TEMA 6

Introducción a Symfony 8

Arquitectura Moderna & Full Stack Development

Symfony 8

Framework Core + PHP 8.4

Docker

Nginx, PHP-FPM, Database

Doctrine

ORM & Database Abstraction

AssetMapper

Tailwind, Stimulus, Turbo

Novedades de Symfony 8

Configuración PHP Nativa

Adiós YAML/XML. Symfony 8 adopta arrays de PHP con "array shapes" para una configuración tipada, autocompletable y más rápida.

```
return [
  'framework' ⇒ [
    'secret' ⇒ '%env(APP_SECRET)%',
  ],
];
```

Atributos en Todo

Configuración descentralizada usando Atributos de PHP 8. Rutas, Seguridad, Entidades y Servicios se definen junto al código.

```
#[Route('/blog', name: 'blog_list')]
#[IsGranted('ROLE_USER')]
public function index(): Response
```

AssetMapper

Gestión de assets moderna sin Node.js ni Webpack. Usa estándares web (Import Maps) para cargar JS y CSS directamente en el navegador.

```
import { Controller } from '@hotwired/stimulus';
// Carga directa desde vendor/
```

Rendimiento Core

Requiere PHP 8.4+. Código base más ligero, eliminación de código legacy y optimizaciones profundas en el contenedor de servicios.

```
JIT Compiler Friendly
Type System robusto
```

Entorno de Desarrollo

Infraestructura (Docker)

Nginx
Servidor Web, Proxy Inverso

PHP-FPM 8.4
Intérprete PHP, Extensiones Symfony

Base de Datos
MySQL / PostgreSQL (Persistencia)

Symfony CLI & Composer

```
# 1. Crear proyecto nuevo
user@dev:~$ composer create-project symfony/skeleton my_app

# 2. Verificar requisitos del sistema
user@dev:~$ symfony check:requirements

# 3. Levantar servidor local (con TLS)
user@dev:~$ symfony server:start -d

# 4. Ejecutar comandos dentro del contenedor
user@dev:~$ docker compose exec php bin/console list
```

Estructura de Proyecto

- assets/
- bin/
- config/
- public/
- **src/**
- **templates/**
- tests/
- var/
- vendor/
- compose.yaml
- composer.json
- importmap.php
- symfony.lock

src/

PHP 8.4

Lógica de negocio. Contiene Controladores, Entidades, Repositorios y Servicios. Todo bajo el namespace `App\`.

templates/

Twig

Vistas de la aplicación. Archivos `.html.twig` que definen la estructura visual y componentes reutilizables.

assets/

AssetMapper

Frontend moderno sin Node.js. Archivos CSS, imágenes y controladores Stimulus servidos directamente.

config/

Config

Configuración de rutas, servicios, seguridad y paquetes. Centraliza el comportamiento del framework.

Controladores y Enrutamiento

src/Controller/BlogController.php — PHP 8.4

```php
<?php
namespace App\Controller;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractControlle
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Attribute\Route;
class BlogController extends AbstractController {
    #[Route('/blog/{slug}', name: 'blog_show')]
    public function show(string $slug, PostRepository $posts):
        $post = $posts->findOneBySlug($slug);
        if (!$post) { throw $this->createNotFoundException(); }
        return $this->render('blog/show.html.twig', ['post' =>
    }
}
```

@ Atributos PHP 8

Definimos la ruta `/blog/{slug}` en el método. Symfony extrae `$slug` de la URL.

Autowiring (DI)

Declaramos `PostRepository` como argumento; el contenedor lo inyecta automáticamente.

Objeto Response

Todo controlador devuelve `Response`. El método `render()` genera HTML usando Twig.

