



## Création d'un Bundle

Sources :

<https://symfony.com/doc/current/bundles.html>

### 1. Préparation du travail

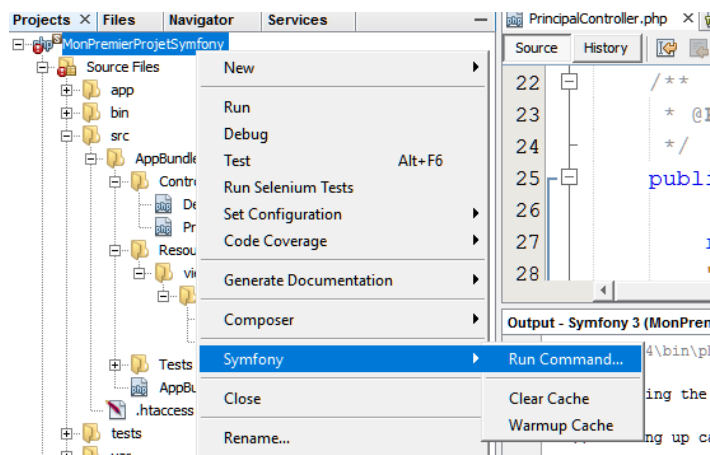
- ✓ On va compléter le projet MonPremierProjetSymfony
- ✓ Répertoire racine : **Bdls** (il s'agit du nom du namespace)
- ✓ Nom du Bundle : on le crée pour ce tutoriel Tutoriel donc on l'appellera **TutorielBundle**

Bundle étant le suffixe obligatoire

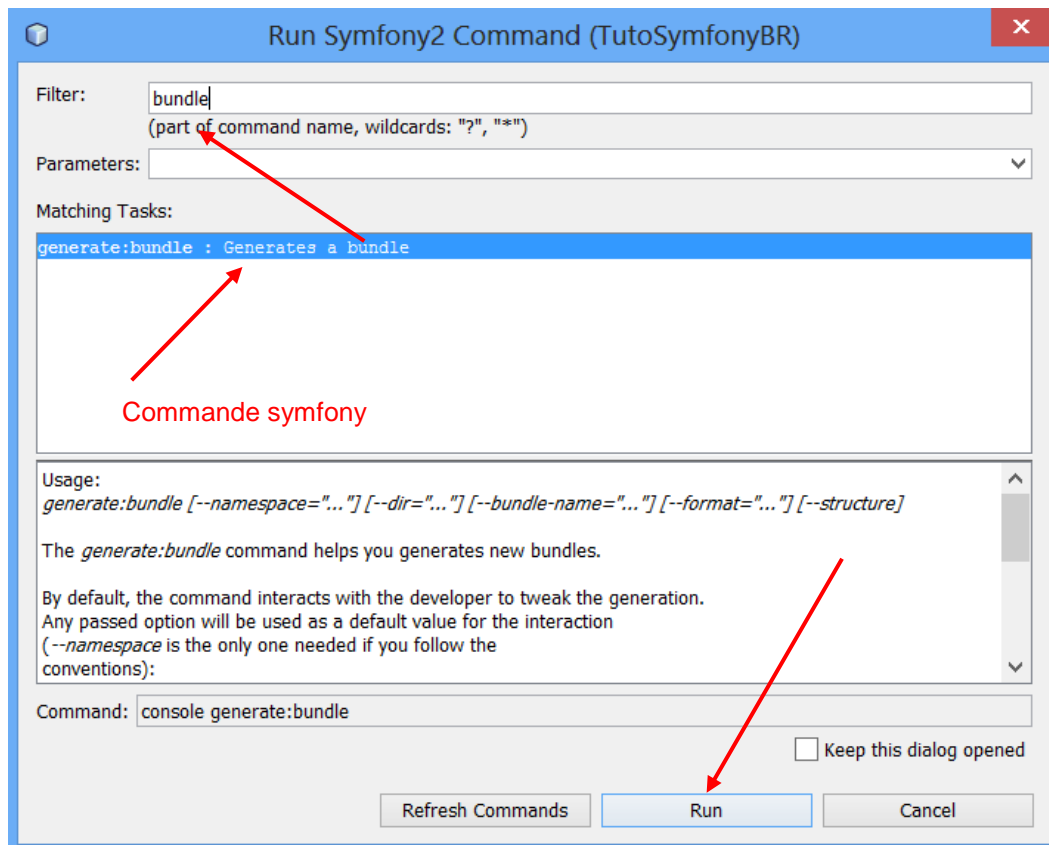
- ✓ Le format du fichier de configuration sera yml
- ✓ On va utiliser la console symfony3 intégrée à Netbeans

