



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

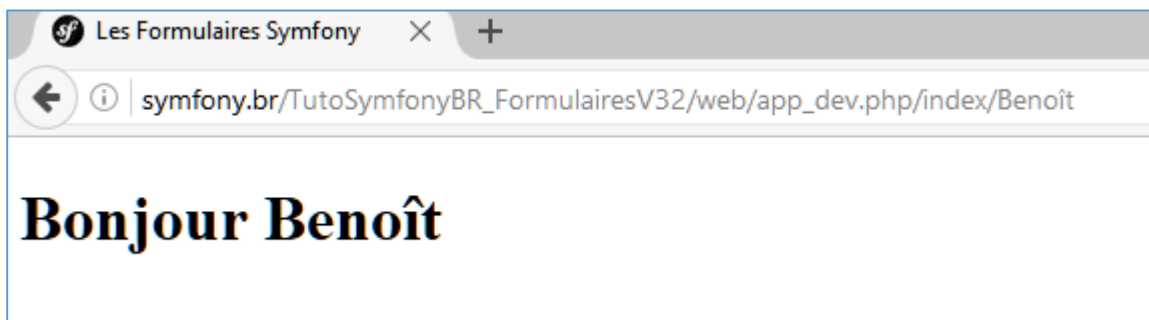
- ✓ Créer un nouveau projet Symfony 3.2
Nom : TutoSymfonyBR_FormulairesV32
- ✓ Créer un nouveau projet TutoSymfonyBR_FormulairesV32 sous Netbeans with existing source
- ✓ Créer le contrôleur **TpForm** (classe TpFormController) avec la commande Symfony (routes par annotation) et la route /index/{nom} . Cette route appelle l'action indexAction du contrôleur TpFormController qui appelle la vue index.html.twig. La variable nom est une chaîne de caractères qui accepte des noms composés du type Dupont-Dupond
- ✓ Supprimer le contrôleur DefaultController
- ✓ Mettre un use sur les classes Response et Request dans le fichier TpFormController.php

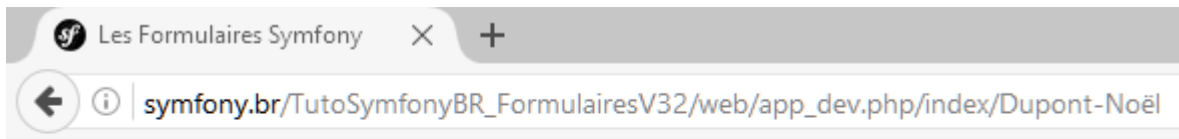
Faire un test avec l'URL

http://symfony.br/TutoSymfonyBR_FormulairesV32/web/app_dev.php/index/Benoît

Vous devriez obtenir :

Ou encore :





Bonjour Dupont-Noël

Solution pour la vue index.html.twig :

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

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

{% block body %}
    <h1>Bonjour {{ nom }} </h1>
{% endblock %}
```



Ce qui serait TOP, ce serait de mettre le titre des formulaires de l'application en paramètre de l'application....

Pour faire ceci :

Modifier le fichier config.yml et rajouter l'entrée globals :

```
# Twig Configuration
twig:
    debug:                "%kernel.debug%"
    strict_variables:      "%kernel.debug%"
    globals:
        twigTitle: '%twig.title%'
```

Implémenter le paramètre dans le fichier parameters.yml :

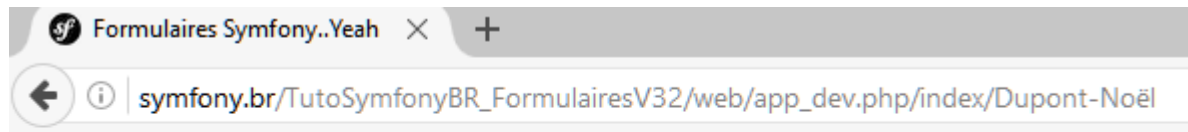
```
secret: 283fbaeafb6d9ecb69add534f0eeaa8f73bcebe0
twig.title: "Formulaires Symfony..Yeah"
```

Et on le rajoute dans le fichier parameters.yml.dist :

```
#twig parameters
twig.title: "Formulaires Symfony..Yeah"
```

Modifier le template index.html.twig pour intégrer cette variable globale :

Et voilà le résultat :



Bonjour Dupont-Noël

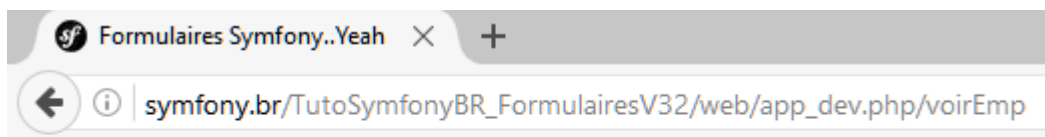
Après ce petit exercice de révision, créer la route /voirEmp qui

- ✓ S'appellera lectureCtrl_voirEmp
- ✓ appellera l'action voirEmpAction du contrôleur TpFormController
- ✓ affichera le template voirEmp.html.twig

Tester avec l'url

http://symfony.br/TutoSymfonyBR_FormulairesV32/web/app_dev.php/voirEmp

Vous devriez obtenir ceci :



Bonjour



Ouf, on y est !!!

On peut commencer à coder, notre environnement est en place

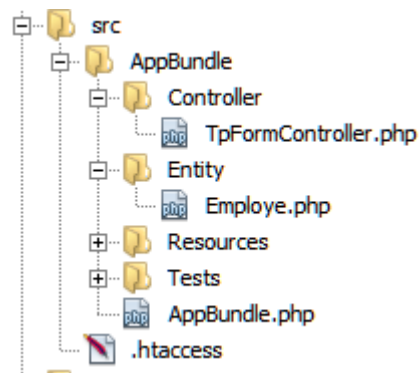
3. Génération automatique de formulaire à partir d'une classe

Créer le dossier AppBundle\Entity

Dans ce dossier créer la classe Employe (fichier Employe.php). Elle sera dans l'espace de noms AppBundle\Entity



Attention à bien respecter la casse



On crée la classe :

