



Les Formulaires

1. Présentation du travail

L'idée est de travailler sur les formulaires.

Vous allez construire un formulaire avec des templates hérités, inclure

- ✓ Du css
- ✓ Des images
- ✓ Des hyperliens

2. Préparation du travail

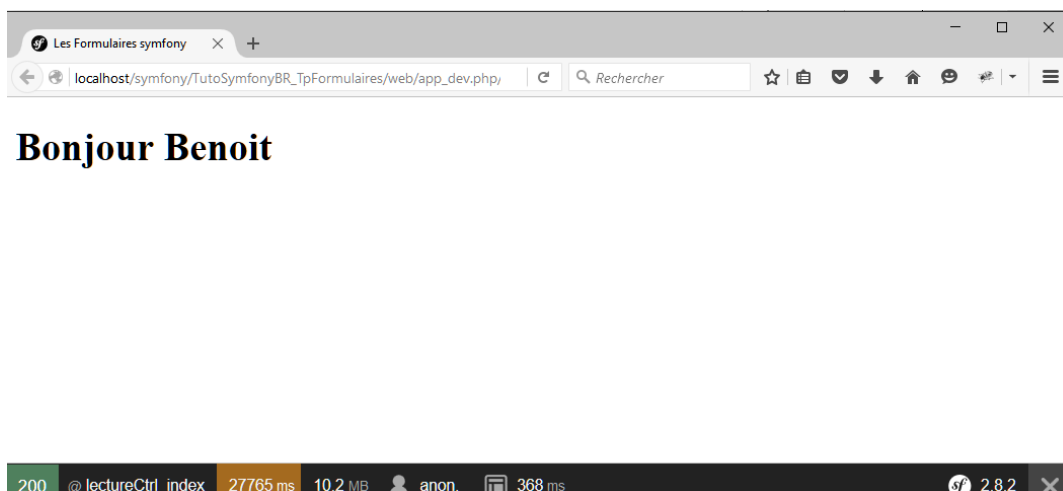
- ✓ Créer un nouveau projet Symfony 2.8
Nom : TutoSymfonyBR_TpFormulaires
- ✓ Créer un nouveau projet TutoSymfonyBR_TpFormulaires sous Netbeans with existing source
- ✓ Créer le contrôleur **LectureCtrl** (classe LectureCtrlController) avec la commande Symfony2 (routes par annotation) et la route /index/{nom} . Cette route appelle l'action indexAction du contrôleur LectureCtrlController qui appelle la vue index.html.twig

Indice

Mettre aussi un use sur la classe Response et Request

Faire un test avec l'URL http://localhost/.../web/app_dev.php/index/Benoît

Vous devriez obtenir :



Solution pour la vue view.html.twig :

```
index.html.twig x LectureCtrlController.php x
Source History
1 { % extends "::base.html.twig" % }
2
3 { % block title % } Les Formulaires symfony { % endblock % }
4
5 { % block body % }
6     <h1>Bonjour {{ nom }}</h1>
7 { % endblock % }
```

Un peu de ménage :

Supprimer

- ✓ Le contrôleur par défaut du bundle AppBundle : DefaultController
- ✓ Le dossier Default du dossier views (si il y est)

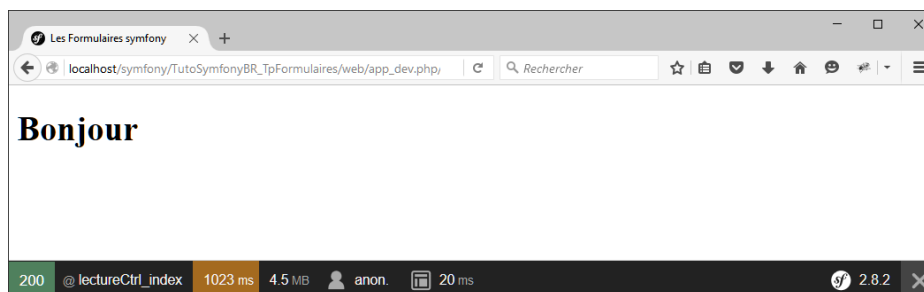
Modifier la route /index/{name} et la remplacer par /index. De plus, elle appellera l'action index du contrôleur lectureCtrlController

