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

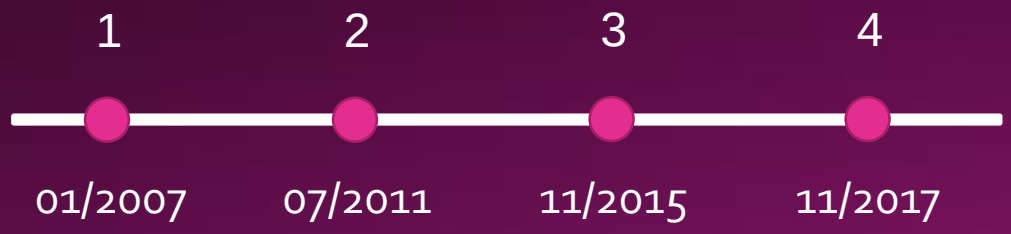
- Routeur
- Moteur de template
- ORM
- Pare-feu
- Système de cache
- Système de log
- Gestionnaire d’utilisateurs
- Module d’internationalisation
- …

Les versions Symfony

- Basé sur SemVer

1 . 3 . 1

MAJOR . MINOR . PATCH



- Courante : 4.0.6 – 07/2018
- LTS : 3.4.6 - 11/2020
- Développement : 4.1.0 - 05/2018

