

The background is a dark blue gradient. On the left, there is a large, semi-transparent circular image of a circuit board. Overlaid on this and the background are several geometric shapes: a blue parallelogram and a light green parallelogram in the upper left, and a series of white, 3D-looking circuit traces in the upper right.

Symfony

Concepts de base

TABLE DES MATIÈRES

Frameworks

Pattern Model - Vue - Contrôleur

Symfony

Introduction

Architecture

Routing

Controller

TWIG

Entités

Frameworks

- Framework = “Cadre de travail”

- **PHP**

- Symfony
- Laravel
- Code Igniter

- **Java**

- Java EE (ou J2EE)
- Spring Boot

- **Python**

- Django

- **C#**

- ASP .NET

- **Javascript**

- Angular
- VueJS
- React

- **HTML/CSS**

- Bootstrap
- Vuetify



Pourquoi utiliser un framework ?

Qu'est-ce qu'un framework ?

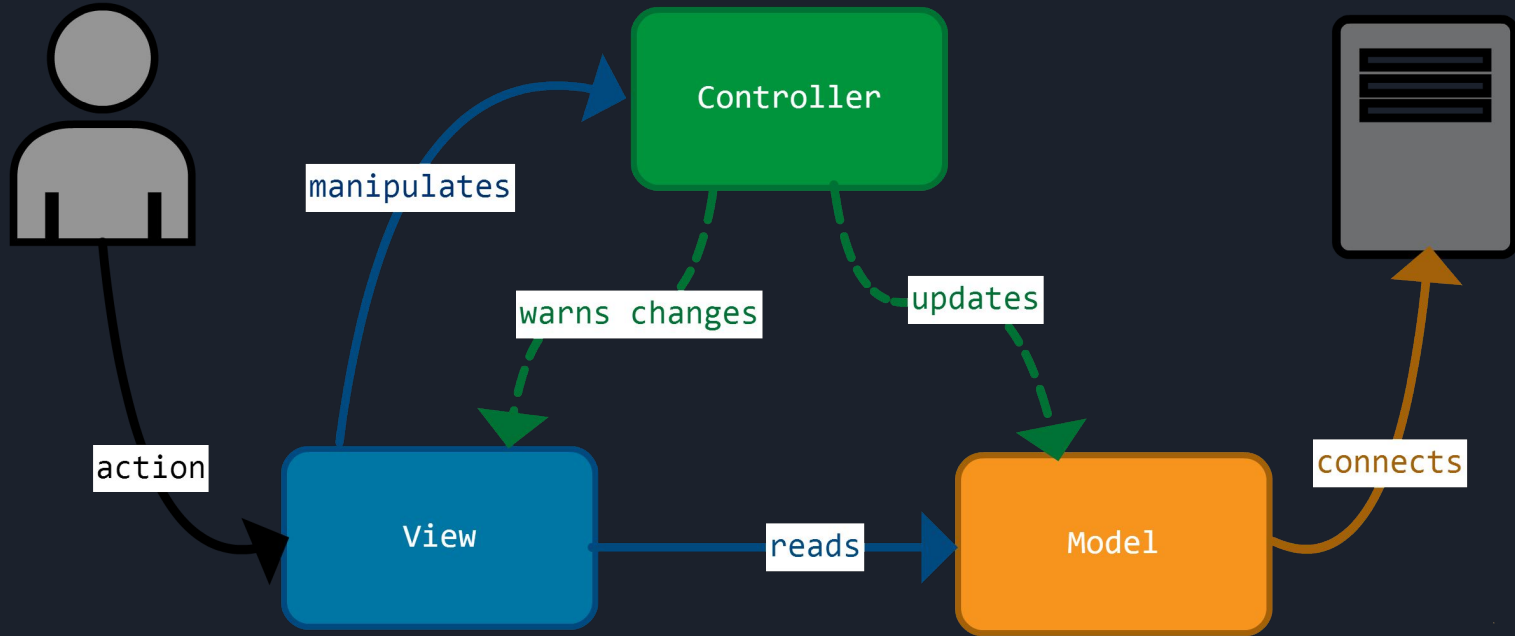
- Architecture applicative
- Ensemble d'outils
- Composants logiciels (bibliothèque logicielle réutilisable)
- Pratiques et normes de développement