Faire les modifications au niveau de la méthode indexAction du contrôleur et dans le template index.html.twig

Tester avec l'url

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

Vous devriez obtenir ceci :



Ouf, on y est !!!
On peut commencer à coder, notre environnement est en place

3. Génération automatique de formulaire à partir d'une classe

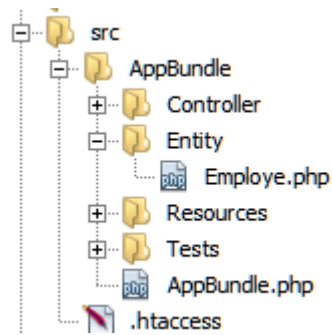
Créer le dossier GestLectureBundle\Entity

Benoît ROCHE

Page 2 sur 21

BR_Symfony_TpFormulaire16V01.docx

Dans ce dossier créer la classe Employe (fichier Employe.php)



```
namespace Bdls\GestLectureBundle\Entity;

class Employe {

    private $num;
    private $nom;
    private $prenom;
    private $dateNaissance;

    public function getNum() { ...3 lines }

    public function setNum($num) { ...3 lines }

    public function getNom() { ...3 lines }

    public function setNom($nom) { ...3 lines }

    public function getPrenom() { ...3 lines }

    public function setPrenom($prenom) { ...3 lines }

    public function getDateNaissance() { ...3 lines }

    public function setDateNaissance($dateNaissance) {
        $this->dateNaissance = $dateNaissance;
    }

    // Pas de constructeur à faire pour l'instant
    function __construct($num, $nom, $prenom, $dateNaissance) { ...6 lines }
}
```

Ajouter un use sur la classe Employe dans le fichier controller

(use Bdl\GestLectureBundle\Entity\Employe)

```
use Bdl\GestLectureBundle\Entity\Employe;

class LectureCtrlController extends Controller {
```

Et écrire le code de l'action indexAction :

```
/**
 * @Route(
 *     path="/index",
 *     name="lectureCtrl_index"
 * )
 */
public function indexAction() {
    $employe = new Employe(1, "Dupont", "Jean", new \DateTime('1988-12-25'));
    $formulaire = $this->createFormBuilder($employe)
        ->add('num', 'number')
        ->add('nom', 'text')
        ->add('prenom', 'text')
        ->add('dateNaissance', 'date')
        ->add('Enregistrer', 'submit')
        ->getForm();
    return $this->render('AppBundle:LectureCtrl:index.html.twig',
        array('leFormulaire' => $formulaire->createView()));
}
```

Modifier le fichier index.html.twig dans le dossier views/LectureCtrl de sorte qu'il contienne ceci :

```
{% extends "::base.html.twig" %}

{% block title %}Les Formulaires symfony{% endblock %}

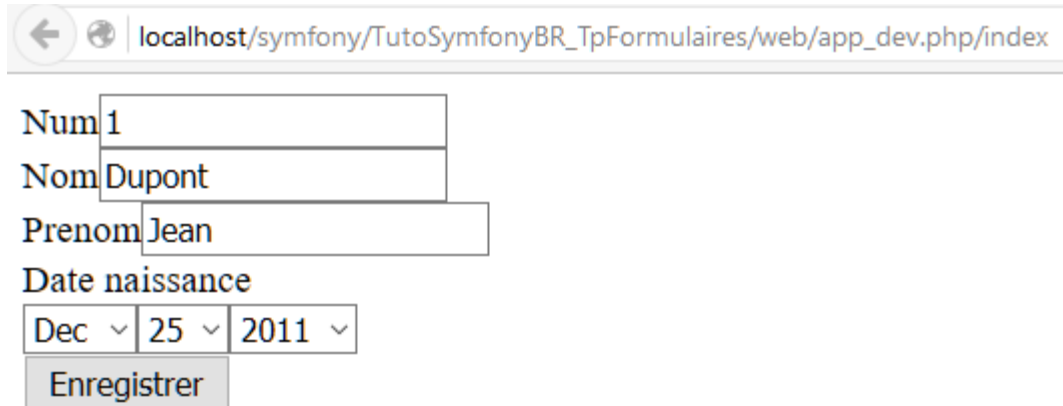
{% block body %}
    {{ form(leFormulaire) }}
{% endblock %}
```

