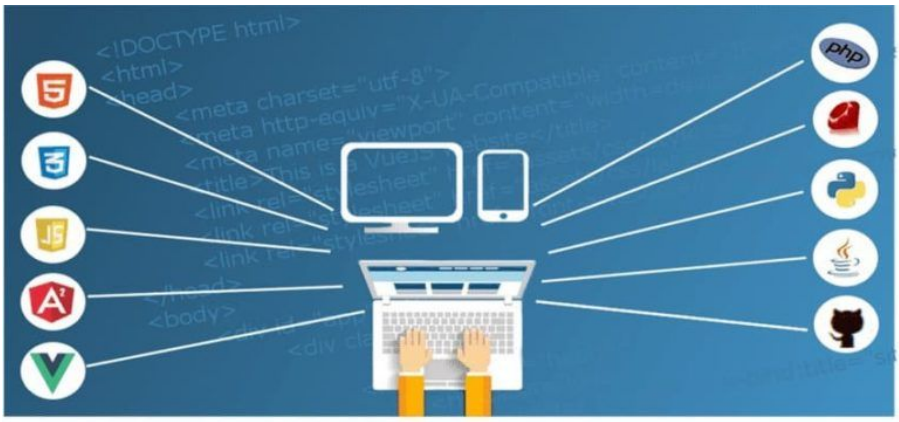
ISPC
INSTITUTO SUPERIOR
POLITÉCNICO CÓRDOBA

FRAMEWORKS - MVC



¿Qué es un framework web?

Trataremos de explicar en forma sencilla en que consiste un framework para sistemas Web y las características generales de estos frameworks.

Introducción:

El concepto framework se emplea un muchos ámbitos del desarrollo de sistemas software, no solo en el ámbito de aplicaciones Web. Podemos encontrar frameworks para el desarrollo de aplicaciones médicas, de visión por computador, para el desarrollo de juegos, etc.

En general, con el término framework, nos estamos refiriendo a:
"Una estructura software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. "
En otras palabras, un framework se puede considerar como una aplicación genérica incompleta y configurable a la que podemos añadirle las últimas piezas para construir una aplicación concreta.

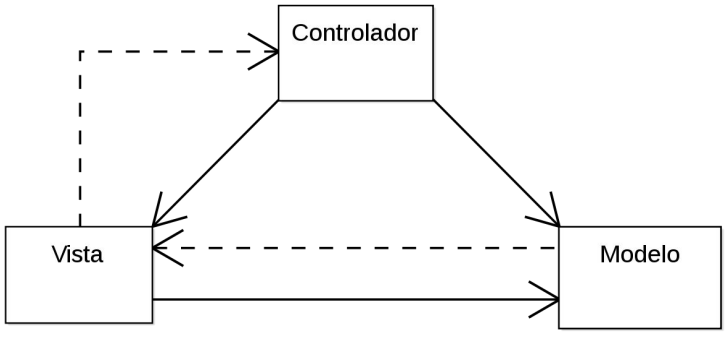
Los objetivos principales que persigue un framework son:
1- acelerar el proceso de desarrollo
2- reutilizar código ya existente
3- promover buenas prácticas de desarrollo como el uso de patrones.

Un framework Web, por tanto, podemos definirlo como un conjunto de componentes que componen un diseño reutilizable que facilita y agiliza el desarrollo de sistemas Web. A la vez que que hace posible que los costos se reduzcan porque:

1- disminuye considerablemente las pruebas y los errores en cada uno de los programas

2- simplifica los procesos y evitar que se esté expuesto a tantas horas de trabajo

Para comprender como trabajan los frameworks Web existentes es imprescindible conocer el patrón MVC.



El patrón **Modelo-Vista-Controlador** es una guía para el diseño de arquitecturas de aplicaciones que ofrezcan una fuerte interactividad con usuarios.
Este patrón organiza la aplicación en tres modelos separados:
1- el primero es un modelo que representa los datos de la aplicación y sus reglas de negocio
2- el segundo es un conjunto de vistas que representa los formularios de entrada y salida de información
3- el tercero es un conjunto de controladores que procesa las peticiones de los usuarios y controla el flujo de ejecución del sistema.

La mayoría, por no decir todos, de los framewroks para Web implementan este patrón.

