



UD-09: Introducción a Symfony

Desarrollo Web en Entorno Servidor

Curso 2020/2021



Introducción

- Conocer el lenguaje PHP nos ofrece muchas posibilidades, aunque nos deja también algo desprovistos de guía.
- Una de las principales críticas que se hacen de PHP es que su código puede llegar a ser bastante caótico si no se siguen ciertas pautas.
- Por este motivo entran en juego los frameworks, que proporcionan herramientas que ayudan a automatizar ciertas partes del proceso de creación de software.
- En el caso de los frameworks PHP, ayudan también a crear una estructura guiada para desarrollar la aplicación de una forma ordenada.
- De entre todos los frameworks posibles, elegimos Symfony por ser uno de los más populares en la actualidad, y por ofrecer una orientación muy sólida hacia el patrón MVC con el que se construyen buena parte de las aplicaciones web hoy en día.

Introducción

- Un framework es una herramienta que proporciona una serie de módulos que ayudan a organizar y desarrollar un producto software.
- En el caso concreto de los frameworks PHP, la mayoría de ellos proporcionan una serie de comandos o herramientas para crear proyectos con una estructura determinada (normalmente, siguiendo el patrón MVC), de forma que ya dan una base de trabajo hecha, y facilidades para poder crear el modelo de datos, la conexión a la base de datos, las rutas de las diferentes secciones de la aplicación, etc.



Introducción

Actualmente existe una gran variedad de frameworks PHP que elegir para desarrollar nuestras aplicaciones. Algunos de los más populares son:

- **Laravel** ha ganado bastante popularidad en los últimos años. Busca desarrollar proyectos de forma elegante y simple. Cuenta con una amplia comunidad detrás.
- **Symfony** cuenta con más camino hecho que Laravel, y una estructura más consolidada. Está muy orientado al patrón MVC, y a una estructura rígida y correcta de hacer aplicaciones.
- **CodeIgniter** es más ligero que los anteriores, pero también con un amplio grupo de seguidores. Aunque sufrió una etapa de abandono, ha vuelto a coger fuerza en los últimos años, quizá por su simplicidad de uso.
- **CakePHP** es otro framework similar a CodeIgniter en cuanto a simplicidad y facilidad de uso. Tiene una amplia comunidad también detrás que le da soporte.
- **Zend** es otro framework popular, aunque quizá con menor visibilidad que los anteriores hoy en día.



Introducción

- Casi todos los frameworks PHP tienen una serie de características comunes:
 - ✓ Uso del patrón MVC para desarrollar sus proyectos
 - ✓ Inyección de dependencias para gestionar recursos, tales como conexiones a bases de datos, o elementos compartidos por toda la aplicación
 - ✓ La posibilidad de desarrollar tanto webs completas como servicios REST accesibles desde diversos clientes, etc.
- A la hora de decantarnos por uno u otro framework, no nos deberíamos dejar engañar por la popularidad del mismo, en términos de cuota de mercado.
- En ese terreno, Symfony y Laravel probablemente sean los más beneficiados, pero la curva de aprendizaje en ellos puede que sea más pronunciada que en otros a priori más sencillos, como CodeIgniter o CakePHP.

Introducción

- Symfony constituye un enorme conjunto de herramientas y utilidades que simplifican el desarrollo de las aplicaciones web.
- Es uno de los frameworks PHP más populares entre los usuarios y las empresas, ya que permite que los programadores sean mucho más productivos a la vez que crean código de más calidad y más fácil de mantener.
- Emplea el tradicional patrón de diseño MVC (modelo-vista-controlador) para separar las distintas partes que forman una aplicación web.
 - ✓ El modelo representa la información con la que trabaja la aplicación y se encarga de acceder a los datos.
 - ✓ La vista transforma la información obtenida por el modelo en las páginas web a las que acceden los usuarios.
 - ✓ El controlador es quien coordina todos los demás elementos y transformar las peticiones del usuario en operaciones sobre el modelo y la vista.



Instalando Symfony a través de Composer

- Para empezar a trabajar con el framework Symfony, su propia web ofrece algunas alternativas, pero recomienda instalar Symfony a través de la herramienta Composer, que es un gestor de dependencias que permite instalar distintos módulos o librerías en un proyecto.
- Composer contiene una BD online con muchas librerías disponibles centralizadas, con lo que podemos indicar cuál/es queremos para cada proyecto concreto, y Composer las descarga e instala por nosotros.
- Es algo muy similar a otros gestores como NPM (Node Package Manager), que se aplica a librerías para Node o Javascript en general.