La balise twig va afficher la vue envoyée à l'aide de la variable formulaire

```
$formulaire = $this->createFormBuilder($employe)
```

```
twig', array('leFormulaire' => $formulaire->createView()));
```

Tester avec l'url : http://localhost...web/app_dev.php/index



localhost/symfony/TutoSymfonyBR_TpFormulaires/web/app_dev.php/index

Num 1

Nom Dupont

Prenom Jean

Date naissance

Dec ▾ 25 ▾ 2011 ▾

Enregistrer

... pas beau, mais efficace

...On peut aussi faire un peu différent :

Modifier la vue view.html.twig

```
{% extends "::base.html.twig" %}
```

```
{% block title %}Premiere{% endblock %}
```

```
{% block body %}
    {{form_start(leFormulaire) }}
    <p> {{ form_row(leFormulaire.num, {'label' : 'Numéro :'}) }} </P>
    <p><H2>{{ form_row(leFormulaire.nom, {'label' : 'Nom de l\'employee'}) }}
    {{ form_row (leFormulaire.prenom,
    {'label' : 'Prénom de l\'employee'}) }}

    </H2> </p>
    <p>{{ form_row(leFormulaire.dateNaissance) }}</p>
    {{form_end(leFormulaire) }}
{% endblock %}
```

Le résultat :



Numéro : 1

Nom de l'employe Dupont

Prénom de l'employe Jean

Date naissance

Dec 25 2011

Enregistrer

C'est pas top, Mais on a introduit de la mise en forme On verra, on peut appliquer des styles.
Le code twig



On a introduit la notion d'Entity Très important, on les reverra lors de l'accès aux bases de données avec Doctrine.

4. Passer d'autres paramètres

On peut bien entendu passer d'autres paramètres au formulaire.

Créer la route suivante dans le contrôleur LectureCtrl :

nom = lectureCtrl_ afficheEmpParam

route : /afficheEmpParam

action : afficheEmpParam Action

Attention, le nom de la vue a changé !!!

L'idée : afficher la vue afficheEmpParam.html.twig

Premiere

localhost/symfony/TutoSymfonyBR_TpFormulaires/web/app_dev.php/

Numéro : 1

Nom de l'employe Dupont

Prénom de l'employe Jean

Date naissance
Dec 25 2011

Enregistrer

Son sport préféré : RUGBY

ses voitures préférées :

Ferrari

Aston Martin

Porsche

Indices :

Le sport est dans la variable *sport*. On peut aussi ne passer aucun sport !
 Les autos sont stockées dans le tableau *autos*
 Il faudra mettre en place une boucle twig
 Il faudra tester la présence de la variable *sport*

Solution :

Le code de la vue autreParam.html.twig :

```
{% extends "::base.html.twig" %}

{% block title %}Premiere{% endblock %}

{% block body %}
    {{form_start(leFormulaire) }}
    <p> {{ form_row(leFormulaire.num, {'label' : 'Numéro :'}) }} </P>
    <p><H2>{{ form_row(leFormulaire.nom, {'label' : 'Nom de l\'employe'}) }}
    {{ form_row (leFormulaire.prenom, {'label' : 'Prénom de l\'employe'}) }}
    </H2> </p>
    <p>{{ form_row(leFormulaire.dateNaissance) }}</p>
    {{form_end(leFormulaire) }}
    <p><h3> Son sport préféré :
    {% if(sport=='Rugby') %}
        {{sport|upper}}
    {% else %}
        {{'Aucun intérêt'}}
    {% endif %} </h3></p>
    <p><H3> ses voitures préférées : </h3>
    {% for uneAuto in autos %}
    <li><h4>{{ uneAuto }}</h4></li>
    {% endfor %}
    </p>
{% endblock %}
```

