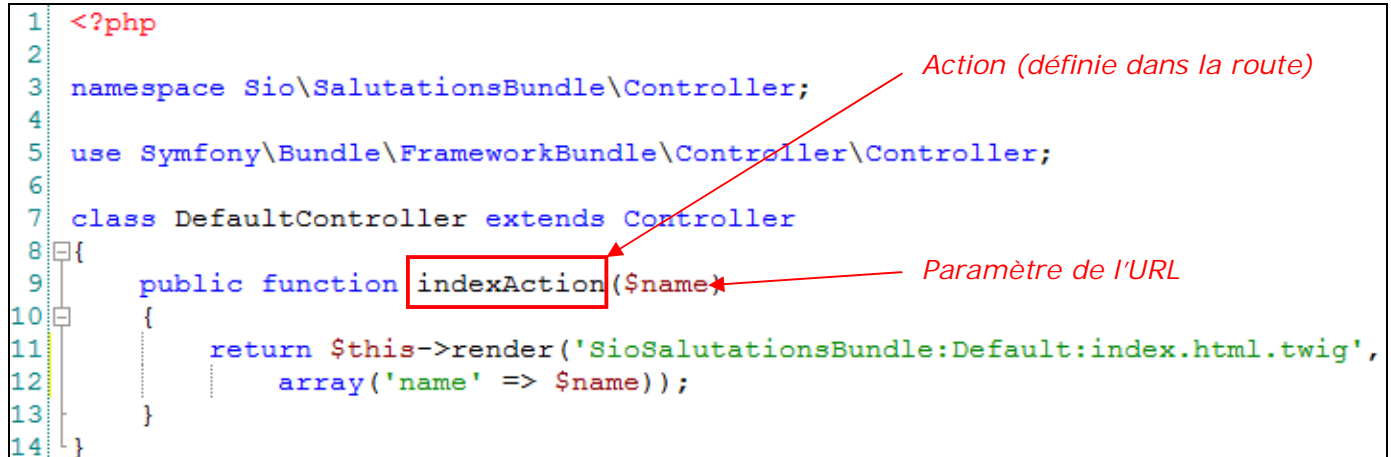
```

1  sio_salutations:
2      resource: "@SioSalutationsBundle/Resources/config/routing.yml"
3      prefix:   /

```


- Symfony2 a créé un contrôleur par défaut avec une action « index »

Fichier `/src/Sio/SalutationsBundle/Controller/DefaultController.php`



```

1  <?php
2
3  namespace Sio\SalutationsBundle\Controller;
4
5  use Symfony\Bundle\FrameworkBundle\Controller\Controller;
6
7  class DefaultController extends Controller
8  {
9      public function indexAction($name)
10     {
11         return $this->render('SioSalutationsBundle:Default:index.html.twig',
12             array('name' => $name));
13     }
14 }

```

Examinons d'un peu plus près la dernière instruction :

```

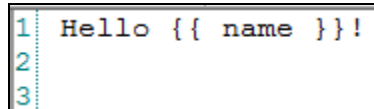
return $this->render('SioSalutationsBundle:Default:index.html.twig',
    array('name' => $name));

```

La méthode `render` permet d'appeler un template. Cette méthode prend en paramètre le nom du template, éventuellement un tableau de données à traiter par le template puis retourne un objet de type `Response` (réponse HTTP) qui est alors affiché.

- Symfony2 a créé un template pour le bundle

Fichier `/src/Sio/SalutationsBundle/Resources/views/Default`



```

1  Hello {{ name }}!
2
3

```

Et oui, c'est tout !

Enfin, bon, ce n'est pas parfait, car nous n'avons pas respecté les standards Html (Doctype, head, body...).

Nous allons maintenant pouvoir adapter tout ceci pour atteindre nos objectifs.

Mais avant de poursuivre, faisons un petit zoom sur l'URL.

`http://localhost/votreAlias/AppliMVC1/web/app_dev.php/hello/jeannot`

Qu'est-ce que cette partie ?

Nous avons vu (dans la partie théorique) que le contrôleur frontal était le point d'entrée de l'application. Or, ce contrôleur frontal se situe dans le répertoire `/web` et il s'agit du fichier `app_dev.php`.

