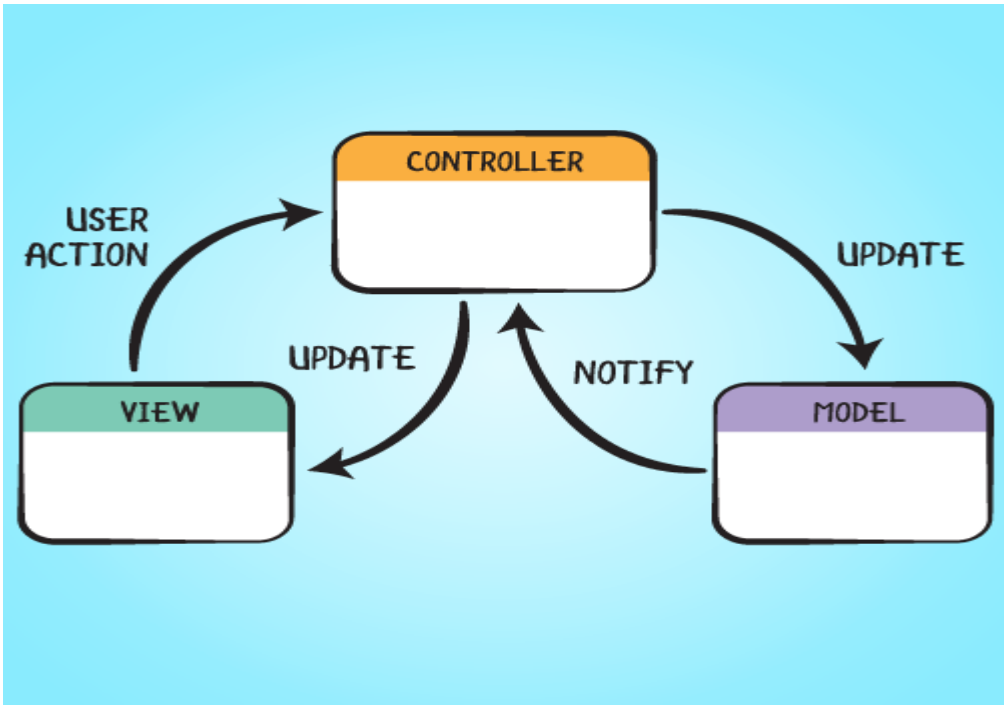
TEMA

5

2ºDAW

Modelo vista controlador



1 Introducción al MVC

El Modelo Vista Controlador (Model View Controller-MVC) es un patrón de diseño (de software) que se encarga de separar la lógica de negocio de la interfaz de usuario y es el mas utilizado en aplicaciones web, framework, etc, ya que facilita la funcionalidad, mantenibilidad, y escalabilidad del sistema, de forma comoda y sencilla, a la vez que ayuda no mezclar lenguajes de programación en el mismo código, el conocido “código espagueti”.

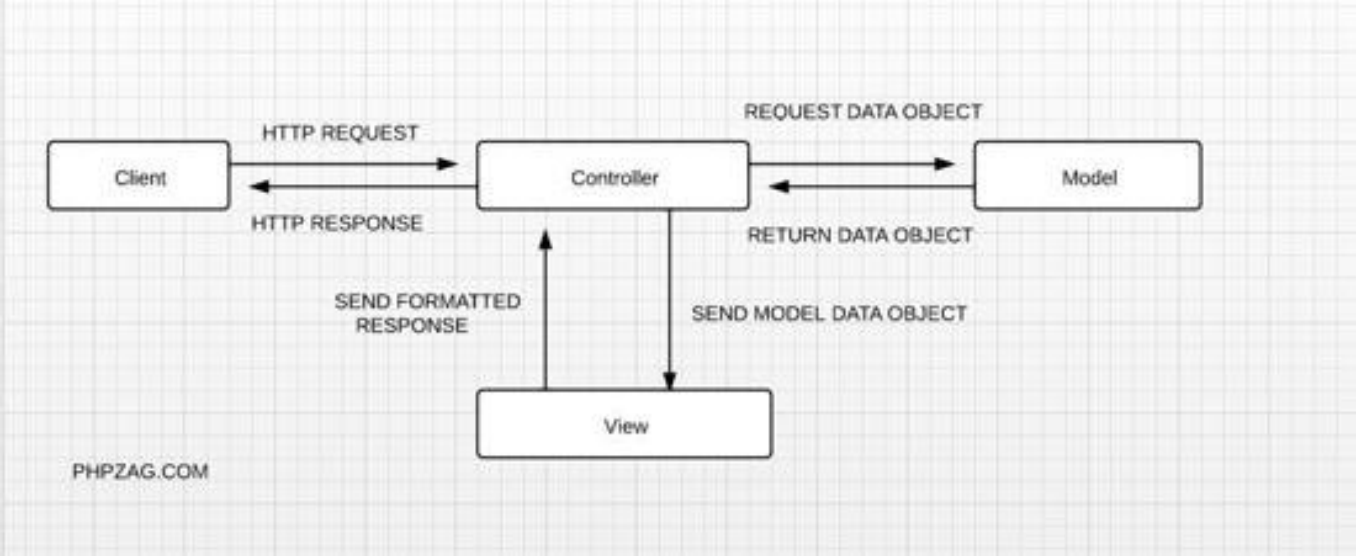
MVC divide las aplicaciones en tres niveles de abstracción:

1. **Modelo**: es la lógica de negocios. Es decir las clases y métodos que se comunican directamente con la base de datos.
2. **Vista**: es la encargada de mostrar la información al usuario, con de forma gráfica y legible.
3. **Controlador**: el intermediario entre la vista y el modelo, se encarga de controlar las interacciones del usuario en la vista, pide los datos al modelo y los devuelve de nuevo a la vista para que esta los muestre al usuario. Es decir las llamadas a clases y métodos, y los datos recibidos de formularios.

¿Como funciona el MVC?

El funcionamiento básico del patrón MVC, puede resumirse en:

1. El usuario realiza una petición.
2. El controlador captura la petición.
3. Hace la llamada al modelo correspondiente.
4. El modelo sera el encargado de interactuar con la base de datos.
5. El controlador recibe la información y la enviá a la vista.
6. La vista muestra la información.



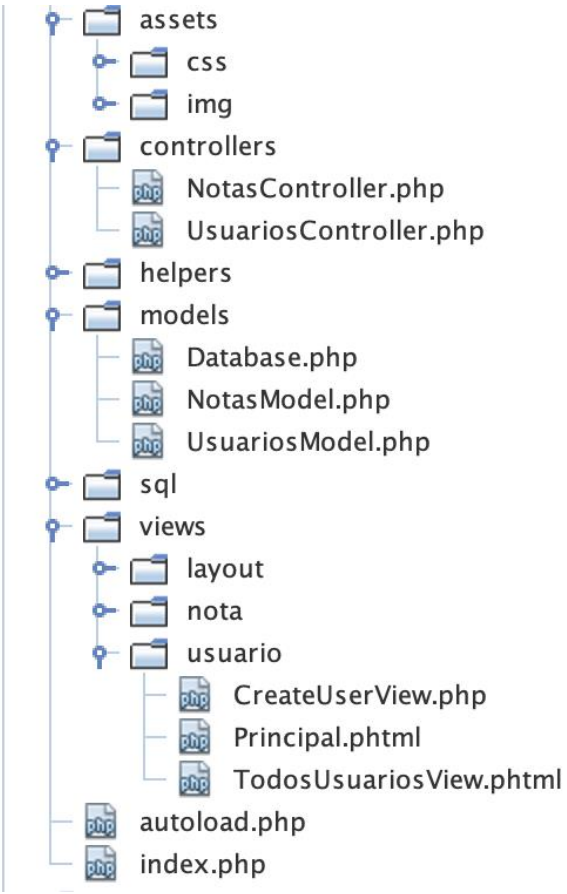
2 Como implementar el MVC en PHP

Para explicar el modelo vista controlador vamos a ver un ejemplo de como mostrar información de todos los usuarios de su tabla

```
CREATE TABLE usuarios(
id          int(255) auto_increment not null,
nombre      varchar(100) not null,
apellidos   varchar(100) not null,
email       varchar(255) not null,
password    varchar(255) not null,
fecha       date not null,
CONSTRAINT pk_usuarios PRIMARY KEY(id),
CONSTRAINT uq_email UNIQUE(email)
)ENGINE=InnoDb;

CREATE TABLE notas(
id              int(255) auto_increment not null,
usuario_id      int(255) not null,
titulo          varchar(255) not null,
descripcion     MEDIUMTEXT,
fecha           date not null,
CONSTRAINT pk_entradas PRIMARY KEY(id),
CONSTRAINT fk_entrada_usuario FOREIGN KEY(usuario_id) REFERENCES usuarios(id) ON DELETE CASCADE
)ENGINE=InnoDb;
```

Para implementar el MVC es imprescindible crear una estructura de ficheros parecida a esta:



En los frameworks, como por ejemplo Symfony, el patrón MVC viene dado de la siguiente manera: por cada entidad de la base de datos se crea un controlador, que no es más que una clase en PHP con sus respectivos métodos **llamadas acciones** (index, update, register etc..), estos métodos llaman bien sea a las **vistas** o a otros métodos del **modelo**.

En la carpeta **Views** (vista) se crea también varias subcarpetas por lo general **una por cada entidad de la base de datos de tu proyecto**, es decir si tienes una tabla que se llama usuarios, ventas etc., se debería crear una carpeta por cada entidad, ahora bien, dentro de cada carpeta, están las vistas, aunque no es un estándar pero vas a tener un archivo en el que mostrarás todos los usuarios, otro para editar un usuario etc., de acuerdo a tus necesidades.

Veamos un ejemplo típico del uso del MVC con PHP.

index.php

```php
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>bhkh</title>
        <link rel="stylesheet" href="assets/css/styles.css" />
    </head>
    <body>
        <?php
        session_start();
    // Clase necesaria para la conexión con la BD de todos los controladores
        require_once 'models/Database.php';