Code de l'action du contrôleur :

```
/**
 * @Route("/afficheEmpParam",
 * name="appBundle_afficheEmpParam")
 *
 */
public function afficheEmpParamAction() {
    $employe = new Employe(1, "Dupont", "Jean", new \DateTime('1988-12-25'));
    $formulaire = $this->createFormBuilder($employe)
        ->add('num', 'number')
        ->add('nom', 'text')
        ->add('prenom', 'text')
        ->add('dateNaissance', 'date')
        ->add('Enregistrer', 'submit')
        ->getForm();
    $sport = 'Rugby';
    $autos = array('Ferrari', 'Aston Martin', 'Porsche');
    return $this->render('AppBundle:LectureCtrl:afficheEmpParam.html.twig',
        array('leFormulaire' => $formulaire->createView(),
            'sport' => $sport, 'autos' => $autos));
}
```

On remarque le test et les boucles Merci twig !!!

<http://twig.sensiolabs.org/doc/tags/index.html>

5. Beaucoup plus loin :

Vous l'aurez remarqué, nos formulaires ne sont pas beaux ...

On va faire de beaux formulaires avec du css. Pourquoi pas aussi du javascript !!!

On va utiliser le principe de l'héritage de templates !!!

Le principe :

- ✓ un template père qui contient le design du site. **Il sera constitué de blocks.**
- ✓ Des templates fils qui vont remplir ces blocks en utilisant la notion d'héritage.

Par exemple, on trouvera comme blocks :



- ✓ Le header,
- ✓ le footer,
- ✓ le centre,
- ✓ éventuellement le menu sur la gauche

Par convention, le template père s'appelle layout.html.twig et se trouve dans le dossier views des ressources du bundle



Répondez aux questions suivantes :

Dans votre projet actuel, y a-t-il un template par défaut ?

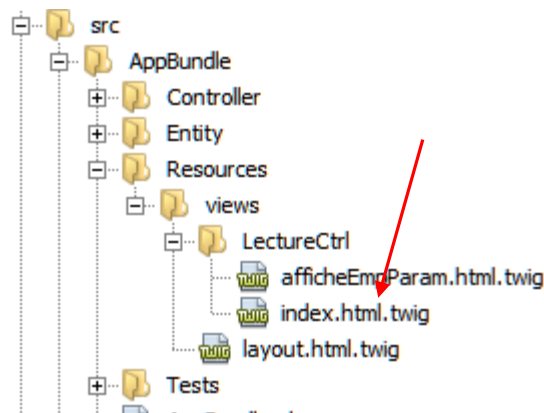
Si oui quel est-il ? où est-il ? Pourquoi ?

Peut-on faire différemment ?

L'idée est donc d'avoir un template père que l'on placera dans les ressources du bundle AppBundle.

Si vous avez plusieurs bundles et que vous travaillez toujours avec le même modèle de template, vous pouvez placer le template père dans un dossier ...app\Resources\views\xxx

Par convention, on appellera le template père layout.html.twig et on va le placer dans le dossier views des ressources du bundle AppBundle :



Et voici à quoi il ressemble ... [voir](#)

```

<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        {% block stylesheets %} {% endblock %}
        <title>{% block title %}{% endblock %}</title>
    </head>
</head>
<body>
    <header id="headerSite">
        <aside id="logo">
        </aside>
        <nav id="menuHeader">
            <ul>{% block liens %} {% endblock %}</ul>
        </nav>
        <div class="separateurHorizontal"></div>
    </header>
    <section id="descriptifDivers">
        <h1>{% block head %} {% endblock %} </h1>
        <h2>{% block body %} {% endblock %}</h2>
        <p>{% block widgets %} {% endblock %}</p>
        <H1> Bravo pour l'exercice !!!!</H1>
    </section>
    <footer id="footerSite">
        {% block footer %}{% endblock %}
    </footer>
</body>
</html>

```

Et voici le template qui hérite de ce template père ... [voir](#)

```

{% extends 'AppBundle::layout.html.twig' %}
{% block title %} Deux {% endblock %}
{% block head %} tete à l'envers {% endblock %}
{% block body %} <H1>{{ uneEntree }}</H1> {% endblock %}
{% block widgets %}{{ form(leFormulaire) }} {% endblock %}
{% block liens %}
    <li><a href={{path('lectureCtrl_index')}}> Index</a></li>
    <li><a href={{path('AppBundle_afficheEmpParam')}}> Exercice Paramètres</a></li>
    <li><a href={{path('AppBundle_afficheEmpTwig')}}> Exercice Twig</a></li>
{%endblock%}

