

Práctica Profesionalizante I

Unidad 7

Introducción a Symfony



Agenda

- **Presentación**
- **Instalación**
- **Estructura de un proyecto**
- **Primera página**



Framework

Definición

Un framework es un conjunto estandarizado de:

- conceptos
- prácticas
- criterios



para resolver problemas nuevos de índole similar

Framework

Ventajas

- Ahorra tiempo
- Facilita el desarrollo colaborativo
- Mayor seguridad
- Gran cantidad de herramientas
- Buenas prácticas



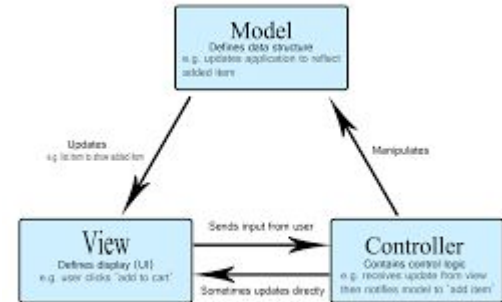
Symfony

- Framework php para aplicaciones web
- Utiliza patrón MVC
- Madurez en el mercado
- Gran comunidad de desarrolladores



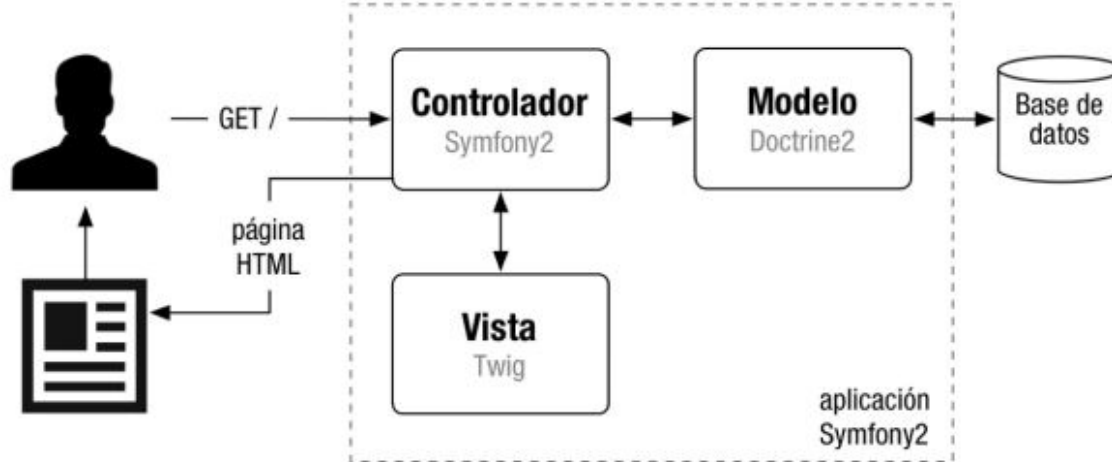
Patrón MVC

- Separa lógica de negocios de la interfaz
- Ayuda a la mantenibilidad y escalabilidad
- Divide la aplicación en tres niveles:
 - Modelo
 - Vista
 - Controlador



Symfony

Patrón MVC



Symfony

Instalación

```
composer create-project symfony/skeleton:"^6.4.2" myapp
```

```
cd myapp
```

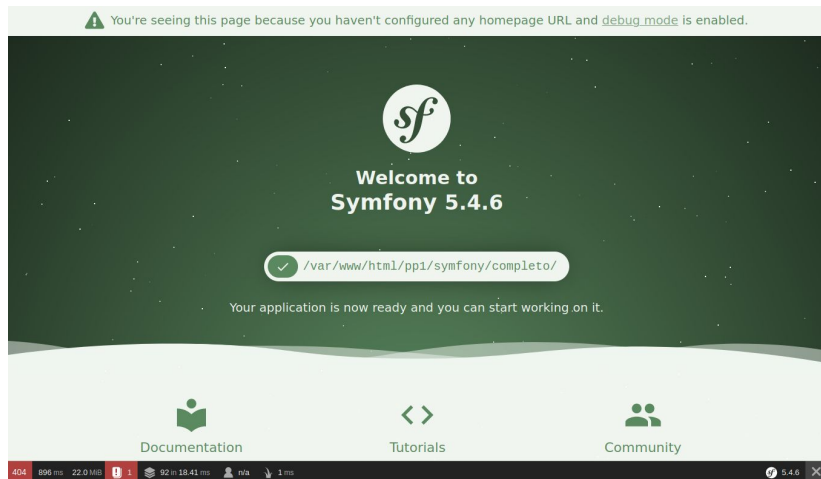
```
composer require symfony/webapp-pack
```

```
composer require symfony/apache-pack
```