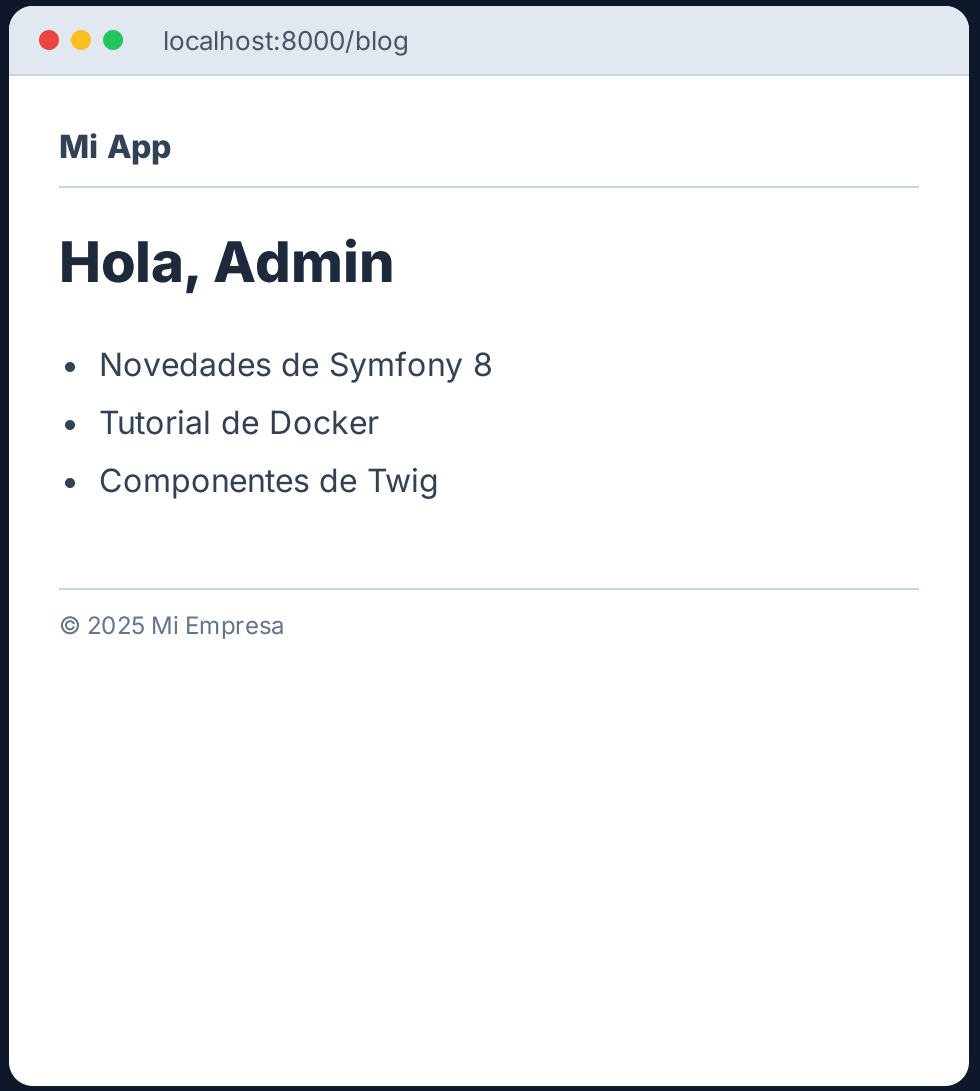
Vistas con Twig

templates/base.html.twig

```twig
<html>
  <body>
    <nav>Mi App</nav>
    {% block body %}{% endblock %}
    <footer>© 2025</footer>
  </body>
```

templates/blog/index.html.twig

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

{% block body %}
  <h1>Hola, {{ user.name }}</h1>

  <ul>
  {% for post in posts %}
    <li>{{ post.title }}</li>
  {% endfor %}
  </ul>
{% endblock %}
```



Datos con Doctrine ORM

src/Entity/Product.php

```php
#[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 ?float $price = null;

    // Getters & Setters ...
}
```

Table: product

id	INT (PK, AI)
name	VARCHAR(255)
price	DOUBLE

```
$ php bin/console make:entity
```

El comando interactivo genera la clase y los atributos automáticamente.

Consultas con Repositorios

Métodos Estándar

```php
// Buscar por ID
$product = $repository->find(1);

// Buscar por criterios
$user = $repository->findOneBy(['email' => 'admin@x.com']);

// Buscar todos
$all = $repository->findAll();
```

QueryBuilder (Personalizado)