### 2. Création du Bundle

Appel de la console :



Dans la partie Filter, on peut rentrer un mot clé, la console affichera dans la zone Matching Tasks les commandes se rapportant au mot clé Bundle



Dans la fenêtre du bas on va répondre aux questions.

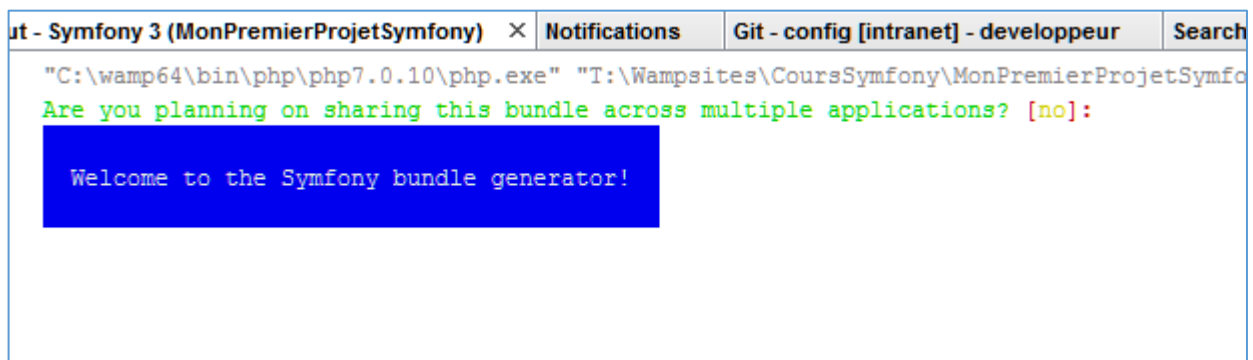
Dans certains cas, il proposera des réponses par défaut, il suffira de taper sur enter pour valider le choix par défaut.

Dans la fenêtre Output de Netbeans apparait la console de saisie des paramètres du Bundle que l'on va créer.

Premier paramètre : il nous demande si le bundle sera commun à plusieurs applications.

Valeur par défaut : no

Laisser comme ça et valider




Deuxième paramètre : le nom du namespace qui va déterminer le nom du Bundle



Surtout ne pas créer les dossier à l'avance dans l'explorateur de fichiers, sans quoi l'application ne fonctionnera pas en mode production.

```
Bundle name:
Your application code must be written in bundles. This command helps
you generate them easily.


Give your bundle a descriptive name, like BlogBundle.
Bdls/TutorielBundle|
```



On valide par enter :

Le nom du Bundle est suggéré, il faut valider pour respecter les règles de nommage.

```
Give your bundle a descriptive name, like BlogBundle.
Bdls/TutorielBundle
Bundle name [BdlsTutorielBundle]:
In your code, a bundle is often referenced by its name. It can be the
concatenation of all namespace parts but it's really up to you to come
up with a unique name (a good practice is to start with the vendor name
Based on the namespace, we suggest BdlsTutorielBundle.
```



Il nous est demandé de fournir le dossier cible (target directory)

Par convention, comme on l'a vu, on place nos bundles dans le répertoire src/. C'est ce que Symfony nous propose. On valide par enter :

```
Target Directory [src/]:
Bundles are usually generated into the src/ directory. Unless you're
doing something custom, hit enter to keep this default!
```

|

---

Vient ensuite un choix important, celui du format du fichier de configuration. Plusieurs formats sont possibles :

