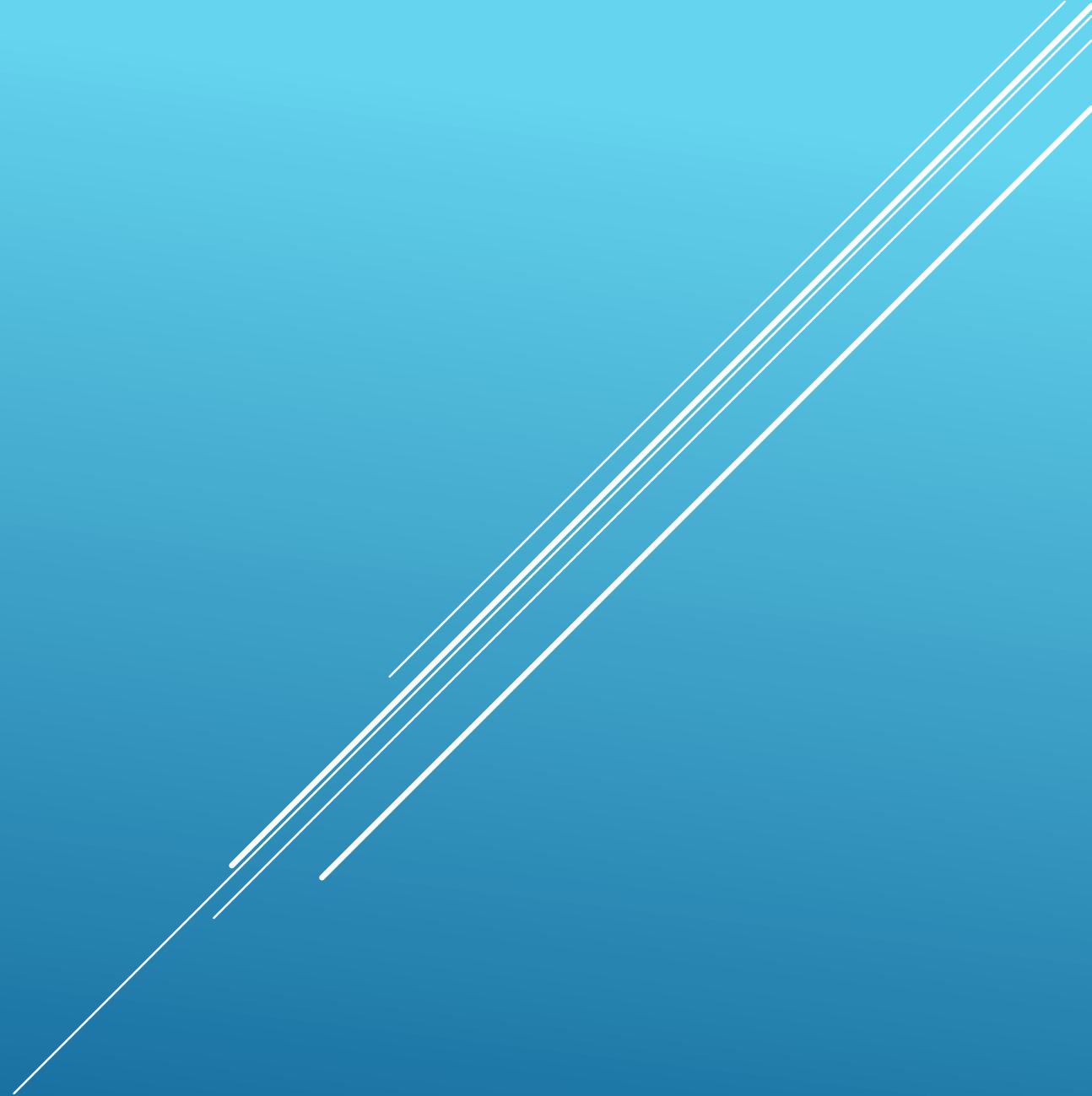


PLANTILLAS

MOTOR TWIG



INTRODUCCIÓN

Una plantilla es una mezcla de texto, y código php utilizada para representar la información.

