

# Les Traits dans Symfony

Ou comment faire de l'héritage horizontal

## Les Traits dans Symfony : oui mais pourquoi ?

- Qu'est ce qu'un trait en php et pourquoi des traits ?
  - Php ne permet pas de faire l'héritage multiple
  - Par contre il est parfois pratique de rajouter des fonctionnalités communes dans différentes classes (qui elles mêmes héritent vraisemblablement d'une autre classe)
  - Ces fonctionnalités peuvent être partagées dans des classes qui ne font pas partie de la même arborescence d'héritage
  - C'est vieux comme le monde (depuis php 5.4)

# Les Traits dans Symfony

- La solution

## LES TRAITS

# Les Traits dans Symfony

- Dans Symfony il y en a quelques uns
  - Grep -R “Trait” vendor

```
Security/Http/Firewall/ExceptionListener.php:use Symfony\Component\Security\Http\Util\TargetPathTrait;  
Security/Http/Firewall/ExceptionListener.php:    use TargetPathTrait;  
Security/Http/Authentication/DefaultAuthenticationSuccessHandler.php:use Symfony\Component\Security\Http\Util\TargetPathTrait;  
Security/Http/Authentication/DefaultAuthenticationSuccessHandler.php:    use TargetPathTrait;  
Security/Http/Util/TargetPathTrait.php: * Trait to get (and set) the URL the user last visited before being forced to authenticate.  
Security/Http/Util/TargetPathTrait.php:trait TargetPathTrait  
Security/Http/Tests/Util/TargetPathTraitTest.php:use Symfony\Component\Security\Http\Util\TargetPathTrait;  
Security/Http/Tests/Util/TargetPathTraitTest.php:class TargetPathTraitTest extends \PHPUnit_Framework_TestCase  
Security/Http/Tests/Util/TargetPathTraitTest.php:    $obj = new TestClassWithTargetPathTrait();  
Security/Http/Tests/Util/TargetPathTraitTest.php:    $obj = new TestClassWithTargetPathTrait();  
Security/Http/Tests/Util/TargetPathTraitTest.php:    $obj = new TestClassWithTargetPathTrait();  
Security/Http/Tests/Util/TargetPathTraitTest.php:class TestClassWithTargetPathTrait  
Security/Http/Tests/Util/TargetPathTraitTest.php:    use TargetPathTrait;
```

# Les Traits dans Symfony

- TargetPathTrait

```
namespace Symfony\Component\Security\Http\Util;

use Symfony\Component\HttpFoundation\Session\SessionInterface;

/**
 * Trait to get (and set) the URL the user last visited before being forced to authenticate.
 */
trait TargetPathTrait

    /**
     * Sets the target path the user should be redirected to after authentication.
     *
     * Usually, you do not need to set this directly.
     *
     * @param SessionInterface $session
     * @param string           $providerKey The name of your firewall
     * @param string           $uri         The URI to set as the target path
     */
    private function saveTargetPath(SessionInterface $session, $providerKey, $uri)
    {
        $session->set('_security.'.$providerKey.'.target_path', $uri);
    }

    /**
     * Returns the URL (if any) the user visited that forced them to login.
     *
     * @param SessionInterface $session
     * @param string           $providerKey The name of your firewall
     *
     * @return string
     */
    private function getTargetPath(SessionInterface $session, $providerKey)
    {
        return $session->get('_security.'.$providerKey.'.target_path');
    }

    /**
```

# Les Traits dans Symfony

- Dans Symfony il y en a quelques uns
  - Grep -R "Trait" vendor

```
Symfony/Bundle/FrameworkBundle/Kernel/MicroKernelTrait.php:trait MicroKernelTrait
Symfony/Bundle/FrameworkBundle/Controller/Controller.php:use Symfony\Component\DependencyInjection\ContainerAwareTrait;
Symfony/Bundle/FrameworkBundle/Controller/Controller.php:    use ContainerAwareTrait;
Symfony/Bundle/FrameworkBundle/Controller/TemplateController.php:use Symfony\Component\DependencyInjection\ContainerAwareTrait;
Symfony/Bundle/FrameworkBundle/Controller/TemplateController.php:    use ContainerAwareTrait;
Symfony/Bundle/FrameworkBundle/Controller/RedirectController.php:use Symfony\Component\DependencyInjection\ContainerAwareTrait;
Symfony/Bundle/FrameworkBundle/Controller/RedirectController.php:    use ContainerAwareTrait;
Symfony/Bundle/FrameworkBundle/Tests/Kernel/MicroKernelTraitTest.php:class MicroKernelTraitTest extends \PHPUnit_Framework_TestCase
Symfony/Bundle/FrameworkBundle/Tests/Kernel/ConcreteMicroKernel.php:use Symfony\Bundle\FrameworkBundle\Kernel\MicroKernelTrait;
Symfony/Bundle/FrameworkBundle/Tests/Kernel/ConcreteMicroKernel.php:    use MicroKernelTrait;
```

# Les Traits dans Symfony

- ContainerAwareTrait

```
/*
 * This file is part of the Symfony package.
 *
 * (c) Fabien Potencier <fabien@symfony.com>
 *
 * For the full copyright and license information, please view the LICENSE
 * file that was distributed with this source code.
 */

namespace Symfony\Component\DependencyInjection;

/**
 * ContainerAware trait.
 *
 * @author Fabien Potencier <fabien@symfony.com>
 */
trait ContainerAwareTrait
{
    /**
     * @var ContainerInterface
     */
    protected $container;

    /**
     * Sets the container.
     *
     * @param ContainerInterface|null $container A ContainerInterface instance or null
     */
    public function setContainer(ContainerInterface $container = null)
    {
        $this->container = $container;
    }
}
```