- ✓ Le **XML** est un langage de balises génériques, c'est-à-dire qu'il t'est possible de créer des entrées de ton choix, en respectant un schéma que tu auras mis en place. C'est un langage qui est très utilisé pour les web services (à titre d'exemple), car le rendu peut-être interprété par toutes les plateformes/OS. C'est malheureusement très verbeux, contrairement aux autres propositions que tu as citées 😊! (encore que la configuration en PHP ...)
- ✓ Le **YML** est un langage bien lisible par l'homme. Les entrées (ou lignes de configuration dans le cadre de Symfony) se font par indentation. On peut donc configurer des propriétés d'objets très facilement, en quelques lignes de code !
- ✓ Les **annotations** ont la chance de pouvoir, comme le code PHP, d'être centralisées directement dans les fichiers PHP (Au dessus des controllers, des actions ou des attributs d'une classe... ). Doctrine2 utilise ce principe-là, et il faut avouer que c'est très flexible et pratique !

- ✓ le **PHP**, pas besoin de s'étendre, la réponse est limpide, mais bon: Symfony2 est développé sur une base de PHP, le tout est donc parsé en PHP, il est donc possible de tout configurer en PHP ... logique 😊

Au final:

- ✓ **XML**: Verbeux, très flexible, plus complexe et moins lisible que le YML ou les annotations
- ✓ **YML**: Très lisible, configurable (Attention aux espaces ! 😊)
- ✓ **Annotations**: Centralisées, modifiable en quelques secondes lorsque l'on traite un certain fichier (par exemple une Entity, la gestion de pre/post update ... pas besoin d'ouvrir un fichier externe pour modifier tout ceci !)  
Seul point: faire TRES attention aux quotes et double quotes ! 😊
- ✓ **PHP**: Ça reste du PHP, c'est donc peut-être un peu plus long de configurer (du fait que l'on doit accéder à certaines classes, et objets), mais c'est donc flexible, et performant ! 😊

Côté performances, il faut de toute façon parser le code, qu'il soit dans un fichier externe (xml, yml) ou depuis le PHP.

Il faudrait faire des benchmarks afin de voir quelle méthode est la plus performante, mais en fin de compte, n'est-ce pas plutôt une question de préférences et de lisibilité? 😊

On préférera ici le format annotation qui est maintenant recommandé par Symfony (depuis Symfony 2.4). et on valide par enter

```
Configuration format (annotation, yml, xml, php) [annotation]:  
What format do you want to use for your generated configuration?
```

Enfin, la console nous indique que le bundle est créé :

```
Configuration format (annotation, yml, xml, php) [annotation]:  
What format do you want to use for your generated configuration?
```

Bundle generation

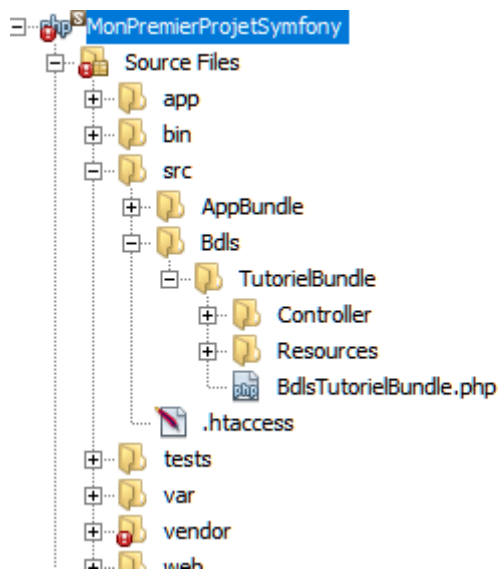
```
> Generating a sample bundle skeleton into T:\Wampsites\CoursSymfony\MonPremierProjetSymfony\app\src/Bdls/TutorielBundle  
created .\app\src/Bdls/TutorielBundle/  
created .\app\src/Bdls/TutorielBundle/BdlsTutorielBundle.php  
created .\app\src/Bdls/TutorielBundle/Controller/  
created .\app\src/Bdls/TutorielBundle/Controller/DefaultController.php  
created .\app\tests/BdlsTutorielBundle/Controller/  
created .\app\tests/BdlsTutorielBundle/Controller/DefaultControllerTest.php  
created .\app\src/Bdls/TutorielBundle/Resources/views/Default/  
created .\app\src/Bdls/TutorielBundle/Resources/views/Default/index.html.twig  
created .\app\src/Bdls/TutorielBundle/Resources/config/  
created .\app\src/Bdls/TutorielBundle/Resources/config/services.yml  
> Checking that the bundle is autoloading  
> Enabling the bundle inside T:\Wampsites\CoursSymfony\MonPremierProjetSymfony\app\AppKernel.php  
updated .\app\AppKernel.php  
> Importing the bundle's routes from the T:\Wampsites\CoursSymfony\MonPremierProjetSymfony\app\config\routing.yml file  
updated .\app\config\routing.yml  
> Importing the bundle's services.yml from the T:\Wampsites\CoursSymfony\MonPremierProjetSymfony\app\config\config.yml file  
updated .\app\config\config.yml
```

