Taller de Symfony2

Sergio Gómez (@sgomez)

CAPÍTULO 1

Introducción

Si quieres puedes saltarte este capítulo e ir directamente al segundo, dónde está explicado todo el proceso de instalación de la aplicación de ejemplo.

1.1 ¿Qué es Symfony2?

Según la wikipedia (http://es.wikipedia.org/wiki/Symfony) **Symfony2** es un completo framework diseñado para optimizar el desarrollo de las aplicaciones web mediante algunas de sus principales características. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación web.

Symfony está desarrollado completamente en PHP 5.3. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y Microsoft SQL Server. Se puede ejecutar tanto en plataformas *nix (Unix, Linux, etc.) como en plataformas Windows.

1.2 Características

- Fácil de instalar y configurar en la mayoría de plataformas (y con la garantía de que funciona correctamente en los sistemas Windows y *nix estándares).
- Independiente del sistema gestor de bases de datos. Su capa de abstracción y el uso de Propel, permiten cambiar con facilidad de SGBD en cualquier fase del proyecto.
- Utiliza programación orientada a objetos, de ahí que sea imprescindible PHP 5.
- Sencillo de usar en la mayoría de casos, aunque es preferible para el desarrollo de grandes aplicaciones Web que para pequeños proyectos.
- Aunque utiliza MVC (Modelo Vista Controlador), tiene su propia forma de trabajo en este punto, con variantes del MVC clásico como la capa de abstracción de base de datos, el controlador frontal y las acciones.
- Basado en la premisa de “convenir en vez de configurar”, en la que el desarrollador sólo debe configurar aquello que no es convencional.

- Sigue la mayoría de mejores prácticas y patrones de diseño para la web.
- Preparado para aplicaciones empresariales y adaptable a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- Código fácil de leer que incluye comentarios de phpDocumentor y que permite un mantenimiento muy sencillo.
- Fácil de extender, lo que permite su integración con las bibliotecas de otros fabricantes.
- Una potente línea de comandos que facilitan generación de código, lo cual contribuye a ahorrar tiempo de trabajo.

1.3 ¿De qué va el taller?

Vamos a desarrollar desde el principio una aplicación muy sencilla para gestionar las inscripciones a una serie de actividades.

El taller repasará brevemente los conceptos de la arquitectura Modelo-Vista-Controlador y como encajan los componentes de Symfony2 en ella.

Con esa breve introducción pasaremos a la parte práctica, que consiste en ir desarrollando la aplicación poco a poco.

CAPÍTULO 2

Preparando nuestro sistema

Aunque Symfony2 puede funcionar en cualquier plataforma que soporte LAMP (http://en.wikipedia.org/wiki/LAMP_(software_bundle)) vamos a centrar las explicaciones en la última versión de Ubuntu. Si usas otro sistema u otra distribución deberás adaptar las instrucciones a las características de tu equipo.

2.1 Requisitos técnicos

Los requisitos para instalar **Symfony2** en un entorno de desarrollo son:

2.1.1 Obligatorios:

- PHP en su versión 5.3.2 o superior
- Soporte de sqlite3, json, ctype
- Tener configurada la opción `date.timezone` en el archivo php.ini

2.1.2 Opcionales

Las siguientes opciones no son obligatorias, pero son muy recomendables porque casi con seguridad van a ser utilizadas:

- Módulo php-xml instalado.
- Librería libxml versión 2.6.21 o superior.
- PHP tokenizer habilitado.
- Funciones mbstring habilitadas.
- iconv habilitado.
- En *NIX, soporte de POSIX.
- Módulo intl con soporte ICU 4+
- Caché de código PHP como APC o XCODE.
- Tener las siguientes opciones de php desactivadas: `short_open_tag`, `magic_quotes_gpc`, `register_globals` y `session.autostart`.
- El software git de control de versiones.

2.1.3 Doctrine

Doctrine es un Object-Relational Mapper para PHP. Proporciona una capa de persistencia para objetos en PHP y se situa entre nuestra aplicación y nuestra base de datos. En el

taller lo explicaremos con más detalle, pero ahora nos basta con saber que lo necesitamos. Para usar **Doctrine** es necesario tener instalada los drivers PDO para PHP.

2.1.4 Otras aplicaciones

Se recomienda también instalar:

- **phpmyadmin** para administrar las bases de datos que creemos.
- **Google Chrome** o **Firefox** con **Firebug** en su última versión.
- **Netbeans** como entorno de desarrollo (si tenemos memoria suficiente, netbeans la debora).