```

namespace AppBundle\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) { ...3 lines }

    function __construct($num, $nom, $prenom, $dateNaissance) {
        $this->num = $num;
        $this->nom = $nom;
        $this->prenom = $prenom;
        $this->dateNaissance = $dateNaissance;
    }
}

```

Ajouter un use sur la classe Employe dans le fichier TpFormcontroller
(use AppBundle\Entity\Employe)

```

use AppBundle\Entity\Employe;

class TpFormController extends Controller
{

```

Et écrire le code de l'action voirEmpAction :

```
public function voirEmpAction() {
    $employe = new Employe(1, "Dupont", "Jean", new \DateTime('1988-12-25'));
    $formulaire = $this->createFormBuilder($employe)
        ->add('num', IntegerType::class)
        ->add('nom', TextType::class, array('label'=>'Votre nom :'))
        ->add('prenom', TextType::class, array('label'=>'Votre prénom :'))
        ->add('dateNaissance', DateType::class, array('label'=>'Votre date de naissance :'))
        ->add('Enregistrer', SubmitType::class)
        ->getForm();
    return $this->render('AppBundle:TpForm:voirEmp.html.twig', array('leFormulaire' => $formulaire->createView()));
}
```

Modifier le fichier voirEmp.html.twig dans le dossier
AppBundle\Resources\views\TpForm\voirEmp.html.twig

de sorte qu'il contienne ceci :

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

{% block title %}{{ twigTitle }}{% endblock %}

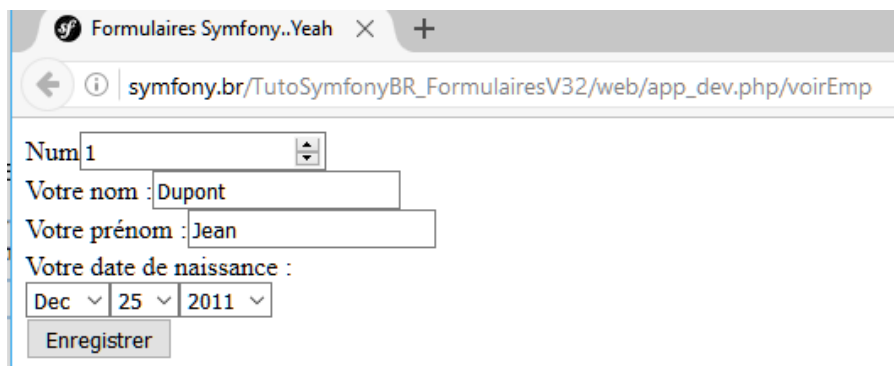
{% block body %}
    {{ form(leFormulaire) }}
{% endblock %}
```

Le template voirEmp.html.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://symfony.br/TutoSymfonyBR_FormulairesV32/web/app_dev.php/voirEmp



... pas beau, mais efficace



On peut accéder à chacun des champs du formulaire pour les mettre en forme. Pour ceci, on va utiliser la fonction twig `form_start()`.

Modifier la vue `view.html.twig`

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

{% block title %}{{ twigTitle }}{% endblock %}

{% block body %}
    {{form_start(leFormulaire) }}
    <H4>
        {{ form_row(leFormulaire.num,{'label' : 'Numéro :'} ) }}
        {{ form_row(leFormulaire.nom) }}
        {{ form_row(leFormulaire.prenom) }}
        {{ form_row(leFormulaire.dateNaissance) }}
    </H4>
    <div>
</div>
        {{form_end(leFormulaire) }}
        <H5> Fin du formulaire </h5>
    {% endblock %}
```

... Ainsi, on accède individuellement à chacun des champs du formulaire et on attribut au champ numéro un label.

Le résultat :

Numéro : 1

Votre nom : Dupont

Votre prénom : Jean

Votre date de naissance :
Dec 25 2011

Enregistrer

Fin du formulaire

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



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_voirEmpParam
- ✓ route : /voirEmpParam
- ✓ action : voirEmpParamAction

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

L'idée : afficher la vue voirEmpParam.html.twig suivante qui tient compte du sport passé :

Formulaires Symfony..Yeah X +

symfony.br/TutoSymfonyBR_FormulairesV32/web/app_dev.php/voirEmpParam

Numéro : 1

Votre nom : Dupont

Votre prénom : Jean

Votre date de naissance : Dec 25 2011

Enregistrer

Fin du formulaire

Son sport préféré : **RUGBY**

ses voitures préférées :

Ferrari

Aston Martin

Porsche

Formulaires Symfony..Yeah X +

symfony.br/TutoSymfonyBR_FormulairesV32/web/app_dev.php/voirEmpParam

Numéro : 1

Votre nom : Dupont

Votre prénom : Jean

Votre date de naissance : Dec 25 2011

Enregistrer

Fin du formulaire

Son sport préféré : **Aucun intérêt**

ses voitures préférées :

Ferrari

Aston Martin

Porsche

URL : http://symfony.br/TutoSymfonyBR_FormulairesV32/web/app_dev.php/voirEmpParam

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 voirEmpParam.html.twig :

