



# Symfony

Par Robin Delbaere

# Qu'est ce que Symfony ?

- Framework PHP
- Développé par SensioLabs
- Orienté Objet
- Architecture MVC

# D'autres alternatives ?



# Dans la balance

- Avantages
  - Open Source
  - Communauté
  - Documentation
  - Mise à jour
  - Version LTS
  - Performance
- Inconvénients
  - Performance
  - Apprentissage
  - Ligne de commande

# Les composants

- Routeur
- Moteur de template
- ORM
- Pare-feu
- Système de cache
- Système de log
- Gestionnaire d'utilisateurs
- Module d'internationalisation
- ...

# Les versions Symfony

- Basé sur SemVer



# Composer

- Gestionnaire de dépendances
- Non spécifique à Symfony
- Gère les versions et l'autoload
- Référencement des packages : <http://www.packagist.org>

# Créer un projet

- Installer Composer
  - <https://getcomposer.org/download/>

- Générer un projet
  - Préconfiguré pour le web
  - Configuration minimale

```
λ composer create-project symfony/website-skeleton alma
```

```
λ composer create-project symfony/skeleton alba
```

# Fil Rouge



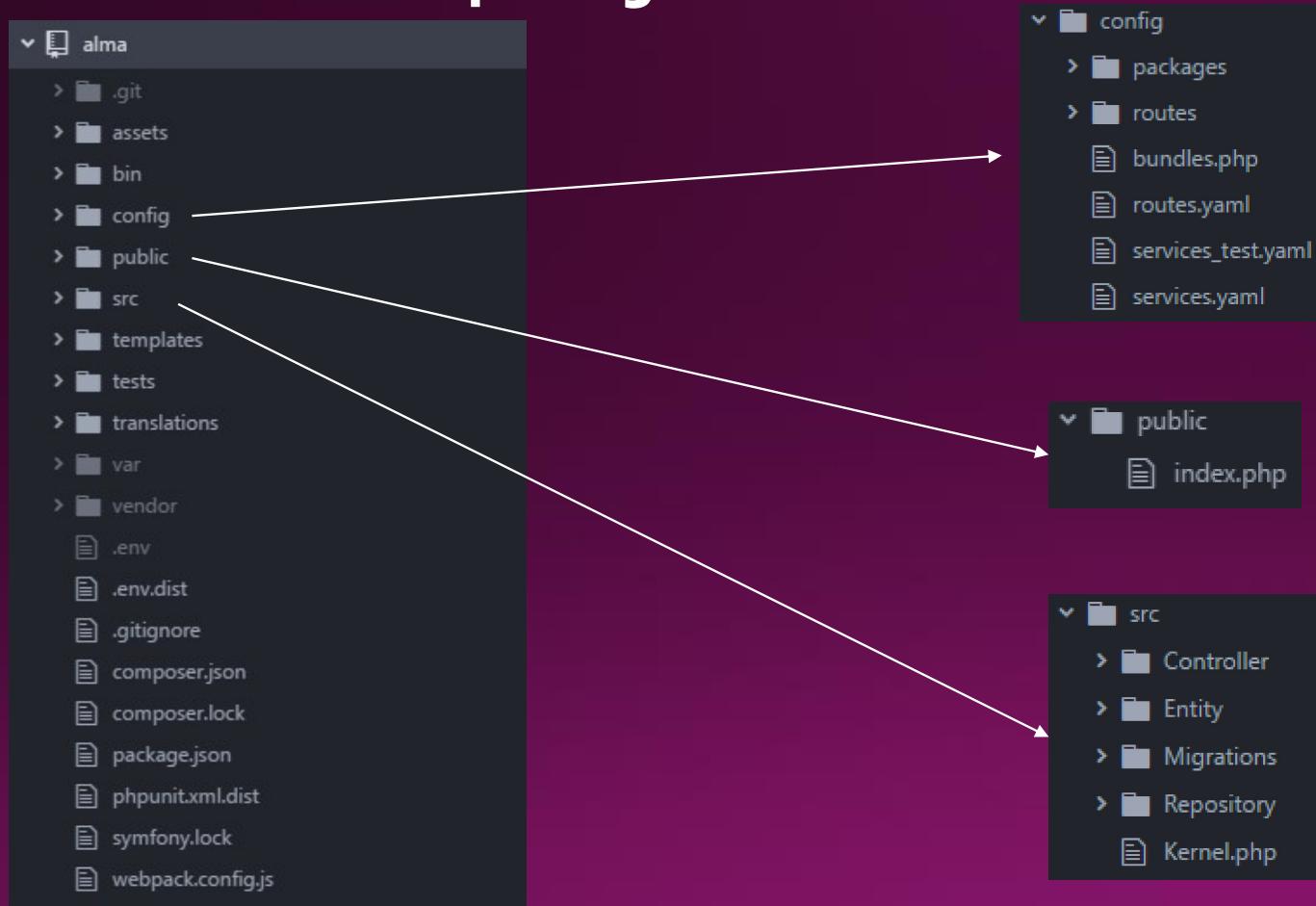
BeerTime

- Lister les évènements
- Afficher les détails d'un évènement
- Créer un évènement
- Afficher le nombre d'évènement
- Recherche par nom
- Tri des évènements
- Rejoindre un évènement
- ...

