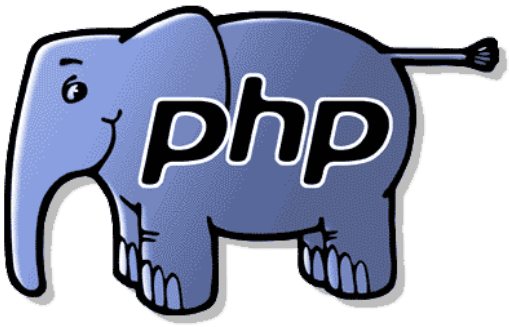


conocimientos básicos necesarios para el desarrollo de web-apps.

Acerca de....

Trabajo y presentación del alumno José Ramón Cid Constela para la asignatura Proyecto Integrado sobre herramientas y metodología MVC sobre PHP para el desarrollo de aplicaciones web.

Contenido del documento

- El antes y el ahora de las aplicaciones web
- ¿ ASP o PHP ?
- ¿ SQL o MySQL ? ¡ o Cassandra !
- Metodologías OOP y MVC
- Herramientas: IDE y otros
- Propuesta

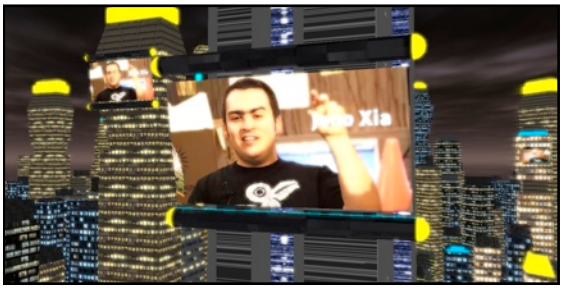
La historia de los navegadores

Aparecieron en el año 1990 con Internet Explorer y Netscape (desbancado a los pocos años), no fue hasta 2001-2002 cuando aparecieron Firefox o Mozilla (antes eran dos navegadores distintos) que se empezaron a desarrollar tecnologías sobre la web, renderizado de páginas y estándares HTML y CSS tal y como los conocemos ahora. A continuación vemos cómo era Google en esos primeros años de la web y lo que son capaces de hacer ahora los navegadores con las tecnologías existentes y las que están en desarrollo.

Antes

¿Cómo eran las aplicaciones web? CSS muy básico si es que lo había, y los diseños estaban basado en tablas, cosa que ahora es inimaginable que una marca o empresa por pequeña que sea haga una web con dicha estructura. El funcionamiento de apoyaba sobre las bases de datos que corrían detrás.

The Flight of the Navigator, de Mozilla



The Flight of the Navigator, de Mozilla

Kingdoms of camelot,
un juego de Facebook en HTML5

Ahora

Pero el cuento cambia cuando dejamos el pasado de lado y miramos a las tecnologías web presentes, estas son: CCS3, HTML5 y WEBGL. La base de datos de cara al usuario final pasa a un segundo plano y las aplicaciones se centran en la experiencia del usuario final.

Nuevas tecnologías web:

- CSS3: añade efectos de bordes redondeados, sombras y muchos más.
- HTML5: nuevas etiquetas para audio, vídeo dibujos 2D y 3D y para mejorar la estructura del código.
- WEBGL: Gráficos 3D avanzados

Los ejemplos que se muestran a la izquierda requieren de navegadores modernos, ya que los antiguos y desactualizados no se ciñen a los nuevos estándares.

- Desarrollado por Mozilla, para demostrar la capacidad de su navegador web Firefox 4 (en fase beta a día de hoy): El vuelo del navegador:
  (Necesita Google Chrome o Firefox 4)
  bit.ly/mozillaflight

- Galería de Google (en realidad de la empresa y fans) para demostrar las capacidades de Google Chrome: Chrome Experiments:
  (Necesita Google Chrome)
  bit.ly/chromeexp

HTML5 es una tecnología que ya se está usando, como por ejemplo en aplicaciones de juegos, (ver imágen lateral) no pensemos en esta tecnología como algo futuro.

Desarrollando nuestra propia web-app.

Antes de empezar con cualquier proyecto, como en cualquier otra ingeniería, se debería realizar (cosa que muy poco a menudo ocurre es así en pequeñas y medianas empresas) se debe realizar un estudio y planteamiento de soluciones y necesidades a cubrir para el o los problemas que puedan surgir y el desarrollo en general. En tema de aplicaciones web una pregunta básica que nos debemos hacer es: ¿ Linux o Windows ? Puesto que de ello dependerán: costes (licencias), servidores a utilizar, librerías disponibles, soporte, etc.