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

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

  - Configuration minimale

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

Fil Rouge

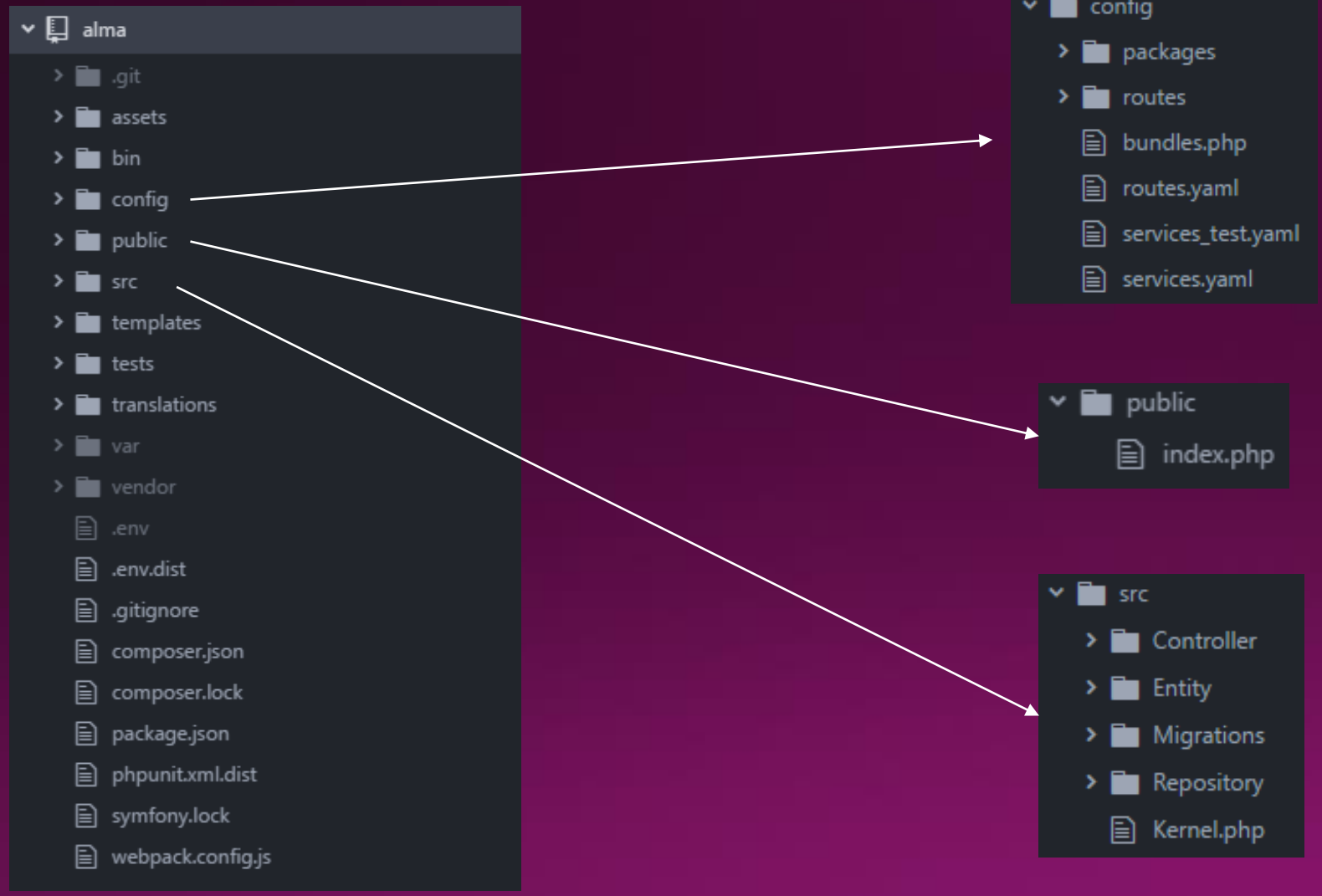
BeerTime

- 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
- …

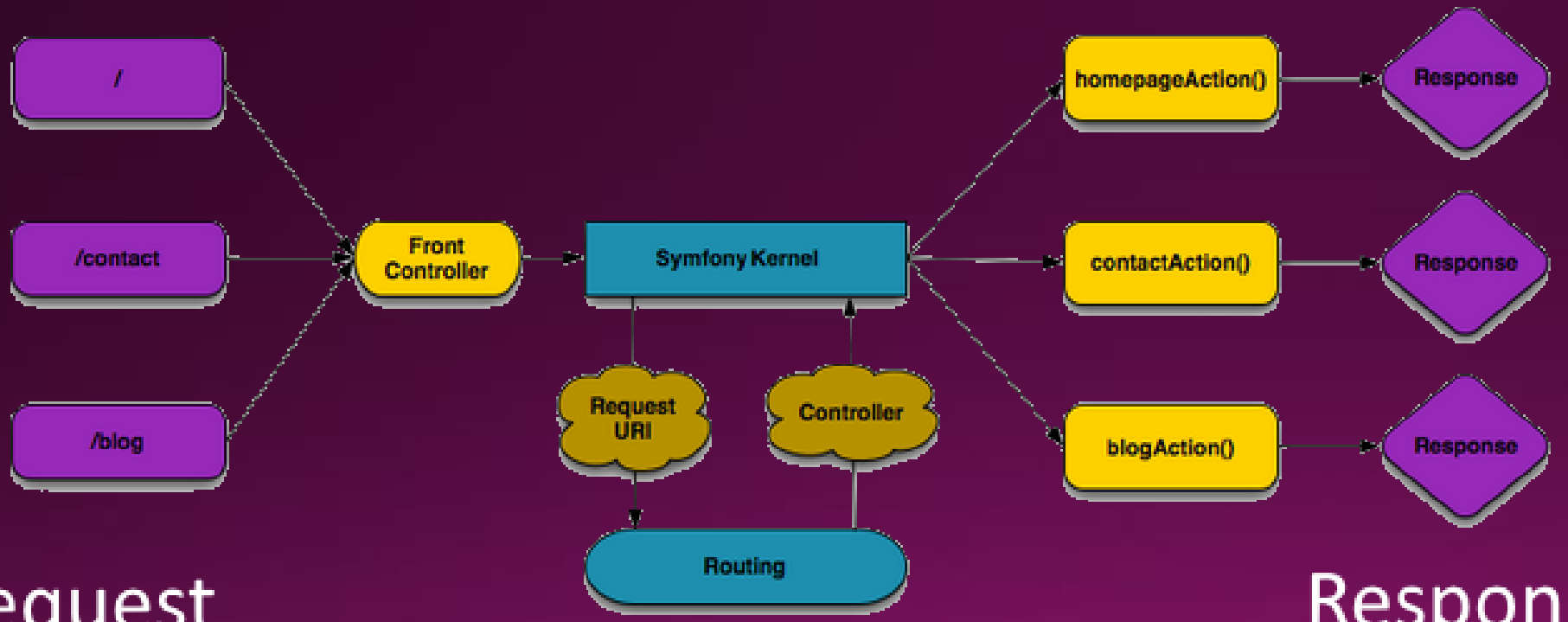
Pratique

- Créer un projet Symfony
- Créer un modèle de donnée

Structure du projet



Workflow



Request

Response

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

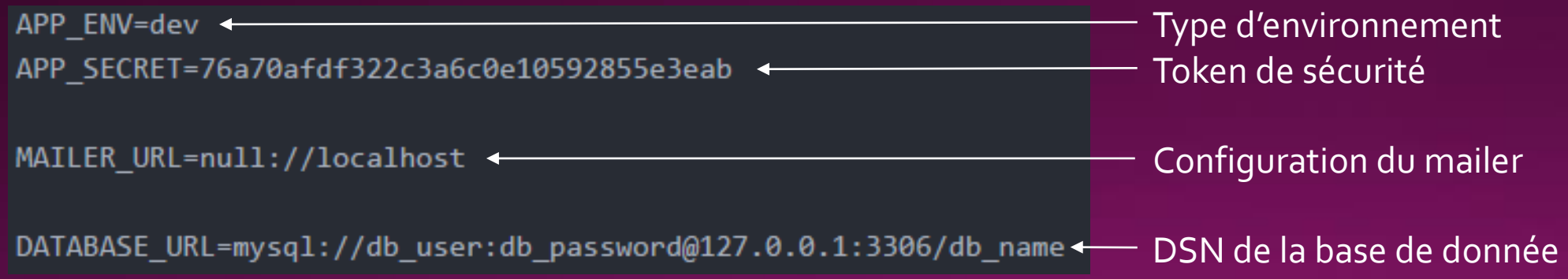
```
λ php bin/console doctrine:schema:update --dump-sql
```

  - 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



Mise en place des routes

- Créer une route

Annotation

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

YAML `config\routes.yaml`

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

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

```yaml
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

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

- Définir une valeur par défaut

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

- Définir des exigences

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

```yaml
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 containeur de contrôleur `src\Controller`

```php
<?php
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\Controller;

class MainController extends Controller{

}
```

- Créer un point d'entrée

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

Les différents type de réponse

- Une réponse standard

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

- Une réponse JSON

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

- Une redirection

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

- Un template

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

Du contexte dans un contrôleur

- Injecter le requête

```php
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

```php
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/`

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

Les variables en Twig

- Affichage

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

- Concaténation

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

- Opération

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