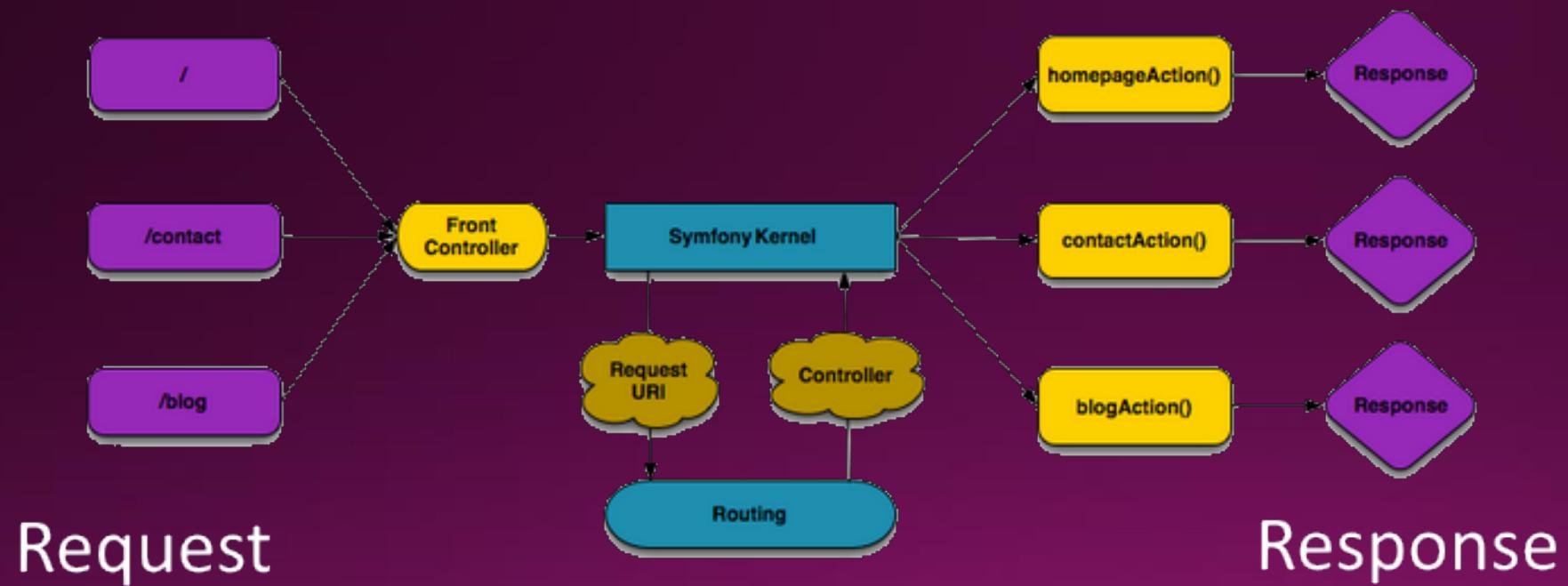
# Pratique

- Créer un projet Symfony
- Créer un modèle de donnée

# Structure du projet



# Workflow



# La console Symfony

- Usage

- `λ php bin/console`

- Fonctionnalités

- Génération de code

- `λ php bin/console make:controller`

- Gestion du cache

- `λ php bin/console cache:clear`

- Manipulation de la base de données

- `λ php bin/console doctrine:schema:update --dump-sql`

- Information de débogage

- `λ php bin/console debug:router --env=prod`

# La configuration d'environnement

- Gestion via deux fichiers
  -  .env configuration de l'environnement courant
  -  .env.dist modèle de configuration

```
APP_ENV=dev ←           Type d'environnement  
APP_SECRET=76a70afdf322c3a6c0e10592855e3eab ←   Token de sécurité  
  
MAILER_URL=null://localhost ←           Configuration du mailer  
  
DATABASE_URL=mysql://db_user:db_password@127.0.0.1:3306/db_name ← DSN de la base de donnée
```

# Mise en place des routes

- Créer une route

Annotation

```
/**  
 * @Route("/", name="main")  
 */  
public function index() { =
```

YAML `config\routes.yaml`

```
main:  
    path: /  
    controller: App\Controller>MainController::index
```

Nom	Description
methods	Méthodes autorisées
schemes	Forcer le protocole
host	Restreindre l'hôte

```
main:  
    path: /  
    controller: App\Controller>MainController::index  
    methods: [GET,POST]  
    schemes: [https]  
    host: 127.0.0.1
```

# Les routes dynamiques

- Définir un paramètre dans une route

```
hello:  
    path: /hello/{username}  
    controller: App\Controller\MainController::hello
```

- Définir une valeur par défaut

```
hello:  
    path: /hello/{username}  
    controller: App\Controller\MainController::hello  
    defaults:  
        username: Paul
```

- Définir des exigences

```
hello:  
    path: /hello/{username}  
    controller: App\Controller\MainController::hello  
    requirements:  
        username: Paul|Xul|Alice|Traquenard
```

```