Qué tecnologías elegir.

Opciones:

- En cuanto a servidor de aplicaciones:
  ASP o PHP
- En cuanto a SGBD
  SQL o MySQL (Cassandra y muchos otros)

Siguiendo el ejemplo de los líderes

Para no entrar en debate sobre qué tipo de tecnologías usar (privadas o públicas,de pago o privadas, libres o patentadas, etc) y de las que tratar en el documento, simplemente introducirlas como las tecnologías que usan los líderes del mercado de páginas y aplicaciones web como pueden ser Facebook, Twitter o Tuenti. Estas son: PHP, MySQL / Cassandra, patrones de diseño MVC y algunos de los IDE`s de desarrollo que utilizan.

PHP

- Libre, abierto, GRATIS.
- Multiplataforma. ASP sólo para WIN.
- Soporte para varios servidores web.
- Fácil acceso a Bases de Datos.
- Mucha documentación (Ejemplos, manuales.)
- Integración perfecta entre Apache-PHP-MySQL.
- Posee una sintaxis bastante clara.
- Fácil aprendizaje.
- Seguro.
- Popular.
- Programación orientada a objetos.

MySQL Comparison

- MySQL > 50 GB Data
  Writes Average : ~300 ms
  Reads Average : ~350 ms
- Cassandra > 50 GB Data
  Writes Average : 0.12 ms
  Reads Average : 15 ms

Fuente: slidesha.re/cassandraatnosql

SQBD

Depende de la tecnología que utilices, y de los recursos económicos con los que contemos. Los líderes del mercado han utilizado desde sus comienzos bases de datos MySQL, debido a:

- Costes reducidos.
- Cumplen con los estándares SQL.
- Integración con PHP.

SGBD Avanzado

A la larga, los líderes han cambiado su sistema gestor de bases de datos a sistemas más potentes hablando de software no de hardware. El premio se lo lleva Facebook con su SGBD no relacional llamado Cassandra.

- Desarrollada por Facebook
- No relacional -> NoSQL
- Estructura de único fichero
- Escalabilidad horizontal

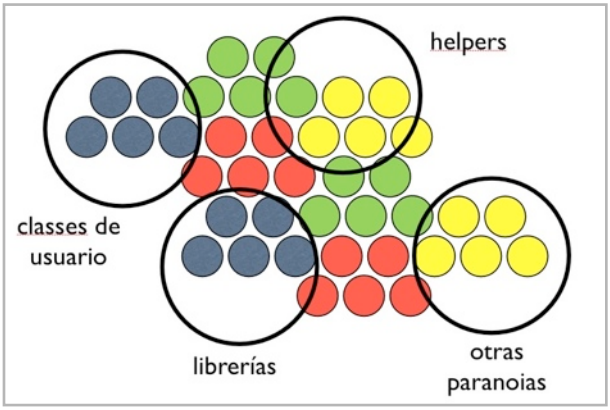
Meses después de su primera versión Twitter y otros servicios online como Digg se cambiaron a Cassandra.

Qué patrón de desarrollo seguir

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. - Wikipedia

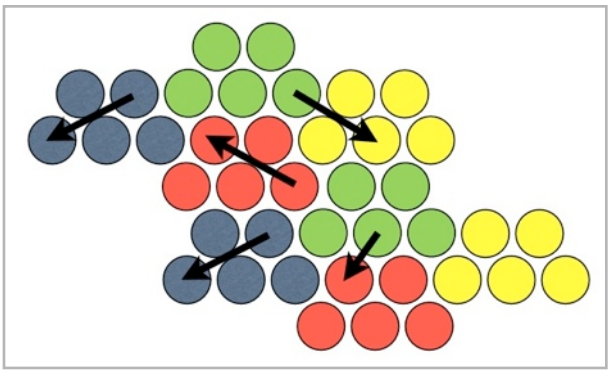
OOP, Programación orientada a objetos.

A continuación se muestra cómo funciona este patrón de diseño y los problemas que plantea a la larga bien por problemas de documentación o por que la aplicación está adquiriendo una tamaño grande.



