HÉTIC

LA GRANDE ÉCOLE DE L'INTERNET

Symfony

Yann Le Scouarnec

HÉTIC

Symfony

HÉTIC

Symfony? C'est quoi?

- Symfony est une organisation open source dépendant de SensioLabs qui développe des dizaines de composants logiciels utilisés par de nombreux projets (Framework Symfony, Laravel, Drupal, etc.)

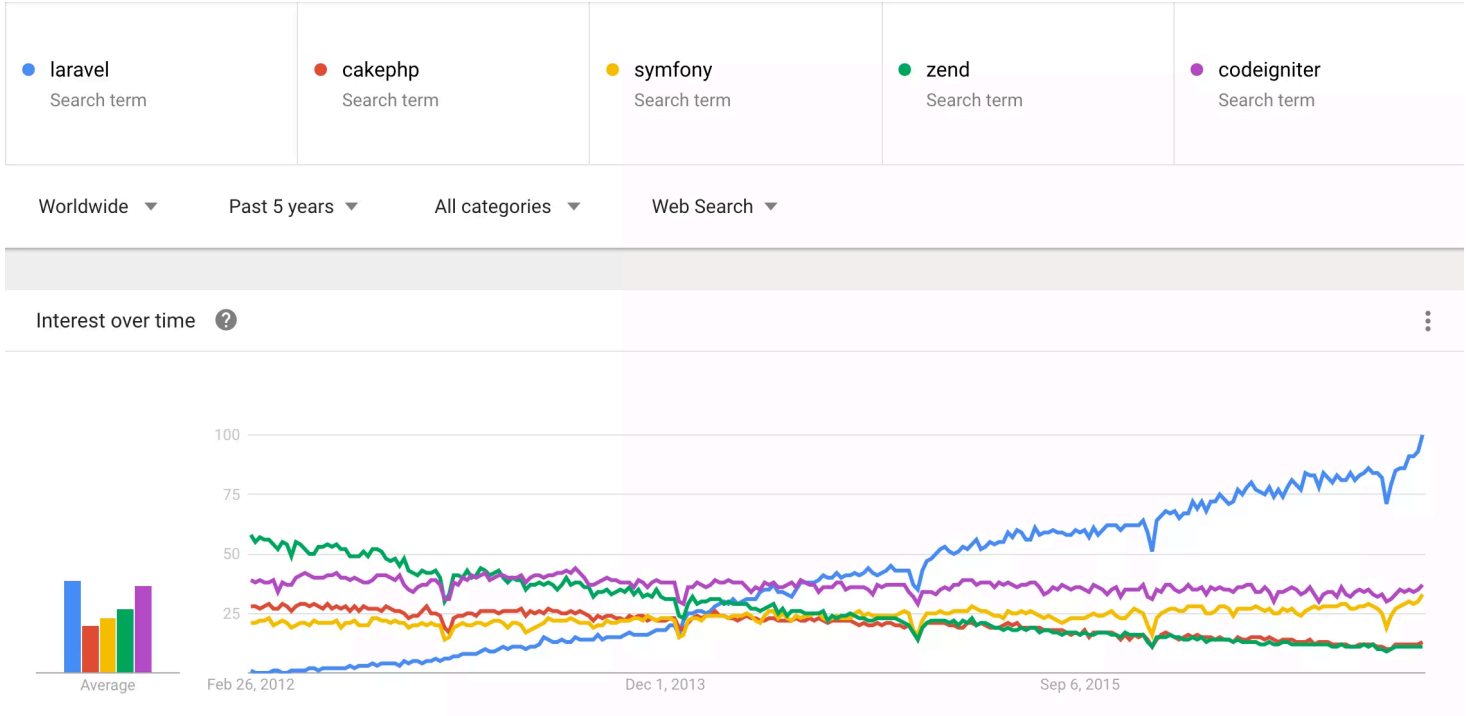
HÉTIC

Symfony… ?