hello:  
    path: /hello/{username}  
    controller: App\Controller\MainController::hello  
    requirements:  
        username: "[a-zA-Z]{2,15}"
```

# Les contrôleurs

- Retourne impérativement une réponse HTTP
- Créer un conteneur de contrôleur `src\Controller`

```
<?php  
namespace App\Controller;  
  
use Symfony\Bundle\FrameworkBundle\Controller\Controller;  
  
class MainController extends Controller{  
  
}
```

- Créer un point d'entrée

```
public function index(){  
    return new Response('Hello');  
}
```

# Les différents type de réponse

- Une réponse standard

```
return new Response( 'Hello', 200, array( 'Content-Type' => 'text/plain; charset=utf-8' ) );
```

- Une réponse JSON

```
return new JsonResponse( array( 'status' => true, 'data' => 'Bonjour' ) );
```

- Une redirection

```
return $this->redirectToRoute( 'hello', array( 'username' => 'Dieu' ));
```

- Un template

```
return $this->render( 'main/index.html.twig' );
```

# Du contexte dans un contrôleur

- Injecter le requête

```
public function index( Request $request ){
    $getParameters = $request->query->all();
    $postParameters = $request->request->all();
    $ajax = $request->isXmlHttpRequest();

    return $this->render( 'main/index.html.twig' );
}
```

- Injecter les paramètres

```
public function hello( $username ){
    return $this->render('main/hello.html.twig', array(
        'username' => $username
    ));
}
```

# Pratique

- Créer les routes et les contrôleurs associés pour les pages suivantes
  - Accueil
  - Liste des événements
  - Affichage d'un événement
  - Création d'un événement
  - Rejoindre un évènement

# Les templates

- Utilisation du moteur Twig
- Type de template multiple (HTML, JSON, ...)
- Nommage `{name}.{type}.twig`
- Dans le dossier `templates/`

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Hello</title>
    </head>
    <body>
        <h1>Hello {{ username }}</h1>
    </body>
</html>
```

# Les variables en Twig

- Affichage