```

Qui est appelé par la méthode afficheEmpTwigAction :

http://localhost/symfony/TutoSymfonyBR_TpFormulaires/web/app_dev.php/afficheEmpTwig

```

/**
 * @Route("/afficheEmpTwig",
 * name="AppBundle_afficheEmpTwig")
 *
 */
public function afficheEmpTwigAction() {

    //on test si l'objet request est de type post (formulaire envoyé de plus par post
    $request = $this->get('request');
    if ($request->getMethod() === 'POST') {
        $a = 0;
    }
    $employe = new Employe(1, "Dupont", "Jean", new \DateTime('1988-12-25'));
    $employe->setNum($request->request->get('nom'));
    // $employe->setNum(1);

    $employe->setPrenom("Jean");
    $employe->setDateNaissance(new \DateTime('today'));

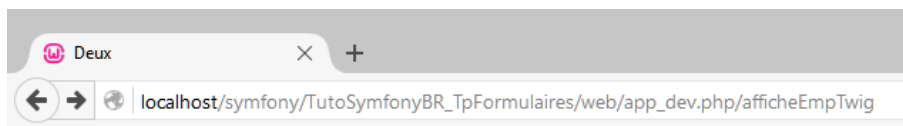
    $formulaire = $this->createFormBuilder($employe)
        ->add('num', 'number')
        ->add('nom', 'text')
        ->add('prenom', 'text')
        ->add('dateNaissance', 'date')
        ->add('Enregistrer', 'submit')
        ->getForm();

    return $this->render('AppBundle:LectureCtrl:afficheEmpTwig.html.twig',
        array('uneEntree' => 'je viens du controller',
            'leFormulaire' => $formulaire->createView()));
}

```

Vous devriez obtenir ceci :

http://symfony.br/TutoSymfonyBR_Formulaire/web/app_dev.php/afficheEmpTwig



- [Index](#)
- [Exercice Paramètres](#)
- [Exercice Twig](#)

tete à l'envers

je viens du controller

Num	<input type="text"/>
Nom	<input type="text" value="Dupont"/>
Prenom	<input type="text" value="Jean"/>
Date naissance	
	<input type="text" value="Feb"/> <input type="text" value="25"/> <input type="text" value="2016"/>
<input type="button" value="Enregistrer"/>	

Bravo pour l'exercice !!!!

Que remarque-t-on, en dehors du fait que ce formulaire est laid ?

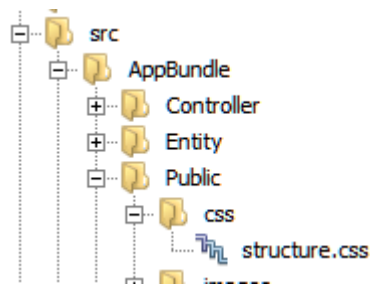
- ✓ La correspondance des blocks. Les fils se substituent aux pères. On pourra aussi faire cohabiter le contenu avec le contenu des pères
- ✓ Les liens href : on utilise la fonction twig path pour indiquer la route correspondant au lien... eh oui, on ne peut pas mettre de lien "en dur" On doit indiquer une route !!!



Mais, ce n'est toujours pas beau !!!

On va rajouter une feuille de style !!!
(voir [fichier structure.css](#) fourni)

On place ce fichier dans le dossier :
...\TutoSymfonyBR_Formulaire\src\AppBundle\Public\css\structure.css



Vous allez créer la méthode `AfficheEmpTwigCssAction`, qui répondra à la route `/AfficheEmpTwigCss`
 Et qui renverra la vue `AfficheEmpTwigCss.html.twig`
 Vous ferez bien entendu du copier/coller