- Symfony est aussi un des framework PHP les plus utilisés en France

HÉTIC

Parts de marché Symfony



HÉTIC

Références

- https://www.sitepoint.com/the-state-of-php-mvc-frameworks-in-2017/

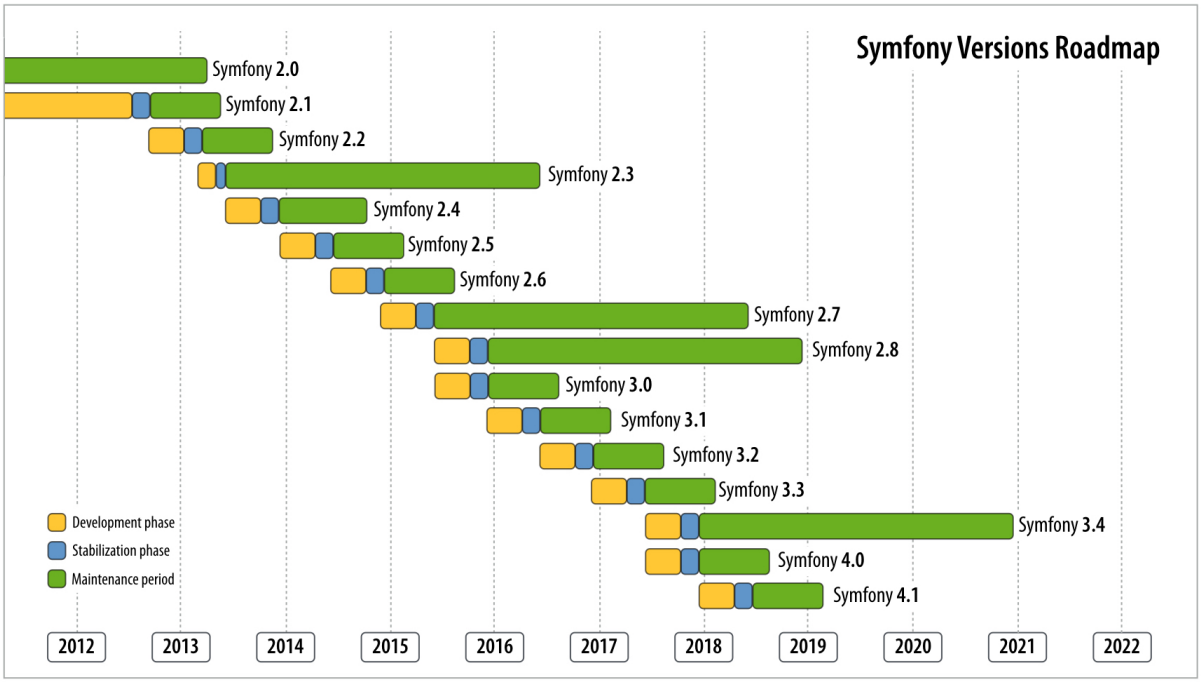
HÉTIC

Un framework c'est quoi?

- Un framework est une surcouche à un langage, développé dans ce même langage, pour permettre le développement de certaines solutions ciblées

HÉTIC

Cycle de vie Symfony



HÉTIC

Les versions

- À ce jour, les versions suivantes sont supportées par SensioLabs
  - Symfony 2.7 (milieu 2018)
  - Symfony 2.8 (fin 2018)
  - Symfony 3.4 (fin 2021)
  - Symfony 4.0 (milieu 2018)

HÉTIC

Cycle de vie

- https://symfony.com/doc/current/contributing/community/releases.html

HÉTIC

PRÉREQUIS

HÉTIC

Prérequis

- Git
- PHP 7.1
  Accessible dans le terminal
- MySQL 5.7
  Accessible dans le terminal
- PhpStorm

HÉTIC

Plugins PhpStorm

- Symfony plugin
- .ignore
- PHP annotations
- PHP composer.json support
- PHP Toolbox

HÉTIC

COMPOSER

HÉTIC

Composer

- Outils de gestion des dépendances
- Boite à outils de développement

HÉTIC

Dépendances

- Le concept de dépendance recouvre toutes les librairies sur lesquelles s'appuie votre projet (connexion de bases de données, envoi d'email, génération de PDF, etc.)