Instalando Symfony a través de Composer

- Composer puede instalarse localmente para cada proyecto web, o de forma global para todo el sistema, que es la opción más recomendable en el caso de querer gestionar varios proyectos en nuestro equipo.
- Para hacer esto último, lo primero que haremos es descargar Composer en el directorio actual

```
php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
```

- Verificar el instalador SHA-384.

```
php -r "if (hash_file('sha384', 'composer-setup.php') ===  
'756890a4488ce9024fc62c56153228907f1545c228516cbf63f885e036d37e9a59d27d63f46af1d4d07ee0f76181c7d3') { echo  
'Installer verified'; } else { echo 'Installer corrupt'; unlink('composer-setup.php'); } echo PHP_EOL;"
```

- Ejecutar el instalador

```
php composer-setup.php
```

- Quitar el instalador

```
php -r "unlink('composer-setup.php');"
```




Instalando Symfony a través de Composer

- Una vez descargado Composer, moveremos el archivo descargado (composer.phar) a /usr/local/bin, con el nombre "composer", para que al ejecutarlo lo encuentre en el PATH del sistema.

```
sudo mv composer.phar /usr/local/bin/composer
```

- Podemos comprobar la versión instalada de Composer mediante:

```
composer --version
```

```
miguel@miguel-VirtualBox: ~  
Archivo Editar Ver Buscar Terminal Ayuda  
miguel@miguel-VirtualBox:~$ composer --version  
Composer version 2.0.8 2020-12-03 17:20:38  
miguel@miguel-VirtualBox:~$
```

- En este punto es posible que nos pida habilitar la carpeta temporal en *php.ini*. Solo tendrás que buscar la línea indicada y descomentarla.



Instalando Symfony a través de Composer

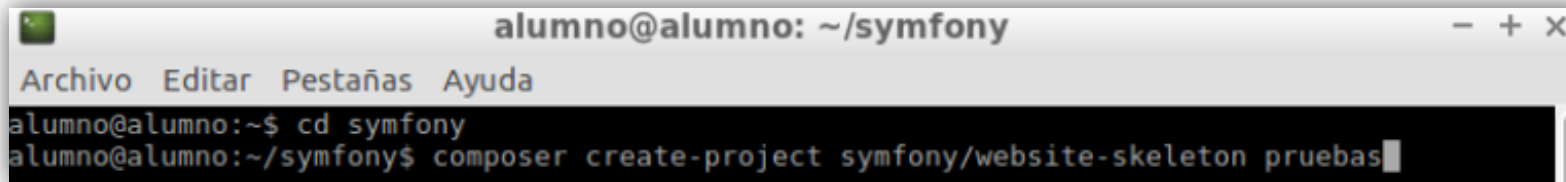
- Para crear un proyecto Symfony, podemos descargar directamente el framework e instalarlo en nuestra carpeta de aplicación web, o bien utilizar Composer (*opción recomendada*), en cuyo caso existen dos alternativas:
 - ✓ La primera opción es escribir en consola lo siguiente:

```
composer create-project symfony/skeleton nombreproyecto
```
 - ✓ La segunda (*la que usaremos*) es escribir lo siguiente:

```
composer create-project symfony/website-skeleton nombreproyecto
```
- Se diferencian en que la primera crea un proyecto con la estructura mínima (nos tocaría luego instalarle varias cosas a mano), mientras que la segunda ya incluye una serie de dependencias básicas, como el motor de plantillas Twig y el ORM Doctrine. Por este motivo, suele utilizarse la segunda opción.