```

{% block body %}
    {{form_start(leFormulaire) }}
    <H4>
        {{ form_row(leFormulaire.num,{'label' : 'Numéro :'}) }}
        {{ form_row(leFormulaire.nom) }}
        {{ form_row(leFormulaire.prenom) }}
        {{ form_row(leFormulaire.dateNaissance) }}
    </H4>
    <div>
    </div>
    {{form_end(leFormulaire) }}
    <H5> Fin du formulaire </h5>
    <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("/voirEmpParam",
 * name="lectureCtrl_voirEmpParam")
 *
 */
public function voirEmpParamAction() {
    $employe = new Employe(1, "Dupont", "Jean", new \DateTime('1988-12-25'));
    $formulaire = $this->createFormBuilder($employe)
        ->add('num', IntegerType::class)
        ->add('nom', TextType::class, array('label'=>'Votre nom :'))
        ->add('prenom', TextType::class, array('label'=>'Votre prénom :'))
        ->add('dateNaissance', DateType::class, array('label'=>'Votre date de naissance :'))
        ->add('Enregistrer', SubmitType::class)
        ->getForm();

    $sport = 'Rugby';
    $autos = array('Ferrari', 'Aston Martin', 'Porsche');
    return $this->render('AppBundle:TpForm:voirEmpParam.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 et utiliser bootstrap en mode CDN !!!

Le principe :

On va créer un héritage à 2 niveaux :

- ✓ un template général qui contient le design du site. ***Il sera constitué de blocks.***
- ✓ Un template bundle qui héritera du bundle de l'application et contiendra des informations propres au bundle
- ✓ Un template de l'action qui héritera de du template du bundle

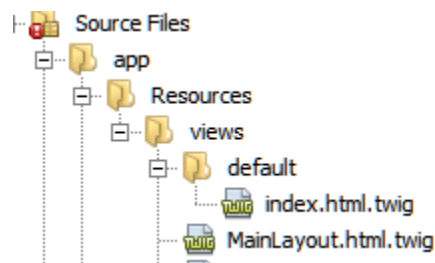
Le template général

Par exemple, on trouvera comme blocks :

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

Pour la clarté du cours, le template de l'application s'appellera mainLayout.html.twig

Il se trouvera obligatoirement dans les dossier `app/Resources/views`



Le fichier est [ici](#)

```
{# app/Resources/views/mainLayout.html.twig #}
```

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<meta charset="utf-8">
```

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

```
<title>{% block title %}{{ twigTitle }}{% endblock %}</title>
```

```
{% block stylesheets %}
```

```
{# On récupère bootstrap en mode CDN (depuis le site #)}
```

```
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
```

```
{% endblock %}
```

```
</head>
```

```
<body>
```

```
<div class="container">
```

```
<div id="header" class="jumbotron">
```

```
<h3>Les Formulaires en symfony 3 et Bootstrap</h3>
```

```
<h4>
```

```
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam consequat lorem ligula, at scelerisque tortor mattis eu. Donec ut elit semper enim efficitur fringilla. Quisque lorem sapien, feugiat egestas felis at, pretium tempor enim.
```

```
</h4>
```

```
<a class="btn btn-primary btn-lg" href="//symfony.com/doc/current/templating.html">
```

```
Documentation Symfony templates </a>
```

```
</a>
```

```
</p>
```

```
</div>
```

```
<div class="row">
```

```
<div id="menu" class="col-md-3">
```

```
<h3>Les annonces</h3>
```

```
<ul class="nav nav-pills nav-stacked">
```

```
<li><a href="{{path('lectureCtrl_index', {'nom': 'Benoit'})}}> Index</a></li>
```

```
<li><a href="{{path('lectureCtrl_voirEmp')}}> Exercice Paramètres</a></li>
```

```
<li><a href="{{path('lectureCtrl_voirEmp')}}> Exercice Twig</a></li>
```

```
</ul>
```

```
</div>
```

```
<div id="content" class="col-md-9">
```

```
{% block body %}
```

```
{% endblock %}
```

```
</div>
```

```
</div>
```

```
<hr>
```

```
<footer>
```

```
<p>Mentions légales </p>
```

```
<p>Nous sommes le {{ 'now' | localizeddate('full', 'none') }} </p>
```

```
</footer>
```

```
</div>
```

```
{% block javascripts %}
```

```
{# les liens CDN vers le bootstrap Twitter #}
```

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
```

```
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>*
```

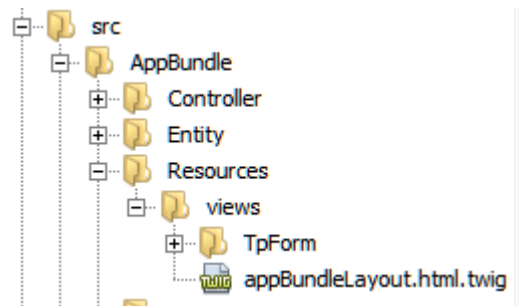
```
{% endblock %}
```

```
</body>
```

```
</html>
```

Le template du bundle

Pour la clarté du cours, le template du bundle s'appellera appBundleLayout.html.twig
Il se trouvera obligatoirement dans le dossier `src\AppBundle\Resources\views`



Le fichier est [ici](#)

```
{# src\AppBundle\Resources\views\appBundleLayout.html.twig #}
{% extends "::mainLayout.html.twig" %}

{% block title %}
    AppBundle - {{ parent() }}
{% endblock %}

{% block body %}

    {# Titre commun à chaque action du Bundle #}
    <h2>exercice formulaires</h2>

    <hr>

    {# block dans lequel va s'insérer le contenu issu des actions #}
    {% block mesActions_body %}
    {% endblock %}

{% endblock %}
```

Le template de l'action

On va créer une nouvelle action :

- ✓ nom = lectureCtrl_voirEmpTwig
- ✓ route : /voirEmpTwig
- ✓ action : voirEmpTwigAction