        require_once 'helpers/utils.php';
        //Carga todos los controladores automaticamente
        //require_once 'autoload.php';
        require_once 'controllers/UsuariosController.php';
    if(!isset($_GET['c'])|| !isset($_GET['a'])){

        $controlador = new UsuariosController();
        $controlador->index();
    }else{
        $nombre_controlador = $_GET['c'].'Controller';
        if(class_exists($nombre_controlador)){
            $controlador = new $nombre_controlador();
            if(method_exists($controlador, $_GET['a'])){
                $action = $_GET['a'];
                $controlador->$action();
        }else{
                echo "La pagina que buscas no existe";
        }
}else{
        echo "La pagina que buscas no existe";
}

    }?>
    </body>
</html>
```

Servicios web (web services)

database.php (Realizamos la conexión a la BD y devolvemos conexión)

```php
<?php

class  Database {
 private static $pdo_db = 'mysql:host=localhost;dbname=notas_master;charset=utf8';
 private static $pdo_user = 'root';
 private static $pdo_password = "pestillo";

public  function conectaDB()  {

try {
// mysql es el gestor y myDB se sustituye por el nombre de BD
    $conexion= new PDO(self::$pdo_db, self:: $pdo_user, self:: $pdo_password);
    // Establece los atributos de los reportes de errores
    $conexion->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    }
catch(PDOException $e)
    {
    echo ( "Error de conexión: " . $e->getMessage());
    }
return $conexion;
}
}
?>
```

model/UsuariosModel.php

```php
<?php

class UsuariosModel extends Database{
    private $id;
    private $nombre;
    private $apellidos;
    private $email;
    private $password;
    private $fecha;
    private $conn;

    public function __construct($nombre=null,$apellidos=null,$email=null,$password=null){

        $this->conn= parent::conectaDB();
        if (isset($nombre)){$this->nombre=$nombre;}
        if (isset($apellidos)){$this->apellido=$apellidos;}
        if (isset($email)){$this->email=$email;}
        if (isset($password)){$this->password=$password;}
    }
    public function get_all(){
        $consulta=$this->conn->query("SELECT * FROM usuarios ORDER BY id DESC");
       return $consulta;
    }
    public function save(){
                $sql = "INSERT INTO usuarios VALUES(NULL, '{$this->getNombre()}', '{$this->getApellidos()}', '{$this->getEmail()}', '{$this->getPassword()}', CURDATE());";
                $save = $this->conn->exec($sql);

                $result = false;
                if($save){
                    $result = true;
                }
                return $result;
        }

//Todas estas funciones set y get  se generan automaticamente
    function getId() {
        return $this->id;
    }

    function setId($id) {
        $this->id = $id;
    }

        function getFecha() {
        return $this->fecha;
    }

    function setFecha($fecha) {
        $this->fecha = $fecha;
    }


        function getNombre() {
        return $this->nombre;
    }

    function getApellidos() {
        return $this->apellidos;
    }
```