La imagen previa se trata de una recomendación personal para la distribución de directorios y directorios. Siguiendo esto y la lógica de la OOP, las classes (una por cada bola de color) se dividen en categorías como pueden ser:

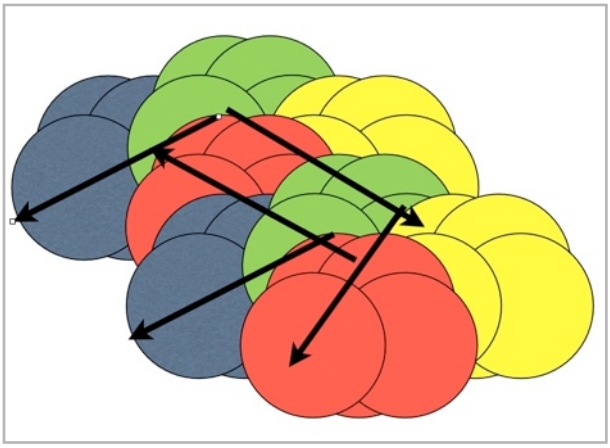
- **Classes de usuario:** usuarios, productos, facturas, etc
- **Helpers:** Manejadores de cadenas de texto, cálculo de facturas, etc
- **Librerías:** classes a mayores que amplían las funcionalidades de la aplicación. Ejemplo: sistemas o APIs de comunicación con redes sociales.



El error humano que se describe a continuación es el primer motivo que debería hacernos pensar en usar patrones MVC, y es la comunicación entre classes. Estas deberían ser independientes unas de otras (esta es una de las características de OOP: posibilidad de reutilizar código, de forma que las classes sean totalmente independientes).

Llegados a este punto de desarrollo de nuestra aplicación, nos encontramos con decenas de classes gestoras, helpers y otros que se comunican entre ellas. Y el problema crece y crece a medida que nuestra

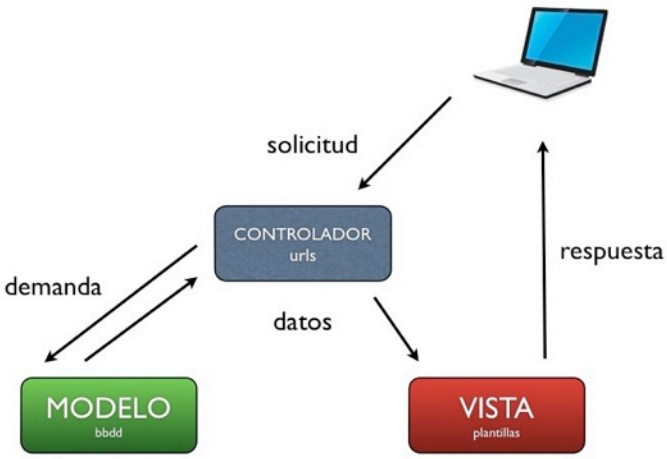
aplicación requiere más funcionalidades, corrección de errores, etc.



Horror.

MVC

Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que separa los datosde una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón de llamada y retorno MVC (según CMU), se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página. El modelo es elSistema de Gestión de Base de Datos y la Lógica de negocio, y el controlador es el responsable de recibir los eventos de entrada desde la vista. - Wikipedia



En la imagen de la izquierda se muestra el esquema de funcionamiento del modelo MVC.

Un ejemplo de funcionamiento para explicar la lógica del modelo MVC seria:

Queremos mostrar todos los productos de nuestro carrito de la compra. La url de acceso a dicha sección podría ser:

ejemplo.com/carrito/productos

MVC interpreta esta url de la siguiente forma:

ejemplo.com/carrito/productos

clase método

```
// 1.1
class Carrito_controller
{
  function productos()
  {
    $this->view->load('carrito');
  }
}
```

```
// 1.2
class Carrito_controller
{

  function agrega($id_producto)
  {
    $this->model('carrito')->agrega
($id_producto);
    $this->view->load('carrito');
  }

}
```

En la figura 1.1 se muestra cómo sería la clase y método que atiende a esta solicitud.

Como vemos, MVC ofrece un genial nivel de abstracción con el que creando tantos ficheros clase como secciones tiene la web podremos lograr un nivel de organización muy alto.

La línea $this->view->load(...) indica:

- **$this:** Sobre la clase en cuestion (la cual contiene todas las librerías de nuestro MVC)...
- **->view:** ... utilizar la librería que controla las vistas (la parte visual de la aplicación) ...
- **->load()**: indica el fichero de la vista a cargar.

Otra posible url sería por ejemplo aquella que agrega un producto a nuestro carro de la compra:

Esta vez el método de la clase carrito sería algo parecido al que se refleja en la figura 1.2

Se trata de un ejemplo adaptado al framework 'CodeIgniter', en el cual, los parámetros pasados por URL llegan al método según el orden de la url.

En la primera línea del método se carga el modelo 'carrito' (una clase) de la cual se llama al método agrega al cual se le pasa directamente el parámetro que recibe el controlador.

Fin

Autor: José Ramón Cid Constela
Email: jose@jrcconstela.com
Web: jrcconstela.com

Presentación disponible en: bit.ly/jrdaiphp