Gestión de la Información

David Corredor Lacha

Departamento de Lenguajes y Sistemas Informáticos

david.corredor@ua.es

Gestión de la Información

MVC El paradigma Modelo-Vista-Controlador en el diseño de aplicaciones Web

- 1) Introducción.
  - El Modelo.
  - La Vista.
  - El Controlador.
- 2) Aplicación en Frameworks y desarrollos Web sobre PHP.
- 3) Métodos y ejemplos de implementación.
  - Ejemplos de implementaciones en PHP.
  - Uso de URLs intuitivas y amigables

1 Introducción

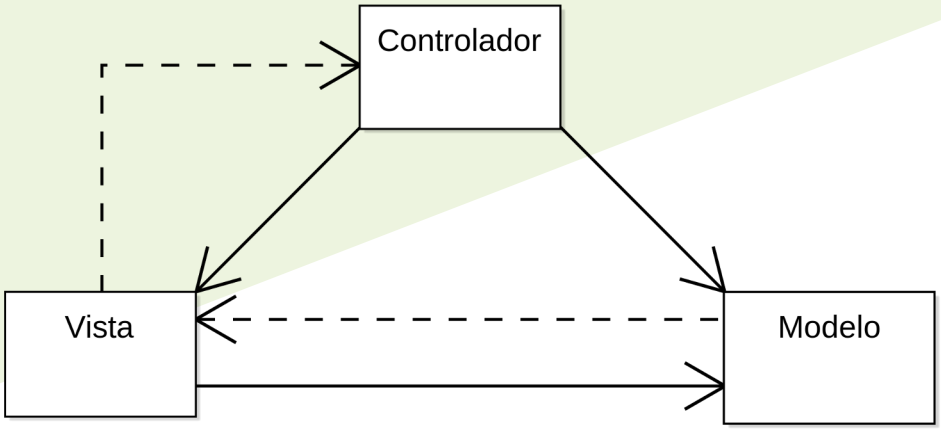
Introducción al MVC

- El MVC, o Model-View-Controller es un paradigma de arquitectura de software orientado a separar, en una aplicación determinada, el tratamiento/proceso de los datos/acciones, de la presentación de los resultados obtenidos.

- Fue una de las primeras ideas en el campo de las interfaces gráficas de usuario. Se usa desde hace décadas, y además tiene grandes ventajas a la hora de realizar proyectos de aplicaciones Web.

Introducción al MVC

- De forma general, el paradigma se podría definir como la interacción entre tres componentes. El ***Modelo***, que engloba las acciones orientadas al tratamiento de los datos. La ***Vista***, que engloba las acciones orientadas a presentar los datos obtenidos por el modelo, y el ***Controlador***, que se encarga de conectar e interactuar con los dos anteriores.



Introducción al MVC: El Módelo

- El Módelo podemos decir que generalmente es la parte “dura” de la arquitectura. Recibe unos determinados parámetros desde el Controlador, y realiza las tareas que se requieran (aritmética, tratamiento de ficheros, manejo de datos, interacción con bases de datos, etc…)

- Lo que no hace nunca el Modelo, es ocuparse de mostrar o presentar resultados. Simplemente realiza las acciones y calcula dichos resultados, enviándolos a la Vista (generalmente a través del Controlador) y dejando que esta se encargue de su presentación.

- Cuando el trabajo del Modelo finaliza, si ha generado algún tipo de información que necesita ser presentada, esta es enviada a la Vista para su tratamiento.

- Podríamos decir que el Modelo se encarga de toda la parte de BackOffice (la que no se ve).

Introducción al MVC: La Vista

- La Vista es la parte de la arquitectura que se encarga de presentar los datos al consumidor (usuario o sistema externo), y para ello los transforma o procesa para que sean presentados en la estructura y forma que se requiera.

- La Vista nunca realiza la obtención de los datos a presentar. Solo les aplica la estructura necesaria para que adopten el formato que se pide en su presentación

- Podríamos decir que la Vista se encarga de toda la parte Visual. Lo que se ve. (Al menos cuando está interactuando con un humano, pues si es el caso de una comunicación M2M (máquina a máquina), su trabajo es formatear de la forma pactada la información a enviar.)