L'objectif d'un framework

- Squelette
- Conception faite pour faciliter le travail du développeur



Pattern Model - Vue - Contrôleur





Symfony - One to rule them all

- Flexibilité
- Grandes performances
- Extensibilité
- Débogage
- Tests

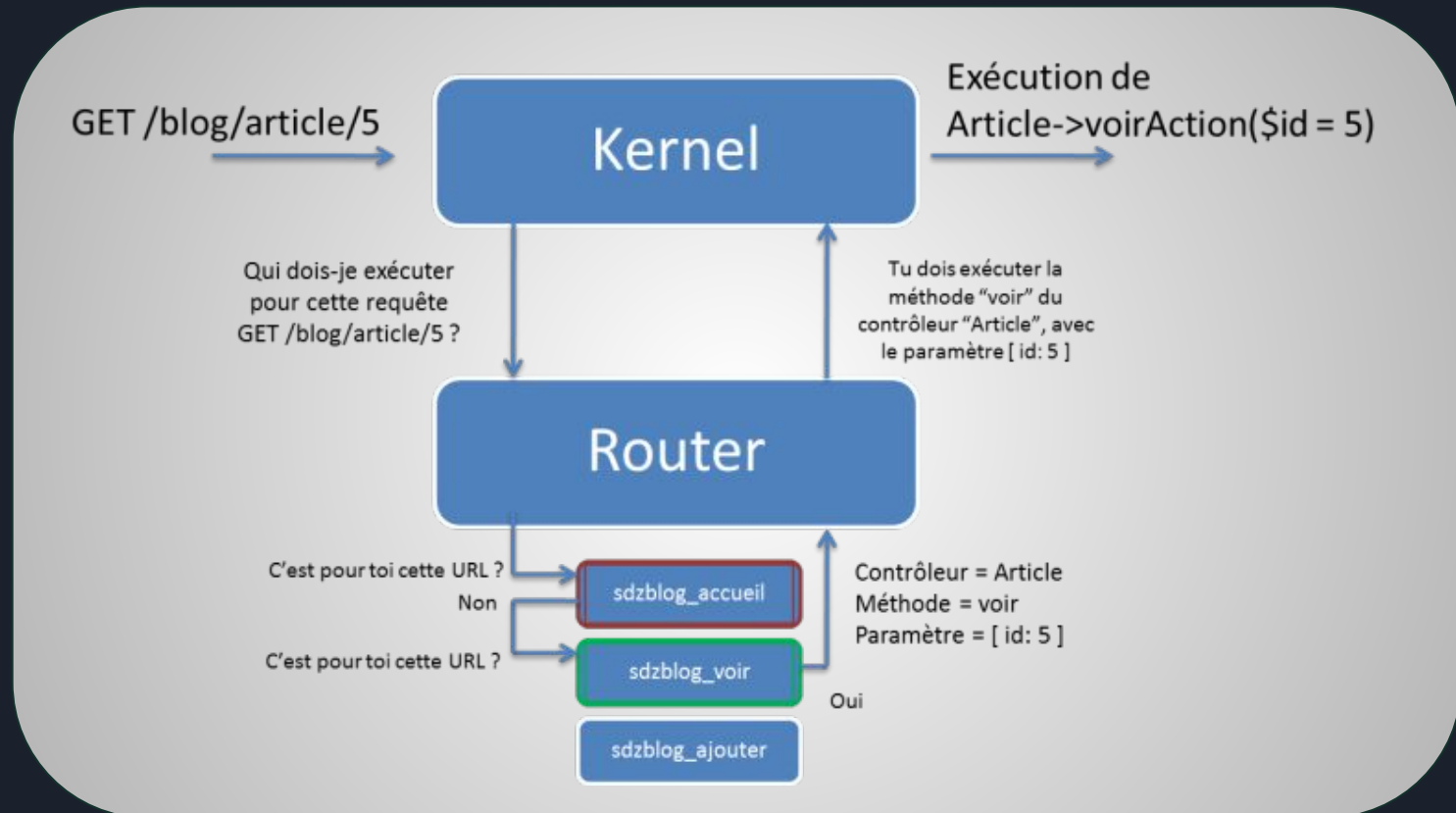




Introduction - Les points forts

- Abstraction de la base de données
- Couche d'abstraction du cache
- Gestion des formulaires
- Gestion des utilisateurs
- Internationalisation
- Moteur de template

Architecture - Fonctionnement simplifié





Architecture - Éléments de base

- ❖ Routes (YAML / Annotations / PHP)
- ❖ Controllers (PHP)
- ❖ Vues (TWIG)
- ❖ Entités (PHP)
- ❖ Formulaires (PHP)



Architecture - Arborescence

- ❖ bin/
- ❖ config/
- ❖ migrations/
- ❖ public/
- ❖ src/
- ❖ templates/
- ❖ var/
- ❖ vendor/
- ❖ composer.json
- ❖ .env
- ❖ .env.local



Routing - Intro

Une route doit être, au minimum, configurée avec :

- Le nom de la route
- L'URL

Configuration supplémentaire :

- Méthodes HTTP
- Possibilité d'adapter les paramètres selon la langue



Routing Annotations

```
class BlogController extends AbstractController
{

    #[Route("/blog", name="blog_list")]
    public function list()
    {
        // ...
    }

    #[Route("/blog/{slug}", name="blog_show")]
    public function show($slug)
    {
        // ...
    }
}
```



Routing - Paramétrage avancé

```
// src/Controller/BlogController.php
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;

class BlogController extends AbstractController
{
    /**
     * @Route("/blog/{page}", name="blog_list", requirements={"page"="\d+"})
     */
    public function list($page)
    {
        // ...
    }
}
```



Controllers - Intro

Un controller traite les requêtes.

Les controllers sont composés de :

- **un namespace**