- Assigner

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

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

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

```php
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

```twig
<!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>
```

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

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

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

- Inclusion

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

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

Des variables globales pour Twig

- Quelques variables globales disponible

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`

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

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

Twig – Filtres & fonctions utiles

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

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

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 évènement 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 containeur de service

- Créer un service `src\Service`

```php
<?php
namespace App\Service;

class MediaService{

}
```

- Injecter un service

```php
use App\Service\MediaService;

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

- Lister les services

```
λ 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

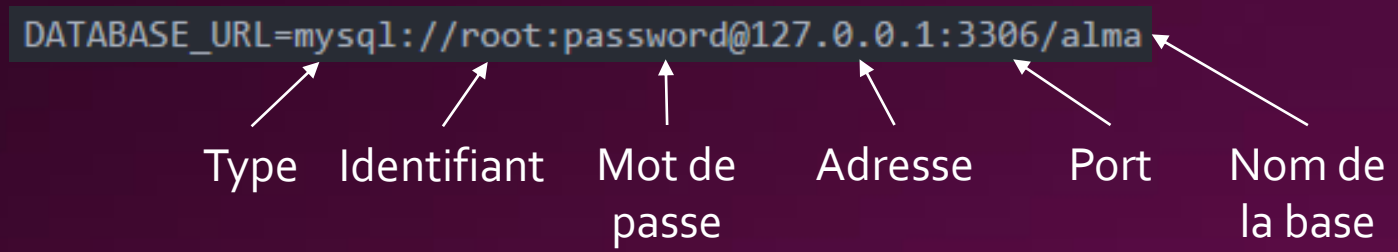
DataBase Abstraction Layer

ORM

Object Relational Mapper

Configuration de Doctrine

- Information de connexion
  - Configuration du DSN `.env`



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

- Création 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!
```

```php
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()
    {
        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

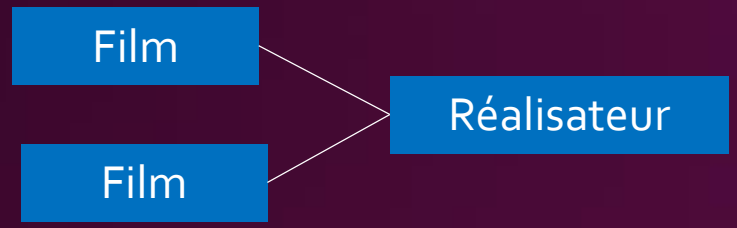
```php
/**
 * @ORM\OneToOne(targetEntity="Review")
 */
private $review;
```

- Bidirectionnel

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

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

Les relations entre entités - ManyToOne



- Unidirectionnel

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

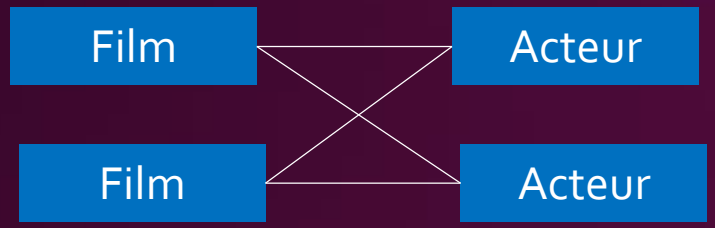
- Bidirectionnel

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