Introducción al MVC: El Controlador

- El Controlador se puede definir como el alma de toda la estructura. Recibe del exterior eventos y datos, a través de los cuales debe decidir qué tarea debe realizar el modelo. Le llama y le pasa los datos necesarios (recibidos o almacenados) para que realice su tarea.

- Cuando el Modelo termina su tarea y devuelve resultados, el Controlador selecciona qué tarea de la Vista debe procesarlos, y le llama pasándole los parámetros recibidos para que los presente.

- Podríamos decir que el Controlador recibe órdenes desde el exterior y hace de intermediario entre el Modelo y la Vista para realizar las tareas solicitadas y comunicar los resultados.

2 Aplicación en FrameWorks y Desarrollos Web sobre PHP.

- La implementación más habitual se basa en el uso de una estructura de directorios estandarizada.

- Por un lado tenemos dos directorios. Uno para todos los módulos de tipo “modelo” y otro para todos los módulos de tipo “vista”. Por ejemplo los directorios “Modelos” y “Vistas”.

- Por otro lado tenemos un programa básico, el controlador, que será el que cargue los objetos de los modelos y las vistas según convenga.

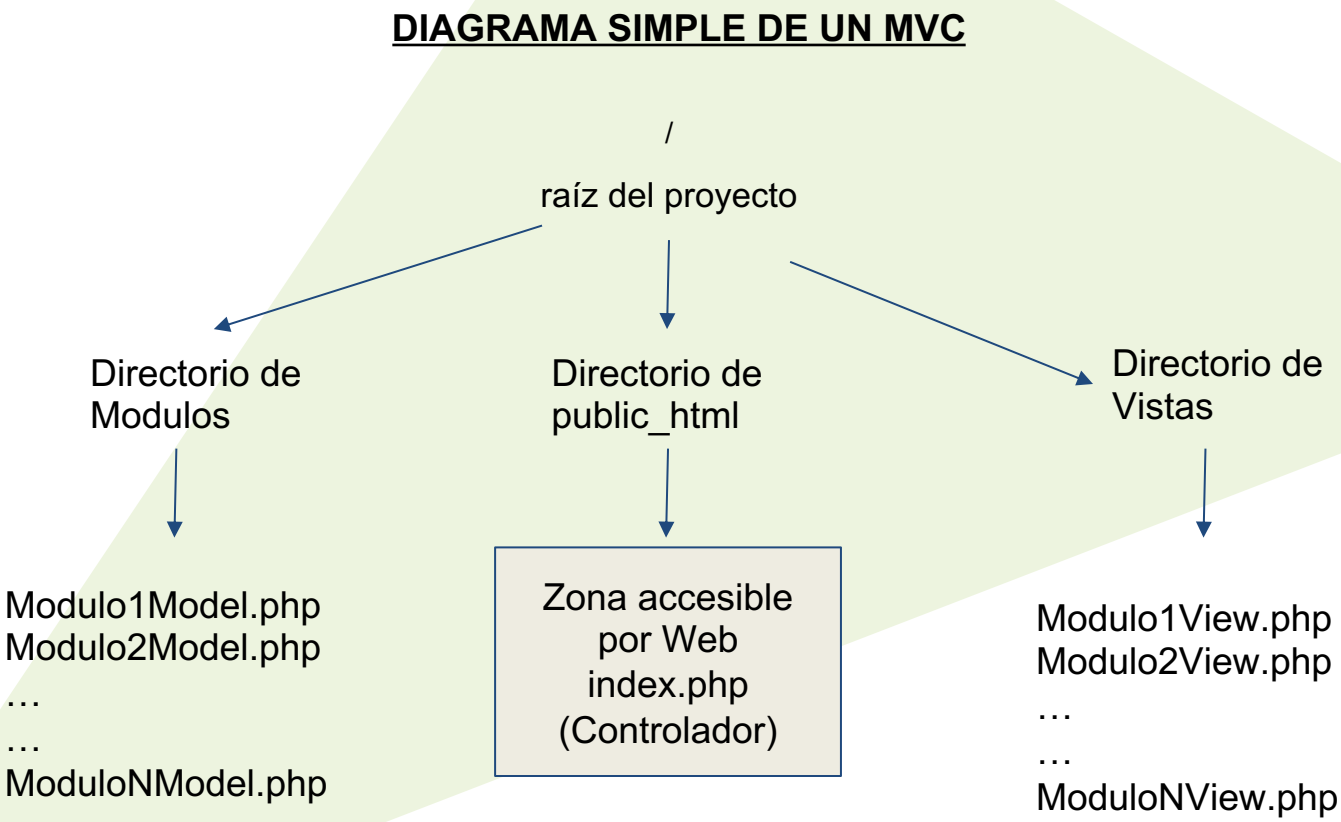
Aplicación en un desarrollo Web con PHP

- En PHP, los módulos y las vistas se traducen a Clases. En general, cada clase corresponde a una página web.

- Cada Clase tiene su propio archivo (una clase por archivo), el cual se suele llamar igual que la clase.

- Las clases de Vista y Modelo relacionadas, se suélen llamar igual, o a veces incluyendo algún acrónimo que las distinga. Por ejemplo MuestraClienteView y MuestraClienteModel

Aplicación en un desarrollo Web con PHP



Ventajas de la aplicación del MVC en un desarrollo Web

- Incrementa la seguridad: Solo es necesario tener un único archivo accesible desde la Web (el controlador).
- Simplifica el desarrollo y el mantenimiento.
- Permite el reaprovechamiento de los mismos módulos para distintas vistas, según el tipo de cliente (HTML, XML, JSON, WebServices, etc…)
- Acelera el desarrollo en equipos de trabajo, al necesitar mínima coordinación entre los desarrolladores de los módulos (backoffice) y los de las vistas (frontoffice)

3 Métodos y ejemplos de Implementación

Métodos y ejemplos de implementación

- Ya hemos visto las ventajas que nos ofrece el paradigma MVC, así como que básicamente, se trata de una estructura de directorios donde residen los módulos, y un controlador (normalmente el único archivo accesible por la web y que normalmente será el index.php), que los va utilizando.

