INTRODUCCIÓN A YII2

Ricardo Pérez López

IES Doñana, curso 2017-18

FRAMEWORKS,

MICROFRAMEWORKS Y LIBRERÍAS

FRAMEWORKS

Un ***framework*** es un software que proporciona una funcionalidad genérica que el programador puede cambiar selectivamente escribiendo código específico para una aplicación en particular.

Un *framework* proporciona una forma estándar de programar y desplegar aplicaciones.

Un *framework* es un entorno de software universal y reutilizable que proporciona funcionalidades concretas como parte de una plataforma software más grande para facilitar el desarrollo de aplicaciones.

Los *frameworks* pueden incluir programas de soporte, compiladores, librerías de código, conjuntos de utilidades y APIs que, juntos, forman los diferentes componentes que permiten desarrollar un proyecto o sistema.

FRAMEWORKS PARA APLICACIONES WEB

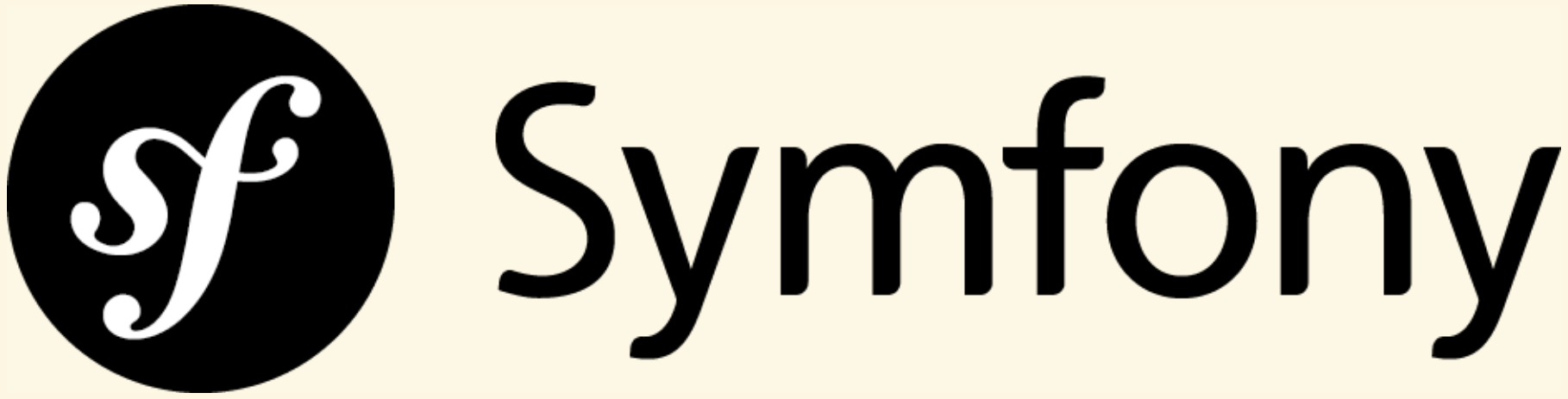
Un ***framework* para aplicaciones web** es un *framework* diseñado para facilitar el desarrollo de sitios web dinámicos, aplicaciones web y servicios web.

Este tipo de *frameworks* intenta aliviar el exceso de trabajo asociado con actividades comunes usadas en desarrollos web.

Por ejemplo, muchos *frameworks* proporcionan bibliotecas para acceder a bases de datos, estructuras para plantillas y gestión de sesiones, y con frecuencia facilitan la reutilización de código.

EJEMPLOS DE FRAMEWORKS WEB PARA PHP

- Symfony
- Laravel
- CakePHP
- CodeIgniter
- Yii



https://symfony.com

Ventajas

- El más popular, junto con Laravel
- Maduro y estable
- Especialmente indicado para proyectos grandes
- Mucha documentación
- Muy demandado

Inconvenientes

- Curva de aprendizaje algo pronunciada
- Con proyectos pequeños cuesta arrancar
- Su rendimiento no es especialmente alto

https://laravel.com

Ventajas

- Está de moda
- Gran comunidad de usuarios
- Muy demandado

Inconvenientes

- Curva de aprendizaje pronunciada
- Su rendimiento es bastante pobre
- Actualizaciones complejas

https://cakephp.org

Ventajas

- Maduro y moderno al mismo tiempo
- Muchos elementos de serie
- Bien diseñado
- No es especialmente difícil de aprender
- Interesante equilibrio entre potencia, comunidad y facilidad de uso

Inconvenientes

- No es tan utilizado como otros *frameworks* (aunque tampoco es de los menos)
- Su rendimiento no es su principal baza (aunque tampoco es de los peores)

https://codeigniter.com

Ventajas

- Rendimiento espectacular
- Simple y sencillo
- Fácil de aprender
- Buena comunidad de entusiastas usuarios