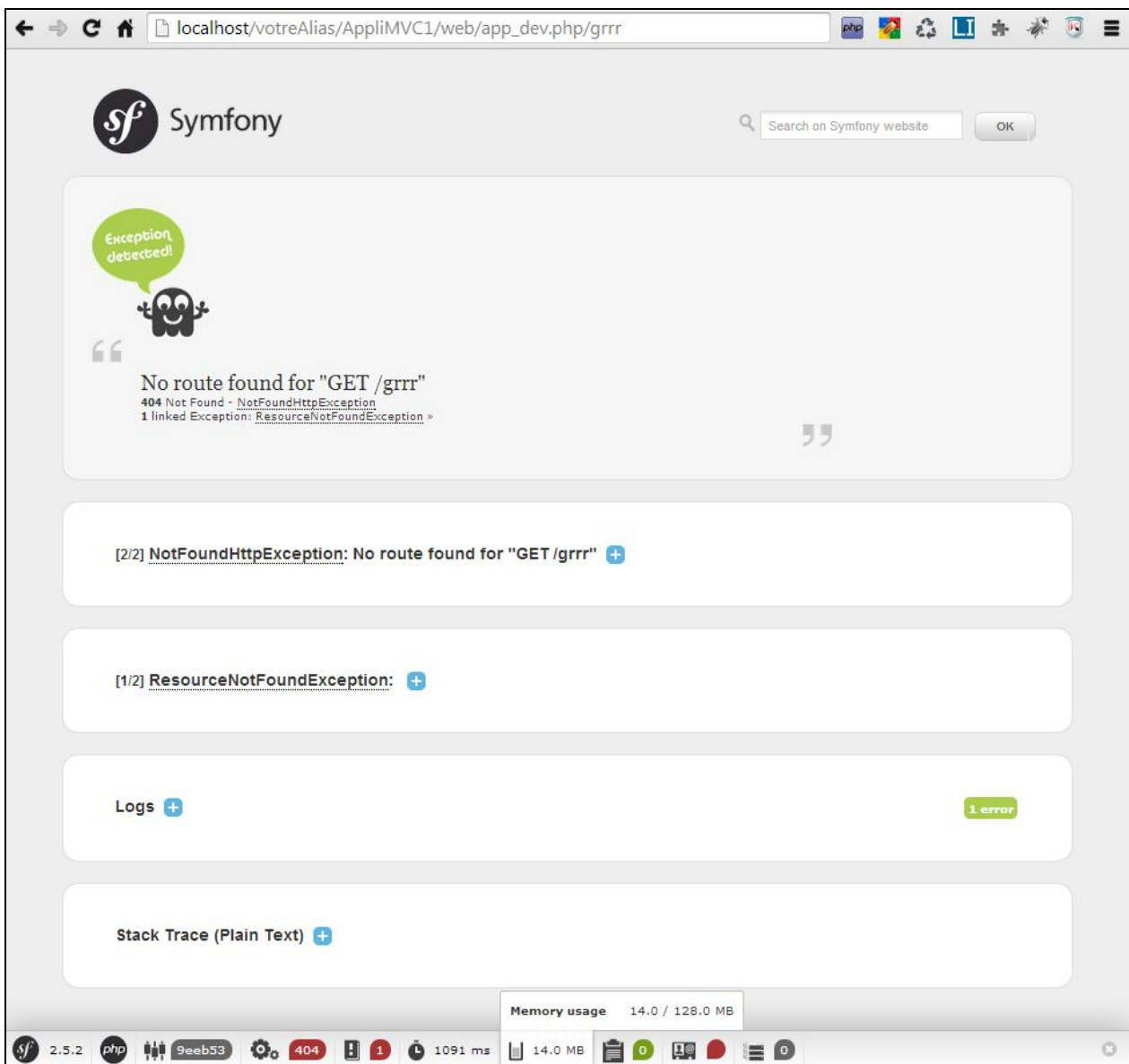
Pourquoi deux contrôleurs frontaux ? Tout simplement parce que Symfony2 propose deux contrôleurs frontaux : un pour l'environnement de développement (app_dev.php) et un pour l'environnement de production (app.php). 10/18

L'objectif est de répondre au mieux au besoin de chacun :

- Un développeur a besoin d'informations sur la page afin de l'aider à développer. En cas d'erreur, il veut tous les détails pour pouvoir déboguer facilement. Il n'a pas besoin de rapidité.
- Un visiteur normal n'a pas besoin d'informations particulières sur la page. En cas d'erreur, l'origine de celle-ci ne l'intéresse pas du tout, il veut juste retourner d'où il vient. Par contre, il veut que le site soit le plus rapide possible à charger.