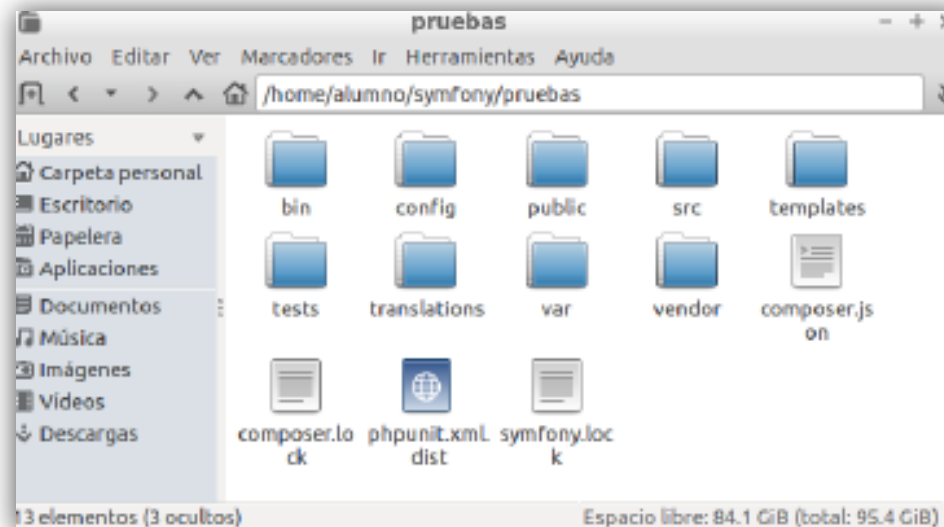
Nuestro primero proyecto

Como ejemplo, elige una carpeta y crea un proyecto llamado *pruebas*:



```
alumno@alumno: ~/symfony
Archivo  Editar  Pestañas  Ayuda
alumno@alumno:~$ cd symfony
alumno@alumno:~/symfony$ composer create-project symfony/website-skeleton pruebas
```

Tras un rato, se crearán las carpetas del proyecto (la primera vez tarda más que el resto).

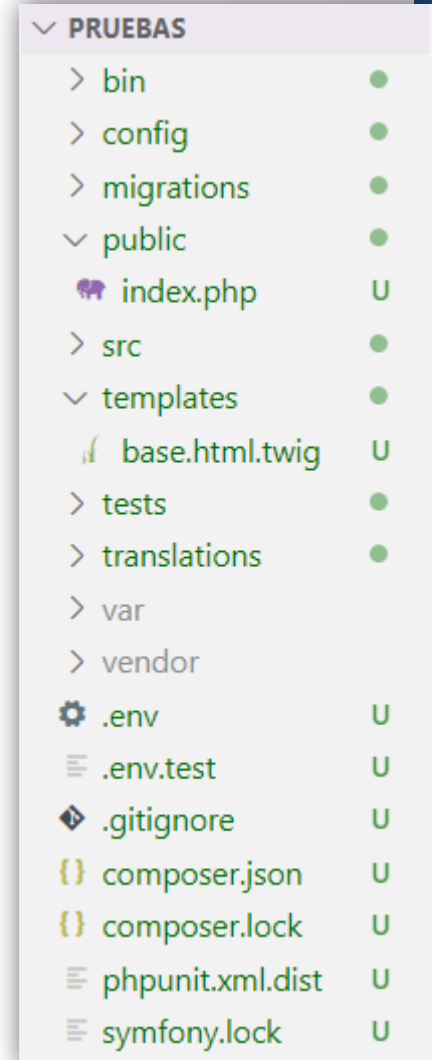
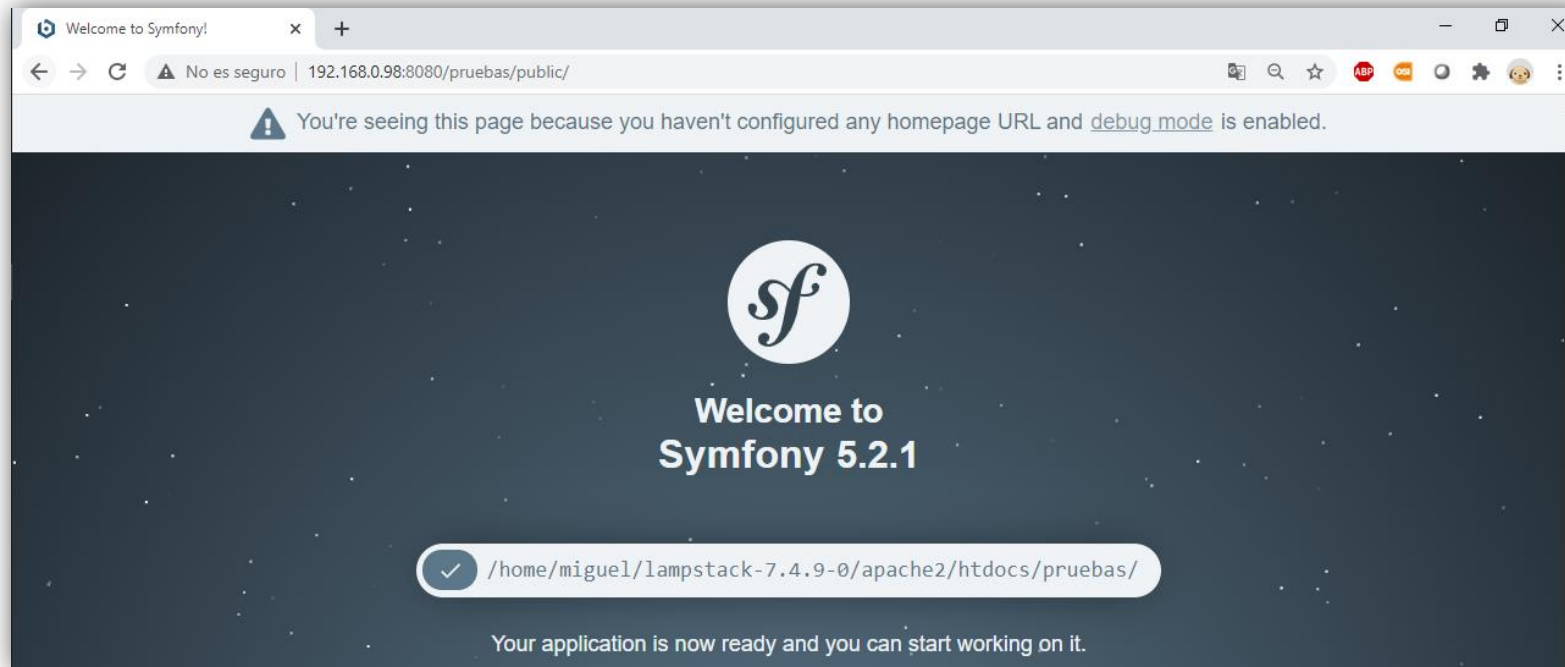