El lenguaje que incluye symfony es twig, que permite escribirlas de forma legible y concisa, y este lenguaje está preparado para trabajar con plantillas, utilizar variables, sentencias de control, etc.

Como hemos visto en los controladores anteriores, se debe devolver una respuesta que nosotros, hasta ahora, hemos hecho con un fragmento de código html. Esta opción no es la más adecuada.

INTRODUCCIÓN

Lo ideal es definir las páginas por separado(vistas), y desde los controladores redireccionar a las vistas, pasándoles los datos necesarios para mostrarlos.

Para ello utilizaremos el método render, que nos permite indicar la plantilla a la que queremos dirigir la petición y un array asociativo con los datos que el controlador desea enviarla.

En nuestro controller podemos ver que se importa la clase:

```
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController
```

La cual se extiende a la que acabamos de crear en nuestro controller mediante:

```
extends AbstractController
```

Esta sintaxis nos permite incluir dentro de ella la renderización y otros helpers necesarios que iremos viendo.

INTRODUCCIÓN

En **la última línea** se ve la renderización y en ella se invoca la plantilla llamada nombrefichero.html.twig.

El nombre de las plantillas tiene dos extensiones el primero hace alusión al formato del documento y el segundo al motor utilizado para la plantilla.

La extensión twig indica que la plantilla está ejecutada por el motor de la plantilla Symfony twig. Y como se puede observar se le esta pasando un valor, que es el que debe mostrar.

```
=<?php
```

```
namespace App\Controller;
```

```
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
```

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

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

```
class TestController extends AbstractController
```

```
{
```

```
    #[Route('/test', name: 'app_test')]
    public function index(): Response
```

```
    {
        return $this->render('test/index.html.twig', [
            'controller_name' => 'TestController',
        ]);
    }
}
```

Hello TestController!

This friendly message is coming from:

- Your controller at `C:/wamp64/www/miAppSymfony/s`
- Your template at `C:/wamp64/www/miAppSymfony/te`

Como podemos ver estamos pasando a la plantilla ,mediante el índice asociativo 'controller_name', el valor que se visualiza que es 'TestController'

SINTÁXIS DE TWIG EN LA PLANTILLA

Twig tiene tres tipos especiales de sintaxis:

- `{{ ... }}` "Dice algo": imprime una variable o el resultado de una expresión en la plantilla.
 - `{% ... %}` "Hace algo": una etiqueta que controla la lógica de la plantilla; Se utiliza para ejecutar sentencias como bucles y condicionales.
 - `{# ... #}` "Comentar algo": Se utiliza para agregar comentarios de una o varias líneas. El contenido de los comentarios no se incluye en las páginas representadas.
- Twig también contiene filtros , que modifican el contenido antes de ser renderizado. `{{title | upper}}`.

SINTÁXIS DE TWIG EN LA PLANTILLA

La plantilla se debe ubicar en el directorio templates/**nombredirectorio**.

Que por convenio, el nombredirectorio se debe llamar igual que el del controlador pero en minúsculas.

La plantilla deberá tener un contenido similar a:

```
{% extends 'base.html.twig' %}  
{% block body %}  
Texto a visualizar {{valor}}  
{% endblock %}
```


SINTÁXIS DE TWIG EN LA PLANTILLA

La primera línea sirve para especificar que esta plantilla , utiliza otra plantilla que se usará como la plantilla base donde vamos a reescribir (Layout). Con la etiqueta `extends` extendemos la plantilla base. Es decir `base.html.twig` se usará como estructura común de otras muchas plantillas para evitar repetir código html.

El resto del código es un bloque en el que se imprime un texto y el valor pasado desde el controlador.

La herencia de plantillas nos ahorra escribir código repetido en otras, de manera que podremos inyectar este código desde las plantillas hijas con sus valores específicos.

Twig nos permite reescribir por medio de Bloques (`block`), de manera que extendiendo la plantilla base la heredamos, y solo variamos lo que necesitamos.