Essayez pour voir la différence : appelez la page suivante, qui n'existe pas : /grrr

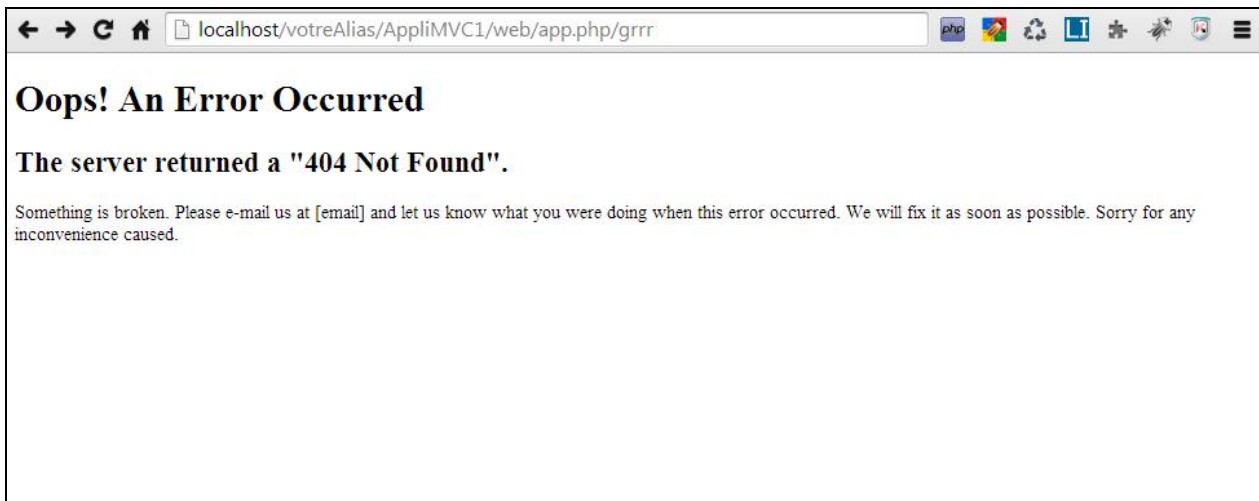
- En mode « dev », vous obtenez :



Dans ce mode, vous obtenez plein d'information sur l'origine de l'erreur qui vont pour aider à déboguer notre application.

Notez la présence de la barre d'outils en bas de la fenêtre !

- En mode « prod », vous obtenez :



Dans ce mode, les erreurs sont enregistrées dans le fichier `/app/logs/prod.log`.

Rassurez-vous, il est possible de changer les URL de votre site :

Plutôt que d'avoir quelque chose du genre : <http://monSite/web/app.php/blog>

Il est en effet préférable d'avoir : <http://monSite/blog>

Pour cela, il existe deux méthodes possibles :

- L'URL Rewriting
- Les VirtualHost

Mais pour l'instant, ce n'est pas notre priorité.

Configurer correctement notre éditeur

Pour éviter tout problème ultérieur, configurez correctement votre éditeur :

- Encodage des fichiers en UTF-8 sans BOM par défaut
- Remplacement des tabulations par des espaces

Gérer nos routes

Revenons donc à nos moutons : nous souhaitons gérer 3 types d'URL

L'url / qui devra afficher « Salut, toi ! »

L'url /bonjour/ qui devra afficher « Bonjour ! »

L'url /bonjour/Jeanot qui devra afficher « Bonjour Jeanot ! »

Cela signifie qu'il faut définir 3 routes pour notre Bundle.

Nous allons utiliser un contrôleur spécifique que l'on appellera `BonjourSauceSio` !

C'est juste pour vous montrer comment en créer un nouveau mais nous aurons pu évidemment utiliser le contrôleur par défaut.