- Pero ¿Cómo se implementa este método a nivel de programación PHP? Vamos a ver algunas formas generales de hacerlo, a las cuales se les pueden realizar pequeñas variaciones para adaptarlas a situaciones o proyectos específicos.

Ejemplos de implementaciones en PHP

- En el método más básico, el controlador (index.php) selecciona el método y vista a utilizar a través de un parámetro recibido mediante GET. Dicho parámetro es un string que estará relacionado (o se podrá relacionar mediante una tabla), con el módulo de Modelo/Controlador a utilizar. El resto de parámetros los podemos recoger en POST.

- Por ejemplo, si el parámetro es "*id*", un ejemplo de llamada al controlador podría ser "*index.php?id=MuestraFactura*". Esto haría que el controlador utilice el modelo asociado a ese id (por ejemplo Modelos/MuestraFactura.php), y para mostrar los resultados utilice la vista correspondiente (por ejemplo Vistas/MuestraFactura.php).

Ejemplos de implementaciones en PHP

- ¿Por qué el parámetro de selección de módulo se debe recoger en GET y no en POST?
  - Cuando se trabaja en un proyecto web es muy importante siempre tener una visión comercial de lo que estamos haciendo. Salvo aplicaciones privadas, normalmente nos va a interesar que nuestra web la visite mucha gente, y para eso es crucial llevarnos bien con los buscadores (google, yahoo, etc…)
  - Si recogemos el parámetro en POST en vez de con GET, la url de nuestra web no cambiaría nunca (siempre sería index.php), y eso a los buscadores no les gusta, y nos penalizarán en las búsquedas (tendremos menos visitas a nuestra web).

Ejemplos de implementaciones en PHP

Ejemplo de un index.php como controlador

Archivo index.php

```php
<?php
define("BASE_DIR","/Mi/Directorio/Raiz/Del/Proyecto");
define("MOD_DIR",BASE_DIR."/modelos/");
define("VIEW_DIR",BASE_DIR."/vistas/");

#Recoge por GET el identificador de la página a mostrar.
if (array_key_exists("ID",$_GET)){
   $pagina=$_GET["ID"];
   $datos=$_POST;
}else{ // Clase a cargar por defecto)
   $pagina="inicio";
   $datos=array();
}
#Carga el modelo y lo ejecuta
$modelo="${pagina}Model";
require (MOD_DIR."$modelo.php");
$m=new $modelo;
$resultado=$m->Ejecuta($datos);
#Carga la vista y la ejecuta
$vista="${pagina}Vista";
require (VIEW_DIR."$vista.php");
$v=new $vista;
$v->Ejecuta($resultado);
?>
```

Ejemplos de implementaciones en PHP

- En algunos casos nos puede interesar no pasar como parámetro los nombres exactos de los módulos, y utilizar una matriz de conversión interna para obtenerlos.