- **une méthode par route**
 - **chaque méthode doit retourner une réponse**



Controllers - Exemple simple

```
// src/Controller/LuckyController.php

namespace App\Controller;

use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

class LuckyController
{
    #[Route("/Lucky/number/{max}", name="app_Lucky_number")]
    public function number($max)
    {
        $number = random_int(0, $max);

        return new Response(
            '<html><body>Lucky number: '.$number.'</body></html>'
        );
    }
}
```



Controllers - Redirections

```
use Symfony\Component\HttpFoundation\RedirectResponse;

// ...
public function index()
{
    // redirects to the "homepage" route
    return $this->redirectToRoute('homepage');

    // does a permanent - 301 redirect
    return $this->redirectToRoute('homepage', [], 301);

    // redirect to a route with parameters
    return $this->redirectToRoute('app_lucky_number', ['max' => 10]);

    // redirects externally
    return $this->redirect('http://symfony.com/doc');
}
```




Controllers - Exceptions

```
use Symfony\Component\HttpFoundation\Exception\NotFoundHttpException;

//...

public function index()
{
    $product = ...;
    if (!$product) {
        throw $this->createNotFoundException('Le produit n'existe pas');
    }
    return $this->render(...);
}
```



TWIG - Les vues

- Moteur de template
- Syntaxe simplifiée
- Filtres
- Héritage / Inclusion
- Méthodes



Une page en PHP

```
<!DOCTYPE html>
<html>
  <head>
    <title>Welcome to Symfony!</title>
  </head>
  <body>
    <h1><?= $page_title ?></h1>
    <ul id="navigation">
      <?php foreach ($navigation as $item): ?>
        <li><a href="<?= $item->getHref() ?>">
          <?= $item->getCaption() ?>
        </a>
      </li>
      <?php endforeach ?>
    </ul>
  </body>
</html>
```