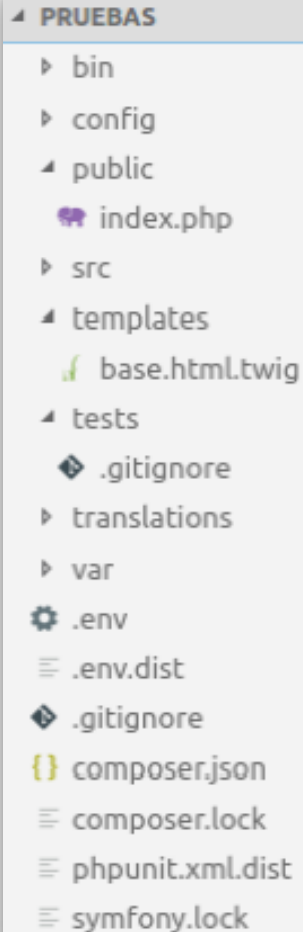
Nuestro primero proyecto

Tras completar el comando anterior, se habrá creado una estructura con varias carpetas y archivos dentro de */rutaElegida/pruebas*.

Si abrimos esta carpeta desde VS Code, podremos ver la estructura del proyecto en el panel izquierdo.



Estructura general de un proyecto



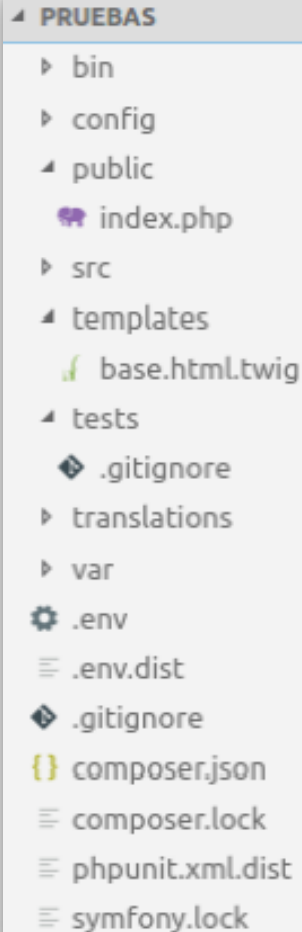
```
PRUEBAS
├── bin
├── config
├── public
│   └── index.php
├── src
├── templates
│   └── base.html.twig
├── tests
│   ├── .gitignore
│   └── translations
├── var
├── .env
├── .env.dist
├── .gitignore
├── composer.json
├── composer.lock
├── phpunit.xml.dist
└── symfony.lock
```

bin: contiene algunos ejecutables, como `console` para escribir comandos de consola, o `phpunit` para lanzar las pruebas unitarias.

config: contiene los archivos de configuración para los diferentes ámbitos en que se desarrolle el proyecto. Por defecto se definen 3 ámbitos, que son *dev* (para desarrollo), *prod* (para puesta en producción) y *test* (para pruebas), aunque podemos añadir los que queramos. Cada ámbito contiene unos archivos de configuración YAML (veremos más adelante en qué consiste este formato), para poder especificar opciones concretas. Por ejemplo, en el ámbito *dev* nos puede interesar que se muestre toda la información necesaria por pantalla, o a un archivo, mientras que en el ámbito *prod* primará más la eficiencia. Emplearemos esta carpeta para configurar rutas y servicios, como veremos próximamente.

public: contiene la parte pública, o estática de la página (css, js,...). En versiones anteriores de Symfony se llamaba *web*.

