

Práctica Profesionalizante I

Unidad 13

Ruteo y controladores



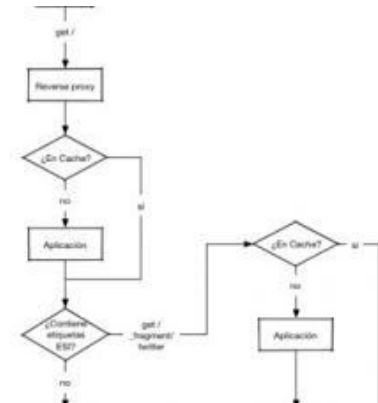
Agenda

- **Introducción**
- **Ruteo**
- **Controlador**
- **Servicios**
- **Objetos especiales**



Ruteo

- Determina el controlador a ejecutar
- URL amigables
- Consta de una url y un nombre
- Flexibilidad ante cambios



Ruteo

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

```
namespace App\Controller;
```

```
use Symfony\Component\HttpFoundation\Response;
```

```
use Symfony\Component\Routing\Annotation\Route;
```

```
class LuckyController
```

```
{
```

```
    #[Route('/lucky/number', name: 'app_lucky_number')]
```

```
    public function number(): Response
```

```
    {
```

```
        $number = random_int(0, 1000);
```

```
        return new Response(
```

```
            '<html><body>Lucky number: '.$number.'</body></html>'
```

```
        );
```

```
    }
```

```
}
```

Ruta



Nombre

www.misitio.com/lucky/number

Ruteo - Parámetros

Definen rutas variables

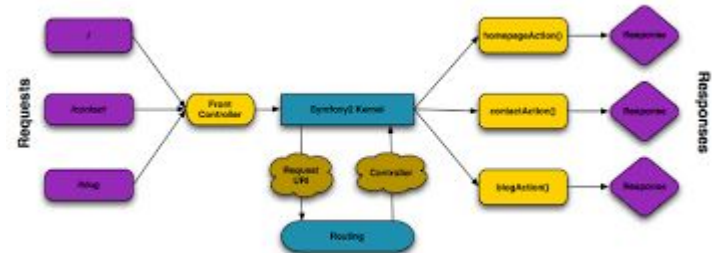
Ejemplo:

/lucky/number/{max}

Uso:

/lucky/number/1

/lucky/number/1234



Ruteo - Parámetros

Se crean variables que se pasan al controlador

```
// 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(int $max): Response
    {
        $number = random_int(0, $max);

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

Ruteo - Parámetros

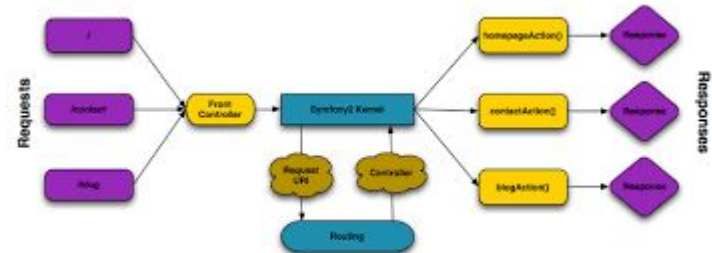
Varias variables en una ruta

Ejemplo:

/blog/posts-about-{category}/page/{pageNumber}

Uso:

/blog/posts-about-10/page/23



Práctica

Realizar la práctica 1 de la unidad



Ruteo

Se pueden restringir los métodos para las rutas

```
#[Route('/api/posts/{id}', methods: ['GET'])]
```

```
#[Route('/api/posts/{id}', methods: ['POST'])]
```

Ruteo - Orden de evaluación

```
class BlogController extends AbstractController
{
    /**
     * Matches /blog exactly
     */
    #[Route('/blog', name: 'blog_list')]
    public function list()
    {
        // ...
    }

    /**
     * Matches /blog/*
     */
    #[Route('/blog/{slug}', name: 'blog_show')]
    public function show($slug)
    {
        // ...
    }
}
```

Ruteo - Parámetros con prioridad

```
class BlogController extends AbstractController
{
    #[Route('/blog/{slug}', name: 'blog_show')]
    public function show(string $slug): Response
    {
        // ...
    }

    #[Route('/blog/list', name: 'blog_list')]
    public function list(): Response
    {
        // ...
    }
}
```

/blog/list - Entra en el primer controlador

Ruteo - Parámetros con prioridad

```
class BlogController extends AbstractController
{
  #[Route('/blog/{slug}', name: 'blog_show')]
  public function show(string $slug): Response
  {
    // ...
  }

  #[Route('/blog/list', name: 'blog_list', priority: 2)]
  public function list(): Response
  {
    // ...
  }
}
```

Ruteo - Validación de parámetros