L'équivalent en TWIG

```
<!DOCTYPE html>

<html>

  <head><title>Welcome to Symfony!</title></head>

  <body>

    <h1>{{ page_title }}</h1>

    <ul id="navigation">

      {% for item in navigation %}

        <li><a href="{{ item.href }}">{{ item.caption }}</a></li>

      {% endfor %}

    </ul>

  </body>

</html>
```



TWIG - Syntaxe

```
{% structure %}
```

```
{{ variable.propriete }}
```

```
{# commentaires #}
```

```
{{ variable|filtre }}
```

```
{{ variable }}
```

```
{{ methode(...) }}
```



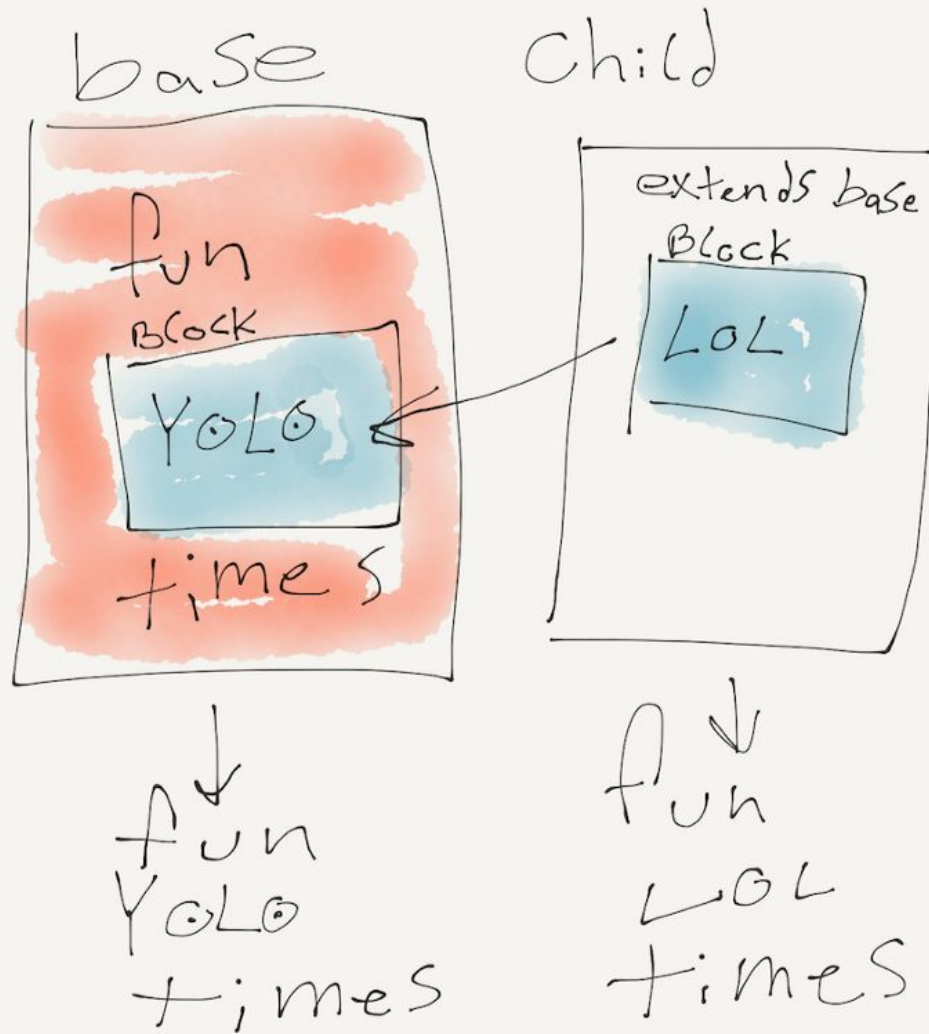
TWIG - Générer des urls

```
<a href="{{ path('welcome') }}">Home</a>
```

```
{% for article in articles %}  
  
    <a href="{{ path('article_show', {'slug': article.slug}) }}">  
        {{ article.title }}  
    </a>  
  
{% endfor %}
```