Estructura general de un proyecto



```
PRUEBAS
├── bin
├── config
├── public
│   ├── index.php
│   ├── src
│   ├── templates
│   │   └── base.html.twig
│   ├── tests
│   │   └── .gitignore
│   ├── translations
│   └── var
├── .env
├── .env.dist
├── .gitignore
├── composer.json
├── composer.lock
├── phpunit.xml.dist
└── symfony.lock
```

src: contiene el código fuente PHP. Tanto los *controladores* de la aplicación, como las clases de nuestro *modelo* de datos.

templates: contiene las plantillas para las vistas, es decir páginas que se preprocesarán por nuestro motor de plantillas, Twig.

tests: se destina al desarrollo de pruebas.

translations: para opciones de internacionalización.

var: para archivos temporales, como los de caché o de log.

vendor: para instalar *bundles* de terceros que descarguemos. Lo veremos al final del trimestre. *Esta carpeta, por su tamaño, tendréis que sacarla fuera de vuestro proyecto en las entregas.*



La consola de Symfony

Hemos visto que hay una carpeta **bin** con dos ejecutables. Veremos uno de ellos: **console**, que nos puede ayudar en ciertas tareas, como crear entidades, examinar servicios, etc.

Puedes usar este ejecutable desde *terminal del sistema* siguiendo estos pasos:

- 1) Situarnos en la carpeta del proyecto
- 2) Escribir `php bin/console XXXXX` (siendo `XXXXX` comandos a utilizar que iremos viendo).

Si quieres experimentar, prueba la siguiente orden, para obtener un listado de los servicios que hay actualmente predefinidos en Symfony (ya veremos en qué consiste esto de los servicios).

```
php bin/console debug:autowiring
```



Entregar nuestros proyectos Symfony

Puedes comprobar que la carpeta `/home/alumno/symfony/pruebas` del proyecto que hemos creado, ocupa bastantes megas. Gran parte de ese tamaño se debe a las dependencias que por defecto se instalan al crear el proyecto con la opción `website-skeleton`. Además, el tamaño crecerá más si instalamos alguna otra dependencia adicional a las que ya vienen.

Obviamente, distribuir una carpeta de ese tamaño no es lo más recomendable. En lugar de eso, lo que se suele hacer es **quitar la subcarpeta vendor**, que es donde se instalan todas las dependencias externas. Este hecho no supone ningún problema, dado que se puede regenerar la carpeta con el siguiente comando (desde la carpeta principal del proyecto Symfony en cuestión):

```
composer install
```

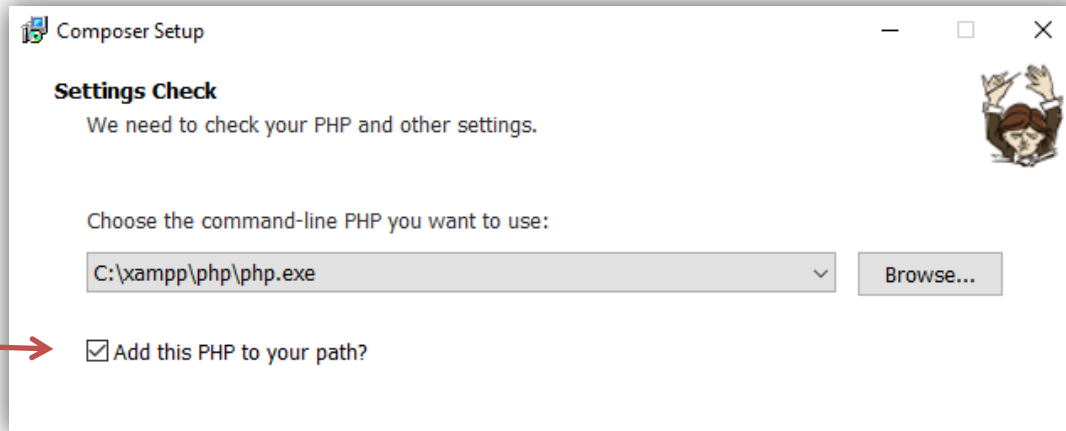
que simplemente vuelve a crear la carpeta vendor y reinstala dentro las dependencias que figuran en el archivo `composer.json`.