```
class BlogController extends AbstractController
{
  #[Route('/blog/{page}', name: 'blog_list')]
  public function list(int $page): Response
  {
    // ...
  }

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

Ruteo - Validación de parámetros

```
class BlogController extends AbstractController
{
  #[Route('/blog/{page}', name: 'blog_list', requirements: ['page' => '\d+'])]
  public function list(int $page): Response
  {
    // ...
  }

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

Ruteo - Parámetros opcionales

```
class BlogController extends AbstractController
{
    #[Route('/blog/{page}', name: 'blog_list')]
    public function list(int $page = 1): Response
    {
        // ...
    }
}
```

Coincide con /blog y /blog/5

Ruteo - Generación de URL

```
class BlogController extends AbstractController
{
  #[Route('/blog/', name: 'blog_index')]
  public function index(): Response
  {
    // ...
  }

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

Ruteo - Generación de URL

```
class MainController extends AbstractController
{
    public function show($slug)
    {
        // /blog/my-blog-post

        $url = $this->generateUrl(
            'blog_show',
            array('slug' => 'my-blog-post')
        );
    }
}
```

Ruteo - Generación de URL

```
<a href="{{ path('blog_index') }}">Homepage</a>
```

```
{# ... #}
```

```
{% for post in blog_posts %}
```

```
  <h1>
```

```
  <a href="{{ path('blog_post', {slug: post.slug}) }}">{{ post.title }}</a>
```

```
  </h1>
```

```
  <p>{{ post.excerpt }}</p>
```

```
{% endfor %}
```

Práctica

Realizar la práctica 2 de la unidad



Práctica

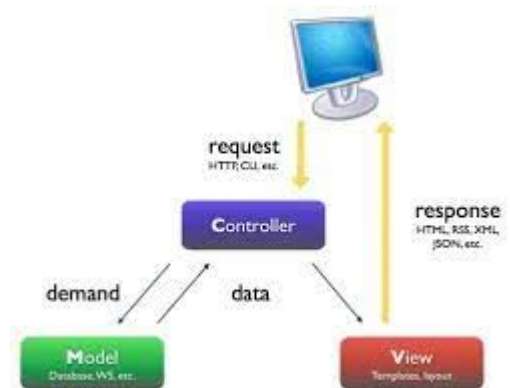
Realizar la práctica 3 de la unidad



Controlador

Es una función PHP

- Toma información del Request
- Construye un Response y lo devuelve
 - HTML
 - Json
 - Imagen



Controlador

```
// 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(int $max): Response
    {
        $number = random_int(0, $max);

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


Controlador

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

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

class ProductoController extends AbstractController
{
    #[Route('/', name: 'listar_productos')]
    public function listarProductos(ProductoManager $productoManager): Response
    {
        $productos = $productoManager->getProductos();

        return $this->render('producto/lista.html.twig', ['productos' =>
$productos]);
    }
}
```



Este método devuelve un Response

Procesamiento de parámetros - PHP

- Parámetros por URL - \$_GET

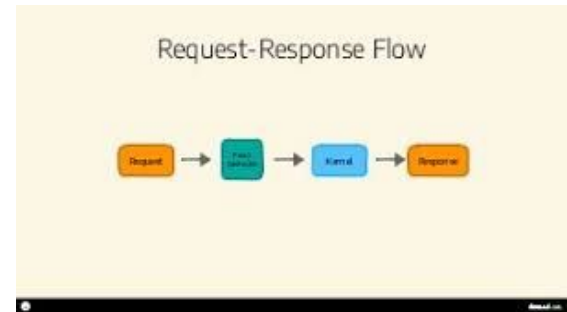
```
<a href="destino.php?saludo=hola&texto=Esto es una variable texto">Mi enlace</a>
```

- Parámetros por formularios - \$_POST

```
<body>  
  <form method="post" action="destino2.php">  
    Nombre<br>  
    <input type="text" name="nombres"><br>  
    Apellidos<br>  
    <input type="text" name="apellido"><br>  
    <input type="submit">  
  </form>  
</body>
```

Controlador Objeto Request

- Encapsula la petición del usuario
- Contiene el valor de las variables
 - \$_GET
 - \$_POST
 - \$_SERVER
 - \$_FILES



Objeto Request

```
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;

public function index(Request $request): Response
{
    // retrieves GET variables
    $request->query->get('saludo'); // $_GET['saludo']

    // retrieves POST variables
    $request->request->get('nombres'); // $_POST['nombres']

    // retrieves SERVER variables
    $request->server->get('HTTP_HOST');

    // retrieves an instance of UploadedFile identified by foo
    $request->files->get('foo');

    // retrieves an HTTP request header, with normalized, lowercase keys
    $request->headers->get('host');
    $request->headers->get('content-type');
}
```