Symfony Instalación

Para probar la instalación:

`http://localhost/myapp/public`

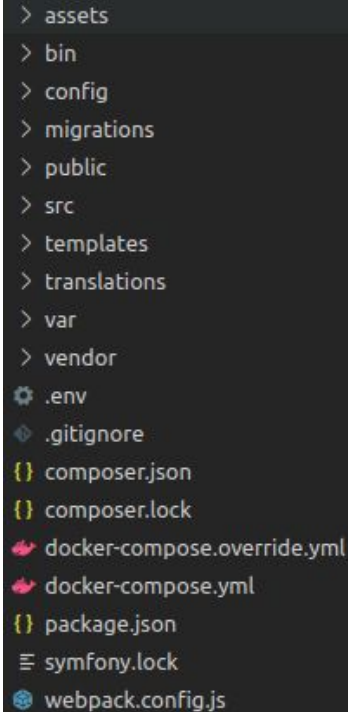


Práctica

Realizar el ejercicio 1 de la guía de trabajos prácticos

Symfony

Estructura de un proyecto

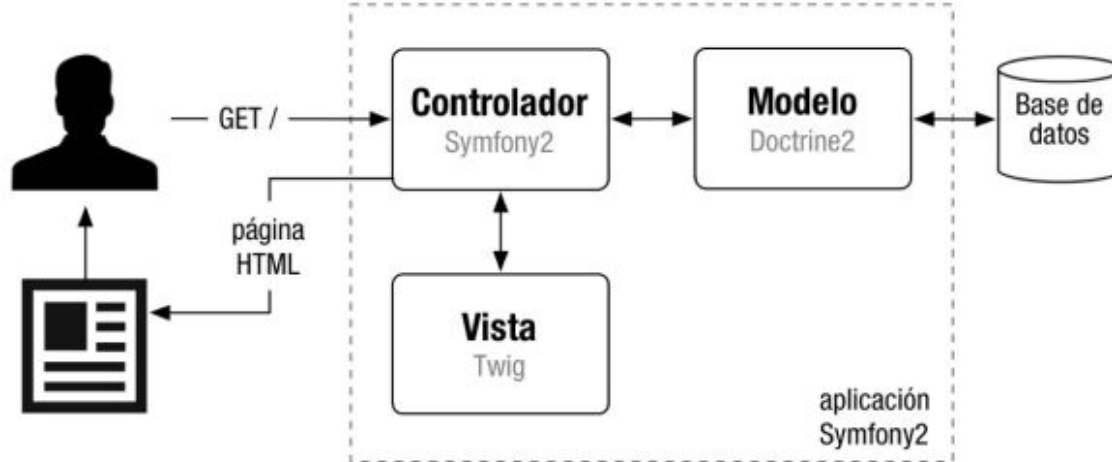


```
> assets
> bin
> config
> migrations
> public
> src
> templates
> translations
> var
> vendor
⚙ .env
🔒 .gitignore
{} composer.json
{} composer.lock
🐳 docker-compose.override.yml
🐳 docker-compose.yml
{} package.json
≡ symfony.lock
🌐 webpack.config.js
```

A screenshot of a file explorer window showing the directory structure of a Symfony project. The list includes directories like assets, bin, config, migrations, public, src, templates, translations, var, vendor, and files like .env, .gitignore, composer.json, composer.lock, docker-compose.override.yml, docker-compose.yml, package.json, symfony.lock, and webpack.config.js.