HÉTIC

Démonstration

HÉTIC

Exercice

- Installer composer
  https://getcomposer.org/download/

HÉTIC

composer init

- composer init initialise le projet composer et vous guide dans la création du fichier composer.json

HÉTIC

composer require

- Permet de rajouter une dépendance grâce à un assistant CLI qui met à jour le fichier composer.json, télécharge la dépendance et met à jour le fichier composer.lock et les fichiers d'autoload de composer

HÉTIC

Fichier composer.json

- Ce fichier doit impérativement être commité dans le repository Git de votre projet
- Le fichier composer.json décrit les dépendances de votre projet, l'auteur, la licence, les directives d'autoload et les scripts à exécuter

HÉTIC

composer update

- Cette commande va :
  - Chercher les versions des dépendances les plus pertinentes basées sur les masques de version et votre version PHP
  - Créer le fichier composer.lock

HÉTIC

composer update

- Télécharger les dépendances et leurs dépendances dans le répertoire vendor
- Mettre à jour les fichiers d'autoload

HÉTIC

Fichier composer.lock

- Ce fichier doit impérativement être commité dans le repository Git de votre projet
- Contient les détails permettant de dupliquer les versions sélectionnées par l'update
- Ce fichier ne doit absolument pas être modifié à la main

HÉTIC

composer install

- Cette commande a deux fonctionnements en fonction du contexte:
  - Installe les dépendances listées dans le fichier composer.lock
  - Ou lance composer update si le fichier composer.lock n'est pas présent

HÉTIC

composer dumpautoload

- Régénère tous les fichiers qui gèrent l'inclusion automatique dans le répertoire vendor

HÉTIC

Le répertoire vendor

- Ce répertoire et son contenu ne doivent en aucun cas être modifiés à la main. En effet, dès que composer exécute une mise à jour, il écrase et régénère la totalité du répertoire vendor

HÉTIC

Composer

- Composer est l'outil qui installe:
  - Symfony
  - Ses dépendances
  - Les bundles utilisés par votre projet
- Tout le contenu du répertoire vendor est exclu de votre repository Git

HÉTIC

Composer

- L'utilisation de la commande composer install permet de répliquer le contexte du projet et de garantir l'identicité des versions des dépendances entre tous les postes de travail de l'équipe de développement

HÉTIC

Exercice

- Créer un projet
- Initialiser le projet
- Ajouter la dépendance de dev symfony/var-dumper
- Créer un répertoire src/Human/
- Linker Mamal\\ à src/Human/
- Class Mamal\Human
- Créer un index.php à la racine du projet
- Instancier la class Mamal\Human et dumper l'instance

HÉTIC

SYMFONY FLEX

HÉTIC

Flex

- Installé avec la commande suivante dans le projet Symfony courant:
  composer require symfony/flex
- Plugin composer qui permet d'installer des groupes de dépendances cohérents basés sur des recipes définies par l'équipe Symfony ou la communauté

HÉTIC

Installer Flex

```
~/sites/sf4(branch:master*) » composer require symfony/flex                    yann@PHPBunker
Using version ^1.0 for symfony/flex
./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)
Package operations: 21 installs, 0 updates, 0 removals
  - Installing symfony/flex (v1.0.61): Loading from cache
  - Installing symfony/polyfill-mbstring (v1.6.0): Loading from cache
  - Installing symfony/console (v4.0.3): Loading from cache
  - Installing symfony/routing (v4.0.3): Loading from cache
  - Installing symfony/http-foundation (v4.0.3): Loading from cache
  - Installing symfony/event-dispatcher (v4.0.3): Loading from cache
  - Installing psr/log (1.0.2): Loading from cache
  - Installing symfony/debug (v4.0.3): Loading from cache
  - Installing symfony/http-kernel (v4.0.3): Loading from cache
  - Installing symfony/finder (v4.0.3): Loading from cache
  - Installing symfony/filesystem (v4.0.3): Loading from cache
  - Installing psr/container (1.0.0): Loading from cache
  - Installing symfony/dependency-injection (v4.0.3): Loading from cache
  - Installing symfony/config (v4.0.3): Loading from cache
  - Installing psr/simple-cache (1.0.0): Loading from cache
  - Installing psr/cache (1.0.1): Loading from cache
  - Installing symfony/cache (v4.0.3): Loading from cache
  - Installing symfony/framework-bundle (v4.0.3): Loading from cache
  - Installing symfony/yaml (v4.0.3): Loading from cache
  - Installing symfony/dotenv (v4.0.3): Loading from cache
Writing lock file
Generating autoload files
Executing script cache:clear [OK]
Executing script assets:install --symlink --relative public [OK]
```