Elle fera la même chose que l'action voirEmpParamAction



Ne pas oublier de changer de template...

```
/**
 * @Route("/voirEmpTwig",
 * name="lectureCtrl_voirEmpTwigm")
 */
public function voirEmpTwigAction() {
    $employe = new Employe(1, "Dupont", "Jean", new \DateTime('1988-12-25'));
    $formulaire = $this->createFormBuilder($employe)
        ->add('num', IntegerType::class)
        ->add('nom', TextType::class, array('label' => 'Votre nom :'))
        ->add('prenom', TextType::class, array('label' => 'Votre prénom :'))
        ->add('dateNaissance', DateType::class, array('label' => 'Votre date de naissance :'))
        ->add('Enregistrer', SubmitType::class)
        ->getForm();
    $sport = 'Tennis';
    $autos = array('Ferrari', 'Aston Martin', 'Porsche');
    return $this->render('AppBundle:TpForm:voirEmpTwig.html.twig', array('leFormulaire' => $formulaire->createView(),
        'sport' => $sport, 'autos' => $autos));
}
```

Le template :

Le fichier est [ici](#)

```

{# src\AppBundle\Resources\views\TpForm\voirEmpTwig.html.twig #}

{% extends "AppBundle::appBundleLayout.html.twig" %}

{% block title %}
    voirEmpTwig - {{ parent() }}
{% endblock %}

{% block mesActions_body %}

    <h2>Gestion des employés</h2>

    {{ form_start(leFormulaire) }}
    <h4>
        {{ form_row(leFormulaire.num, {'label' : 'Numéro :'}) }}
        {{ form_row(leFormulaire.nom) }}
        {{ form_row(leFormulaire.prenom) }}
        {{ form_row(leFormulaire.dateNaissance) }}
    </h4>
    <div>
    </div>
    {{ form_end(leFormulaire) }}
    <h5> Fin du formulaire </h5>
    <p><h3> Son sport préféré :
        {% if(sport=='Rugby') %}
            {{sport|upper}}
        {% else %}
            {'Aucun intérêt'}
        {% endif %} </h3></p>
    <h3> ses voitures préférées : </h3>
    {% for uneAuto in autos %}
    <li><h4>{{ uneAuto }}</h4></li>
    {% endfor %}
{% endblock %}

```

Il ne nous reste plus qu'à tester :

http://symfony.br/TutoSymfonyBR_FormulairesV32/web/app_dev.php/voirEmpTwig

Et voilà le résultat :

voirEmpTwig - appBundle - F... X +

symfony.br/TutoSymfonyBR_FormulairesV32/web/app_dev.php/voirEmpTwig

Rechercher

Les Formulaires en symfony 3 et Bootstrap

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam consequat lorem ligula, at scelerisque tortor mattis eu. Donec ut elit semper enim efficitur fringilla. Quisque lorem sapien, feugiat egestas felis at, pretium tempor enim.

[Documentation Symfony templates »](#)

Menu

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

exercice formulaires

Gestion des employés

Numéro : 1

Votre nom : Dupont

Votre prénom : Jean

Votre date de naissance : 25 déc. 2011

[Enregistrer](#)

[Fin du formulaire](#)

Son sport préféré : Aucun intérêt

ses voitures préférées :

- Ferrari
- Aston Martin
- Porsche

Mentions légales

Nous sommes le mardi 13 décembre 2016

Qui est appelé par la méthode afficheEmpTwigAction :

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



Avez-vous remarqué la date en bas ?
Comment est-elle arrivée ici ...?

Dans le layout principal, on a rajouté le champ twig suivant :

```
<p>Mentions légales </p>  
<p> Nous sommes le {{ 'now' | localizeddate('full', 'none') }} </p>
```

Vous comprenez, pour afficher la date courante, on doit filtrer le mot 'now' en appliquant un format et surtout en affichant la date dans un format donné.

Si on laisse les choses en l'état, il y a de grandes chances que votre date soit formatée en anglais...
Du type :

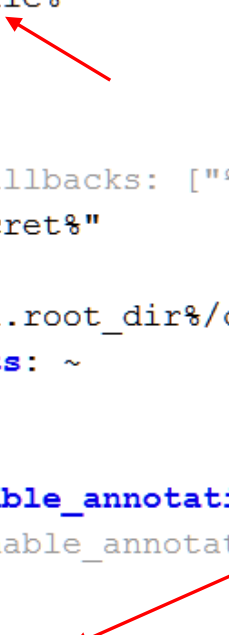
On va donc définir la langue française dans les paramètres de l'application.

- ✓ Fichier config.yml de l'application :

On renseigne les paramètres default_locale en général et pour le framework :

```
parameters:
    default_locale: "%locale%"

framework:
    #esi: ~
    #translator: { fallbacks: ["%locale%"] }
    secret: "%secret%"
    router:
        resource: "%kernel.root_dir%/config/routing.yml"
        strict_requirements: ~
    form: ~
    csrf_protection: ~
    validation: { enable_annotations: true }
    #serializer: { enable_annotations: true }
    templating:
        engines: ['twig']
    default_locale: "%locale%"
```



Il faut maintenant donner une valeur à ces paramètres :

- ✓ Fichier parameters.yml :

```
twig_title: 'Formulaires Symfony..Yeah'
locale: fr
```

- ✓ Mais aussi, le fichier parameters.yml.dist :

```
mailer_password: ~
locale: fr
```

Ce fichier permet de garder une trace des paramètres et il se synchronisera avec le fichier parameters.yml si celui-ci venait à être reconstruit, par exemple après un composer update.

- ✓ Mais ce n'est pas tout ... Pour utiliser ces fonctionnalités de twig, il faut utiliser les extensions de twig. Pour les mettre en place, il faut charger les extensions grâce à composer.