```php
public function findExpensive($price)
{
    return $this->createQueryBuilder('p')
        ->andWhere('p.price > :val')
        ->setParameter('val', $price)
        ->orderBy('p.id', 'ASC')
        ->getQuery()
        ->getResult();
}
```

Métodos Mágicos

Doctrine ofrece métodos listos para usar para operaciones comunes: buscar por ID o campos simples sin SQL.

DQL vs SQL

QueryBuilder genera **DQL** (sobre *Objetos*), haciendo el código agnóstico de la base de datos.

`SELECT * FROM product WHERE ...`

Formularios en Symfony

1. Definición (UserType)

```php
public function buildForm(FormBuilderInterface $builder, array $options)
{
    $builder
        ->add('email', EmailType::class)
        ->add('password', PasswordType::class)
        ->add('save', SubmitType::class, [
            'label' => 'Registrar'
        ]);
}
```

Abstracción Total: Defines los campos y tipos en una clase PHP separada, reutilizable en cualquier parte.

2. Procesamiento (Controller)

```php
#[Route('/register', name: 'app_register')]
public function register(Request $request): Response
{
    $user = new User();
    $form = $this->createForm(UserType::class, $user);

    $form->handleRequest($request);

    if ($form->isSubmitted() && $form->isValid()) {
        // Guardar en BD...
        return $this->redirectToRoute('home');
    }

    return $this->render('register.html.twig', [
        'form' => $form
    ]);
}
```

Seguridad y Control de Acceso

Configuración Global

`security.yaml` define los **Firewalls** (autenticación) y el **Access Control** (autorización por URL).

config/packages/security.yaml

```yaml
security:
    firewalls:
        main:
            lazy: true
            provider: app_user_provider
            form_login:
                login_path: app_login
                check_path: app_login

    access_control:
        - { path: ^/admin, roles: ROLE_ADMIN }
        - { path: ^/profile, roles: ROLE_USER }
```

Lógica Granular (Voters)

Para permisos complejos (ej: "¿Es el dueño de este post?"), usamos clases **Voter**.

src/Security/PostVoter.php

```php
protected function voteOnAttribute($attribute, $post, $token): bool
{
    $user = $token->getUser();