TWIG

Comprendre l'héritage





TWIG - Héritage - Vue mère

```
{# templates/base.html.twig #}  
<!DOCTYPE html>  
<html>  
  <head>  
    <meta charset="UTF-8">  
    <title>{% block title %}Bienvenue sur RIL.fr{% endblock %}</title>  
  </head>  
  <body>  
    <div id="sidebar">  
      {% block sidebar %}  
        <ul>  
          <li><a href="/">Accueil</a></li>  
        </ul>  
      {% endblock %}  
    </div>  
  
    <div id="content">  
      {% block body %}{% endblock %}  
    </div>  
  </body>  
</html>
```




TWIG - Héritage - Vue fille

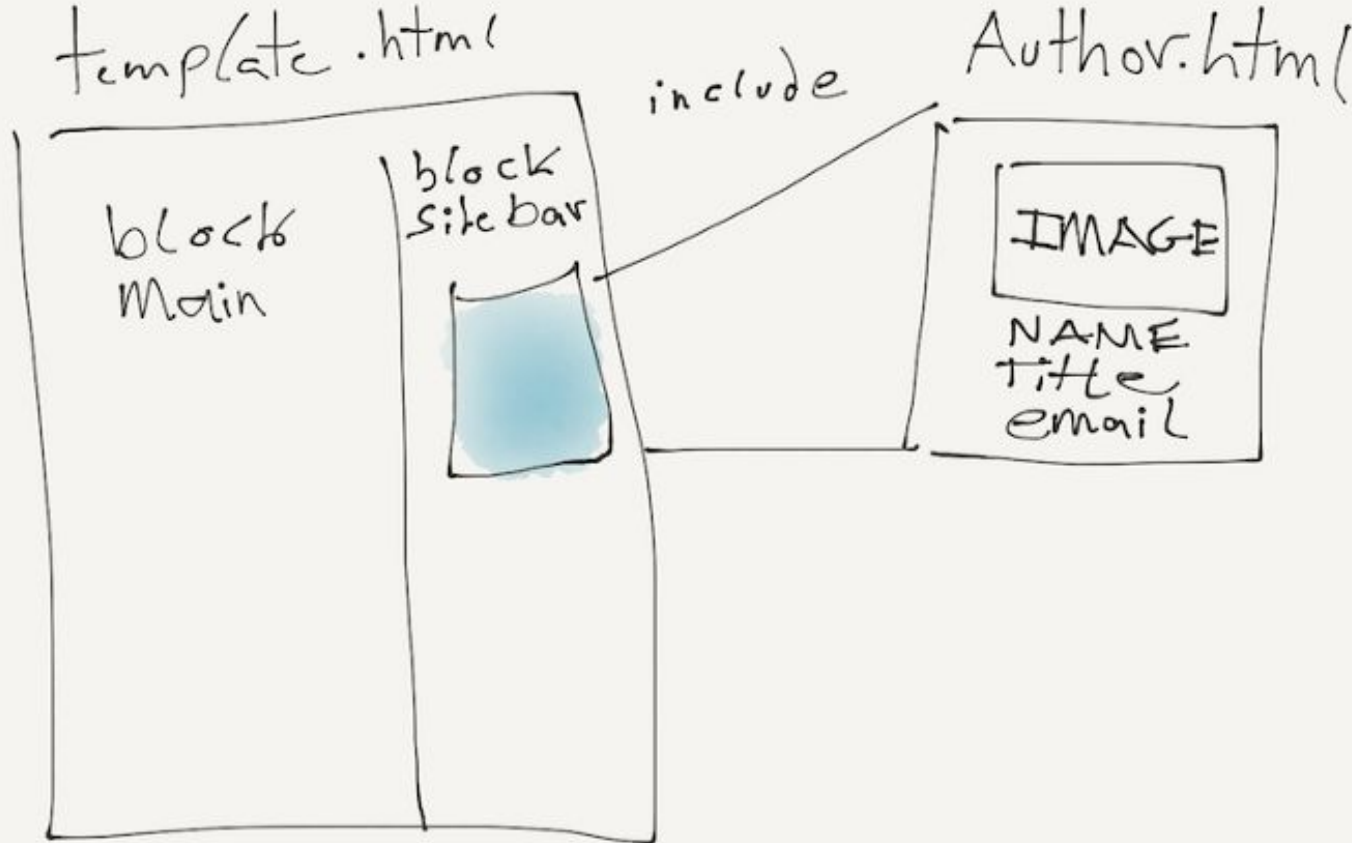
```
{# templates/blog/index.html.twig #}

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


{% block title %}Ma liste d'articles{% endblock %}


{% block body %}
    {% for article in articles %}
        <h2>{{ article.title }}</h2>
        <p>{{ article.body }}</p>
    {% endfor %}
{% endblock %}
```