La plantilla base tendrá el aspecto siguiente:

SINTÁXIS DE TWIG EN LA PLANTILLA

Como vemos se limita a crear un esqueleto de un archivo html.

Crea varios elementos block para que las plantillas que la hereden puedan escribir sus hojas de estilo, titulo, su contenido de body, etc.

De esta manera solo se redefinen las partes que sean necesarias.

También nos permite **recoger el código** del bloque definido en la plantilla padre, para eso se utiliza **el método parent()**.

La plantilla index(dentro del directorio recién creado) tendrá el aspecto siguiente:

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

{% block title %}Hello TestController!{% endblock %}

{% block body %}
<style>
    .example-wrapper { margin: 1em auto; max-width: 800px; width: 95%; font: 18px/1.5 sans-serif; }
    .example-wrapper code { background: #F5F5F5; padding: 2px 6px; }
</style>

<div class="example-wrapper">
    <h1>Hello {{ controller_name }}! ☐</h1>

    This friendly message is coming from:
    <ul>
        <li>Your controller at <code>C:/wamp64/www/miAppSymfony/src/Controller/TestController.php</code></li>
        <li>Your template at <code>C:/wamp64/www/miAppSymfony/templates/test/index.html.twig</code></li>
    </ul>
</div>
{% endblock %}
```

EJEMPLO

Vamos a crear una vista para nuestro controlador gobernada por la ruta /hola, el controlador quedará modificado, en lugar de responder con Response, se debe hacer con render, y quedará :

```
#[Route('/hola/{edad}/{nombre}', name: 'app_hola', requirements:['nombre'=>'[a-zA-Z]{2,50}'])]  
public function hola($nombre,int $edad): Response  
{  
    $nomb=$nombre;  
    Return $this->render('test/hola.html.twig', ['nombre'=>$nombre,'edad'=>$edad ]);  
}
```

EJEMPLO

La vista se llamará `hola.html.twig`, y estará en la carpeta `templates/test`, tal y como vemos en el controlador

Solo vamos a poner un `title` que diga : primera plantilla.

Y en el `body` vamos a dar la bienvenida con el nombre y la edad que vamos a pasar en el controlador en un `h2`.

La plantilla sería:

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

{% block title %}primera plantilla{% endblock %}

{% block body %}|

<h2>Bienvenido {{nombre}} tienes {{edad}} años y es tu primera plantilla twig </h2>

{% endblock %}
```



localhost/miAppSymfony/public/index.php/hola/12/Ana

Bienvenido Ana tienes 12 años y es tu primera plantilla twig

CONCEPTOS BÁSICOS DE TWIG:

BUCLE FOR

Recorre cada elemento de una secuencia, esta secuencia puede ser tanto indeterminada, como una secuencia de número o de letras determinadas, y lo hace mediante el operador “..”

<h1>SECUENCIA INDETERMINADA </h1>

{% for user in USUARIOS%

{{ user}}

{% endfor %}

<h1>SECUENCIA DETERMINADA NUMERICA</h1>

{% for i in 0..10 %}

* {{ i }}

{% endfor %}

<h1>SECUENCIA DETERMINADA ALFABETICA</h1>

{% for letter in 'a'..'z' %}

{{ letter }}

{% endfor %}

El operador .. puede tomar cualquier expresión en ambos lados:

{% for letter in 'a' | upper..'z' | upper %} → Convierte en mayúsculas todo el abecedario

{{ letter }}

{% endfor %}

Donde el valor lo da la variable que esta después del in y lo recoge la variable de antes de l in

CONCEPTOS BÁSICOS DE TWIG:

BUCLE FOR(VBLE LOOP)

Dentro de un bloque de bucle for puedes acceder a algunas variables especiales:

Variable	Descripción
<code>loop.index</code>	La iteración actual del bucle. (indexada en 1)
<code>loop.index0</code>	La iteración actual del bucle. (indexada en 0)
<code>loop.revindex</code>	El número de iteraciones a partir del final del bucle (indexadas en 1)
<code>loop.revindex0</code>	El número de iteraciones a partir del final del bucle (indexadas en 0)
<code>loop.first</code>	True si es la primera iteración
<code>loop.last</code>	True si es la última iteración
<code>loop.length</code>	El número de elementos en la secuencia