Everything is OK! Now get to work :).

Done.

On voit l'ensemble des fichiers créés....il y a en a un certain nombre !!!

On va vérifier l'arborescence créée sous le dossier src :



C'est bon, la racine de nos bundles ou encore l'espace de nom Bdl's a été créé et dessous, on voit le dossier conteneur du bundle TutorielBundle.

### 3. Débogage

---

En mode développement, symfony propose une toolbar en bas de page.

Quand on a saisi l'url

[http://symfony.br/BRMonPremierProjetSymfony/web/app\\_dev.php/hello/les\\_symfoniens](http://symfony.br/BRMonPremierProjetSymfony/web/app_dev.php/hello/les_symfoniens), la toolbar

La toolbar apparaît ! Parcourez là et repérez les informations qu'elle fournit.



Testez avec l'url :

[http://localhost/TutoSymfonyBR/web/app.php/hello/les\\_Symphoniens](http://localhost/TutoSymfonyBR/web/app.php/hello/les_Symphoniens)

Elle n'apparaît pas !



#### 4. Pour en arriver là



On va résumer ce qu'il s'est passé lors de la création du bundle afin de mieux comprendre.

Symfony a généré le bundle en plusieurs phases :

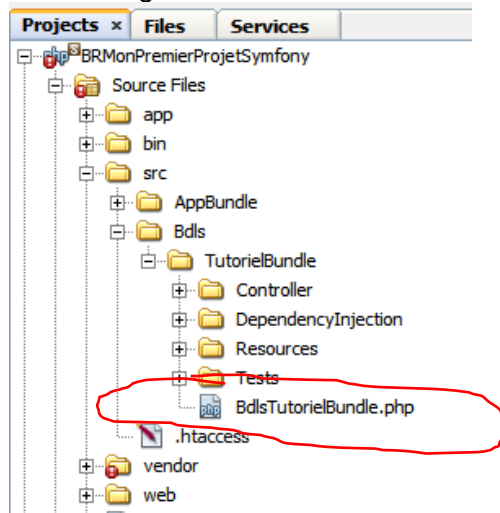
- ✓ Création de la structure du bundle
- ✓ Référencement du bundle auprès du kernel (noyau)
- ✓ Enregistrement des routes dans le routeur

Voyons ça en détail :

- ✓ Création de la structure du bundle

On peut voir dans l'arborescence du projet les dossiers et fichiers créés dans le dossier  
T:\Wampsites\Symfony\BRMonPremierProjetSymfony\src\Bdls\TutorielBundle

Le seul fichier obligatoire est le fichier contenant la classe du bundle : BdlsTutorielBundle.php



On remarque :

- ✓ Qu'il dérive de la classe Bundle
- ✓ Qu'il est vide. On n'interviendra pas sur ce fichier, on laisse symfony le gérer

```
1 <?php
2
3 namespace Bdls\TutorielBundle;
4
5 use Symfony\Component\HttpKernel\Bundle\Bundle;
6
7 class BdlsTutorielBundle extends Bundle
8 {
9 }
```

- ✓ Référencement du bundle auprès du kernel (noyau)

Rappelons-nous la démarche pour supprimer le bundle par défaut Acme.... On est allé supprimer une ligne dans le fichier app/AppKernel.php.