➤ Le routeur principal n'a pas besoin d'être modifié :

```
sio_salutations:
  resource: "@SioSalutationsBundle/Resources/config/routing.yml"
  prefix: /
```

Fichier /app/config/routing.yml

➤ Il faut ajouter les nouvelles routes au fichier de routes de notre bundle :

```
sio_salutations_homepage:
  path:      /hello/{name}
  defaults: { _controller: SioSalutationsBundle:Default:index }

sio_home:
  path:      /
  defaults: { _controller: SioSalutationsBundle:BonjourSauceSio:index }

sio_bonjour:
  path:      /bonjour/
  defaults: { _controller: SioSalutationsBundle:BonjourSauceSio:bonjour }

sio_bonjourNominatif:
  path:      /bonjour/{name}
  defaults: { _controller: SioSalutationsBundle:BonjourSauceSio:bonjourNominatif }
```

Fichier /src/sio/SalutationsBundle/Resources/config/routing.yml

On crée une route pour chacune des URL que l'on souhaite gérer.

Rappels

- nom de la route : doit être unique
- path : url
- defaults : contrôleur à appeler

SioSalutationsBundle:BonjourSauceSio:bonjourNominatif

nom du Bundle nom du contrôleur nom de l'action



**PAS DE TABULATION DANS LES FICHIERS YML,
UNIQUEMENT DES ESPACES !!!!!!!**

Définir le contrôleur

13/18

Il faut maintenant créer le contrôleur et ses méthodes.

Sous `/src/Sio/SalutationsBundle/Controller/`, créer un nouveau script PHP nommé obligatoirement `BonjourSauceSioController.php`.

Règle

- tous les contrôleurs ont leur nom qui se termine par Controller

Ce script contient la classe du contrôleur avec ses 3 méthodes :

```
<?php
namespace Sio\SalutationsBundle\Controller;
use Symfony\Bundle\FrameworkBundle\Controller\Controller;
class BonjourSauceSioController extends Controller
{
    public function indexAction()
    {
        return $this->render('SioSalutationsBundle:BonjourSauceSio:index.html.twig');
    }

    public function bonjourAction()
    {
        return $this->render('SioSalutationsBundle:BonjourSauceSio:bonjour.html.twig');
    }

    public function bonjourNominatifAction($name)
    {
        return $this->render('SioSalutationsBundle:BonjourSauceSio:bonjourNominatif.html.twig',
            array('name' => $name));
    }
}
```

➤ `namespace Sio\SalutationsBundle\Controller;`

Il faut indiquer le namespace de la classe `BonjourSauceSioController`

➤ `use Symfony\Bundle\FrameworkBundle\Controller\Controller;`

Il faut indiquer que l'on utilise le namespace de la classe de base des contrôleurs

➤ `class BonjourSauceSioController extends Controller`

Notre classe hérite (« étend » dans le jargon PHP ou Java) de la classe de base `Controller`

- Ensuite on trouve nos 3 méthodes correspondant aux 3 actions définies dans le fichier de routes. Seule la dernière reçoit un paramètre (le nom du visiteur). Chacune appelle une vue qui lui est propre (nom identique à celui de l'action !)

Créer nos vues

14/18

Nous allons faire les choses bien et tout d'abord créer un « layout », autrement dit le modèle de base de nos pages.

Celui-ci doit se trouver dans le dossier `/app/resources/views` pour pouvoir être utilisé par tous les bundles de l'application.

Nous aurions pu le créer sous `/src/sio/SalutationsBundle/Resources/views` puisque nous n'avons qu'un seul bundle mais nous allons faire comme si !

- Créez le fichier `/app/resources/views/layout.html.twig` comme ci-dessous :

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>{% block title %}Salutations{% endblock %}</title>
    {% block stylesheets %}
    <link rel="stylesheet" href="{{ asset('css/styles.css') }}"
      type="text/css" />
    {% endblock %}
  </head>
  <body>
    <h1>Ma première application MVC avec Symfony</h1>
    {% block body %}{% endblock %}
    {% block javascripts %}
    <script src="//ajax.googleapis.com/ajax/libs/jquery/2.1.1/jquery.min.js">
    </script>
    {% endblock %}
  </body>
</html>
```

- Le bloc title
Il nous permettra d'afficher un titre propre à chaque page.
- Le bloc stylesheets
Nous allons utiliser de base une feuille de styles commune à toutes les pages. Comme le lien vers la feuille de styles est défini dans un bloc twig, il sera possible de redéfinir ce bloc, pour chaque vue individuellement. `{{ asset('css/styles.css') }}` signifie que le fichier `styles.css` se situe dans le dossier `/web/css/`.
- Le bloc body
Ce bloc est vide par défaut. Il sera défini par les vues de notre bundle.
- Le bloc javascripts
Ce bloc contient un lien vers la bibliothèque jquery disponible sur google. Nous n'utiliserons pas jQuery dans nos vues. Nous surchargerons donc ce bloc pas un bloc vide mais nous pourrions tout aussi bien ajouter un script JQuery propre à chacune de nos vues.

- Créez le fichier `/web/css/styles.css` suivant :