CONCEPTOS BÁSICOS DE TWIG: BUCLE FOR(VBLE LOOP)

Las variables `loop.length`, `loop.revindex`, `loop.revindex0` y `loop.last` únicamente están disponibles para arrays PHP, u objetos que implementen la interfaz `Countable`. Tampoco están disponibles cuando iteras con una condición.

TWIG :BUCLE FOR CLAUSULA ELSE

La cláusula else en un bucle se ejecuta si no se llevó a cabo la iteración debido a que la secuencia esté vacía, puedes reproducir un bloque sustituto utilizando else:

```
<ul>
{% for user in users %}
    <li>{{ user}}</li>
{% else %}
    <li><em>usuario no encontrado</em></li>
{% endfor %}
</ul>
```

TWIG :BUCLE FOR(ITERANDO EN LAS CLAVES)

De forma predeterminada, un bucle itera con los valores de la secuencia. Puedes iterar en las claves con el filtro keys:

```
<h1>Members</h1>
<ul>
{% for key in users | keys %}
<li>{{ key }}</li>
{% endfor %}
</ul>
```

TWIG :BUCLE FOR(ITERANDO EN CLAVES Y VALORES)

También puedes acceder tanto a las claves como a los valores

```
<h1>Members</h1>
<ul>
{% for key, user in users %}
<li>{{ key }}: {{ user }}</li>
{% endfor %}
</ul>
```

TWIG : IF

Es comparable con las declaraciones if de PHP.

En la forma más simple:

```
{% if variable == true %}  
<p>entro por aquí.</p>  
{% endif %}
```

Para ramificación múltiple puedes utilizar elseif y else como en PHP

```
{% if vble==1 %}  
    si vale 1 por aquí.  
{% elseif vble==2 %}  
    si vale 2 por aquí  
{% else %}  
    cualquier otro valor por aquí  
{% endif %}
```

EJEMPLO-GUIADO-1

Vamos a definir una vista que imprimirá las asignaturas que hay en 2º de DAW desde el controlador que generamos en nuestro proyecto

Desde el controlador que tenemos generado:

Debemos definir:

1. La ruta que va a gobernar el acto de visualizar las asignaturas
2. El array de asignaturas que son los datos que quiero visualizar mediante la vista, y se los paso en el render.
3. Se visualizarán dentro de una lista ordenada

Los Módulos de 2 DAW son:

1. DWEC
2. DES
3. DWES
4. ENTORNOS GRÁFICOS
5. EIE
6. INGLÉS

CONTROLADOR:

El controlador quedará:

```
#[Route('/asignaturas', name: 'app_asignaturas')]  
public function modulo(): Response  
{  
    $asignatura=array('dwec','des','dwes','entornos gráficos','eie','inglés');  
    Return $this->render('asignatura/index.html.twig', array('keyasignatura'=>$asignatura ));  
}
```

PLANTILLA

La plantilla será:

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

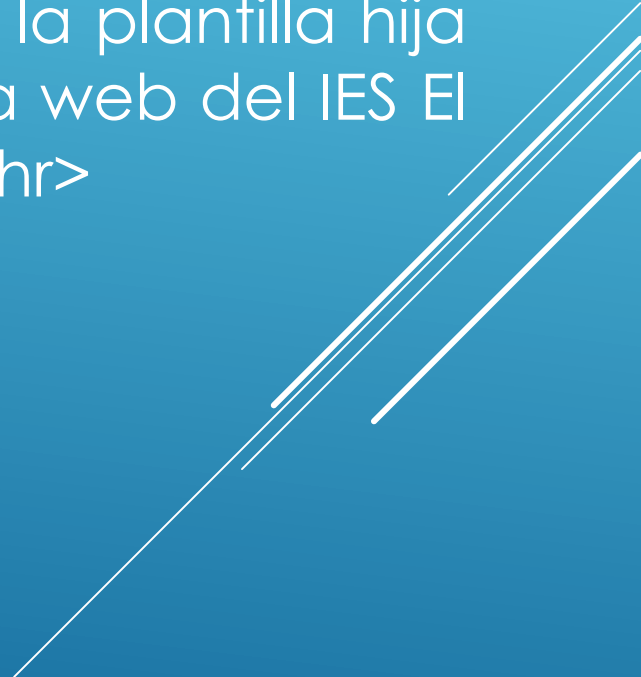
{% block title %}ASIGNATURAS{% endblock %}