-

Instalación en servidor Windows

- Instala Composer desde su [web oficial](#) (*Windows Installer*)



- Cierra toda consola de comandos y abre una nueva. Revisa la versión con `composer -v`

```
C:\Users\Miguel>composer -v

Composer version 2.0.8 2020-12-03 17:20:38
```

Instalación en servidor Windows

- Accede a la carpeta en que tienes las aplicaciones en tu servidor local (*htdocs*)

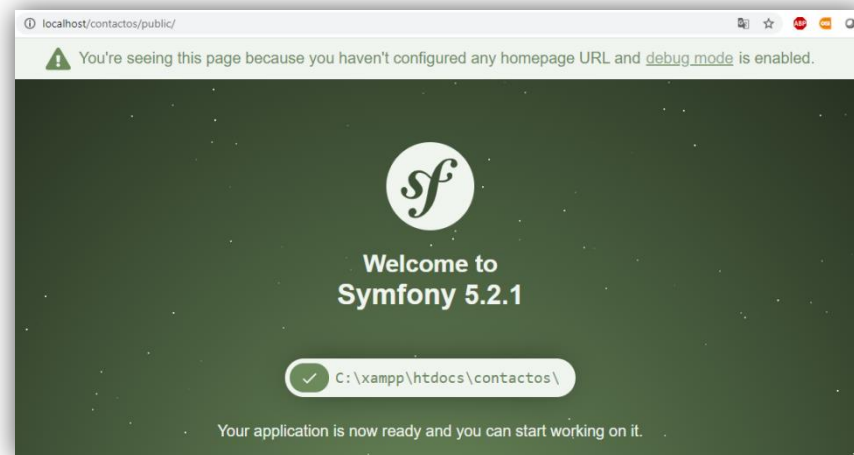
```
C:\Users\Miguel>cd C:\xampp\htdocs  
C:\xampp\htdocs>
```

- Ejecuta el siguiente comando, que instalará Symfony y descargará todos los paquetes y dependencias, antes de generar un proyecto llamado **contactos**.

`composer create-project symfony/website-skeleton contactos`

- Inicia el servidor
- Accede a `localhost/contactos/public`

Si deseas, puedes crear el proyecto en una carpeta llamada 'symfony', por ejemplo. Ten en cuenta que, de este modo, cambiará la url también.



EJERCICIO 2

- Crea en local (no máquina virtual) un proyecto *Symfony* llamado ***contactos***.
- Debes entregar 2 capturas de pantalla (**NO recortes**), que muestren:
 - ✓ El navegador con la URL y la página de bienvenida de Symfony.
 - ✓ Muestra la versión de Composer, para demostrar que está instalado.





ANEXO: YAML (I)

La configuración de los proyectos Symfony, por defecto, se define mediante archivos YAML. Estos archivos son una alternativa al formato XML, normalmente más largos y engorrosos, y al propio formato PHP, más complicado de escribir.

Son archivos que se pueden ejecutar sobre la marcha. Esto tiene el inconveniente de que los posibles errores (de sintaxis, por ejemplo) no se mostrarán hasta que la aplicación no se ejecute.

El estándar YAML es muy amplio, pero para Symfony sólo es necesario emplear un subconjunto reducido. La información se estructura en parejas **clave: valor**.

La jerarquía de la información se establece mediante indentaciones, pero no realizadas con el tabulador, sino con la barra espaciadora (4 espacios por nivel).

```
default_table_options:  
  charset: utf8mb4  
  collate: utf8mb4_unicode_ci
```



ANEXO: YAML (II)

Puede haber grupos de elementos **clave: valor**, formando arrays. En el caso de arrays normales, van entre corchetes, y en el caso de asociativos, entre llaves:

```
roles: [ROLE_USER, ROLE_ADMIN]
users:
  admin: { password: adminpass, roles: [ROLE_ADMIN] }
  user: { password: userpass, roles: [ROLE_USER] }
```

Los comentarios se indican con una almohadilla a principio de línea.

```
# In-memory users
users:
```

La primera clave de cada archivo YAML es su clave principal, que contiene al resto de claves y valores (el resto, van indentadas 'n' niveles respecto a ésta).

Cuando se ejecuta la aplicación, Symfony convierte estos archivos YAML en archivos PHP con un código equivalente.