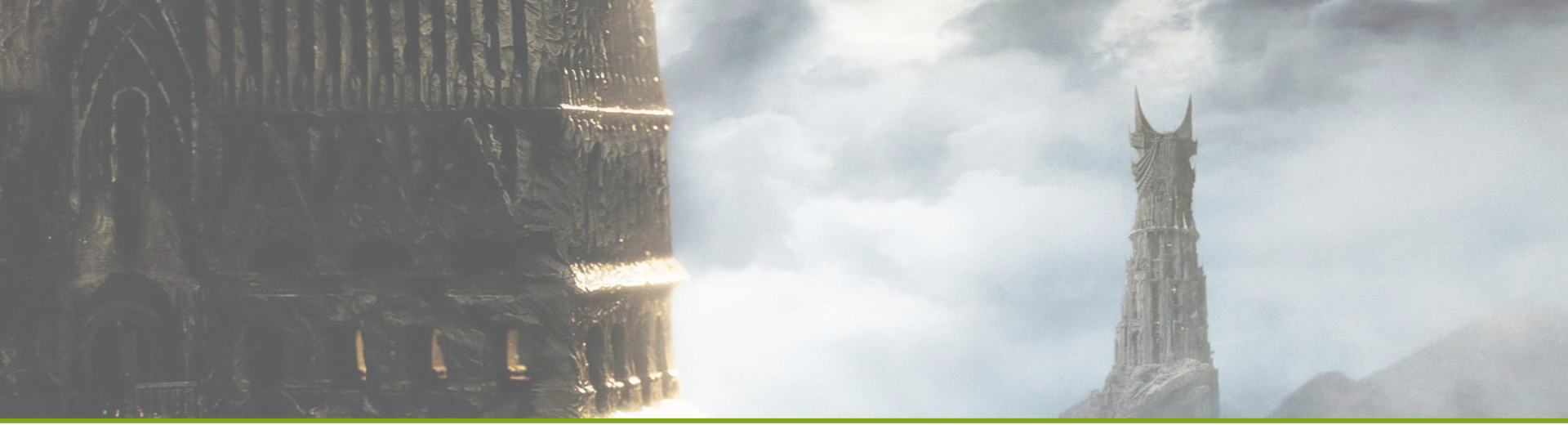
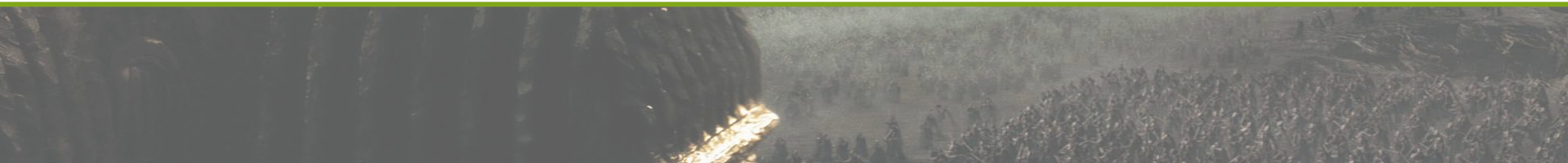


Formation Symfony

Version 3.4 & 4.0

IPSSI - février 2018



Présentation

Geoffrey CHAMEROY

Lead Developer chez Darkmira

gchameroy@darkmira.fr

Auto Entrepreneur

Nous aborderons

- Les bases de la POO
- L’architecture Symfony
- Création d’un nouveau projet
- Les contrôleurs
- Les routes
- Les vues
- Les services
- Les formulaires
- Les rôles et sessions

La notation

QCM de 40 questions (⅓ note)

- Les bases de la POO
- Les éléments Symfony

Projet (⅔ note)

- Réalisation d'un projet sous Symfony 3.4

darkmira

Les bases de la POO

- Les classes
- Les propriétés
- Les méthodes
- Les constantes de classe
- Le constructeur
- Les interfaces
- Les traits
- Les classes abstraites

Les classes

```php
<?php

class User
{
    /**
     * @return string
     */
    function getName(): string
    {
        return 'Geoffrey';
    }
}

$user = new User();

```

Les propriétés

```php
<?php

class User
{
    /** @var string */
    private $name = 'Geoffrey';

    /** @var int */
    public $yolo = 1;

    /**
     * @return string
     */
    function getName(): string
    {
        return $this->name;
    }
}

$user = new User();
var_dump($user->getName());
var_dump($user->yolo);
```

Les méthodes

```php
<?php

class User
{
    /** @var string */
    private $name = 'Geoffrey';

    /**
     * @return string
     */
    public function getName(): string
    {
        return $this->getPrivateName();
    }

    /**
     * @return string
     */
    private function getPrivateName(): string
    {
        return $this->name;
    }
}

```

Les constantes de classe

```php
<?php

class User
{
    const NAME = 'Geoffrey';

    /**
     * @return string
     */
    public function getName(): string
    {
        return self::NAME;
    }
}
```

Le constructeur

```php
<?php

class User
{
    /** @var string */
    private $name;

    public function __construct(string $name)
    {
        $this->name = $name;
    }

    /**
     * @return string
     */
    public function getName(): string
    {
        return $this->name;
    }
}

$user = new User( name: 'Geoffrey');
```

Les interfaces

```php
<?php

class User implements UserInterface
{
    /**
     * @inheritdoc
     */
    public function getName(): string
    {
        return 'Geoffrey';
    }
}

interface UserInterface
{
    /**
     * @return string
     */
    public function getName(): string;
}
```

Les traits

```php
<?php

class User
{
    use SoftDeletable;

    /**
     * @inheritdoc
     */
    public function getName(): string
    {
        return 'Geoffrey';
    }
}
```

```php
trait SoftDeletable
{
    /** @var \DateTime */
    private $deletedAt;

    /**
     * @return bool
     */
    public function isDeleted(): bool
    {
        return $this->deletedAt ? true : false;
    }
}
```

Les classes abstraites

```php
<?php

class User extends UserAbstract
{

}

abstract class UserAbstract
{
    /** @var string */
    private $name = 'Geoffrey';

    /**
     * @return string
     */
    public function getName(): string
    {
        return $this->name;
    }
}

```

Revenons à Symfony !