Dans le dossier de l'application, entrer la commande suivante :

composer require twig/extensions

et enfin, modifier le fichier services.yml pour créer le service qui instancie et met à disposition les extensions de twig :

```
services:

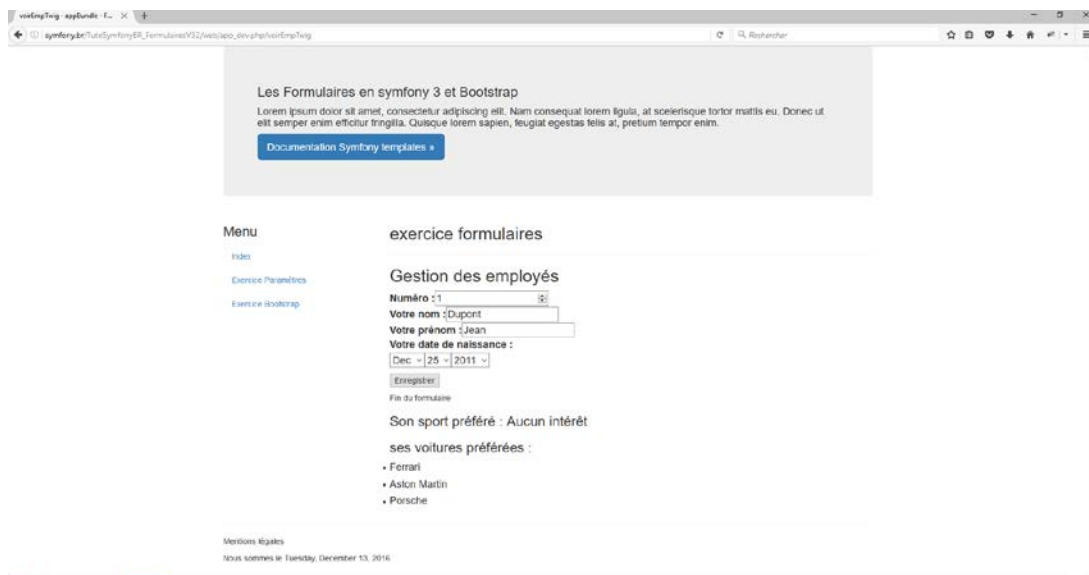
    twig.extension.intl:
        class: Twig_Extensions_Extension_Intl
        tags:
            - { name: twig.extension }
```



Prenez garde à bien écrire les fichiers twig..... Parfois vous insèrerez une tabulation au lieu des 4 espaces, vous ne vous en rendrez pas compte car la coloration syntaxique sera bonne.

Vider le cache avant de tester votre template :

http://symfony.br/TutoSymfonyBR_FormulairesV32/web/app_dev.php/voirEmpTwig



6. Travail avec des ressources (assets) : css, js, png

On va reprendre le travail qu'on vient de faire, pour ce faire, on va créer une nouvelle action :

- ✓ nom = lectureCtrl_voirEmpAssets
- ✓ route : /voirEmpAssets
- ✓ action : voirEmpAssetsAction

On va également créer :

- ✓ Un nouveau template de l'application : layout.html.twig
- ✓ Un nouveau template du bundle : appBundle.html.twig

- ✓ Un nouveau template pour l'action : voirEmpAssets.html.twig

Bootstrap et jquery en CDN, c'est bien, mais pas top si on perd la connexion internet Donc on va mettre les ressources en local.

L'idée ... On ne va plus utiliser bootstrap et jquery en CDN, mais bel et bien installer les fichiers en local.

En plus de Bootstrap et jquery, qui seront des ressources globales à l'application, on utilisera un fichier css spécifique au bundle AppBundle

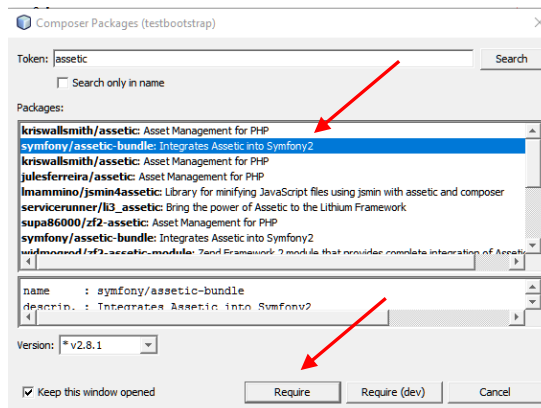


Comme on l'a vu en début de cours, il est nécessaire que les ressources (Assets) soient placées dans le dossier web, le seul qui soit exposé sur le web.

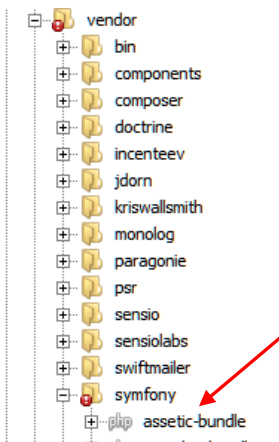
Une pratique recommandée par symfony est de placer les ressources où on veut dans l'arborescence. Le bundle **AsseticBundle** va se charger de gérer les ressources en optimisant leur chargement et en les compressant.

Installation de Assetic :

Dans la console composer : Add dependency : Chercher symfony/assetic-bundle



Le bundle a bien été chargé :



Configuration :

Ajouter le bundle dans le fichier app/AppKernel.php :

```
$bundles = [  
    new Symfony\Bundle\FrameworkBundle\FrameworkBundle(),  
    new Symfony\Bundle\SecurityBundle\SecurityBundle(),  
    new Symfony\Bundle\TwigBundle\TwigBundle(),  
    new Symfony\Bundle\MonologBundle\MonologBundle(),  
    new Symfony\Bundle\SwiftmailerBundle\SwiftmailerBundle(),  
    new Doctrine\Bundle\DoctrineBundle\DoctrineBundle(),  
    new Sensio\Bundle\FrameworkExtraBundle\SensioFrameworkExtraBundle(),  
    new AppBundle\AppBundle(),  
    new Symfony\Bundle\AsseticBundle\AsseticBundle(),
```

Puis le fichier app/config.yml :

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

Le mot clé assetic sur la première colonne.