controller/UsuariosController.php

```php
<?php
require_once("models/UsuariosModel.php");
class UsuariosController{

        public function index(){

                require_once("views/usuario/Principal.phtml");
        }

        public function TodosUsuarios(){

           $usuario=new UsuariosModel();

           $todosUsuarios=$usuario->get_all();
           //Llamada a la vista
           require_once("views/usuario/TodosUsuariosView.phtml");
        }

        public function save(){

           if(!isset($_POST['submit1'])){
                require_once 'views/usuario/CreateUserView.php';
           }else{

                    $nombre = isset($_POST['nombre']) ? $_POST['nombre'] : false;
                      $apellidos = isset($_POST['apellidos']) ? $_POST['apellidos'] : false;
                          $email = isset($_POST['email']) ? $_POST['email'] : false;
                          $password = isset($_POST['password']) ? $_POST['password'] : false;

                          if($nombre && $apellidos && $email && $password){
                          $usuario = new UsuariosModel($nombre,$apellidos,$email,$password);
                                  // OTRA FORMA DE INSERTAR UN USUARIO
                              ////$usuario = new UsuariosModel();
                              //$usuario->setNombre($nombre);
                                  //$usuario->setApellidos($apellidos);
                                  //$usuario->setEmail($email);
                                  //$usuario->setPassword($password);

                                  $save = $usuario->save();
                                  if($save){
                                          $_SESSION['register'] = "complete";
                                  }else{
                                          $_SESSION['register'] = "failed";
                                  }
                          }else{
                                  $_SESSION['register'] = "failed";

                          }
                header("Location:index.php?c=Usuarios&a=save");
                        }

}}
```

El controlador contiene lo que se da en llamar la lógica de negocio de la aplicación debe tener siempre esta estructura llamada al modelo y a la vista, si hubiera mas modelos y vistas se sigue haciendo así con todos.

view/usuario/TodosUsuariosView.phtml

```php
    <?php

    while($user = $todosUsuarios->fetchObject() ){

        echo $user->nombre." - ".$user->email." - ". $user->fecha." <br/>";
    }
?>
```

view/usuario/CreateUserView.phtml

```php
<h1>Registrarse</h1>

<?=Utils::mostrarError('register'); ?>
<form action="index.php?c=Usuarios&a=save" method="POST">
        <label for="nombre">Nombre</label>
        <input type="text" name="nombre" required/>

        <label for="apellidos">Apellidos</label>
        <input type="text" name="apellidos" required/>

        <label for="email">Email</label>
        <input type="email" name="email" required/>

        <label for="password">Contraseña</label>
        <input type="password" name="password" required/>

        <input type="submit" value="Registrarse" />
</form>
```

Los ficheros de la vista según el estándar de Zend Framework debemos usar .phtml, pero se podría sin ningún problema usar la extensión .php
Muchos dicen que es recomendable usar CamelCase en los nombres de los ficheros y las clases, a efectos prácticos da igual usarlo que no, incluso algunos frameworks como Codeigniter nos sugieren que usemos nombres como «wellcome_model» por eso no he utilizado CamelCase. Si puedes y quieres abusa del CamelCase, porque así lo dicen los estándares.

helpers/utils.php

```php
<?php

class Utils{

        public static function mostrarError($name):string{
           $error="";
            if (isset($_SESSION[$name])){
                 if  ($_SESSION[$name] == 'complete'){
                     $error='<strong class="alert_green">Registro completado correctamente</strong>';
                }else{
                    $error='<strong class="alert_red">Registro fallido, introduce bien los datos</strong>';
                }
                    $_SESSION[$name] = null;
                    unset($_SESSION[$name]);

        }
        return $error;
}}
?>
```