2.2 Instalando el software básico

Primero vamos a proceder a descargar los paquetes que vamos a necesitar para poder desarrollar con **symfony2** en nuestro equipo:

```
bash$ sudo apt-get install apache2 php5 php5-cli php5-sqlite php5-mysql
php5-intl php5-xmlrpc php-apc phpmyadmin mysql-server git
```

En el proceso nos preguntará por una clave para el administrador(root) de mysql. Luego nos la preguntará otra vez para instalar phpmyadmin, además de una clave para la base de datos de phpmyadmin. Podemos poner siempre la misma, pero es importante que no la olvidemos o no podremos crear ninguna base de datos después.

Bug del paquete php5-sqlite: El paquete php5-sqlite tiene un bug que os puede dar problemas, aunque lo normal es que solo dé un warning sin más consecuencias. Para arreglarlo tenéis que borrar un archivo que está añadido de más:

```
bash$ sudo rm /etc/php5/conf.d/sqlite.ini
```

Gracias a Nacho(@thedarkpal) por reportarlo.

2.2.1 Configurando PHP

Ahora, antes de poder empezar a descargar y utilizar **symfony2** vamos a configurar algunas opciones de PHP que son requisito indispensable para poder trabajar. Para ello vamos a crear un archivo (como root) en /etc/php5/conf.d llamado symfony.ini

```
bash$ sudo vim /etc/php5/conf.d/symfony.ini
```

Y lo editamos con el siguiente contenido:

```
; fichero symfony.ini
date.timezone=Europe/Madrid
short_open_tag=Off
magic_quotes_gpc=Off
register_globals=Off
session.autostart=Off
```

Ahora si queremos podemos reiniciar **apache2** para que relea la configuración, pero lo vamos a hacer de todas maneras más adelante.

2.2.2 Instalando la aplicación Actividades

Ya podemos comenzar a instalar la aplicación de ejemplo que vamos a usar para el taller. Para ello vamos a usar la herramienta git que instalamos en uno de los pasos anteriores. El último paso, el de instalación de vendors, tardará un poco porque tiene que descargarse bastante software. Si tenéis una buena conexión, mejor.

```
bash$ mkdir $HOME/Sites
bash$ cd $HOME/Sites
bash$ git clone https://bitbucket.org/perseo/actividades.git
bash$ cd actividades
bash$ cp app/config/parameters.ini.orig app/config/parameters.ini
bash$ php bin/vendors install
```

Si hemos seguido los pasos correctamente, podemos ya comprobar que todo funciona correctamente.

```
bash$ cd $HOME/Sites/actividades
bash$ php app/check.php
********************************
*                              *
*  Symfony requirements check  *
*                              *
********************************

** Mandatory requirements **

 OK        Checking that PHP version is at least 5.3.2
 OK        Checking that the "date.timezone" setting is set
 OK        Checking that app/cache/ directory is writable
 OK        Checking that the app/logs/ directory is writable
 OK        Checking that the json_encode() is available
 OK        Checking that the SQLite3 or PDO_SQLite extension is available
 OK        Checking that the session_start() is available
```

```
 OK        Checking that the ctype_alpha() is available
 OK        Checking that the token_get_all() is available
 OK        Checking that the APC version is at least 3.0.17

** Optional checks **

 OK        Checking that the PHP-XML module is installed
 OK        Checking that the token_get_all() function is available
 OK        Checking that the mb_strlen() function is available
 OK        Checking that the iconv() function is available
 OK        Checking that the utf8_decode() is available
 OK        Checking that the posix_isatty() is available
 OK        Checking that the intl extension is available
 OK        Checking that the intl ICU version is at least 4+
 OK        Checking that a PHP accelerator is installed
 OK        Checking that php.ini has short_open_tag set to off
 OK        Checking that php.ini has magic_quotes_gpc set to off
 OK        Checking that php.ini has register_globals set to off
 OK        Checking that php.ini has session.auto_start set to off

** Optional checks (Doctrine) **

 OK        Checking that PDO is installed
 OK        Checking that PDO has some drivers installed: mysql, sqlite
```

2.2.3 Configurar Apache

Ya tenemos casi listo nuestra aplicación. Ahora nos queda permitir que sea accesible desde un navegador web. Para ello vamos a configurar **apache2** para un entorno de ***DESARROLLO***. Que quede bien claro que esta configuración no está recomendada para un sistema en producción. Simplemente está pensada para facilitar el trabajo.