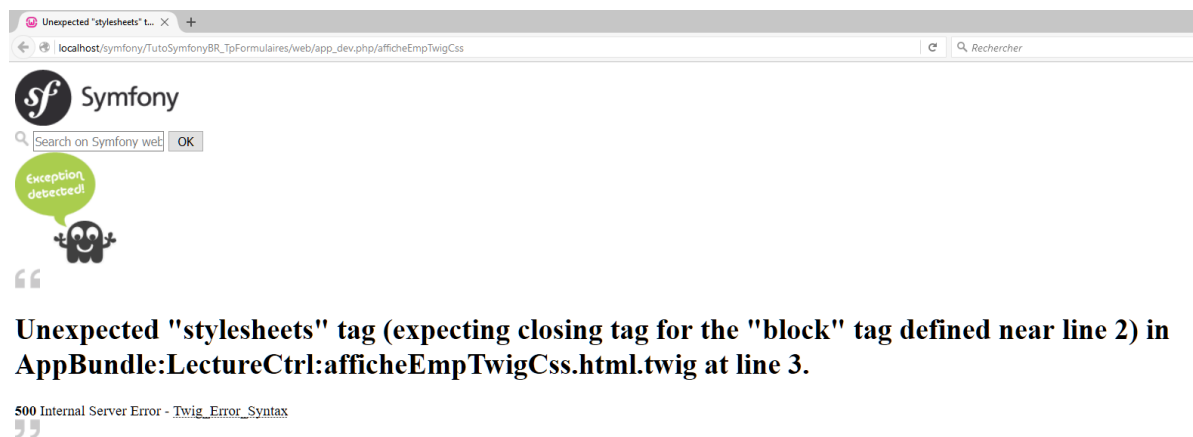
Et mettons le lien en place dans le fichier `layout.html.twig`:

```
{% block css %}{% endblock %}
<title>{% block title %}{% endblock %}</title>
:/head>
```

Et définissons notre fichier css dans le template fils `AfficheEmpTwigCss.html.twig` :

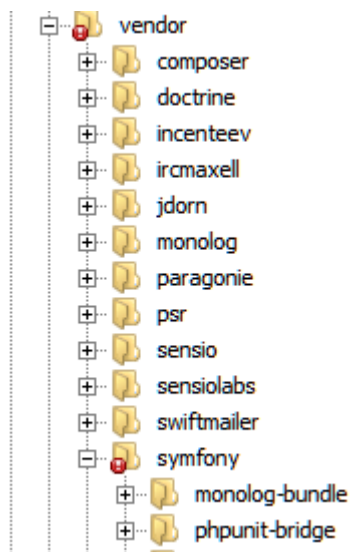
```
{% block css %}
{% stylesheets '@AppBundle/public/css/structure.css' %}
<link rel="stylesheet" href="{{ asset_url }}" type="text/css" />
{% endstylesheets %}
{% endblock %}
```

Vous pouvez avoir cette erreur :