```php
/**
 * @ORM\OneToMany(targetEntity="Movie", mappedBy="director")
 */
private $movies;

public function __construct() {
    $this->movies = new ArrayCollection();
}
```

Les relations entre entités - ManyToMany



- Unidirectionnel

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

- Bidirectionnel

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

```php
/**
 * @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

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

- Création de l'entité

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

- Persistance de l'entité

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

Récupérer des entités

- Récupérer un dépôt

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

- Récupérer toutes les entités

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

- Rechercher des entités

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

- Rechercher une entité

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

- Récupérer une entité par son id

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

Récupération avancée

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

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

- Ajouter une clause WHERE

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

- Définir les critères

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

- Retourner un seul résultat

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

- Compter le nombre d'entrées

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

- Utilisation

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

Interagir avec le cycle de vie Doctrine

Autoriser les callbacks

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

Définir un callback

```php
/**
 * @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

Doctrine – Les migrations

- Mettre à jour la base de données

- Créer une migration

```
λ php bin/console make:migration
```

Génère →

- Déployer une migration

```
λ php bin/console doctrine:migrations:migrate
```

- Annuler une migration

```
λ php bin/console doctrine:migrations:execute [x] --down
```

src/Migrations/Version[x].php

```php
final class Version20181202155335 extends AbstractMigration
{
    public function up(Schema $schema) : void
    {
        // this up() migration is auto-generated, please modi
        $this->abortIf($this->connection->getDatabasePlatform

        $this->addSql('CREATE TABLE post (id INT AUTO_INCREME
    }

    public function down(Schema $schema) : void
    {
        // this down() migration is auto-generated, please mo
        $this->abortIf($this->connection->getDatabasePlatform

        $this->addSql('DROP TABLE post');
    }
}
```

Pratique

- Créer une nouvelle entité pour les commentaires
- Utiliser les migrations pour mettre à jour la base de données

Les formulaires

- Générer un formulaire

```
λ php bin\console make:form
```

- Utiliser le formulaire

```php
public function movie(){
    $movie = new Movie();
    $form = $this->createForm( MovieType::class, $movie );

    return $this->render('main/movie.html.twig', array(
        'form' => $form->createView(),
    ));
}
```

- Afficher le formulaire

```
{{ form_start( form ) }}
{{ form_widget( form ) }}
<button type="submit">Ajouter</button>
{{ form_end( form ) }}
```

src\Form\MovieType.php

```php
<?php
namespace App\Form;

use App\Entity\Movie;
use Symfony\Component\Form\AbstractType;
use Symfony\Component\Form\FormBuilderInterface;
use Symfony\Component\OptionsResolver\OptionsResolver;

class MovieType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder
            ->add('title')
            ->add('director')
            ->add('releaseDate')
            ->add('description')
        ;
    }

    public function configureOptions(OptionsResolver $resolver)
    {
        $resolver->setDefaults([
            'data_class' => Movie::class,
        ]);
    }
}
```

Configurer le formulaire

- Modifier les champs