Noter qu'on a donné le bundle AppBundle dans la liste des bundles à gérer pour Assetic.

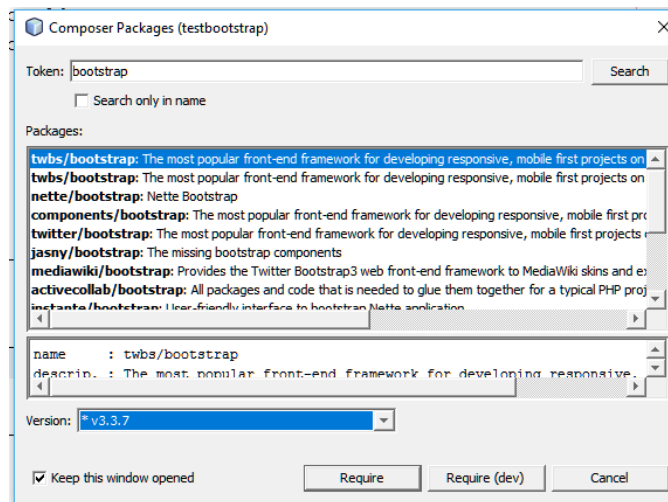


Attention à bien respecter les indentations (4 espaces)
Si les mots clés ne sont pas écrits en bleu, c'est faux !!!

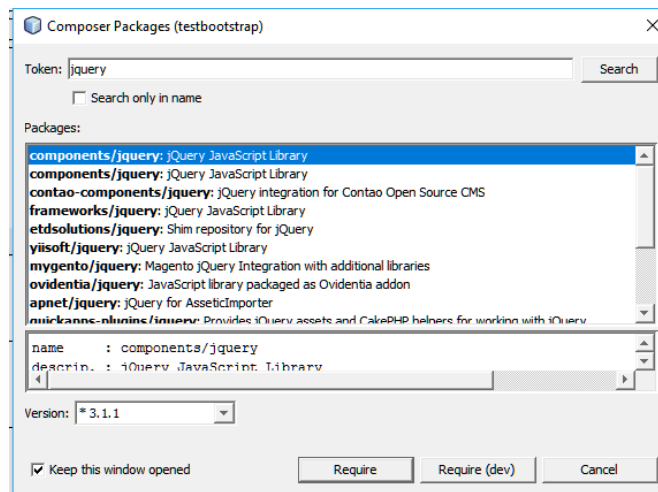
On va utiliser composer pour installer bootstrap et jquery

Installation de bootstrap :

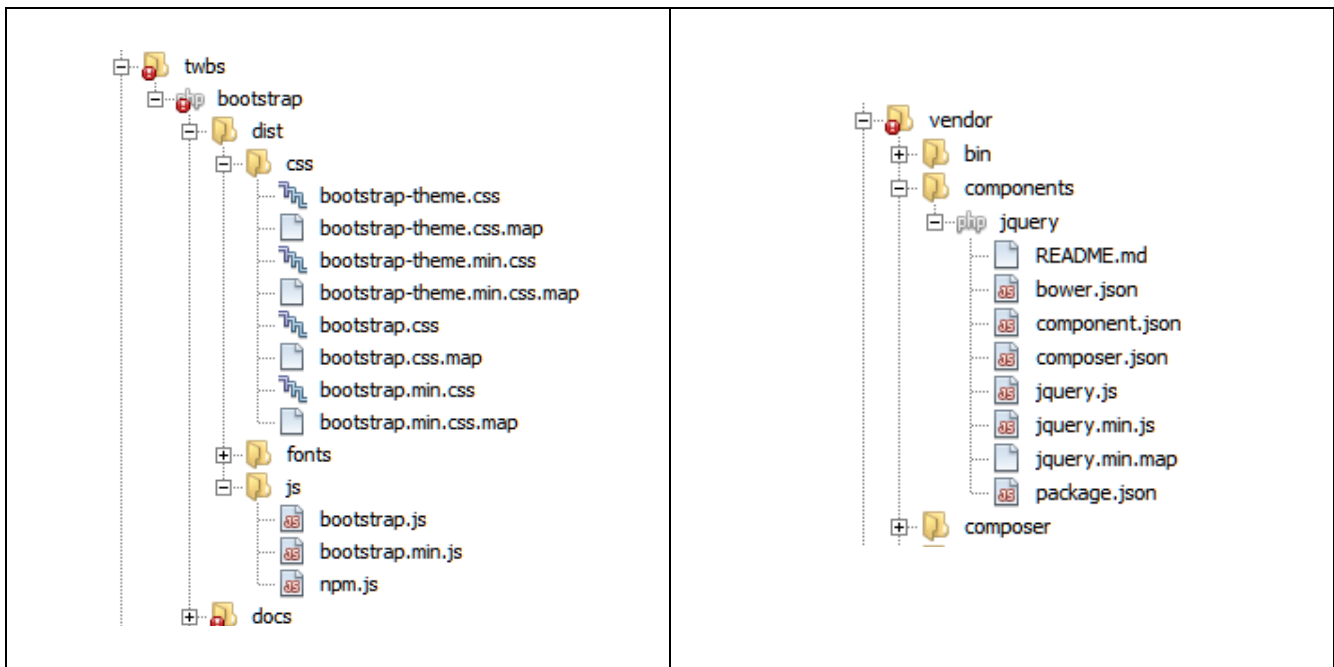
Dans la console composer, choisir : twbs/bootstrap



Pour jquery :



Dans le dossier vendor, on trouvera ces 2 dépendances ici :



Il ne nous reste plus qu'à configurer les dépendances dans le fichier app/config.yml :
On complète la définition de assetic, notamment pour indiquer les chemins des ressources :

```
assetic:
    debug: '%kernel.debug%'
    use_controller: '%kernel.debug%'
    bundles: ["AppBundle"]
    filters:
        cssrewrite: ~
    assets:
        jquery:
            inputs:
                - '%kernel.root_dir%/../vendor/components/jquery/jquery.min.js'
        bootstrap_css:
            inputs:
                - '%kernel.root_dir%/../vendor/twbs/bootstrap/dist/css/bootstrap.min.css'
        bootstrap_js:
            inputs:
                - '%kernel.root_dir%/../vendor/twbs/bootstrap/dist/js/bootstrap.min.js'
```

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

Il ne nous reste plus maintenant qu'à modifier notre template pour prendre en compte bootstrap et jquery.