HÉTIC

Installer une recipe

```
------------------------------------------------------------
~/sites/sf4(branch:master*) » composer req server
./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)

Prefetching 2 packages
  - Connecting (100%)
  - Downloading (100%)

Package operations: 2 installs, 0 updates, 0 removals
  - Installing symfony/process (v4.0.3): Loading from cache
  - Installing symfony/web-server-bundle (v4.0.3): Loading from cache
Writing lock file
Generating autoload files
Symfony operations: 1 recipe (db0393e604d68de66cf3e07403e5cd55)
  - Configuring symfony/web-server-bundle (>=3.3): From github.com/symfony/recipes:master
Executing script cache:clear [OK]
Executing script assets:install --symlink --relative public [OK]

Some files may have been created or updated to configure your new packages.
Please review, edit and commit them: these files are yours.
```

HÉTIC

Flex

- Chaque recipe est un fichier manifest.json dans un repository dédié à un groupe de dépendances

HÉTIC

composer req

- Consulte les recipes disponibles de la team Symfony ou de la communauté et installe les dépendances spécifiées dans le manifest.json du repository de la recipe

HÉTIC

composer remove

- Supprime la recipe de votre projet ainsi que toutes ses dépendances

HÉTIC

STANDARDS DE CODE

HÉTIC

Standards codage PHP

- PSR-1
  http://www.php-fig.org/psr/psr-1/
- PSR-2
  http://www.php-fig.org/psr/psr-2/
- PSR-12
  https://github.com/php-fig/fig-standards/blob/master/proposed/extended-coding-style-guide.md
- Symfony
  http://symfony.com/doc/current/contributing/code/standards.html

HÉTIC

Interfaces standards

- PSR-3 Logger Interface
- PSR-6 Caching Interface
- PSR-7 HTTP Message Interface
- PSR-11 Container Interface
- PSR-13 Hypermedia Links
- PSR-16 Simple Cache

HÉTIC

Twig

- Langage de templating
  https://twig.symfony.com/doc/2.x/coding_standards.html

HÉTIC

Outillage

- PHP CodeSniffer (vérification de la structure code)
  https://github.com/squizlabs/PHP_CodeSniffer
- PHPMetrics (analyse statique du code)
  http://www.phpmetrics.org/

HÉTIC

Installation

- Installer PHP CodeSniffer
- Installer PHP Metrics

SYMFONY 3.4 & 4

HÉTIC

Installation

composer create-project symfony/skeleton my-project

HÉTIC

Démonstration Symfony 3.4

```
composer create-project symfony/skeleton votre-projet 3.4
```

```
Création du squelette d'application symfony 3.4
```

HÉTIC

Démonstration Symfony 3.4

```
cd votre-projet
```

Ouverture du répertoire de projet

HÉTIC

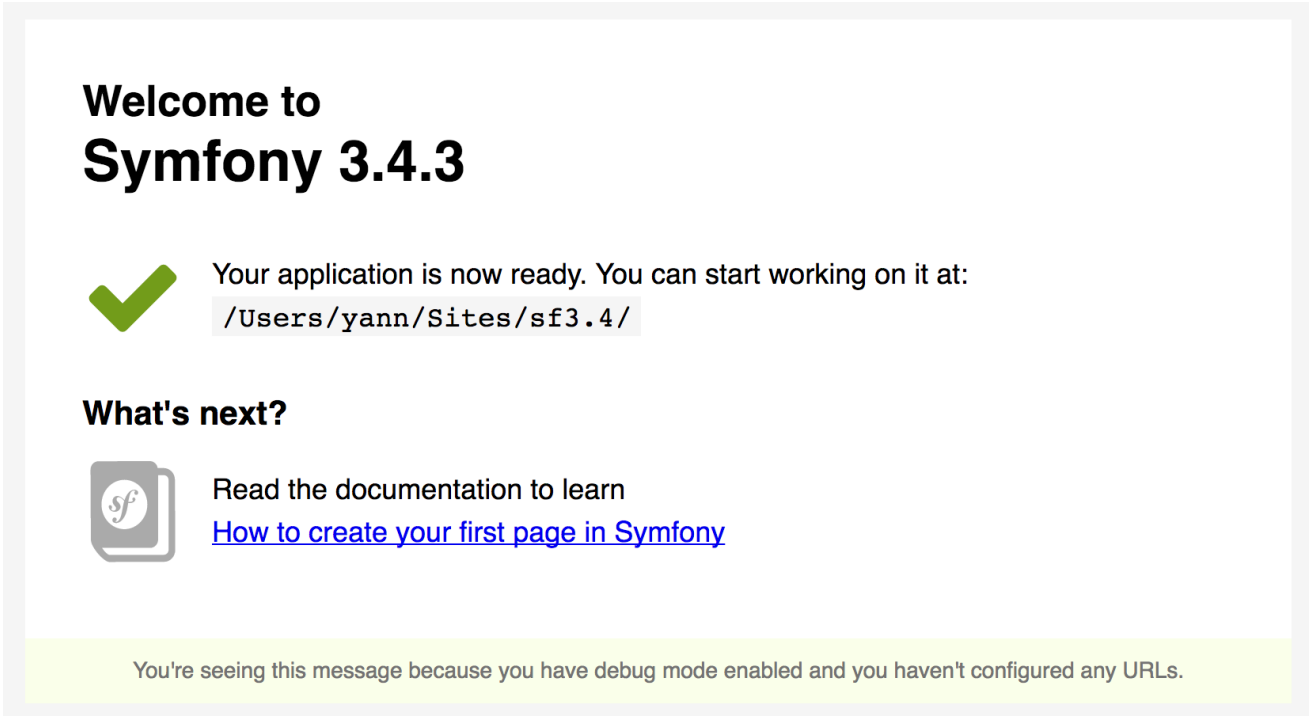
Démonstration Symfony 3.4

```
composer req server
bin/console server:start
```