```php
$builder
    ->add('title', TextType::class, array(
        'label' => 'Titre',
    ))
    ->add('director', null, array(
        'label' => 'Réalisateur',
    ))
    ->add('releaseDate', DateType::class, array(
        'label' => 'Date de sortie',
        'widget' => 'single_text',
    ))
    ->add('description')
;
```

Les propriétés dépendent du type de champ

- Quelques types
  - TextType
  - TextareaType
  - EmailType
  - IntegerType
  - PasswordType
  - ChoiceType
  - EntityType
  - DateType
  - DateTimeType
  - CheckboxType
  - FileType
  - RadioType
  - CollectionType
  - HiddenType

Personnaliser l'affichage du formulaire

- Ouvrir le formulaire

```
{{ form_start( form ) }}
```

- Fermer le formulaire

```
{{ form_end( form ) }}
```

- Affichage complet

```
{{ form_row( form.title ) }}
```

- Par élément

```
{{ form_label( form.title ) }}
{{ form_widget( form.title ) }}
{{ form_errors( form.title ) }}
```

- Ajouter des attributs

```
{{ form_start( form, {
    'attr': {'novalidate': 'novalidate'},
    'method': 'GET',
}) }}
```

- Afficher le formulaire complet

```
{{ form( form ) }}
```

Egalement nommé la méthode « j'ai pas le temps »

Traiter le formulaire

- Intercepter les données

```php
$form->handleRequest( $request );
```

- Vérifier la validité du formulaire

```php
if( $form->isSubmitted() && $form->isValid() ){
    // Sauvegarder l'entité
}
```

- Sauvegarder l'entité

```php
$em = $this->getDoctrine()->getManager();
$em->persist( $movie );
$em->flush();
```

- Exemple

```php
public function movie( Request $request ){
    $movie = new Movie();
    $form = $this->createForm( MovieType::class, $movie );

    $form->handleRequest( $request );
    if( $form->isSubmitted() && $form->isValid() ){
        $em = $this->getDoctrine()->getManager();
        $em->persist( $movie );
        $em->flush();

        return $this->redirectToRoute('main');
    }

    return $this->render('main/movie.html.twig', array(
        'form' => $form->createView(),
    ));
}
```

Validation du formulaire

- Définir des règles pour une entité
- Importation

```php
use Symfony\Component\Validator\Constraints as Assert;
```

- Définir une contrainte

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

- Définir plusieurs contraintes

```php
/**
 * @Assert\NotBlank(message="Vous devez renseigner le réalisateur")
 * @Assert\Length(min=3)
 * @ORM\Column(type="string", length=255)
 */
private $director;
```

- Quelques contraintes
  - NotBlank
  - IsTrue
  - Email
  - Length
  - Url
  - Range
  - EqualTo
  - LessThan
  - GreaterThan
  - Choice
  - File
  - Expression

Les messages Flash

- Notification à usage unique
- Utilise le système de session

- Usage
  - Notification de connexion
  - Confirmation de formulaire
  - Retour d'une action

- Ajouter un message