    if (!$user instanceof User) {
        return false;
    }

    return $post->getAuthor() === $user;
}
```

Generación de Código (Maker Bundle)

`make:entity`
Crea o actualiza entidades Doctrine y repositorios.

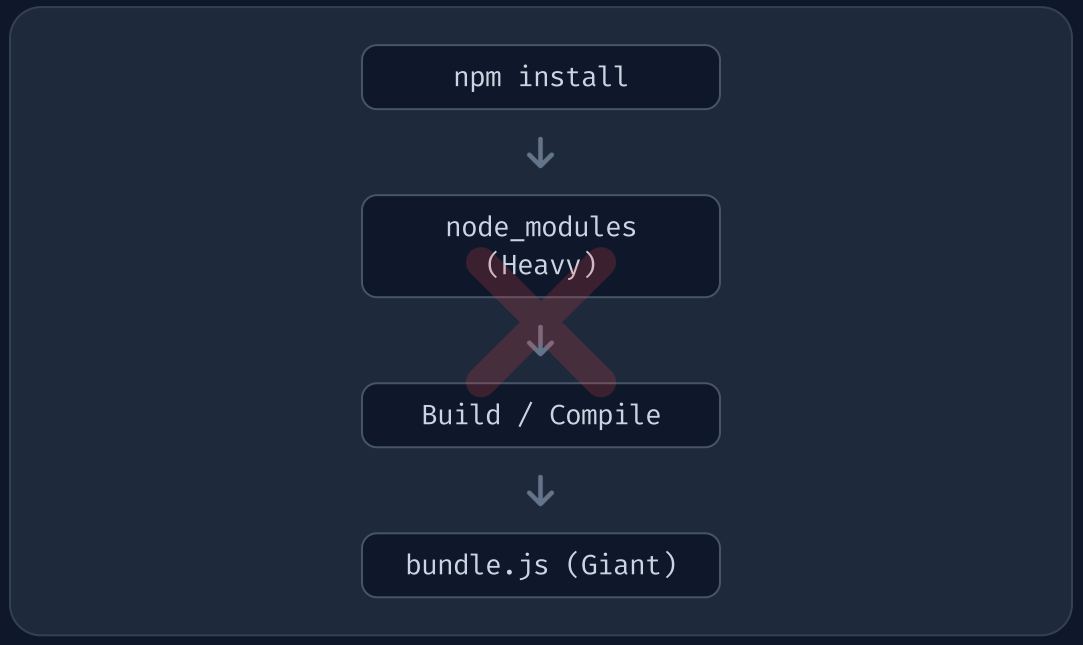
`make:controller`
Genera un controlador, ruta y plantilla Twig básica.

`make:form`
Crea una clase FormType basada en una entidad.

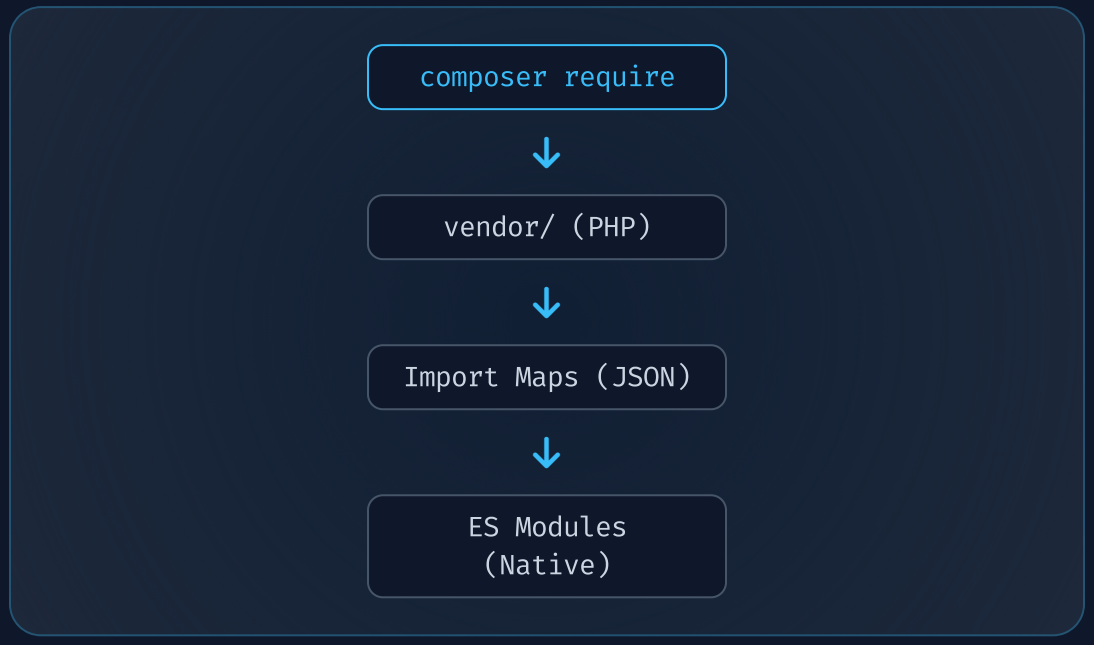
```
user@symfony:~$ php bin/console make:controller

 Choose a name for your controller class (e.g. OrangeGnomeController):
 > BlogController

 created: src/Controller/BlogController.php
 created: templates/blog/index.html.twig

 Success!

 Next: Open your new controller and add some pages!
```

Frontend sin Node.js: AssetMapper

Tradicional (Webpack)



Moderno (AssetMapper)



importmap.php

```php
return [
  'app' => ['path' => './assets/app.js', 'entrypoint' => true],
  '@hotwired/stimulus' => ['version' => '3.2.1'],
];
```

Estilos y Reactividad

Tailwind CSS

Estilado rápido usando clases utilitarias directamente en el HTML. Sin archivos CSS separados.

```
<button class="bg-blue-500 hover:bg-blue-700 text-white fo
  Click Me
</button>
```

Click Me

Stimulus Controllers

JavaScript modesto que conecta el HTML con el comportamiento mediante atributos `data-`.

```
<div data-controller="hello"></div>
```

```
import { Controller } from '@hotwired/stimulus';

export default class extends Controller {
  connect() {
    this.element.textContent = "Hello World!";
  }
}
```