Características:

A continuación enunciamos una serie de características que podemos encontrar en prácticamente todos los frameworks existentes.
Abstracción de URLs y sesiones. No es necesario manipular directamente las URLs ni las sesiones, el framework ya se encarga de hacerlo.
Acceso a datos. Incluyen las herramientas e interfaces necesarias para integrarse con herramientas de acceso a datos, en BBDD, XML, etc..
Controladores. La mayoría de frameworks implementa una serie de controladores para gestionar eventos, como una introducción de datos mediante un formulario o el acceso a una página. Estos controladores suelen ser fácilmente adaptables a las necesidades de un proyecto concreto.
Autentificación y control. Incluyen mecanismos para la identificación de usuarios de acceso. mediante login y password y permiten restringir el acceso a determinas páginas a determinados usuarios.
Internacionalización. Separación entre diseño y contenido

Ventajas de los frameworks web:

1 – Documentación y comunidad.
La cantidad de documentación que podremos encontrar sobre un framework web, suele ser enorme y además con una gran comunidad detrás, **respondiendo preguntas y desarrollando nuevas funcionalidades**.
2 – Reutilización del código.
Uno de los puntos fuertes del los frameworks web es la **modularidad de su código** y la capacidad para poder hacer **múltiples proyectos con el mismo código,** cambiando simplemente los textos.
3 – Arquitectura y metodología.
La mayoría de frameworks web del mercado usan **arquitecturas y metodologías actuales**, como el Modelo-Vista-Controlador.
4 – Plantillas web.
Las plantillas facilitan mucho el trabajo de los desarrolladores web y los frameworks no se quedan atrás en esto. Algunos **frameworks Frontend** como Bootstrap cuentan con grandes cantidades de **plantillas y componentes** desarrollados por su extensa comunidad.
5– Seguridad web.
Los frameworks web suelen contar con **medidas de seguridad para proteger nuestros datos y los de nuestros clientes**, ayudando en gran medida en uno de los temas que lleva de cabeza a grandes empresas de servicios web desde el 2017.
6– Posicionamiento en motores de búsqueda.
Elposicionamiento web SEOon page es muy importante si queremos lograr aparecer en las**primeras posiciones de buscadores como Google**. Por eso muchos frameworks web ya implementan en su estructura código para poder lograrlo más fácilmente.

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.

Modelo vista controlador (MVC)

Modelo Vista Controlador **(MVC)** es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

Se trata de un modelo valido para todo tipo de aplicaciones, sobre distintos lenguajes y plataformas de desarrollo.
Así:

a) El **Modelo** que contiene una representación de los datos que maneja el sistema, su lógica de negocio, y sus mecanismos de persistencia.
b) La **Vista**, o interfaz de usuario, que compone la información que se envía al cliente y los mecanismos interacción con éste.
c) El **Controlador**, que actúa como intermediario entre el Modelo y la Vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno.

En detalle:

El modelo

Es el responsable de:
-Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.
-Define las reglas de negocio (la funcionalidad del sistema). Un ejemplo de regla puede ser: "Si la mercancía pedida no está en el almacén, consultar el tiempo de entrega estándar del proveedor".
-Lleva un registro de las vistas y controladores del sistema.
Si estamos ante un modelo activo, notificará a las vistas los cambios que en los datos pueda producir un agente externo (por ejemplo, un archivo que actualiza los datos, un temporizador que desencadena una inserción, etc.).

El controlador

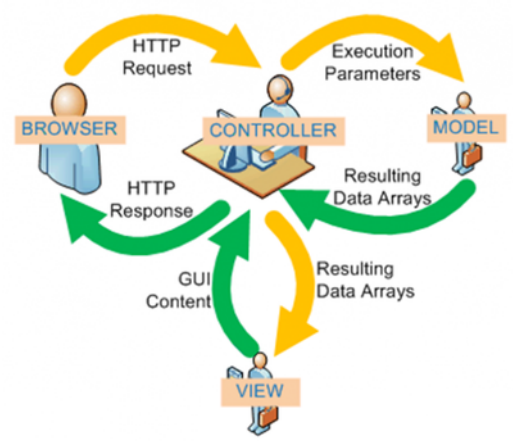
Es responsable de:
-Recibe los eventos de entrada (un clic, un cambio en un campo de texto, etc.).
-Contiene reglas de gestión de eventos, del tipo "SI Evento Z, entonces Acción W". Estas acciones pueden suponer peticiones al modelo o a las vistas.
Una de estas peticiones a las vistas puede ser una llamada al método "Actualizar()". Una petición al modelo puede ser "Obtener_tiempo_de_entrega (nueva_orden_de_venta)".

Las vistas

Son responsables de:
-Recibir datos del modelo y los muestra al usuario.
-Tienen un registro de su controlador asociado (normalmente porque además lo instancia).
-Pueden dar el servicio de "Actualización()", para que sea invocado por el controlador o por el modelo (cuando es un modelo activo que informa de los cambios en los datos producidos por otros agentes).

Proceso

El flujo es el siguiente:



1- El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace, etc.)
2- El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario.
3- El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos (handler) o callback.
4- El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario (por ejemplo, el controlador actualiza el carro de la compra del usuario).
5- La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.