```
<h1>Hello {{ username }}</h1>
```

- Concaténation

```
<h1>{{ 'Hello ' ~ username }}</h1>
```

- Opération

```
<p>{{ ( ratePress + ratePlayer ) / 2 }}</p>
```

- Assigner

```
<p>
    {% set total = ( rate.press + rate.player ) / 2 %}
    {{ total }}
</p>
```

```
return $this->render('main/hello.html.twig', array(
    'username' => $username,
));
```

```
return $this->render('main/hello.html.twig', array(
    'ratePress' => 12,
    'ratePlayer' => 14,
));
```

```
return $this->render('main/hello.html.twig', array(
    'rate' => array(
        'press' => 12,
        'player' => 11,
    ),
));
```

# Les structures de contrôles en Twig

- Structure conditionnelle

```
{% if isConnected %}  
    <p>J'te reconnos ti t'es d'min coin</p>  
{% else %}  
    <p>Ej n'ai jamais vu t'giffe</p>  
{% endif %}
```

```
{% if age < 17 %}  
    <p>Pas de bière pour toi</p>  
{% elseif age < 18 %}  
    <p>Les bières c'est pour bientôt</p>  
{% else %}  
    <p>Prends une girafe, c'est plus efficace</p>  
{% endif %}
```

- Structure de répétition

```
<ul>  
    {% for actor in actors %}  
        <li>{{ loop.index }} - {{ actor }}</li>  
    {% endfor %}  
</ul>
```

```
<p>J'apprends à compter avec PHP</p>  
<ul>  
    {% for i in 0..10 %}  
        <li>{{ i }}</li>  
    {% endfor %}  
</ul>
```

# Séparation des templates

- Héritage

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>{% block title %}Alma{% endblock %}</title>
    {% block stylesheets %}{% endblock %}
  </head>
  <body>
    {% block body %}{% endblock %}
    {% block javascripts %}{% endblock %}
  </body>
</html>
```

```
{% extends 'base.html.twig' %}

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

{% block body %}
  <h1>{{ 'Hello ' ~ username }}</h1>
  {% endblock %}

  <h3>Les bières du moment</h3>
  <ul>
    <li>Paix Dieu</li>
    <li>Tripel Karmeliet</li>
    <li>Anosteké</li>
  </ul>
```

- Inclusion

```
{% include 'main/component.html.twig' %}
```

# Des variables globales pour Twig

- Quelques variables globales disponibles

Nom	Description
app	Information sur l'application
app.user	Utilisateur courant
app.request	Requête courante
app.session	Session courante

- Ajouter des variables globales `config\packages\twig.yaml`

```
twig:  
  # ...  
  globals:  
    apiKey: qf8s98dqs564vq897f2548b25v
```

```
<p>La clé d'API est {{ apiKey }}</p>
```

# Twig – Filtres & fonctions utiles

Filtre	Description	Exemple
upper	Met toutes les lettres en majuscules	<code>{{ 'bonjour'   upper }}</code>
date	Formate la date selon le format donné en argument	<code>{{ date date('d/m/Y') }}</code>
round	Arrondi un nombre flottant	<code>{{ 42.55 round(1) }}</code>
length	Retourne le nombre d'éléments d'un tableau	<code>{{ actors length }}</code>

Fonction	Description	Exemple
asset	Accès aux ressources statique	<code>&lt;img src="{{ asset('images/roadster.jpg') }}" /&gt;</code>
path	Créer un lien vers une route	<code>&lt;a href="{{ path('main') }}&gt;Accueil&lt;/a&gt;</code>
dump	Dumper une variable	<code>{{ dump( app.user ) }}</code>
date	Récupérer la date courante	<code>{{ date() date('d/m/Y') }}</code>

# Pratique

- Créer un template pour la page d'accueil
  - Présentation de la plateforme
  - Lien vers la liste
- Créer un template pour la liste d'événement
  - Les événements seront un tableau PHP transmis à la vue pour affichage
  - Pour chaque événement afficher une pastille (a venir, en cours, passé)
- Créer un template pour les détails d'un événement
  - L'événement sera un tableau PHP transmis à la vue pour affichage
- Utiliser le système d'héritage pour créer un design générique