Primera página

Controlador



Primera página

Controlador

- Definir una ruta
 - Define la url de la página
 - Especifica el controlador a ejecutar
- Crear el controlador
 - Método php
 - Ejecuta la petición entrante
 - Devuelve un objeto Response al usuario

Primera página

Controlador

```
// src/Controller/ProductoController.php

<?php

namespace App\Controller;

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

class ProductoController extends AbstractController
{
    #[Route('/', name: 'listar_productos')] // http://localhost/pp1/carrito/public
    public function listarProductos(): Response
    {
        $html = '<html><body><b>Esta es la lista de productos</b></body></html>';
        return new Response($html);
    }
}
```

Primera página

Controlador

```
class ProductoController extends AbstractController
{
    #[Route('/', name: 'listar_productos')] // http://localhost/pp1/carrito/public
    public function listarProductos(): Response
    {
        $html = '<html><body><b>Esta es la lista de productos</b></body></html>';
        return new Response($html);
    }

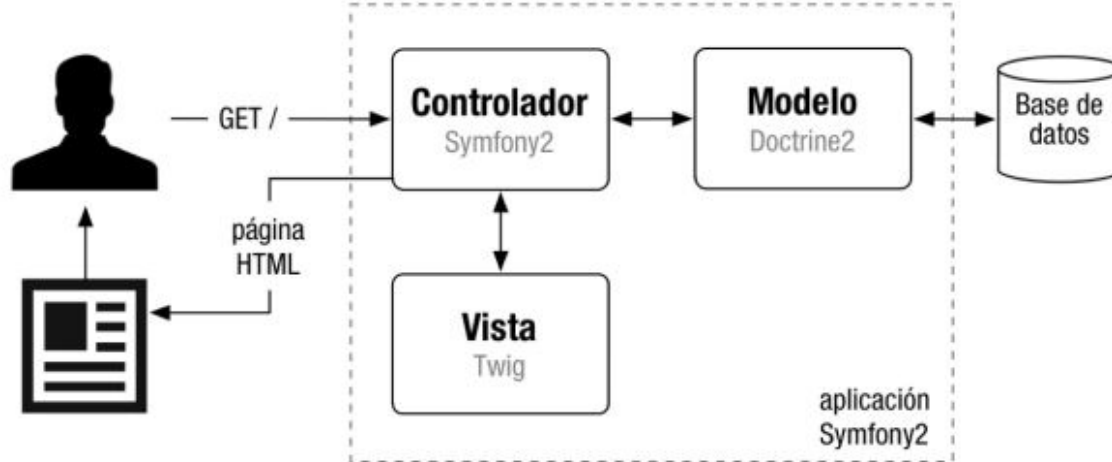
    #[Route('/producto', name: 'detalle_producto')] // http://localhost/pp1/carrito/public/producto
    public function detalleProducto(): Response
    {
        $html = '<html><body><b>Este es el detalle del producto</b></body></html>';
        return new Response($html);
    }
}
```


Práctica

Realizar el ejercicio 2 de la guía de trabajos prácticos

Primera página

Controlador



Primera página

Vista

```
# templates/producto/lista.html.twig
```

```
<html>
  <head>
    <meta charset="UTF-8">
    <title>Lista de productos</title>
  </head>
  <body>
    <h1>Lista de productos </h1>
    <p>Esta es la lista de productos</p>
  </body>
</html>
```

Primera página

Vista

```
// src/Controller/ProductoController.php

<?php

namespace App\Controller;

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

class ProductoController extends AbstractController
{
    #[Route('/', name: 'listar_productos')]
    public function listarProductos(): Response
    {
        return $this->render('producto/lista.html.twig');
    }
}
```

Práctica

Realizar el ejercicio 3 de la guía de trabajos prácticos

Rutas relativas

- Los templates usan imágenes, estilos, etc.
- Esos recursos se guardan en la carpeta public
- La función asset simplifica el armado de rutas

```

```

```
<link href="{{ asset('css/blog.css') }}" rel="stylesheet" />
```

Rutas relativas

```
# ./templates/base.html.twig
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>My app</title>
    <link rel="stylesheet" href="{{ asset('css/estilos.css') }}">
  </head>
  <body>
    {% block body %}{% endblock %}
  </body>
</html>
```

Práctica

Realizar el ejercicio 4 de la guía de trabajos prácticos