
# QUÉ ES FRAMEWORK

**Definición:**

Un **framework** es un marco o esquema de trabajo generalmente utilizado por programadores para realizar el desarrollo de software. Utilizar un framework permite agilizar los procesos de desarrollo ya que evita tener que escribir código de forma repetitiva, asegura unas buenas prácticas y la consistencia del código.

Un framework es por tanto un conjunto de herramientas y módulos que pueden ser reutilizados para varios proyectos. Uno de los DreamWorks más conocidos y utilizados es el .NET Framework de Microsoft para webs.


## Ventajas de los Frameworks

Entre las **ventajas de utilizar un framework** para el desarrollo de software distinguimos:

- El programador ahorra tiempo ya que dispone ya del esqueleto sobre el que desarrollar una aplicación.
- Facilita los desarrollos colaborativos, al dejar definidos unos estándares de programación.
- Al estar ampliamente extendido, es más fácil encontrar herramientas, módulos e información para utilizarlo.

- Proporciona mayor seguridad, al tener gran parte de las potenciales vulnerabilidades resueltas.
- Normalmente existe una comunidad detrás, un conjunto de desarrolladores que pueden ayudar a responder consultas.


**Ejemplos de Frameworks**

Estos son algunos de los frameworks más conocidos:

- **.Net**: es Framework de Microsoft y uno de los más utilizados.
- **Symphony**: proyecto PHP de software libre.
- **Zend Framework**: framework de código abierto para desarrollar aplicaciones web y con servicios web PHP.
- **Laravel**: uno de los frameworks de código abierto más fáciles de asimilar para PHP.
- **Django**: framework de desarrollo web de código abierto escrito en Python.
- **Ruby on Rails**: framework de aplicaciones web de código abierto escrito en el lenguaje de programación Ruby.
- **Angular**: framework de código abierto desarrollado en TypeScript y mantenido por Google.

Qué es un
FRAMEWORK

@OrixSystems

# ENTONCES ¿QUÉ ES UN 'FRAMEWORK'?

Siendo muy simple, es **un esquema (un esqueleto, un patrón) para el desarrollo y/o la implementación de una aplicación**. Sí, es una definición muy genérica, pero también puede serlo un *framework*: sin ir más lejos, el paradigma MVC (Model-View-Controller) dice poco más que "separa en tu aplicación la gestión de los datos, las operaciones, y la presentación". En el otro extremo, otros *frameworks* pueden llegar al detalle de definir los nombres de ficheros, su estructura, las convenciones de programación, etc.

Pongamos un **ejemplo**: una aplicación web que utilice Java como lenguaje programación puede implementarse de multitud de formas, mediante servlets y JSPs. Hay algunas convenciones que es necesario seguir, como usar un fichero de configuración web.xml, pero el programador sigue sin tener un patrón claro a seguir para la creación de servlets, clases, JSPs, etc.


En una primera estandarización, la utilización de una arquitectura MVC aconseja que separemos la lógica de la aplicación (en los servlets) de la presentación (usando JSPs); en concreto, no sería correcto codificar lógica de aplicación o accesos a base de datos dentro de los JSP.

Un paso más allá: utilizando Faces como *framework*, la estructura de la aplicación queda todavía más definida: un único servlet (FacesServlet) va a controlar el flujo de la aplicación; además, el uso de un fichero concreto (faces-config.xml) permite crear la navegación de la aplicación, definir los objetos (beans) pasados como parámetros, etc., todo ello sin necesidad de codificarlo en Java o JSP.


**EN CONCLUSIÓN**

La utilización de un *framework* en el desarrollo de una aplicación implica un cierto coste inicial de aprendizaje, aunque a largo plazo es probable que facilite tanto el **desarrollo** como el **mantenimiento**.

Existen multitud de *frameworks* orientados a diferentes lenguajes, funcionalidades, etc. Aunque la **elección** de uno de ellos puede ser una tarea complicada, lo más probable que a largo plazo sólo los mejor definidos (o más utilizados, que no siempre coinciden con los primeros) permanezcan. Y si ninguno de ellos se adapta a las necesidades de desarrollo, siempre es mejor definir uno propio que desarrollar “al por mayor”.