{% block body %}
<h1>Los Módulos de 2 DAW son:</h1>
<ol>{#va a imprimir los módulos en mayúsculas#}
{%for valor in keysignatura%}{#se pasa el índice del array#}
<li>{{ valor|upper}}</li>
{%endfor%}

{% endblock %}
```

EJEMPLO GUIADO-2

Del ejercicio anterior, generar una plantilla base nueva, a la que añadimos un bloque footer que va a contener únicamente el texto «**para más información**» y un `<hr>`, vamos a hacer que la plantilla hija lo herede y esta última añada un enlace de la página web del IES El Cañaveral debajo de la línea que genera la etiqueta `<hr>`

Several white lines of varying lengths and angles are drawn in the bottom right corner of the slide, creating a modern, abstract graphic element.

Los Módulos de 2 DAW son:

1. DWEC
2. DES
3. DWES
4. ENTORNOS GRÁFICOS
5. EIE
6. INGLÉS

Para más información:

[EL CAÑAVERAL](#)

EJEMPLO GUIADO-2

Para ello en la plantilla **base** de templates añadimos el bloque de footer que quiero que hereden todas mis plantillas, para que a partir de este footer modifiquen lo imprescindible en cada layout final.

Several white lines of varying lengths and angles are drawn in the bottom right corner of the slide, creating a modern, abstract graphic element.

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>{% block title %}Welcome!{% endblock %}</title>
    <link rel="icon" href="data:image/svg+xml,<svg xmlns=%22http://www
    {% block stylesheets %}
    {% endblock %}

    {% block javascripts %}
      {% block importmap %}{{ importmap('app') }}{% endblock %}
    {% endblock %}
  </head>
  <body>
    {% block body %}{% endblock %}
  </body>
</html>

```

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>{% block title %}Welcome!{% endblock %}</title>
    <link rel="icon" href="data:image/svg+xml,<svg xmlns=%22http://www.w3
    {% block stylesheets %}
    {% endblock %}

    {% block javascripts %}
      {% block importmap %}{{ importmap('app') }}{% endblock %}
    {% endblock %}
  </head>
  <body>
    {% block body %}{% endblock %}
    {% block footer %} Para más información: <hr>{% endblock %}
  </body>
</html>

```

HEREDA DE LA NUEVA PLANTILLA Y REESCRIBE EL FOOTER

Esto se consigue añadiendo lo ya existente mediante `{{parent()}}`

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

{% block title %}ASIGNATURAS{% endblock %}

{% block body %}
    <h1>Los Módulos de 2 DAW son:</h1>
    <ol>{#va a imprimir los módulos en mayúsculas#}
    {%for valor in keysignatura%}{#se pasa el índice del array#}
        <li>{{ valor|upper}}</li>
    {%endfor%}
{% endblock %}

{% block footer %}
    {{parent()}}{#hace que herede lo que hay en la plantilla base#}
    <span><a href=www.educa.madrid.org/web/iescanaveral"> EL CAÑAVERAL</span></a>
{% endblock %}
```


REFERENCIAS A RECURSOS

- El componente *Asset* nos permite generar URLs que nos dirijan a recursos estáticos de nuestra aplicación web como hojas de estilos CSS, archivos JavaScript o archivos de imágenes, así como controlar las distintas versiones de estos archivos.
- Para mejorar la presentación y funcionabilidad se utilizan recursos estáticos (hojas estilos, imágenes, ficheros javascript, etc), symfony recomienda copiar los recursos (css, js, jpg, etc.) dentro de la carpeta **public** donde generaremos los directorios donde guardaremos los recursos que queramos utilizar.