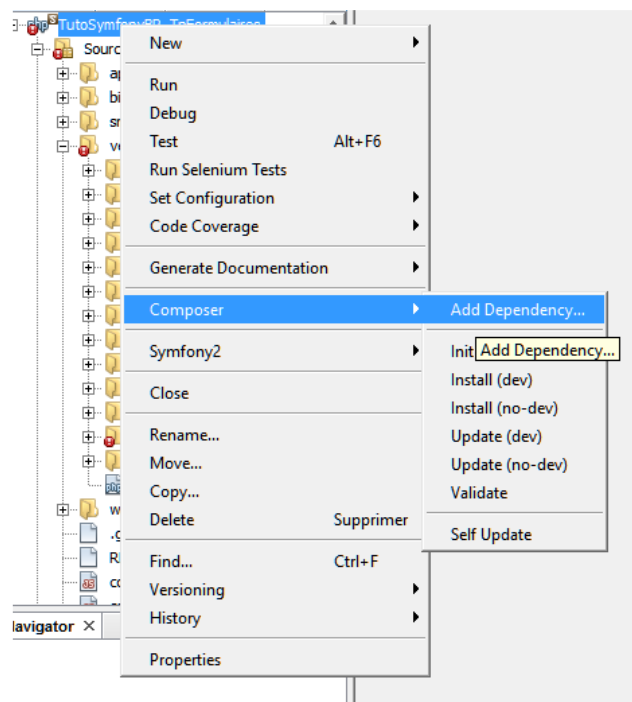


C'est peut-être que le bundle `asseticBundle` n'est pas installé

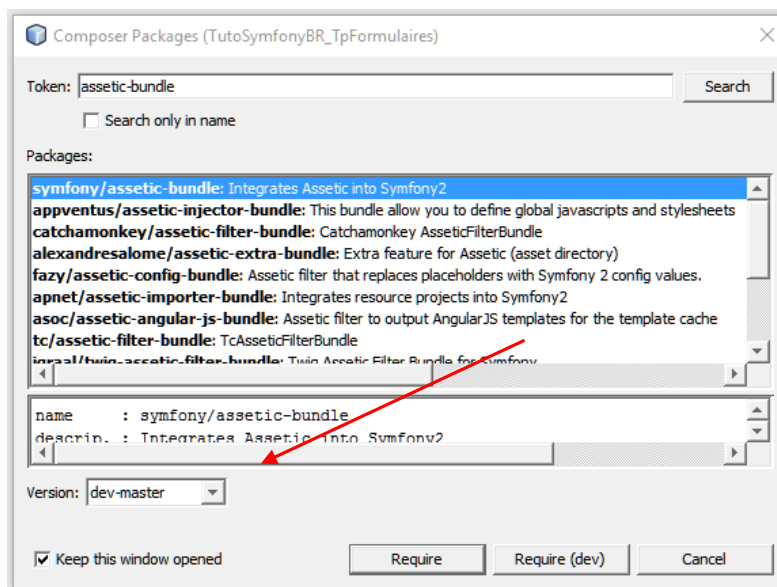
Vérifions dans `vendor/symfony`:



Il n'y est pas, il faut l'installer :



Après une recherche via composer, on peut installer asseticBundle:



Enfin le bundle est installé

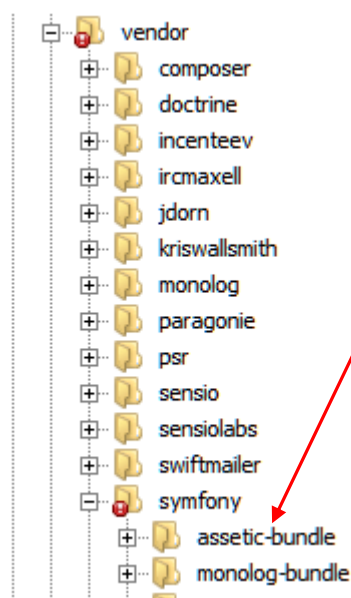
WARNING FrameworkBundle copy

! [NOTE] Some assets were installed via copy. If you make changes to these assets you have to run this command again.

[OK] All assets were successfully installed.

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

Vérifions :



Mais ce n'est pas fini

Il faut enregistrer le bundle dans le fichier de configuration du projet : app/AppKernel.php

```
$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 Doctrine\Bundle\DoctrineBundle\DoctrineBundle(),  
    new Sensio\Bundle\FrameworkExtraBundle\SensioFrameworkExtraBundle(),  
    new AppBundle\AppBundle(),  
    new Symfony\Bundle\AsseticBundle\AsseticBundle()  
);
```

Puis configurer notre bundle pour l'application : app/config/config.yml

```
assetic:  
    debug:          "%kernel.debug%"  
    use_controller: false  
    bundles:        [ ]  
    filters:  
        cssrewrite: ~
```



Attention à bien respecter les indentations (4 espaces)
Si les mots clés ne sont pas écrits en bleu, c'est faux !!!

Ouf ... c'est fait, poursuivons !!

On aurait pu tout mettre dans le layout.html.twig, mais cela dépend de la façon dont vous vous organisez.

On aurait aussi pu inclure plusieurs feuilles de style et des scripts javascript de la même manière !

Testons :

http://symfony.br/TutoSymfonyBR_Formulaire/web/app_dev.php/afficheEmpTwigCss



Une belle erreur Qui nous dit qu'il faut configurer le bundle assetic. En effet, le rôle de ce bundle assetic est de regrouper nos ressources en un seul fichier afin de réduire les temps de chargement.