```php
$this->addFlash(
    'notice',
    'Votre film à été ajouté'
);
```

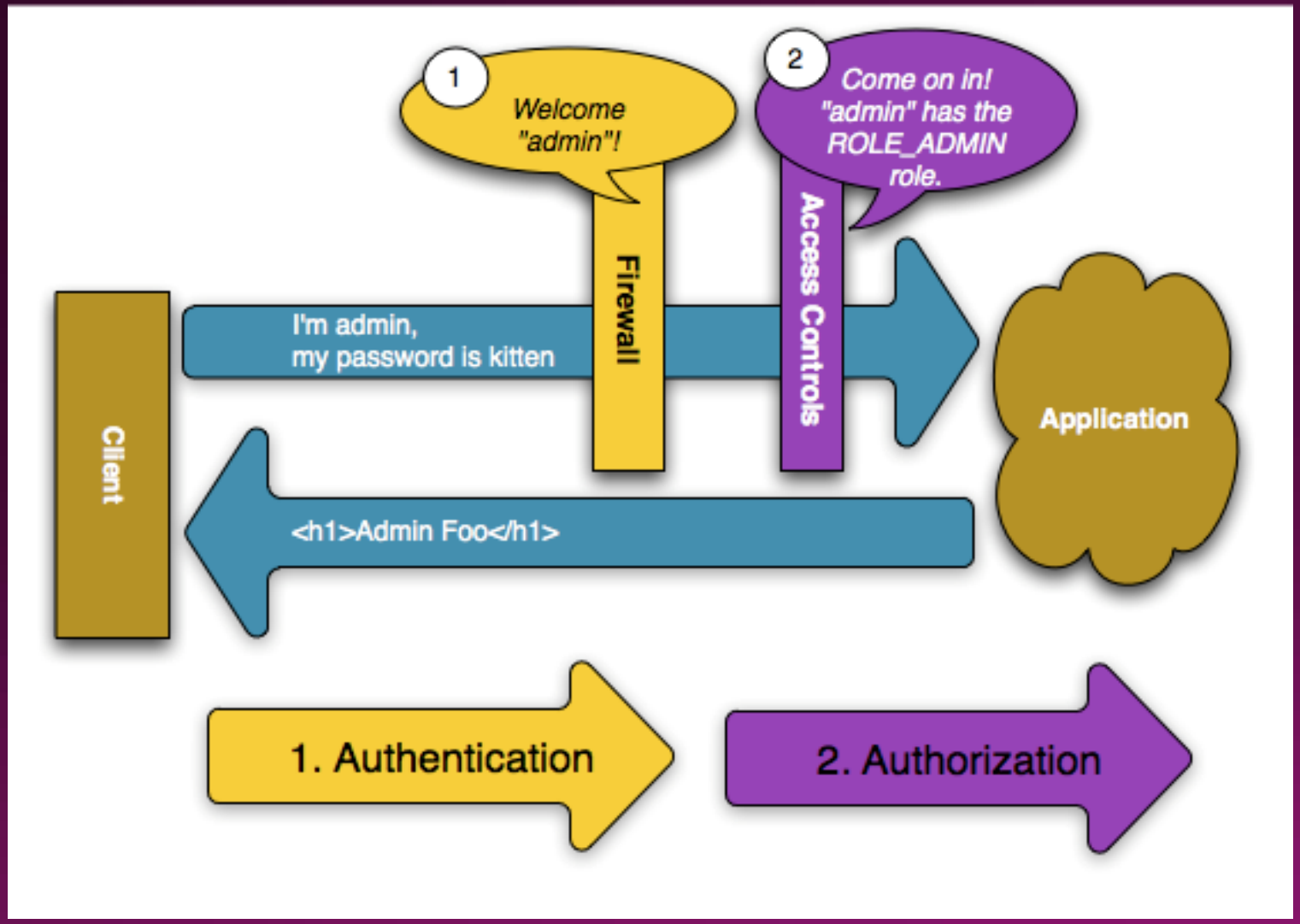
- Afficher les messages

```twig
{% for label, messages in app.flashes %}
    {% for message in messages %}
        <div class="flash-{{ label }}">
            {{ message }}
        </div>
    {% endfor %}
{% endfor %}
```

Pratique

- Créer le formulaire de création d'évènement
  - Créer le FormType
  - Instancier le formulaire dans le contrôleur
  - Afficher le formulaire dans le template
  - Gérer le traitement du formulaire
  - Ajouter les règles de validation suivantes
    - Nom – minimum 3 caractères
    - Date de début – doit être une date future
- Créer un formulaire pour s'inscrire à un évènement

Mécanisme d'authentification



Sécurité

- Configuration `config\packages\security.yaml`

```yaml
security:
    providers:
        in_memory: { memory: ~ }

    firewalls:
        dev:
            pattern: ^/(_(profiler|wdt)|css|images|js)/
            security: false
        main:
            anonymous: true

    access_control:
```

Authentification HTTP simple

- Définir la route

```yaml
manager:
    path: /manager
    controller: App\Controller\MainController::manager
```

- Définir le contrôleur

```php
public function manager(){
    return new Response('Bienvenue sur le manager');
}
```

- Configurer le pare-feu

`config\packages\security.yaml`