Objeto Request

Más información:

https://symfony.com/doc/6.4/components/http_foundation.html#request

Controlador Objeto Response

- El controlador debe devolver un objeto Response
- Contiene lo que debe mandarse al cliente

```
#[Route("/nolucky", name="app_no_lucky")]  
public function noLucky(): Response  
{  
    return new Response('<html><body>No has tenido  
suerte</body></html>');  
}
```

Objeto Response

```
#[Route("/nolucky", name="app_no_lucky")]  
public function noLucky(): Response  
{  
    $respuesta = new Response();  
    $respuesta->setContent('<html><body>No has tenido  
suerte</body></html>');  
    $respuesta->setCodeStatus(Response::HTTP_OK);  
    return $respuesta;  
}
```

Objeto Response

```
#[Route("/lucky", name="app_lucky")]
public function lucky(): Response
{
    $number = random_int(0, 100);
    return $this->render('lucky/lucky.html.twig', ['numero' => $number]);
}
```

Objeto Response

```
public function download(): Response
{
    // send the file contents and force the browser to download it
    return $this->file('/path/to/some_file.pdf');
}
```

Más información:

https://symfony.com/doc/6.4/components/http_foundation.html#response

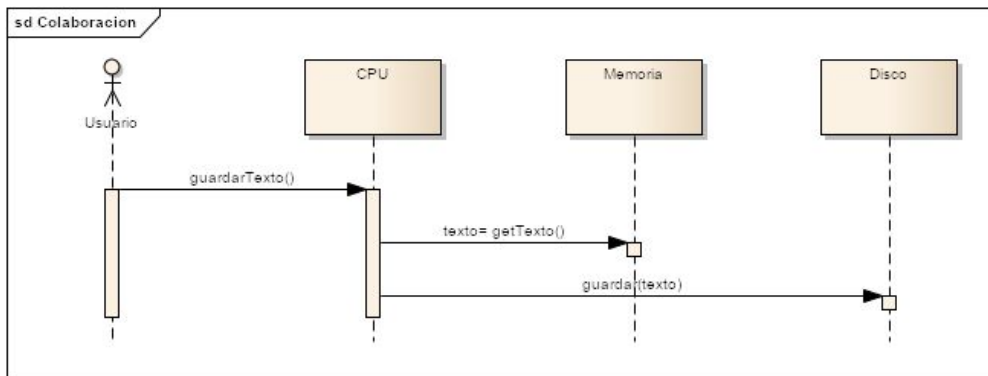
Práctica

Realizar la práctica 4 de la unidad



Inyección de dependencias

- Los objetos colaboran entre sí
- Esto crea dependencias entre objetos



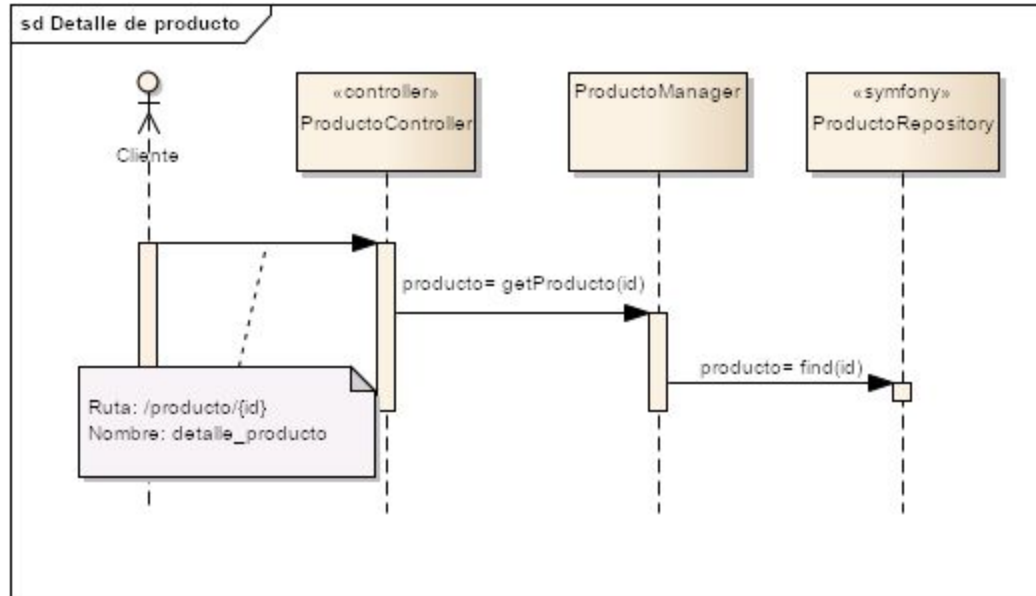
```
<?php
include('cpu.php');
include('memoria.php');
include('disco.php');

$cpu = new CPU();
$memoria = new Memoria();
$disco = new Disco();

$cpu->guardarTexto($memoria, $disco);

echo 'Texto guardado';
?>
```

Inyección de dependencias




- Cuándo se crean los objetos?