Ejercicios

1) Utiliza el ejercicio del ejemplo
   a) Dentro del método index() de UsuarioController tendrás unas vista **principal.phtml** donde mostrarás las funciones del ejemplo en una lista ordenada: **Mostrar todos los usuarios, Crear usuario**
   b) Añade una opción más para que se pueda borrar un usuario a través de un enlace dentro del listado de los usuarios. Tendrás que añadir un nuevo método en el modelo de usuario. Cuando se borre el registro volveremos a mostrar la vista del listado.
   c) Para la tabla nota crea su modelo (NotasModel.php), controlador (NotasController) y vistas (2 vistas **listarTodasNotas.phtml y insertarNotas.phtml** dentro de la carpeta view/notas).
      - Añade el nuevo controlador a index.php
      - Amplia el menú principal con las 2 funciones : Listar todas las notas y insertar nota nueva
      - Cuando **insertes la nota nueva** introduce el id del usuario y no el nombre
      - Cuando **muestre todas las notas** muestra el nombre de usuario en lugar de su id
      - Los campo para insertar son usuario_id, titulo y descripción. La fecha introduce la actual con CURDATE().
      - Añade al modelo los get y set de cada uno de los campos.
      - Realiza un control de errores en el registro de notas mostrando un error general o si se ha insertado correctamente
   d) Para finalizar puedes añadir en el listado de usuarios junto al enlace de borrar otro enlace para listar las notas de ese usuario en concreto.

2) Tienes que una implementar un gestor de ofertas para una pizzería utilizando el Modelo Vista Controlador (MVC). La pantalla principal será un listado de las ofertas (con foto) y un icono de borrar y modificar al lado de cada una. Tambiénqq en esa pantalla habrá un enlace para insertar una nueva oferta

   a) Crea una tabla llamada oferta dentro de la BD pizzeria

```
CREATE TABLE IF NOT EXISTS oferta (
  id int(11) NOT NULL AUTO_INCREMENT,
  titulo varchar(200)  NOT NULL,
  imagen varchar(100)  NOT NULL,
  descripcion varchar(500)  NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB;
```

   b) Dentro de la carpeta Model tendrás 2 ficheros
      - PizzeriaDB.php para realizar la conexión con la BD
      - En OfertasModel.php tendrás funciones de Getter Setter de los (atributos), Insert, delete, getOfertas, getOfertasByid,modificar

   c) Dentro de Controller estará OfertaController.php con los siguientes métodos
      - Index : Mostrará la vista de todas las ofertas con los enlaces correspondientes
      - BorrarOferta
      - NuevaOferta: La vista del registro

- GrabarOferta. Antes de insertar los datos del formulario subiremos la imagen al servidor /uploads/img. Tendras que controlar si es para crear uno nuevo o modificar la oferta y ejercutar el método correspondiente.

```
Guardar la imagen
                if(isset($_FILES['imagen'])){
                        $file = $_FILES['imagen'];
                        $filename = $file['name'];
                        $mimetype = $file['type'];

                        if($mimetype == "image/jpg" || $mimetype == 'image/jpeg' || $mimetype == 'image/png' || $mimetype == 'image/gif'){

                                if(!is_dir('uploads/img')){
                                        mkdir('uploads/img', 0777, true);
                                }

                                move_uploaded_file($file['tmp_name'], 'uploads/img/'.$filename);

                        }
```

- Modificar: Comprueba si existe la variable id (si no reenvia al index) y consigue los datos de la oferta de ese ID y carga la vista del formulario

d) Dentro de la carpeta View/ofertas tendrá 2 vistas

- **RegistroView.phtml:** Formulario para insertar y modificar (Título - Text, Imagen- File, Descripción - Textarea).
  Antes de mostrar el formulario tendrás que comprobar si es para crear uno nuevo o para modificar. Tendrás que cambiar el action y las variables para los value del formulario. El action siempre cargará el método **grabarOferta** pero con id o sin el.
- **ListadoView.phtml:** Mostrara todas las ofertas con sus imágenes