Detenemos el servicio de apache con `sudo service apache2 stop`. Y ahora vamos a editar el archivo /etc/apache2/envvars (no olvidéis hacerlo como root). Debéis cambiar las variables `APACHE_RUN_USER` y `APACHE_RUN_GROUP` para que tengan vuestro nombre de usuario.

```
bash$ sudo service apache2 stop
bash$ sudo vi /etc/apache2/envvars
```

Y ahora buscamos estas líneas y las cambiamos con nuestro nombre de usuario:

```
export APACHE_RUN_USER=sergio
export APACHE_RUN_GROUP=sergio
```

Esta configuración no es obligatoria, pero tiene la ventaja de que si trabajamos en nuestra cuenta de usuario, no tendremos que ir cambiando permisos a los archivos para que **apache2** acceda a ellos. Por otra parte tiene la desventaja de que todo se ejecuta con nuestros mismos permisos, así que cuidado con dejar esta configuración en entornos en producción o en equipos conectados a Internet.

El siguiente paso es agregar el directorio `Sites` que hemos creado, como un repositorio de proyectos web. Para eso editamos como root el archivo /etc/apache2/sites-available/default:

```
<VirtualHost *:80>
    ServerAdmin webmaster@localhost
    UseCanonicalName Off

    VirtualDocumentRoot /home/sergio/Sites/%1/web
    <Directory /home/sergio/Sites/*/web>
            AllowOverride All
    </Directory>

    DocumentRoot /var/www
# [...] El resto igual
```

Y ya, por último, antes de iniciar **apache2** de nuevo necesitamos cargar dos módulos que no vienen activados por defecto:

```
bash$ sudo a2enmod vhost_alias
bash$ sudo a2enmod rewrite
bash$ sudo service apache2 start
```

Corrección: Anteriormente ponía redirect. El nombre correcto del módulo es rewrite. Gracias a @thedarkpal por los comentarios.

La configuración de apache que acabamos de realizar nos permite crear todos los host virtuales que queramos en nuestro directorio Sites. Solo necesitamos una forma de que los navegadores encuentren la IP de esos hosts virtuales. Para ellos vamos a configurar el archivo hosts del sistema para añadir nuestra web:

```
bash$ sudo vi /etc/hosts
```

Y añadimos un nuevo host. Cada vez que queramos añadir un nuevo sitio tendremos que hacer lo mismo.

```
127.0.0.1       localhost
# Añadimos la siguiente línea
127.0.0.1       actividades actividades.local

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Y ya podemos acceder a nuestra web de configuración de **symfony2** a través de la dirección http://actividades.local/app_dev.php/_configurator/

2.2.4 Configurar MySQL

La página que nos acaba de aparecer es la de configuración de la base de datos de nuestra aplicación. Antes de continuar debemos crear una. Podemos hacerlo a través de la aplicación phpmyadmin que ya hemos instalado, o a través de línea de órdenes. Ya que estamos metidos en la dinámica de hacerlo todo por consola, continuemos así. Recordad que os pedirá la contraseña que usásteis en la instalación.

```
bash$ mysqladmin -u root -p create actividades
Enter password:
bash$ mysql -u root -p mysql
Enter password:
mysql> grant all privileges on actividades.* to 'sfactividades'@'localhost'
identified by 'clavesecreta';
Query OK, 0 rows affected (0.00 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)

mysql> quit
Bye
```

Y ya tenemos preparada nuestra base de datos. Solo necesitamos indicar en la web los siguientes datos:

- Como nombre (`Name`) de la base de datos: actividades
- Como usuario (`User`): sfactividades

- Como contraseña (`Password`): clavesecreta (o mejor aún, la clave que hayáis puesto que no sea esta, claro)

Y le damos a `Next step` dos veces hasta que nos diga que nuestra distribución está configurada.

2.3 Todo listo. Comencemos

Ya solo nos queda cargar la base de datos con las tablas y los datos. Nada más fácil. Nos vamos al directorio de nuestra aplicación y ejecutamos lo siguiente:

```
bash$ cd $HOME/Sites/actividades
bash$ php app/console doctrine:schema:create
bash$ php app/console doctrine:fixtures:load
```

¡Y listo! Ya podemos ver nuestra web. Para eso solo tenemos que cargar la dirección del entorno de desarrollo: http://actividades.local/app_dev.php/.

Hay ya usuarios dados de alta. El de administrador se llama `admin` y la contraseña es `admin` también. La aplicación difiere de la del Consejo en que ésta permite crear usuarios, mientras que la nuestra no, ya que usamos los de la Universidad.

Aún queda mucho por ver, pero eso lo dejamos para el taller.

2.4 One last thing

Os recomendamos también que si venís con los portátiles, lo hagáis con Google Chrome instalado. También os recomendamos que vengáis con Netbeans para PHP, ya que por el momento este último es el único IDE que soporta **symfony2**. Los podréis encontrar en sus web oficiales.