Ici on va rajouter l'instruction qui permettra de charger le bundle BdlstutorielBundle.

Voyons ce fichier :

```
<?php

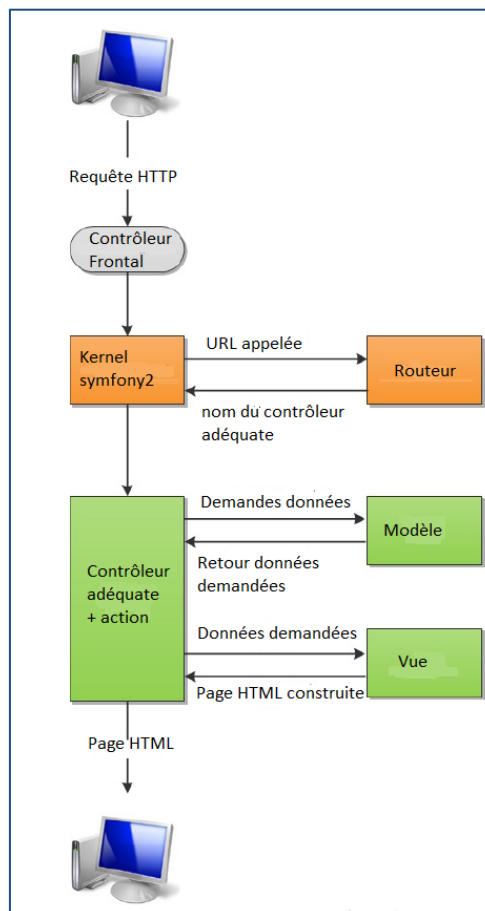
use Symfony\Component\HttpKernel\Kernel;
use Symfony\Component\Config\Loader\LoaderInterface;

class AppKernel extends Kernel
{
    public function registerBundles()
    {
        $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 BdlstutorielBundle\BdlstutorielBundle(),
        ];
    }
}
```

Notre bundle a bien été rajouté, il est donc chargé

✓ Enregistrement des routes dans le routeur

Rappelons-nous ce schéma, le routeur est l'élément qui doit déterminer quel controler (in english) appeler.



En fait, il y a un routeur au niveau application (app/routing.yml) qui va fournir, pour chaque bundle, le type de routing (annotation, yml,...)

Le routeur de l'application :

```
bdls_tutoriel:  
    resource: "@BdlsTutorielBundle/Controller/"  
    type:      annotation  
    prefix:    /
```

On voit que pour le bundle BdlsTutorielBundle, le routage se fait par annotation et qu'il faut scanner chaque contrôleur dans le dossier Controller

Voyons ce fichier :

Tout ceci a été généré automatiquement par symfony.

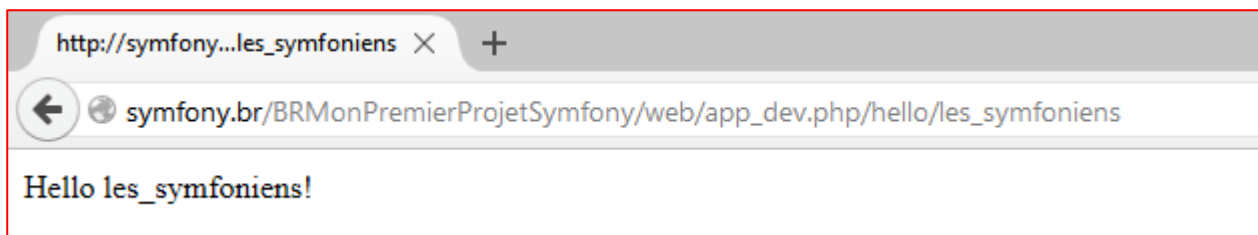
## 1. Test du bundle

---

On saisit l'URL suivante :

[http://symfony.br/BRMonPremierProjetSymfony/web/app\\_dev.php/hello/les\\_symfoniens](http://symfony.br/BRMonPremierProjetSymfony/web/app_dev.php/hello/les_symfoniens)