REFERENCIAS A RECURSOS

Lo que debemos es reescribir en la plantilla base los estilos que queremos aplicar y que las hereden las plantillas hijas que dependen de ella.

```
<head>
```

```
{% block stylesheets %}
```

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

```
{% endblock %}
```

REFERENCIAS A RECURSOS

En public/css/estilos.css estará el archivo que da esta vista del listado.

Los Módulos de 2 DAW son:

1. DWEC
2. DES
3. DWES
4. ENTORNOS GRÁFICOS
5. EIE
6. INGLÉS

Para más información:

[EL CAÑAVERAL](#)

BASE.HTML.TWIG

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>{% block title %}Welcome!{% endblock %}</title>

    {% block stylesheets %}

      <link href="{{ asset('css/estilos.css') }}" rel="stylesheet" type="text/css"/>
    {% endblock %}

    {% block javascripts %}
      {% block importmap %}{{ importmap('app') }}{% endblock %}
    {% endblock %}
  </head>
  <body>
    {% block body %}{% endblock %}
    {% block footer %} <p>Para más información: <hr></p>{% endblock %}
  </body>
</html>_
```

INDEX.HTML.TWIG Y PUBLIC/CSS/ESTILOS.CSS

```
body{background-color:#33ff8d;}  
h1, p{color:#3355ff;}
```

```
{% extends 'base.html.twig' %}  
  
{% block title %}ASIGNATURAS{% endblock %}  
{% block stylesheets %}  
    {{parent()}}_  
{% endblock %}  
{% block body %}  
    <h1>Los Módulos de 2 DAW son:</h1>  
    <ol>{#va a imprimir los módulos en mayúsculas#}  
        {%for valor in keyasignatura%}{#se pasa el índice del array#}  
            <li>{{ valor|upper}}</li>  
        {%endfor%}  
{% endblock %}  
{% block footer %}  
    {{parent()}}{#hace que herede lo que hay en la plantilla base#}  
    <span><a href=www.educa.madrid.org/web/iescanaveral"> EL CAÑAVERAL</span></a>  
{% endblock %}
```

IMÁGENES

Para introducir imágenes sería análogo a lo anterior, en el body introducimos la ruta de nuestro recurso:

```
{% block body %}  
  
{% endblock %}
```

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>{% block title %}Welcome!{% endblock %}</title>

    {% block stylesheets %}

        <link href="{{ asset('css/estilos.css') }}" rel="stylesheet" type="text/css"/>
    {% endblock %}

    {% block javascripts %}
        {% block importmap %}{{ importmap('app') }}{% endblock %}
    {% endblock %}
  </head>
  <body>
    {% block body %}
  {% endblock %}
    {% block footer %} <p>Para más información: <hr></p>{% endblock %}
  </body>
</html>

```

En la plantilla base ya tengo los estilos, y ahora Inserto en el body la imagen para que lo herede todo la plantilla hijas

```

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

{% block title %}ASIGNATURAS{% endblock %}
{% block stylesheets %}
    {{parent()}}
{% endblock %}
{% block body %}
    {{parent()}} <h1>Los Módulos de 2 DAW son:</h1>
    <ol>{#va a imprimir los módulos en mayúsculas#}
    {%for valor in keysignatura%}{#se pasa el índice del array#}
        <li>{{ valor|upper}}</li>
    {%endfor%}
{% endblock %}
{% block footer %}
    {{parent()}}{#hace que herede lo que hay en la plantilla base#}
    <span><a href=www.educa.madrid.org/web/iescanaveral"> EL CAÑAVERAL</span></a>{
{% endblock %}

```

Plantilla Index con parent() ha heredando de la plantilla base los estilos, la imagen en el body y el footer.