Inyección de dependencias

- El controlador puede necesitar de otros objetos
- Se agrega el objeto como parámetro
- Symfony crea el objeto y se lo pasa al controlador
- Es obligatorio tipificar el parámetro

Service Container. Implementation

Symfony 

Dynamic Service Locator

```
public function some()  
{  
    $this->container->get('logger');  
}
```

Inyección de dependencias

```
class ProductoController extends AbstractController
{
    #[Route('/producto/{id}', name: 'detalle_producto')]

    public function detalleProducto(ProductoManager $productoManager, int $id): Response
    {
        $producto = $productoManager->getProducto($id);

        return $this->render('producto/detalle.html.twig', ['producto' => $producto]);
    }
}
```

Inyección de dependencias

```
class ProductoManager {  
  
    private $repository;  
  
    public function __construct(ProductoRepository $repository) {  
        $this->repository = $repository;  
    }  
  
    public function getProductos() {  
        return $this->repository->findAll();  
    }  
  
    public function getProducto(int $id) {  
        return $this->repository->find($id);  
    }  
}
```

Servicios

- Symfony provee muchas funcionalidades
- Estas funcionalidades se llaman servicios
 - envio de mail
 - logueo
 - acceso a base de datos
- Podemos inyectar esos servicios



Servicios

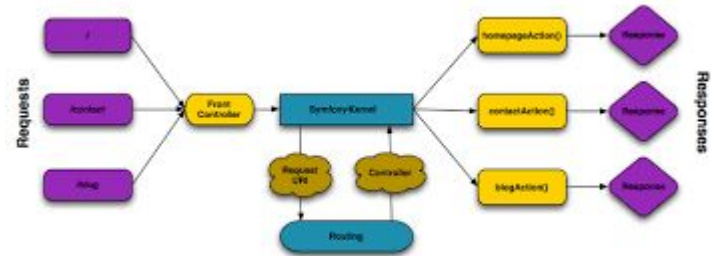
```
use Psr\Log\LoggerInterface;
use Symfony\Component\HttpFoundation\Response;
// ...

#[Route('/lucky', name: 'index')]

public function number(LoggerInterface $logger): Response
{
    $logger->info('We are logging!');
    // ...
}
```


Controlador base

- AbstractController
- Suministra numerosos helpers
 - render
 - generateUrl
 - redirectToRoute
 - redirect



Controlador base

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

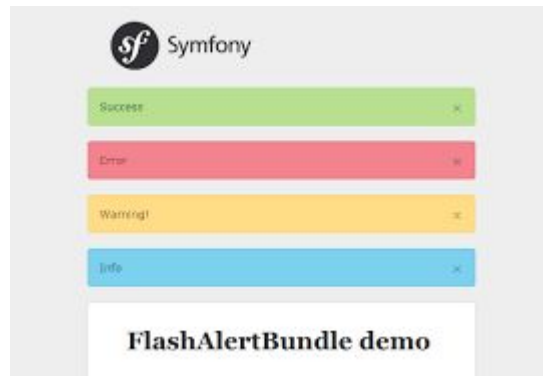
// ...
public function index(): RedirectResponse
{
    // redirects to the "homepage" route
    return $this->redirectToRoute('homepage');

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

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

Controlador Mensajes Flash

- Utilidad para mostrar mensajes al usuario
- Se crean y se pueden utilizar una sola vez



Mensajes Flash

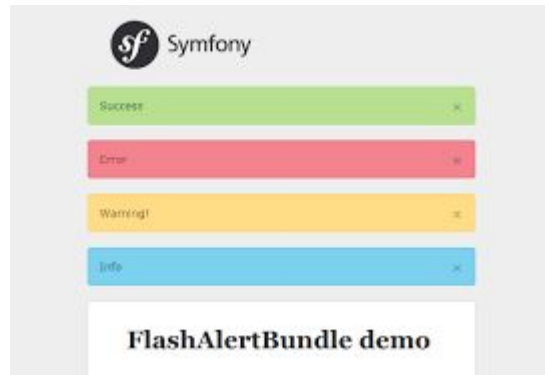
```
public function update(Request $request): Response
{
    // .... lógica del update
    // .... cuando se termina el update

    $this->addFlash(
        'notice',
        'Los cambios se guardaron correctamente'
    );
    $this->addFlash(
        'notice',
        'Por favor vuelva a ingresar al sistema'
    );

    return $this->render(...);
}
```

Mensajes Flash

```
{% for message in app.flashes('notice') %}  
    <div class="flash-notice">  
        {{ message }}  
    </div>  
{% endfor %}
```



Práctica

Realizar la práctica 5 de la unidad