Dans la mesure où nous utiliser ces ressources dans chaque page du projet, nous allons les intégrer dans le layout principal (app\Resources\views\base.html.twig)

Vous l'aurez remarqué, dans ce template on trouve un block stylesheets :

```
<title>{% block title %}welcome!{% endblock %}</title>
{% block stylesheets %} {% endblock %}
```

Il n'y a qu'à le compléter :

```
{% block stylesheets %}
    {% stylesheets
        '@bootstrap_css'
    %}
    <link rel="stylesheet" href="{{ asset_url }}" />
    {% endstylesheets %}
{% endblock %}
```

Le block twig {% stylesheets ... %} contient la ou les chemins vers les ressources

L'élément `<link rel="stylesheet" href="{{ asset_url }}" />` permettra de créer le fichier de ressources utilisé dans le navigateur.

'@bootstrap_css' Fait référence au paramètre du fichier config.yml qui fournit le chemin vers le fichier .css

Idem pour le javascript, on repère au bas du template principal le block :

```
{% block javascripts %}{% endblock %}
```

Et on le complète :

```
{% block javascripts %}
    {% javascripts
        '@jquery'
        '@bootstrap_js'
    %}
    <script src="{{ asset_url }}"></script>
    {% endjavascripts %}
{% endblock %}
```

Le principe de fonctionnement est le même que pour le css

Il ne nous reste plus qu'à tester notre formulaire :

Les Formulaires en symfony 3 et Bootstrap

Documentation Symfony templates »

Menu

- Index
- Exercice Paramètres
- Exercice Bootstrap

exercice formulaires

Gestion des employés

Numéro : 1

Votre nom : Dupont

Votre prénom : Jean

Votre date de naissance : 25 déc. 2011

Enregistrer

Fin du formulaire

Son sport préféré : Aucun intérêt

ses voitures préférées :

- Ferrari
- Aston Martin
- Porsche

Mentions légales

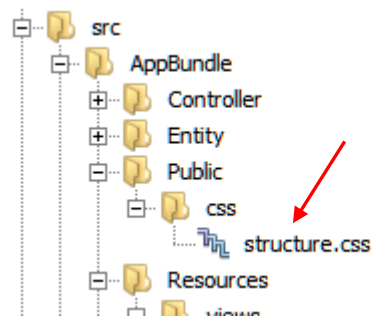
Nous sommes le samedi 17 décembre 2016



Inspectons l'élément

```
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title></title>
  <link rel="stylesheet" href="/TutoSymfonyBR_FormulairesV32/web/app_dev.php/css/fddb9b6_part_1.css">
</head>
<body>
  <div class="container"></div>
  <script src="/TutoSymfonyBR_FormulairesV32/web/app_dev.php/js/a56fa94_part_1.js"></script>
  <script src="/TutoSymfonyBR_FormulairesV32/web/app_dev.php/js/a56fa94_part_2.js"></script>
  <div id="sfwdt1cd8e5" class="sf-toolbar sf-display-none" data-sfurl="/TutoSymfonyBR_FormulairesV32/web
  <script nonce="8eb7fb508196fc0d748ca603fa2bdd92"></script>
  <script nonce="8eb7fb508196fc0d748ca603fa2bdd92"></script>
</body>
</html>
```

Nous pouvons aussi rajouter notre propre feuille de style. Nous allons router le fichier structure.css dans le bundle AppBundle.