Inconvenientes

- *Demasiado* sencillo (le faltan muchos componentes que ya vienen de serie en otros *frameworks*)
- No usa Composer ni es miembro del PHP-FIG.

http://www.yiiframework.com

Ventajas

- Excelente rendimiento
- Fácil de aprender
- Muy completo de serie
- Generador de código integrado incluyendo CRUD con Bootstrap
- Moderno en el desarrollo y uso de las últimas técnicas

Inconvenientes

- No tan conocido ni usado como otros *frameworks*
- Comunidad de usuarios no muy extensa en comparación con otros

MICROFRAMEWORKS

Un *microframework* es un término usado para referirse a un *framework* web minimalista, en contraste con un *framework* completo o ***full-stack***.

Un *microframework* carece de muchas funcionalidades propias de un *framework* web completo, como por ejemplo:

- Cuentas de usuario, autenticación, autorización, roles, etc.
- Abstracción de bases de datos mediante mapeado objeto-relacional.
- Validación y saneado de la entrada.
- Motores de plantillas web.

Normalmente, un *microframework* se encarga de:

1. Recibir una petición HTTP,
2. Encaminar dicha petición al controlador adecuado,
3. Despachar el controlador y
4. Devolver una repuesta HTTP.

A menudo se diseñan específicamente para crear las API de otro servicio o aplicación.

DEFINICIÓN DE MICROFRAMEWORK SEGÚN FLASK

“Micro” does not mean that your whole web application has to fit into a single Python file (although it certainly can), nor does it mean that Flask is lacking in functionality. The “micro” in microframework means Flask aims to keep the core simple but extensible. Flask won’t make many decisions for you, such as what database to use. Those decisions that it does make, such as what templating engine to use, are easy to change. Everything else is up to you, so that Flask can be everything you need and nothing you don’t.

By default, Flask does not include a database abstraction layer, form validation or anything else where different libraries already exist that can handle that. Instead, Flask supports extensions to add such functionality to your application as if it was implemented in Flask itself. Numerous extensions provide database integration, form validation, upload handling, various open authentication technologies, and more. Flask may be "micro", but it's ready for production use on a variety of needs.

EJEMPLO DE APLICACIÓN WEB CON MICROFRAMEWORK

Usando Silex:

```php
require_once __DIR__ . '/../vendor/autoload.php';

$app = new Silex\Application();

$app->get('/hello/{name}', function($name) use ($app) {
    return 'Hello ' . $app->escape($name);
});

$app->run();
```

FRAMEWORKS Y LIBRERÍAS

Los *frameworks* tienen tres características clave que los distinguen de las librerías:

- **Inversión del control**: En un *framework*, a diferencia de lo que pasa con las librerías o las aplicaciones normales de usuario, el flujo de control del programa viene dictado por el propio *framework* y no por el programa que lo usa.
- **Extensibilidad**: Un usuario puede extender el *framework*, normalmente sobreescribiendo código o añadiendo código especializado para proporcionar funcionalidades específicas.
- **No modificable**: El código del *framework*, por lo general, se supone que no se debe modificar, aunque admite extensiones desarrolladas por el programador que lo usa. En otras palabras: el programador usuario del

La diferencia más importante es que **el control se invierte**:

Con una librería:

El código de la aplicación *llama* a la librería, por lo que **el control lo tiene la aplicación**.

Con un framework:

El *framework llama* al código de la aplicación, por lo que **el control lo tiene el framework**.

1. Usa la herramienta Google Trends para determinar la popularidad de los principales *frameworks* para PHP. ¿Cuál es el más popular? ¿Cuál ha tenido un crecimiento más rápido?

2. ¿Crees que es importante la popularidad de un *framework* a la hora de seleccionarlo como herramienta de ayuda al desarrollo de un proyecto web?

3. Busca una comparativa de rendimiento de los principales *frameworks* para PHP y ordénalos según dicha comparativa. ¿Por qué hay algunos que son espectacularmente más rápidos que el resto?

6. Investiga otros *frameworks* web para PHP.
7. Investiga otros *frameworks* web para otros lenguajes como Ruby o Python.

INSTALACIÓN, REQUISITOS Y PUESTA EN MARCHA

LIMIT

bla bla bla

OFFSET

bla bla bla

PATRÓN *MODELO-VISTA-CONTROLADOR* (MVC)

WITH

Common table expressions

PLANTILLA *BÁSICA* VS. *AVANZADA*

BARRA DE DEPURACIÓN

GENERADOR DE CÓDIGO *GII*

ESTILO DEL CÓDIGO