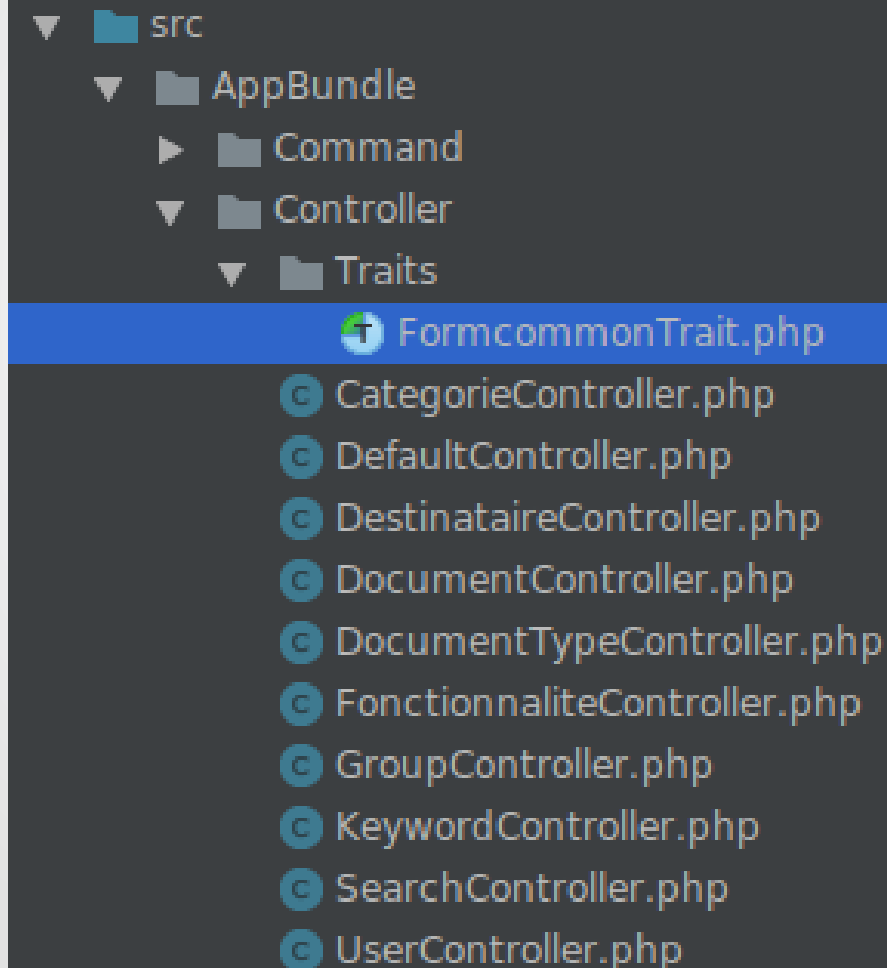
# Les Traits dans Symfony

- Quelques uns dans Doctrine
- Dans d'autres frameworks (comme Cakephp par exemple)
- Bref.... Il y a en a partout.... Peut être parce que c'est pratique.



# Les Traits dans Symfony

- Dans le projet Symfony



```
▼ src
  ▼ AppBundle
    ► Command
    ▼ Controller
      ▼ Traits
        FormcommonTrait.php
        CategorieController.php
        DefaultController.php
        DestinataireController.php
        DocumentController.php
        DocumentTypeController.php
        FonctionnaliteController.php
        GroupController.php
        KeywordController.php
        SearchController.php
        UserController.php
```

# Les Traits dans Symfony

- Le fichier “FormcommonTrait”

```
namespace AppBundle\Controller\Traits;

trait FormcommonTrait
{
    public function formGetErrors($form){
        if($form->isSubmitted() && !$form->isValid()) {
            $message=$this->getFormErrors($form);

            foreach($form->getErrors() as $oneError){
                $message.=$oneError->getMessage();
            }

            $this->addFlash('error', 'Les données indiquées ne sont pas valides '.$message);
        }
    }
}
```

# Les Traits dans Symfony

- Dans un contrôleur

```
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;
use AppBundle\Controller\Traits\FormcommonTrait;

/**
 * @Route("/documenttypes")
 * @Security("has_role('ROLE_GESTIONNAIRE')")
 */

class DocumentTypeController extends Controller
{
    use FormcommonTrait;
    /**
     * @Route("/", name="documenttypes-index")
     */
    public function indexAction(Request $request)
    {
    }
```

# Les Traits dans Symfony

- Et Hop.... La méthode définie dans le trait s'applique à tous les contrôleurs qui l'ont implémenté.
- Pourquoi utiliser ceci ?
  - Parce que je suis fainéant.... Copier la même méthode 7 ou 8 fois me fatigue
  - Parce que je peux faire évoluer la fonctionnalité sans toucher à tous les contrôleurs
  - Et si je veux rajouter des méthodes ou propriétés qui devront être partagées concernant la gestion des formulaires il me suffira de modifier le fichier "FormCommonTrait"

# Les Traits dans Symfony

- Un peu de lecture
  - <http://php.net/manual/fr/language.oop5.traits.php>
  - <https://openclassrooms.com/courses/programmez-en-orientee-objet-en-php/les-traits-2>
  - [http://symfony.com/doc/current/configuration/micro\\_kernels\\_trait.html](http://symfony.com/doc/current/configuration/micro_kernels_trait.html)
  - <http://www.php-fig.org/bylaws/psr-naming-conventions/>
  - <https://www.sitepoint.com/php-traits-good-or-bad/>
  - <http://blog.ircmaxell.com/2011/07/are-traits-new-eval.html>
  -