Installation de la recipe server HTTP

HÉTIC

Démonstration Symfony 3.4



HÉTIC

Démonstration Symfony 3.4

```
composer req annotations
```

Installation de la recipe annotation qui permet à Symfony de lire la configuration stockée dans les annotations de classes, attributs et méthodes.

HÉTIC

Démonstration Symfony 3.4

```
composer req maker
```

Installation de la recipe maker de Symfony. Cette recipe rajoute les commandes de génération en CLI.

HÉTIC

Démonstration Symfony 3.4

```
composer req template
```

Installation de la recipe de templating de Symfony. Cette recipe installe tout ce que Twig nécessite pour fonctionner.

HÉTIC

Démonstration Symfony 3.4

```
bin/console make:controller
```

```
Création d'un controller
MonsterController. Le controller est
une classe spécifique qui gère les
fonctionnalités codées dans Symfony.
```

HÉTIC

Démonstration Symfony 3.4

```
composer req doctrine
```

Installation de la recipe doctrine, l'ORM utilisé par Symfony. C'est une couche d'abstraction permettant de manipuler et stocker les données.

HÉTIC

Démonstration Symfony 3.4

```
pico .env
```

Modification du fichier .env qui contient les informations de connexion à la base de données et plus généralement les paramètres de l'application.

HÉTIC

Démonstration Symfony 3.4

```
sf doctrine:database:create

Création de la base de données
spécifiée dans le fichier .env.
```

HÉTIC

Démonstration Symfony 3.4

```
bin/console make:entity
```

Création de l'entity Monster grâce à un assistant. L'entity est une classe qui liste les attributs qui définissent l'entity et les setters et getters de ces attributs.

HÉTIC

Démonstration Symfony 3.4

```
pico Entity/Monster.php
```

```
Edition de l'entity Monster pour
ajouter des des attributs.
```

HÉTIC

Démonstration Symfony 3.4

```
sf doctrine:schema:update --force
```

Création de la table qui reflète l'entity dans la base de données.

HÉTIC

Démonstration Symfony 3.4

```
composer require admin
```

Installation de la recipe EasyAdmin qui permet d'administrer les données sans développer quoi que ce soit

HÉTIC

Démonstration Symfony 3.4

```
pico config/packages/easy_admin.yaml
```

Edition du fichier de configuration de EasyAdmin pour ajouter l'entity Monster aux entities gérées par EasyAdmin.