```
body {
    background-color: #c7defe;
}
```

- Nous allons maintenant créer les vues de notre. Cela se passe dans le dossier `/src/Sio/SalutationsBundle/Resources/views`.

Examinons notre contrôleur :

```
public function indexAction()
{
    return $this->render('SioSalutationsBundle:BonjourSauceSio:index.html.twig');
}

public function bonjourAction()
{
    return $this->render('SioSalutationsBundle:BonjourSauceSio:bonjour.html.twig');
}

public function bonjourNominatifAction($name)
{
    return $this->render('SioSalutationsBundle:BonjourSauceSio:bonjourNominatif.html.twig',
        array('name' => $name));
}
```

Il faut tout d'abord créer un dossier `BonjourSauceSio` (du même nom que le contrôleur).

Créons maintenant les 3 templates dans ce nouveau dossier.

- o Le template `index.html.twig`

```
1  {% extends "::layout.html.twig" %}
2
3  {% block title %}
4      Accueil - {{ parent() }}
5  {% endblock %}
6
7  {% block body %}
8      <h2>Salut, toi !</h2>
9  {% endblock %}
10
11 {% block javascripts %}
12 {% endblock %}
```

Ligne 1 : indique que l'on hérite du template `layout.html.twig`. Les « `::` » qui précèdent le nom du template signifient qu'il s'agit d'un template qui se trouve sous `/app/Resources/views`.

Lignes 3 à 5 : on redéfinit (= remplace) le bloc `title` du layout de base. L'instruction `{{ parent() }}` permet de récupérer ce qu'il y a dans le bloc `title` du layout de base à savoir « `Salutations` ». La page générée par ce template aura donc pour titre « `Accueil – Salutations` ».

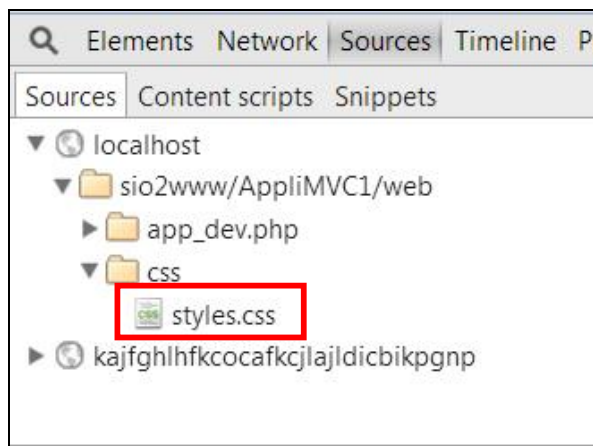
Lignes 7 à 8 : on remplit le bloc `body` (il était vide dans le layout de base).

Lignes 11 à 12 : on redéfinit le bloc `javascripts` du layout de base. Comme ce bloc est vide, on n'aura plus de référence vers la bibliothèque JQuery de Google.

Allez on teste !



- Le titre sur l'onglet est correct (titre du template index.html.twig suivi du titre du template de base).
- La balise <h1> définie uniquement dans le template de base est bien affichée.
- Le corps de la page <body> défini uniquement dans le template fille est bien affiché.
- Nos CSS sont bien prises en compte.
- La bibliothèque JQuery n'est pas chargée. On peut le vérifier par un clic droit au milieu de la page puis « Inspecter l'élément » puis onglet Sources qui affichent tous les fichiers chargés.



On retrouve bien notre fichier de feuilles de styles mais la bibliothèque JQuery est absente. Vous pouvez également le vérifier en affichant le code source de la page.

o Le template **bonjour.html.twig**