TWIG - Comprendre l'inclusion





TWIG - Assets

Package Asset

```
composer require symfony/asset
```

Block des feuilles de style

```
{% block stylesheets %}  
    <link href="{{ asset('css/main.css') }}" rel="stylesheet"/>  
{% endblock %}
```

Insertion d'une image

```

```

Block de scripts JS

```
{% block javascripts %}  
    <script src="{{ asset('js/main.js') }}"></script>  
{% endblock %}
```



Entités


Composants métier

Utilisation de l'ORM (*Object Relational Mapping*) Doctrine

Utilisation améliorée de la commande (make:entity)

Relations entre entités

Contraintes des colonnes faciles à implémenter



Exemple d'entité

```
namespace App\Entity;
```

```
use App\Repository\ProductRepository;  
use Doctrine\ORM\Mapping as ORM;
```

```
#[ORM\Entity(repositoryClass: ProductRepository::class)]
```

```
class Product
```

```
{
```

```
    #[ORM\Id]
```

```
    #[ORM\GeneratedValue]
```

```
    #[ORM\Column]
```

```
    private ?int $id = null;
```

```
    #[ORM\Column(length: 255)]
```

```
    private ?string $name = null;
```

```
    #[ORM\Column]
```

```
    private ?int $price = null;
```

```
    public function getId(): ?int
```

```
{
```

```
        return $this->id;
```

```
}
```

```
}
```



Créer le schéma de base de données

Édition de l'entité ...

Lancement de la commande migration :

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

Fichier “intermédiaire” de migration créé : *src/Migrations/VersionYYYYMMDDHHIISS.php*

Mise à jour de la base de données :

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

Génération des getters et setters :

```
php bin/console make:entity --regenerate
```



Enregistrer une entité

```
// Gestionnaire des entités
$entityManager = $this->getDoctrine()->getManager();

$product = new Product();
$product->setName('Keyboard');
$product->setDescription('Ergonomic and stylish!');

// Indique à Doctrine de sauvegarder le produit
$entityManager->persist($product);

// Exécute les requêtes préparées
$entityManager->flush();

return new Response('Nouveau produit sauvegardé avec l'id : '.$product->getId());
```



Trouver une entité par son identifiant

```
$product = $this->getDoctrine()  
    ->getRepository(Product::class)  
    ->find($id);  
  
if (!$product) {  
    throw $this->createNotFoundException('Aucun produit trouvé');  
}  
  
return new Response('Voici le produit : '.$product->getName());
```




Trouver une entité par d'autres propriétés

```
$repository = $this->getDoctrine()->getRepository(Product::class);

// Récupère un produit avec son identifiant ('id')
$product = $repository->find($id);

// Récupère un seul produit à partir de son nom
$product = $repository->findOneBy(['name' => 'Keyboard']);

// Récupère plusieurs produit à partir de leur prix
$product = $repository->findBy([
    'price' => 100,
]);
```



Merci !



Des questions ?