# Les services

- Classe PHP simple
  - Réalise une tâche spécifique
  - Géré par le conteneur de service
- 
- Créer un service

```
src\Service  
<?php  
namespace App\Service;  
  
class MediaService{  
}
```

- Injecter un service

```
use App\Service\MediaService;  
  
public function getMedias( MediaService $mediaService ){  
    return $mediaService->getAll();  
}
```

- Lister les services

```
\ php bin\Console debug:autowiring
```

# Pratique

- Créer un service pour centraliser la gestion des événements
  - Tableau des évènements
  - Fonction pour récupérer tous les évènements
  - Fonction pour récupérer un évènement

# Doctrine



**DBAL**

DataBase Abstraction Layer

**ORM**

Object Relational Mapper

# Configuration de Doctrine

- Information de connexion

- Configuration du DSN `.env`

```
DATABASE_URL=mysql://root:password@127.0.0.1:3306/alma
```

Type    Identifiant    Mot de passe    Adresse    Port    Nom de la base

- Configurer de Doctrine `config/packages/doctrine.yaml`

- Crédit de la base

```
λ php bin/console doctrine:database:create
```

# Générer une entité

- Utilisation de la commande

```
λ php bin\console make:entity
```

- Utilisation de l'assistant

```
Class name of the entity to create or update (e.g. TinyJellybean):
> Movie
Movie

created: src/Entity/Movie.php
created: src/Repository/MovieRepository.php

Entity generated! Now let's add some fields!
You can always add more fields later manually or by re-running this command.

New property name (press <return> to stop adding fields):
> title

Field type (enter ? to see all types) [string]:
>

Field length [255]:
>

Can this field be null in the database (nullable) (yes/no) [no]:
>

updated: src/Entity/Movie.php

Add another property? Enter the property name (or press <return> to stop adding fields):
>

Success!
```

• Utilisation de la commande

A php bin\console make:entity

```
namespace App\Entity;

use Doctrine\ORM\Mapping as ORM;

/*
 * @ORM\Entity(repositoryClass="App\Repository\MovieRepository")
 */
class Movie
{
    /**
     * @ORM\Id()
     * @ORM\GeneratedValue()
     * @ORM\Column(type="integer")
     */
    private $id;

    /**
     * @ORM\Column(type="string", length=255)
     */
    private $title;

    public function getId(): ?int
    {
        return $this->id;
    }

    public function getTitle(): ?string
    {
        return $this->title;
    }

    public function setTitle(string $title): self
    {
        $this->title = $title;

        return $this;
    }
}
```

# Impacter la base de donnée

- Mise à jour de la base de donnée

- Vérifier

```
λ php bin\console doctrine:schema:update --dump-sql
```

- Appliquer

```
λ php bin\console doctrine:schema:update --force
```

Ne modifiez jamais la structure de la base en passant par PHPMyAdmin

# Les relations entre entités - OneToOne



- Unidirectionnel

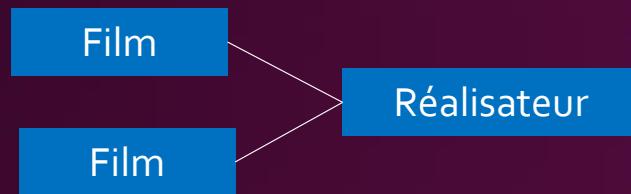
```
/**  
 * @ORM\OneToOne(targetEntity="Review")  
 */  
private $review;
```

- Bidirectionnel

```
/**  
 * @ORM\OneToOne(targetEntity="Review", inverseBy="movie")  
 */  
private $review;
```

```
/**  
 * @ORM\OneToOne(targetEntity="Movie", mappedBy="review")  
 */  
private $movie;
```

# Les relations entre entités - ManyToOne



- Unidirectionnel

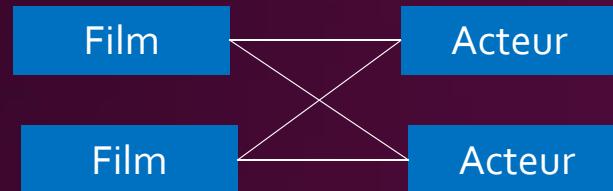
```
/**  
 * @ORM\ManyToOne(targetEntity="Director")  
 */  
private $director;
```