- Por ejemplo, podemos utilizar valores numéricos y relacionarlos con los módulos mediante la matriz.

Ejemplos de implementaciones en PHP

Ejemplo de un index.php como controlador con matriz de conversión

Archivo index.php

```php
<?php
define("BASE_DIR","/Mi/Directorio/Raiz/Del/Proyecto");
define("MOD_DIR",BASE_DIR."/modelos/");
define("VIEW_DIR",BASE_DIR."/vistas/");

#Array de conversión
$modulos=["1"=>"Inicio", "2"=>"MuestraDatos", "3"=>"VerFactura", "4"=>"Otros"];
#Recoge por GET el identificador de la página a mostrar.
if (array_key_exists("ID",$_GET)){
    $pagina=$modulos[$_GET["ID"]];
    $datos=$_POST;
}else{ // Clase a cargar por defecto)
    $pagina=$modulos["1"];
    $datos=array();
}
#Carga el modelo y lo ejecuta
$modelo="${pagina}Model";
require (MOD_DIR."$modelo.php");
$m=new $modelo;
$resultado=$m->Ejecuta($datos);
#Carga la vista y la ejecuta
$vista="${pagina}Vista";
require (VIEW_DIR."$vista.php");
$v=new $vista;
$v->Ejecuta($resultado);
?>
```

Ejemplos de implementaciones en PHP

Ejemplo de un módulo del directorio de Modelos

Archivo MuestraDatosModel.php

```php
<?php
class MuestraDatosModel{

   function Ejecuta($datos_in){

      /*Realiza todas las acciones necesarias con el array $datos_in
       ...
      */

      #Recoge los resultados en un array y lo devuelve
      $salida=["datoX"=>"ValorX", "datosY"=>"ValorY", "datosZ"=>"ValorZ"];
      return $salida;
   }
}
?>
```

Ejemplos de implementaciones en PHP

Ejemplo de un módulo del directorio de Vistas

Archivo MuestraDatosView.php

```
<?php
class MuestraDatosView{

   function Ejecuta($datos_in){

      /*Realiza todas las acciones necesarias previas a la salida de datos
       ...
      */

      #Muestra la página resultado con los datos
      $salida= <<<HTML
<HTML><BODY><H1>Datos de Salida</H1>
<table>
<tr><td>DATO</td><td>VALOR</td></tr>
<tr><td>datoX</td><td>${datos_in["datoX"]}</td></tr>
<tr><td>datoY</td><td>${datos_in["datoY"]}</td></tr>
<tr><td>datoZ</td><td>${datos_in["datoZ"]}</td></tr>
</table></BODY></HTML>
HTML;

      echo $salida;
   }
}
?>
```

Uso de URLs intuitivas y amigables para el .usuario

- Como hemos visto, el método anterior utiliza un parámetro GET para identificar qué módulo (o página web, pues en nuestro caso son sinónimos) debe de cargar el controlador. Esto es conocido como 'página web dinámica'. Es decir, es una página que cambia su salida dependiendo del parámetro pasado.

- Gracias al parámetro GET, conseguimos que las URLs de nuestro proyecto sean indexadas correctamente por los buscadores, pero aun así, estos nos van a penalizar a la hora de mostrar resultados, ya que sus algoritmos hoy día, a la hora de mostrar resultados de una búsqueda, premian las páginas estáticas frente a las dinámicas.

- Además de cara a un usuario, es mucho más amigable tener páginas de tipo miweb.com/MuestraDatos.html que miweb.com/index.php?ID=MuestraDatos

Uso de URLs intuitivas y amigables para el .usuario

¿Cómo podemos conseguir esto?

