ESTRUCTURA DE LAS APLICACIONES MVC

Nuestras aplicaciones tendrán:

- Una carpeta para *vistas* (ya la usábamos)
- Una llamada *clases,* donde pondremos todas las definiciones de clase de manera que cada clase estará en un fichero diferente con el nombre de la clase la extensión php (ejemplo Cliente.php)
- Una carpeta con los *controladores*.
- Una carpeta config con uno o varios ficheros de configuración.
- Un fichero principal *index.html* en la raiz de nuestra aplicación

nota: Ojo porque ahora si llamamos una vista desde un controlador la ruta la no es “vistas/mensaje.php” por ejemplo, sino que será “../vistas/mensaje.php”.

Para poder llevar esto a cabo, y alguna cuestión más a aplicar (como el uso del hash) debemos de aprender algunos conceptos nuevos

Ficheros de Configuración

Es conveniente crear un fichero de configuración que contenga los valores de la aplicación, para que sea facil de reutilizarla, cambiando este fichero (o ficheros). Existen muchas técnicas para esto Pero nosotros nos vamos a basar en el uso de constantes que es el más sencillo de implementar.
Así por ejemplo un fichero de constantes para la configuración de la base de datos *config.php* (es de momento el único que vamos a utilizar de momento) sería:

```
<?php
const BASE='virtualmarket';
const HOST='localhost';
const USUARIO ='root';
const PASS='';
const OPCIONES = array(PDO::MYSQL_ATTR_INIT_COMMAND => "SET
NAMES utf8", PDO::ATTR_ERRMODE =>
PDO::ERRMODE_EXCEPTION,PDO::ATTR_PERSISTENT =>true);
```

Este fichero habrá que incluirlo en la clase de la base de datos y la conexión pasaría a ser:

```
$this->link= new
PDO("mysql:host=".HOST.";dbname=".BASE,USUARIO,
PASS,OPCIONES);
```

Autocarga de classes

Suponemos que tenemos una carpeta, llamada *clases*, en la que hay un archivo para cada clase y que se llama como la clase pero con extenxión php (Ej: para la clase *Cliente*, tendremos un archivo *Cliente.php*). Con el código de abajo se cargará automáticamente cada vez que se haga `new Cliente.`

```
spl_autoload_register(function ($clase) {
    include "clases/$clase.php";
});
```

Almacenar y usar contraseñas con HASH.

No es seguro almacenar las contraseñas de usuarios en la base de datos porque un posible acceso no autorizado a ellas haría que se pudieran compartir y publicar.

En su lugar se almacena un HASH, es una secuencia de carácteres obtenida al aplicar un algoritmo sobre la contraseña, y que es imposible obtener la original en función de ella. Entre las muchas funciones que tine php para obtener el HASH, php.net recomienda el uso de `password_hash` y `password_verify()`.

El HASH debuelto con `password_hash` almacena información sobre el algoritmo que se ha usado y el *salt* utilizado, por lo tanto `password_verify` obtiene esta información del propio HASH. Esto hace que si el algoritmo que usamos habitualmente deja de ser seguro podemos usar otro nuevo y seguiría siendo válida nuestra aplicación.

Ejemplos de uso:

```
$hash=password_hash($pass, PASSWORD_DEFAULT);
```

En este ejemplo hemos cogido el password que tenemos (por ejemplo obtenido de un formulario de registro de usuarios) y obtenemos un hash a partir del el. $hash sería el valor a almacenar en la base de datos

```
if (password_verify($pass, $fila['pass'])) {
    echo '¡La contraseña es válida!';
} else {
    echo 'La contraseña no es válida.';
}
```

En este ejemplo $pass se obtiene de un formulario de validación y `$fila['pass']`es el password que obtenemos de la base de datos.