HÉTIC

Structure du symfony/skeleton

```
.
├── bin
│   └── console
├── composer.json
├── composer.lock
├── config
│   ├── bundles.php
│   ├── packages
│   │   ├── dev
│   │   │   └── routing.yaml
│   │   ├── framework.yaml
│   │   ├── routing.yaml
│   │   └── test
│   │       └── framework.yaml
│   ├── routes.yaml
│   └── services.yaml
├── public
│   └── index.php
├── src
│   ├── Controller
│   └── Kernel.php
├── symfony.lock
└── var
```

HÉTIC

Référence flex

- Flex
  https://symfony.com/doc/current/setup/flex.html
- Recipes
  https://symfony.sh/
- Recipes Flex officielles
  https://github.com/symfony/recipes
- Recipes Flex contributions de la communauté
  https://github.com/symfony/recipes-contrib

HÉTIC

CONTROLLER

https://symfony.com/doc/current/best_practices/controllers.html

HÉTIC

Controller

- Class PHP, étends: Symfony\Bundle\FrameworkBundle\Controller
- Appelle le model (repository) pour récupérer la collection des entities (les données)
- Traite les entities grâce aux services
- "Render" la view et retourne la réponse HTTP
- Doit rester aussi léger que possible

HÉTIC

Nommage des controllers

- Le nom de la classe controller est suffixé par "Controller"
  MonsterController

HÉTIC

Nommage des méthodes

- Le nommage des méthodes Symfony a évolué dans les dernières version
- Symfony < 3.4
  indexAction
- Symfony >= 3.4
  index

HÉTIC

Controller

```php
<?php

namespace App\Controller;

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

class MonsterController extends Controller
{
    /**
     * @Route("/monster", name="monster")
     */
    public function index()
    {
//        return new Response('Welcome to your new controller!');
        return $this->render('Monster/index.html.twig', []);
    }

}
```

HÉTIC

MODEL

https://symfony.com/doc/current/doctrine.html

HÉTIC

Model

- Le model est géré par Doctrine, l'ORM utilisé par Symfony
- Le model est composé du repository et de l'entity

HÉTIC

Repository

- Le repository est une classe particulière appelée par le controller pour exécuter des requêtes
- Le repository vient avec beaucoup de méthodes héritées de Dotrine qui permettent de faire la plupart des requêtes
- Le repository instancie la classe entity liée

HÉTIC

Entity

- L'entity est une classe, elle définit une structure de données simple
- L'entity est une représentation d'une table du schéma de base de données
- Le schéma de bases de données est généré à partir des entities
- Il est possible de générer les entities à partir du schéma de base de données

HÉTIC

Requêter du controller

```
...
/**
 * @return Response

 * @Route("/pizza", name="pizza")
 */
public function index()
{
    $em = $this->getDoctrine()->getManager();
    $pizzaList = $em->getRepository('App:Pizza')->findAll();

    return $this->render('pizza/index.html.twig', ['pizzaList'=>$pizzaList]);
}
...
```

HÉTIC

YAML

http://yaml.org/

HÉTIC

YAML

- Ça veut dire quoi? YAML Ain't a Markup Language
- Format de sérialisation de données
- Utilisé dans Symfony pour la configuration