- Présentation de Symfony
- Création d’un projet SF 3.4
- Architecture du projet
- Le modèle MVC

Présentation Symfony

- Framework PHP
- Programmation Orienté Objet
- Modèle MVC
- Créé par Fabien Potencier
- Communauté réactive
- Nouvelle version tous les 6 mois

Création d’un projet SF 3.4

Prérequis :

- PHP
- Apache
- MySQL
- Composer

Création d’un projet SF 3.4

Téléchargement de l’installeur

Windows

```
php -r "file_put_contents('symfony', file_get_contents('https://symfony.com/installer'));"
```

Mac & Linux

```
sudo mkdir -p /usr/local/bin
sudo curl -LsS https://symfony.com/installer -o /usr/local/bin/symfony
sudo chmod a+x /usr/local/bin/symfony
```

Création d'un projet SF 3.4

Windows

```
php -r "file_put_contents('symfony', file_get_contents('https://symfony.com/installer'));"
php symfony new project_name 3.4
```

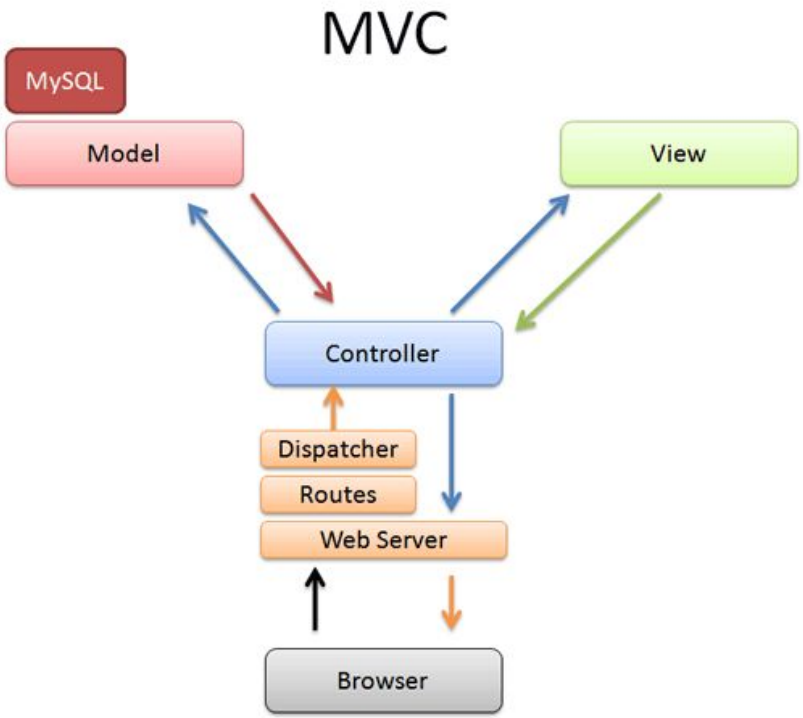
Mac & Linux

```
sudo mkdir -p /usr/local/bin
sudo curl -LsS https://symfony.com/installer -o /usr/local/bin/symfony
sudo chmod a+x /usr/local/bin/symfony
symfony new project_name 3.4
```

Architecture du projet

- app/ => configuration, vues et traductions
- bin/ => Fichiers exécutables
- src/ => Sources PHP
- tests/ => Tests automatiques
- var/ => Fichiers générés (log, cache, etc…)
- vendor/ => Dépendances tierces
- web/ => Répertoire public

Le modèle MVC



Mon 1er “Hello World”

- Le contrôleur
- Le routing
- La vue

Le contrôleur

```php
<?php
// src/AppBundle/DefaultController.php
namespace AppBundle\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\Controller;
use Symfony\Component\HttpFoundation\Response;

class DefaultController extends Controller
{
    // ...

    public function HelloWorldAction()
    {
        return new Response( content: 'Hello World');
    }
}
```

Le routing

```php
<?php
// src/AppBundle/DefaultController.php
namespace AppBundle\Controller;

use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;
use Symfony\Bundle\FrameworkBundle\Controller\Controller;
use Symfony\Component\HttpFoundation\Response;

class DefaultController extends Controller
{
    // ...

    /**
     * @Route("/hello-world", name="hello_world")
     */
    public function HelloWorldAction()
    {
        return new Response( content: 'Hello World');
    }
}
```

La vue

```php
<?php
// src/AppBundle/DefaultController.php
    /**
     * @Route("/hello-world", name="hello_world")
     */
    public function HelloWorldAction()
    {
        return $this->render( view: 'default/hello-world.html.twig');
    }
```

```
{# app\Resources\views\default\hello-world.html.twig #}
<h1>Hello World!</h1>
```

Ajoutons une variable !

- Ajout du paramètre dans la route

```
@Route("/hello/{name}", name="hello_name")
```

- Transfert de la variable dans la vue

```
return $this->render('default/hello-name.html.twig', [
     'name' => $name
]);
```

- Affichage de la variable dans la vue

```
{{ name }}
```