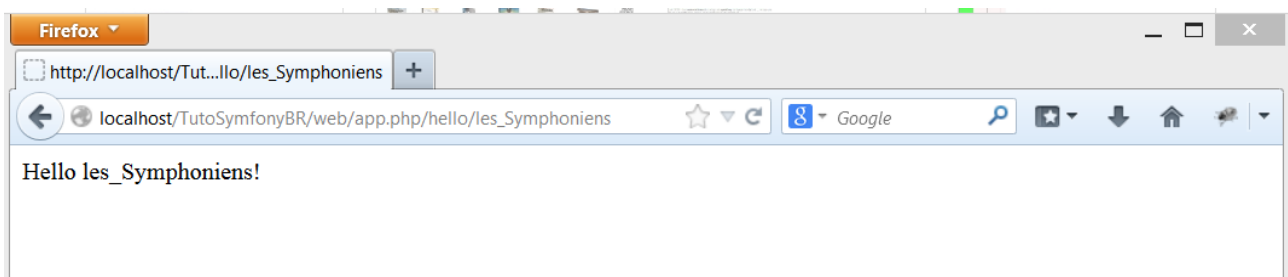
Afficher la page suivante en ne travaillant QUE dans le bundle BdlsTutorielBundle



On va essayer maintenant cette URL :

[http://symfony.br/BRMonPremierProjetSymfony/web/app.php/hello/les\\_symfoniens](http://symfony.br/BRMonPremierProjetSymfony/web/app.php/hello/les_symfoniens)

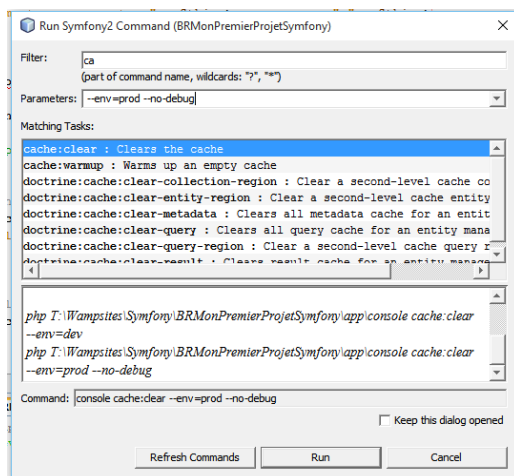
Quelle est la différence avec l'URL précédente ?



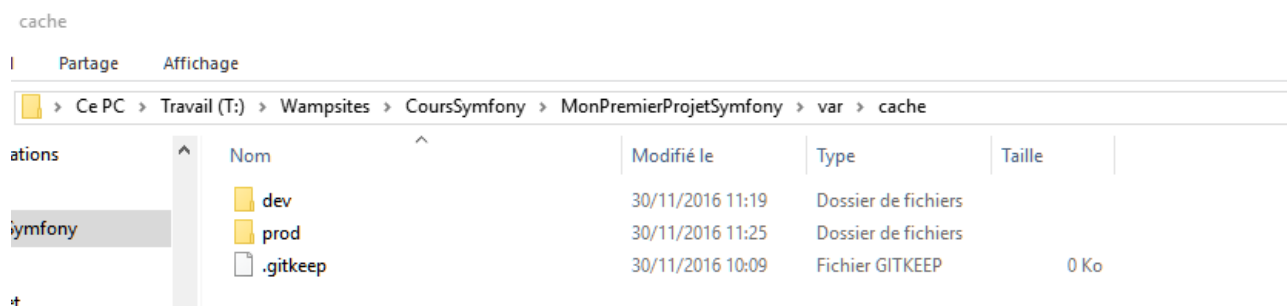




Si la page n'apparaît pas, il faut vider le cache de prod par les commandes symfony.



Si il y a une erreur lors du vidage du cache, il faut vider le cache manuellement en vidant le dossier ....\BRMonPremierProjetSymfony\var\cache



Ce petit exemple fournit un petit aperçu du fonctionnement de symfony.



On verra en détail plus tard les notions

- ✓ de route,
- ✓ de contrôleur
- ✓ d'action