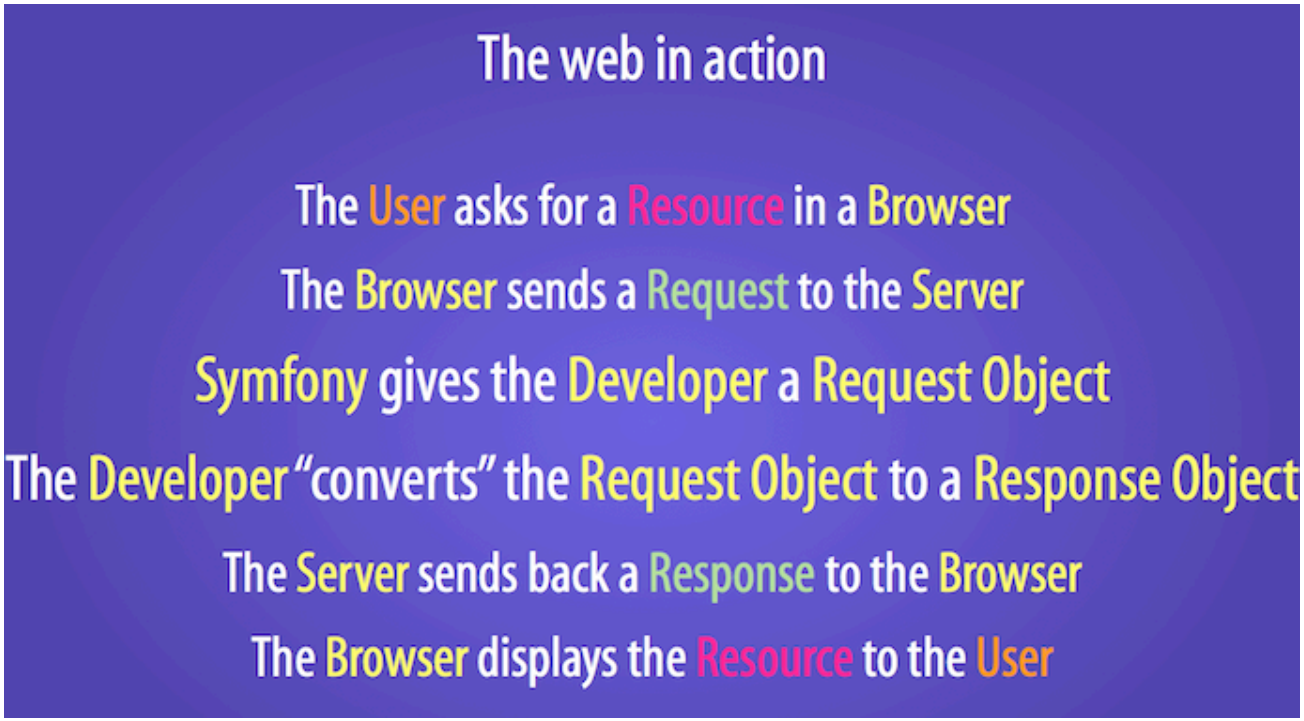
HÉTIC

KERNEL

https://symfony.com/doc/current/components/http_kernel.html

HÉTIC

Kernel



HÉTIC

Kernel

- Responsable du traitement de la requête HTTP et la génération de la réponse HTTP
- Gère les évènements Symfony
- Rarement modifié

HÉTIC

ROUTER

https://symfony.com/doc/current/routing.html

HÉTIC

Router

- Le router est le composent Symfony appelé par le Kernel quand une requête HTTP arrive.
- Le router prends l'URI en entrée et vérifie si une route correspondante a été déclarée

HÉTIC

Router

- Configuration de base config/routes.yaml
- Chaque controller déclare ses routes en annotations

HÉTIC

Lister les routes disponibles

```
bin/console debug:router
```

```
 ------------------ -------- -------- ------ --------------------------
  Name               Method   Scheme   Host   Path
 ------------------ -------- -------- ------ --------------------------
  pokemon            ANY      ANY      ANY    /pokemon
  easyadmin          ANY      ANY      ANY    /admin/
  admin              ANY      ANY      ANY    /admin/
  _twig_error_test   ANY      ANY      ANY    /_error/{code}.{_format}
 ------------------ -------- -------- ------ --------------------------
```

HÉTIC

Tester une route

```
bin/console router:match /admin/


 [OK] Route "easyadmin" matches


+--------------+-------------------------------------------------------------------------+
| Property     | Value                                                                   |
+--------------+-------------------------------------------------------------------------+
| Route Name   | easyadmin                                                               |
| Path         | /admin/                                                                 |
| Path Regex   | #^/admin/$#s                                                            |
| Host         | ANY                                                                     |
| Host Regex   |                                                                         |
| Scheme       | ANY                                                                     |
| Method       | ANY                                                                     |
| Requirements | NO CUSTOM                                                               |
| Class        | Symfony\Component\Routing\Route                                         |
| Defaults     | _controller: EasyAdminBundle:Admin:index                                |
| Options      | compiler_class: Symfony\Component\Routing\RouteCompiler                 |
| Callable     | EasyCorp\Bundle\EasyAdminBundle\Controller\AdminController::indexAction |
+--------------+-------------------------------------------------------------------------+
```

HÉTIC

Routing en YAML

```
index:
    path: /
    defaults: { _controller: 'App\Controller\DefaultController::index' }
```

Routing en annotation

```php
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;

class PokemonController extends Controller
{
    /**
     * @Route("/pokemon", name="pokemon")
     */
    public function index()
    {
        return $this->render('Pokemon/index.html.twig', []);
    }

}
```

HÉTIC

Exercice

- Ajouter une route pour lister les entities créées dans le précédent exercice
- Ajouter une route pour voir les détails d'une des entities existantes

HÉTIC