Mediante una pequeña configuración del servidor Web.

- Para ello tenemos que usar el módulo "***rewrite***" que hoy día incorporan todos los servidores web. Aquí vamos a ver un ejemplo para el servidor Apache, que es el más usual, pero con mínimos cambios se puede utilizar en cualquier otro servidor que tenga activo dicho módulo.

- **Importante:** El administrador del servidor debe tener instalado dicho módulo, y además permitir que los usuarios puedan usarlo dentro de los archivos de configuración local *".htaccess"* que se encuentran en los directorios de los proyectos. (Lo habitual es que por defecto esté instalado y su uso por los usuarios permitido).

Uso de URLs intuitivas y amigables para el .usuario

- El archivo *".htaccess"* (ojo al punto inicial) es un archivo "oculto" , es decir, que no se puede obtener a través del navegador (un usuario final no lo puede ver), y que se coloca en los directorios de nuestro proyecto, conteniendo instrucciones específicas del servidor web que afectan a dichos directorios y a sus directorios hijos. (p.e instrucciones de visibilidad, control de acceso, etc…)

- En el archivo *".htaccess"* de la raíz web de nuestro proyecto (donde se encuentra nuestro archivo *"index.php"*), colocaríamos las dos siguientes órdenes (si el archivo ya existe se añadirían):

```
RewriteEngine On
RewriteRule ^([a-zA-Z0-9]+).html$ index.php?ID=$1
```

Implementación con PHP y Configuración de Servidor

- **RewriteEngine On:** Activa el mod_rewrite para ese directorio.

- **RewriteRule ^([a-zA-Z0-9]+).html$ index.php?ID=$1:** Indica al servidor que cualquier página que se solicite con el formato indicado (un texto formado por letras y números acabado en .html), antes de procesarla, la traduzca como la página index.php?ID=EseTexto.

- Con lo anterior, conseguimos lo que pretendíamos: Por un lado que nuestras páginas sean más amigables tanto para el usuario como para los buscadores (son de la forma VerFactura.html, CrearUsuario.html, …). Y por otro lado, que nosotros podamos tener solo el controlador (index.php) accesible desde web y concentrar en él todas las páginas de nuestro proyecto.

- Las reglas de Rewrite pueden ser mucho más complejas, de forma que incluyan por ejemplo directorios que se traduzcan a valores de nuestro parámetro de selección de módulo.

RESUMEN

- Aunque no es de obligado uso en los proyectos Web, y tu no estas obligado a hacerlo, las ventajas del paradigma MVC han hecho que hoy día tanto la mayoría de frameworks PHP y proyectos Web a medida lo usen, ya que:

  - **Permiten una estandarización y fácil comprensión de un proyecto Web.**
  - **Permite la modularidad y la partición de la aplicación.**
  - **Incrementan la seguridad.**
  - **Consigue un mejor y más productivo trabajo en equipo, aplicando roles específicos en el desarrollo.**
  - **Reaprovecha código, permitiendo una fácil adaptación a cambios (por ejemplo un cambio del front-end no necesita un cambio del back-end.**

TRABAJOS A ENTREGAR

TAREAS DE TRELLO A ENTREGAR EL 2 de MAYO:

- **Esquema de Modelos (MVC)**
- **Esquema de Vistas (MVC)**
- **Controlador (Parte de Implementación PHP)**

Las entregas se realizarán dentro de la carpeta MVC del espacio compartido subiendo toda la estructura (archivos y directorios) del paradigma MVC desarrollado. La entrega debe ser funcional, comentada y bien estructurada, aunque no esté completamente finalizada.

Si no se desea utilizar el paradigma MVC, igualmente se deberá subir toda la estructura con las mismas condiciones que el anterior, así como una pequeña memoria descriptiva de su funcionamiento, que además formará parte del proyecto final.

Bibliografia

- https://es.wikipedia.org/wiki/Modelo-vista-controlador (Descripción general del MVC)
- https://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html (Primer contacto con el MVC)
- https://developers.google.com/search/blog/2008/09/dynamic-urls-vs-static-urls (Dynamic URLs vs. StaticURL for Google)
- https://httpd.apache.org/docs/2.4/mod/mod_rewrite.html (Descripción del módulo rewrite de Apache)