```

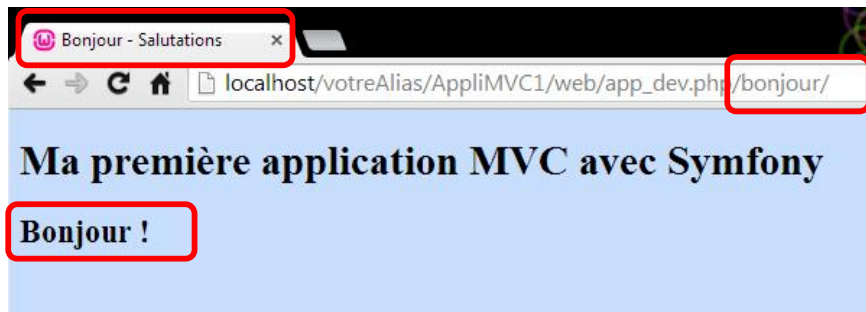
1  {% extends "::layout.html.twig" %}
2
3  {% block title %}
4      Accueil - {{ parent() }}
5  {% endblock %}
6
7  {% block body %}
8      <h2>Bonjour !</h2>
9  {% endblock %}
10
11 {% block javascripts %}
12 {% endblock %}

```

Même chose que pour le précédent. Seul le bloc body change.

On teste !

17/18



- o Le template `bonjourNominatif.html.twig`

```

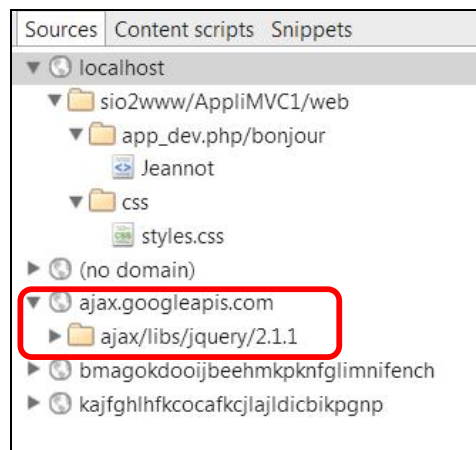
1  {% extends "::layout.html.twig" %}
2
3  {% block title %}
4      Accueil - {{ parent() }}
5  {% endblock %}
6
7  {% block body %}
8      <h2>Bonjour {{ name }} !</h2>
9  {% endblock %}

```

Ici aussi, le bloc body change.

On a également omis le bloc javascripts. Par conséquent, le bloc du layout de base sera pris « en l'état ».

Vérifions.



Un peu de ménage

Je vous l'avais promis, nous allons le faire !

➤ Supprimer le contrôleur par défaut créé dans notre Bundle par Symfony2

- o 1^{ère} étape : modifier le fichier de routes du bundle

Il faut supprimer la route `sio_salutations_homepage`.

On obtient alors :

```
sio_home:
  path:      /
  defaults: { _controller: SioSalutationsBundle:BonjourSauceSio:index }

sio_bonjour:
  path:      /bonjour/
  defaults: { _controller: SioSalutationsBundle:BonjourSauceSio:bonjour }

sio_bonjourNominatif:
  path:      /bonjour/{name}
  defaults: { _controller: SioSalutationsBundle:BonjourSauceSio:bonjourNominatif }
```

- o 2^{ème} étape : Supprimer le contrôleur par défaut

Supprimez le fichier `DefaultController.php` se trouvant dans le répertoire `/src/Sio/SalutationsBundle/Controller/`

- o 3^{ème} étape : Supprimer la vue associée

Supprimez le dossier `/src/Sio/SalutationsBundle/Resources/views/Default/`

- o 4^{ème} étape : Testez (et oui, il faut s'assurer que tout fonctionne encore correctement)

➤ Supprimer le bundle Acme livré avec Symfony2

- o 1^{ère} étape : vérifier le fichier de routage principal

Le fichier `/app/config/routing.yml` ne doit pas référencer le bundle Acme. Normalement, ce n'est pas le cas mais vérifiez tout de même. Voici à quoi il doit ressembler :

```
sio_salutations:
  resource: "@SioSalutationsBundle/Resources/config/routing.yml"
  prefix:   /
```

- o 2^{ème} étape : supprimer le bundle Acme

Là, c'est très simple, il suffit de supprimer le répertoire `/src/Acme`.

- o 3^{ème} étape : Testez !!!

- o Ça plante ? C'est normal. Essayez de corriger les problèmes !

Dans le prochain TP nous nous intéresserons à la couche modèle.