Pour configurer assetic, on va donc aller retourner dans le fichier de configuration
...\TutoSymfonyBR_Formulaire\app\config\config.yml

Et trouver la partie assetic pour lui indiquer quel bundle doit être pris en charge :

```
assetic:
    debug:          "%kernel.debug%"
    use_controller: false
    bundles:         ["AppBundle"]
    filters:
        cssrewrite: ~
```

On relance :

http://symfony.br/TutoSymfonyBR_Formulaire/web/app_dev.php/afficheEmpTwigCss

... ça marche ! Quelle belle interface !



Waouwww ... c'est mieux ... Bon, faudra remplacer les informations par des données plus sérieuses.....

On s'amuse encore un peu ?

On va rajouter un icône

Toujours dans le fichier afficheEmpTwigCss.html, on va rajouter les lignes :

```
{% block icone %}
    <a href={{path('appBundle_afficheEmpParam')}} title='Tutoriel_symphony'>
        {% image '@AppBundle/public/images/clindoeil.png' %}
        
        {% endimage %}
    </a>
{% endblock %}

{% block title %} Deux {% endblock %}
```

Et bien entendu, on a modifié le layout pour lui indiquer le block icone :

```
{% block css %}{% endblock %}
{% block icone %}{% endblock %}
<title>{% block title %}{% endblock %}</title>
.. ..
```



.. à vous de vérifier

http://symfony.br/TutoSymfonyBR_Formulaire/web/app_dev.php/afficheEmpTwigCss

De mieux en mieux !!!



Et puis Une petite image réactive ?

Il suffit de modifier le fichier afficheEmpTwigCss :

```
] {% block icone %}
]     <a href={{path('AppBundle_afficheEmp')}} title='Tutoriel_symphony'>
|         {% image '@AppBundle/public/images/clindoeil.png' %}
|         
|         {% endimage %}
-     </a>
- {% endblock %}
1
```

On teste ?...

C'est cool, non ?

Et pour finir, dans le footer, on va rajouter les mentions légales... dans un block ... au cas où on voudrait le changer dans une vue...

```
{% block footer %}
    <footer id="footerSite">

        <div class="separateurHorizontal"></div>
        <article id="descriptifFooter">
            <header>
                <h2>Mentions légales</h2>
            </header>
            <p>
                Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam dapibus.
            </p>
        </article>
        <article id="partenaireFooter">
            <header>
                <h2>Contacts</h2>
            </header>
            <p>
                Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam dapibus.
            </p>
        </article>

    </footer>
{% endblock %}
```

.. on teste ?

Bravo pour l'exercice !!!!

MENTIONS LÉGALES

Lorem ipsum dolor sit amet,
consectetur adipiscing elit.
Aliquam dapibus.

CONTACTS

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam dapibus.

Et si on redéfinissait ce block dans le template fils (afficheEmpTwigCss.html.twig) ?

```
3  <!-- a met {{path( appendre_afficheEmpParam )}} Afficher un emp  
4  {% endblock %}  
5  {% block footer %} <div class="separateurHorizontal"></div>  
6  <article id="descriptifFooter">  
7  <header>  
8  <h2>Mentions légales</h2>  
9  </header>  
10 <p>  
11     Nouvelles mentions légales  
12 </p>  
13 </article>  
14 <article id="partenaireFooter">  
15 <header>  
16 <h2>Contacts</h2>  
17 </header>  
18 <p>  
19     <li> Monsieur Symfony </li>  
20     <li> Monsieur Twig </li>  
21     <li> Monsieur Doctrine </li>  
22 </p>  
23 </article>  
24 {% endblock %}
```

Résultat :

Bravo pour l'exercice !!!!

MENTIONS LÉGALES Nouvelles mentions légales	CONTACTS <ul style="list-style-type: none">• Monsieur Symfony• Monsieur Twig• Monsieur Doctrine
---	---

... A vous de vous amuser, de voir et de tester ...



Une suggestion

Voir bootstrap (<http://getbootstrap.com/>)

Un framework frontend qui va vous faire gagner du temps pour vos css.....

Bon courage !!!