Los Módulos de 2 DAW son:

1. DWEC
2. DES
3. DWES
4. ENTORNOS GRÁFICOS
5. EIE
6. INGLÉS

Para más información:

[EL CAÑAVERAL](#)


EJERCICIOS:

Todos los elementos comunes de las pantallas visualizadas deberán heredarse de la plantilla base, tanto en contenido como en estilos.

Several thin, white, parallel diagonal lines are positioned in the bottom right corner of the slide, extending from the right edge towards the center.

EJERCICIOS:

Del ejercicio de los números aleatorios visualízalos generando una plantilla que los visualice en forma de tabla, de manera que tanto color del texto naranja (h2) como el resto de colores (caption, td, y valores) serán heredado de la plantilla base, tal y como se muestra en el pantallazo:



Plantillas Symfony:

Números Aleatorios DAW:

TABLA INFORMATIVA

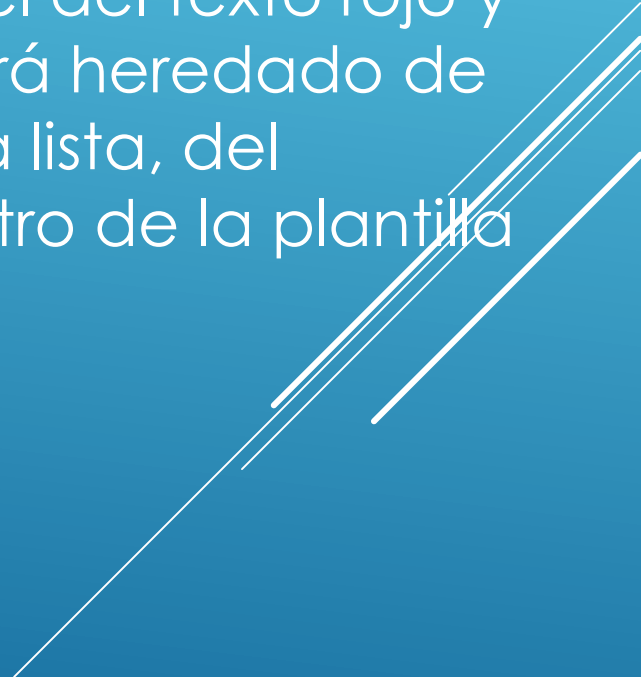
NUMERO	LIMITE INFERIOR	LIMITE SUPERIOR
4	3	6

Para más información:

IES EL CAÑAVERAL CURSO DE SYMFONY PARA 2 DAW

EJERCICIOS:

Del ejercicio del calculo del año de nacimiento, visualiza los resultados generando una plantilla donde se presenten los datos en forma de lista ordenada, de manera que el color de fondo del body, el del texto rojo y naranja correspondientes a h1 y h2 respectivamente, será heredado de la platilla base, y los colores de los elementos pares de la lista, del elemento impar y del fondo de a lista, se reescribirá dentro de la plantilla hija, tal y como se muestra en el pantallazo.

Several white lines of varying lengths and angles are drawn in the bottom right corner of the slide, creating a modern, abstract graphic element.

Plantillas Symfony:

Año de nacimiento

Lista de datos introducidos:

1. Año Actual:2024
2. Año de Nacimiento:2014
3. Edad:10

Para más información:

IES EL CAÑAVERAL CURSO DE SYMFONY PARA 2 DAW

EJERCICIOS:

Del ejercicio del cálculo de las áreas de los polígonos, visualiza los resultados generando una plantilla donde se presenten los datos en forma de tabla de manera que:

La plantilla base se incorporará tanto los estilos de h1, como un footer que constará solo de un `<hr>`

En la plantilla específica(hija) se describirá la tabla que muestra los resultados, y se aplicará color al caption ,y en su footer se añadirá el texto tal y como se muestra:

Plantillas Symfony:

Área de Poligonos

Tabla de datos
introducidos:

poligono	base	altura	área
cuadrado	2	2	4

Para más información:

IES EL CAÑAVERAL CURSO DE SYMFONY PARA 2 DAW