- Bidirectionnel

```
/**  
 * @ORM\ManyToOne(targetEntity="Director", inversedBy="movies")  
 */  
private $director;
```

```
/**  
 * @ORM\OneToMany(targetEntity="Movie", mappedBy="director")  
 */  
private $movies;  
  
public function __construct() {  
    $this->movies = new ArrayCollection();  
}
```

# Les relations entre entités - ManyToMany



- Unidirectionnel

```
/**  
 * @ORM\ManyToMany(targetEntity="Actor")  
 */  
private $actors;  
public function __construct() {  
    $this->actors = new ArrayCollection();  
}
```

- Bidirectionnel

```
/**  
 * @ORM\ManyToMany(targetEntity="Actor", inversedBy="movies")  
 * @ORM\JoinTable(name="cast")  
 */  
private $actors;  
public function __construct() {  
    $this->actors = new ArrayCollection();  
}
```

```
/**  
 * @ORM\ManyToMany(targetEntity="Movie", mappedBy="actors")  
 */  
private $movies;  
public function __construct() {  
    $this->movies = new ArrayCollection();  
}
```

# Pratique

- Créer les entités depuis notre modèle de donnée
- Configuration de la connexion à MySQL
- Créer la base de données
- Mettre à jour la structure de la base de données
- Saisir des données

# Créer des entités

- Récupération de l'EntityManager

```
$em = $this->getDoctrine()->getManager();
```

- Création de l'entité

```
$movie = new Movie();
$movie->setTitle( 'Inception' );
```

- Persistance de l'entité

```
$em->persist( $movie );
$em->flush();
```

# Récupérer des entités

- Récupérer un dépôt

```
$movieRepository = $this->getDoctrine()->getRepository( 'App:Movie' );
```

- Récupérer toutes les entités

```
$movies = $movieRepository->findAll();
```

- Rechercher des entités

```
$movies = $movieRepository->findBy( array(
    'title' => 'Inception'
), array( 'id' => 'DESC' ), 5, 0 );
```

- Rechercher une entité

```
$movie = $movieRepository->findOneBy( array( 'title' => 'Everest' ) );
```

- Récupérer une entité par son id

```
$movie = $movieRepository->find( 1 );
```

# Récupération avancée

- Utilisation des dépôts `src\Repository\{entity}Repository.php`
- Récupération personnalisé

```
public function customQuery(){  
    $stmt = $this->createQueryBuilder('m');  
    return $stmt->getQuery()->getResult();  
}
```

- Ajouter une clause WHERE

```
$stmt->where( 'm.title LIKE :term' );  
$stmt->setParameter( ':term', '%Knight%' );
```

- Définir les critères

```
$stmt->orderBy('m.title', 'ASC');  
$stmt->setFirstResult( 1 );  
$stmt->setMaxResults( 5 );
```

- Retourner un seul résultat

```
return $stmt->getQuery()->getOneOrNullResult();
```

- Compter le nombre d'entrées

```
$stmt->select('count(m)');  
return $stmt->getQuery()->getSingleScalarResult();
```

- Utilisation

```
$movies = $movieRepository->customQuery();
```

# Interagir avec le cycle de vie Doctrine

Autoriser les callbacks

```
/**  
 * @ORM\Entity(repositoryClass="AppBundle\Repository\MovieRepository")  
 * @ORM\HasLifecycleCallbacks()  
 */  
class Movie
```

Définir un callback

```
/**  
 * @ORM\PreUpdate  
 */  
public function updateDate()  
{  
    $this->updatedAt = new \DateTime();  
}
```

Evénement
PrePersist
PostPersist
PreUpdate
PostUpdate
PreRemove
PostRemove
PostLoad

# Pratique

- Dans le service
  - Dynamiser la récupération de tous les évènements
  - Dynamiser la récupération d'un évènement
  - Créer une fonction pour rechercher les évènements par nom
  - Créer une fonction retournant le nombre d'évènement à venir
- Sur la liste des évènements
  - Mettre en place un système de tri pour les évènements
  - Rendre fonctionnel le système de recherche