```yaml
security:
    encoders:
        Symfony\Component\Security\Core\User\User: plaintext

    providers:
        managers:
            memory:
                users:
                    admin:
                        password: supermotdepasse
                    dieu:
                        password: ueid

    firewalls:
        dev:
            pattern: ^/(_(profiler|wdt)|css|images|js)/
            security: false
        manager:
            pattern: ^/manager$
            http_basic: true
        main:
            anonymous: true
```

Encodage du mot de passe

- Modifier l'encodeur

```
encoders:
    Symfony\Component\Security\Core\User\User: bcrypt
```

- Encoder les mots de passes

```
λ php bin/console security:encode-password
```

- Modifier les utilisateurs

```
providers:
    managers:
        memory:
            users:
                admin:
                    password: $2y$13$6P4TUj7dCWG.UNC/TO5ZYukhgaR3xtXXPpMVK16QA4MTcfHO4PzYS
                dieu:
                    password: $2y$13$cgTAIsjaCq.dFxemUOLjAuTCLJdeMZ6KS/gdfldgNYgeGIFUeYSnm
```

Restriction par le rôle

- Définir une hiérarchie

```
role_hierarchy:
    ROLE_ADMIN: ROLE_USER
    ROLE_SUPER_ADMIN: ROLE_ADMIN
```

- Définir le pare-feu

```
main:
    anonymous: true
    http_basic: true
```

- Définir les accès

```
access_control:
    - { path: ^/manager/users, roles: ROLE_SUPER_ADMIN }
    - { path: ^/manager, roles: ROLE_ADMIN }
```

- Configurer les utilisateurs

```
users:
    admin:
        password: $2y$13$6P4TUj7dCWG.UNC/TO5ZYukhgaR3xtXXPpMVK16QA4MTcfHO4PzYS
        roles: 'ROLE_ADMIN'
    dieu:
        password: $2y$13$cgTAIsjaCq.dFxemUOLjAuTCLJdeMZ6KS/gdfldgNYgeGIFUeYSnm
        roles: 'ROLE_SUPER_ADMIN'
```

Un formulaire de login

- Configurer le pare-feu

```yaml
main:
    anonymous: true
    form_login:
        login_path: login
        check_path: login
```

- Définir la route

```yaml
login:
    path: /login
    controller: App\Controller\UserController::login
```

- Définir le contrôleur

```php
public function login( AuthenticationUtils $authenticationUtils )
{
    return $this->render('user/login.html.twig', array(
        'lastUsername' => $authenticationUtils->getLastUsername(),
        'error' => $authenticationUtils->getLastAuthenticationError(),
    ));
}
```

- Le template

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

{% block body %}
{% if error %}
    <p>Erreur lors de la connexion</p>
{% endif %}

<form action="{{ path('login') }}" method="post">
    <input type="text" placeholder="Nom d'utilisateur"
        name="_username" value="{{ lastUsername }}" />

    <input type="password" placeholder="Mot de passe"
        name="_password" />

    <button type="submit">Login</button>
</form>
{% endblock %}
```

Gérer le déconnexion

- Définir la route

```
logout:
    path: /logout
```

- Configurer le pare-feu

```
main:
    anonymous: true
    form_login:
        login_path: login
        check_path: login
    logout:
        path: /logout
        target: /
```

La déconnexion ne fonctionne pas avec l'authentification HTTP simple

Utilisateurs en base de données

- Générer une entité
  - username
  - password
  - roles
- Implémenter l'interface

```
class User implements UserInterface
```

- Surcharge des méthodes

```
public function eraseCredentials(){}

public function getSalt(): ?string{
    return null;
}
```

- Configurer l'encodeur

```
encoders:
    App\Entity\User: bcrypt
```

- Configurer le fournisseur

```
providers:
    db_provider:
        entity:
            class: App\Entity\User
            property: username
```

- Configurer le pare-feu

```
main:
    anonymous: true
    provider: db_provider
    form_login:
        login_path: login
        check_path: login
```

Pratique

- Mettre en place un espace utilisateur
  - Formulaire de login
  - Formulaire d'inscription
  - Bouton de déconnexion
  - Restreindre l'accès à la page d'inscription à un évènement
  - Pré-remplir le champ du nom du participant avec le nom d'utilisateur

Exercices

- Créer un espace administrateur
  - Ajouter une catégorie
  - Ajouter un lieu
- Paginer la liste des évènements
- Donner la possibilité de commenter un événement
  - Après celui-ci, par un utilisateur ayant participé