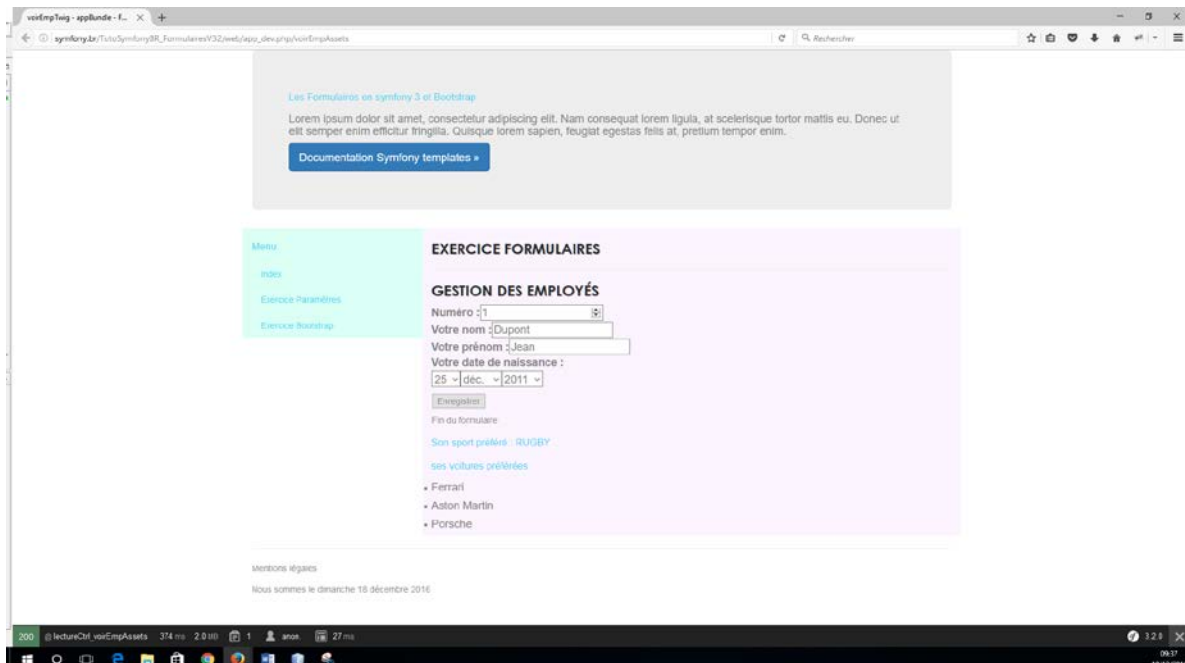


Et modifier la vue du bundle, c'est-à-dire le fichier
src/AppBundle/Resources/views/appBundleLayout.html.twig

```
{% block stylesheets %}
    {{ parent() }}
    {% stylesheets
        '@AppBundle/Public/css/structure.css'
    %}
    <link rel="stylesheet" href="{{ asset_url }}" />
    {% endstylesheets %}
{% endblock %}
```

On voit qu'il faut faire hériter le bloc parent, c'est-à-dire que l'on va conserver les feuilles de style du template parent et on va rajouter notre feuille de style structure.css

On teste :



Les couleurs ne sont pas top ... mais bon, ça marche !!!

7. Rajout d'une image réactive

C'est un exercice que vous allez faire

Vous allez rajouter une image, par exemple le smiley que je vous propose.

Si on clique sur l'image, la première page de ce TP va afficher, à savoir le template index/{nom}

La variable {nom} est la valeur du champ prénom au moment du click sur l'image.

Avec ce formulaire dont on vient de changer la valeur du champ prénom



Menu

- Index
- Exercice Paramètres
- Exercice Bootstrap

EXERCICE FORMULAIRES

GESTION DES EMPLOYÉS

Numéro : 1

Votre nom : Dupont

Votre prénom : marie-hélène

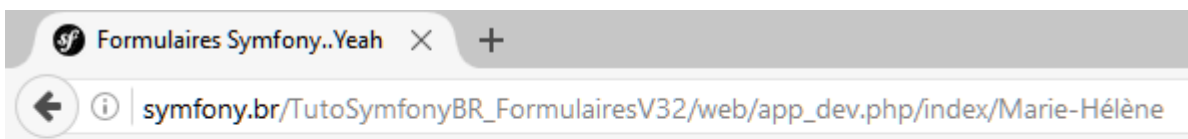
Votre date de naissance : 25 déc. 2011

Enregistrer

Fin du formulaire

Son sport préféré : RUGBY

Résultat à obtenir en cliquant sur l'image :

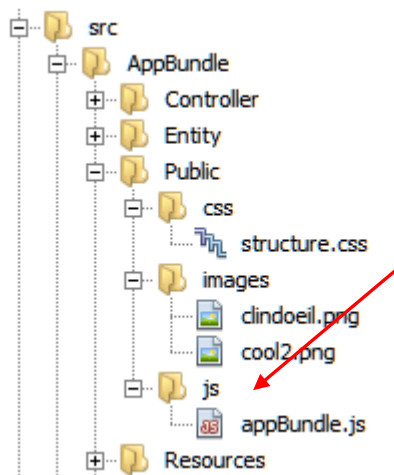


Bonjour Marie-Hélène

Solution :

On va se servir de javascript et jQuery ...

On va créer une nouvelle ressource, à savoir un fichier javascript app



Et dans ce fichier on va créer :

- ✓ Une fonction qui s'exécute au chargement de la page et qui récupère et place la racine de l'url du lien de l'image dans une variable globale au script. Ainsi, on n'aura plus qu'à concaténer la valeur récupérée dans le champ prénom à cette variable pour constituer le nouveau lien.
- ✓ Une fonction qui convertit la chaîne saisie dans le champ form_prenom en camel case pour satisfaire au pattern de l'URL
- ✓ Un événement qui va modifier le lien de l'image à chaque changement de valeur dans le champ form_prenom

```
// fonction qui va s'exécuter au chargement de la page après que le DOM soit chargé
$(function() {
    // on définit la variable qui contient la racine de l'URL à appeler
    // on n'aura plus qu'à rajouter le prénom récupéré au moment du click sur l'image
    // j'ai volontairement décomposé les instructions de la fonction
    var lien = $("#unicode").attr('href');
    var indice= lien.lastIndexOf("/") + 1 ;
    newLien = lien.substring(0,indice);
});
/**
 * Fonction qui va convertir une chaîne en camelcase en respectant les noms composés séparés ;
 */
var camel=function(str) {
    return str.replace(/(?:^|\s|-)\w/g, function(match) {
        return match.toUpperCase();
    });
};

/**
 * Sur événement change de la zone de texte prenom, on modifie le lien de l'image
 */
$("#form_prenom").change(function() {
    $("#unicode").attr('href', newLien + camel($("#form_prenom").val()));
});
```

Enfin, il faut que charger ce script.... On va considérer qu'il est propre à la vue voirEmpAssets.html.twig, on va donc modifier ce fichier et surcharger le bloc twig {% block javascripts %} :

```
<li><h4>{{ uneAuto }}</h4></li>
    {% endfor %}
{% endblock %}
{% block javascripts %}
    {{ parent() }}
    {% javascripts
        '@AppBundle/public/js/appBundle.js'
    %}
<script src="{{ asset_url }}"></script>
{% endjavascripts %}
```

8. Pour aller plus loin dans les formulaires

Je vous conseille pour affiner les formulaires de voir les Form Themes (https://symfony.com/doc/current/form/form_customization.html)



Bon courage !!!