Bases de dades

Aquest document resumeix les operacions bàsiques que podem fer amb PHP i la llibreira PDO quan fem servir base de dades Mysql.

Suposarem sempres que tenim accés a la base de dades i que aquesta i les taules s'han creat ja, bé amb un script o bé mitjançant algun tipus d'utilitat.

Quan aprenem més de PHP o programació web veurem que hi ha altres maneres de treballar. Per ara ens centram en el mínim per poder conèixer el que són les disintes capes d'una aplicació web.

Connexión a la base de datos

Lo primero que debemos hacer antes de empezar a trabajar con las tablas es **conectarnos** a la base de datos, indicando la base de datos a la que nos vamos a conectar, usuario y clave.

Cada página que utilice la base de datos debe tener la conexión, por lo que es recomendable utilizar un archivo `include` para no tener que repetirnos

```
<?php
$pdo = new PDO('DSN', 'username', 'password');
?>

* `DNS` indica la conexión, normalmente `localhost` y el tipo de base de datos `mysql` e
* `username` es el nombre de usuario de la base de datos
* `password` clave de dicho usuario


### Ejemplo:

```php
<?php
$pdo = new PDO('mysql:host=localhost;dbname=test;charset=utf8', 'root', '');
?>
```

Esto abrirá una conexión para acceder a un motor de base de datos compatible con mysql que está en local, contra la base dedatos `test` accediendo comor `root` y sin clave.

Leer datos
```

Para leer datos haremos una consulta de selección contra la tabla que queramos obtener, de modo que vegan todas las columnas (`*`) o bien las que indiquemos en la selección.

Podemos filtrar los datos que queramos con la doncidicón `where` de `sql` .

Una vez establecida la conexión y ejecutado el código, PHP nos devolverá los resultados en forma de array asociativo, que podremos recorrer con un `foreach` . Las claves del array son los nombres de los campos de la tabla.

```
<?php
$sql = "SELECT * FROM users";
foreach ($pdo->query($sql) as $row) {
   echo $row['forename'] . " " . $row['surname'] . "<br />";
   echo "E-Mail: " . $row['email'] . "<br />";
   echo "Created at: " . $row['created_at'] . "<br /><br />";
}
?>
```

Leer una imagen

Guardar imágenes en la base de datos no es la opción más recomendable, pero si lo hacemos, podemos recuperarlas con el método anterior. Las imágenes se reciben codificadas en base64 por lo que para ser utilizadas deberemos descodificarlas.

Supongamos que hemos guardado una imagen en la tabla `users` anterior en el campo `image` . La imagen era un jpg. Para poderla imprimir debermos hacer:

```
<img src="<?php echo base64_encode($row['image']); ?>
```

Insertar datos

Podemos insertar datos de varias maneras, algunas más o menos seguras. Aquí veremos una que, si bien no es la más simple, es una de las más seguras, ya que impide la inyección de código.

Supongamos que tenemos los datos que vienen de nuestro formulario (POST), y ya hemos establecido la conexión con la base de datos en la variable `$pdo`

Los pasos son:

- preparar la consulta indicando los campos que se van a actualizar en `VALUES` (fijaos en los dos puntos)

- mapeamos las variables del sql con las variables de nuestro formulario o programa con `bindParam`
- finalmente ejecutamos el insert

Si todo va bien podemos hacer la redirección si venimos de un formulario (lo veremos más adelante), de momento imprimeremos un resultado.

```
// prepare SQL statement
$statement = $pdo->prepare("INSERT INTO users (forename, surname, email, password)
VALUES (:forename, :surname, :email, :password)");

// bind parameter
$statement ->bindParam(':forename', $forename);
$statement ->bindParam(':surname', $surname);
$statement ->bindParam(':email', $email);
$statement ->bindParam(':password', $password);

if ($statement->execute())
  // lo indicado es hacer la redirección
  echo "New record $forename created successfully<br>";
```

En algunas ocasiones nos interesará obtener el identificar del registro que hemos insertado. Esto se puede hacer

```
$last_insert_id = $pdo->lastInsertId();
echo 'Last Insert ID: '.$last_insert_id;
```

Esto es bastante típico cuando tenemos ids autoincrementales que no se envían por inser sino que es la propia base de datos los que los inserta.

Actualizar datos

La actualización de datos es muy similar a la inserción, salvo que tenemos que indicar qué registro o regisgtros queremos actualizar utilizando `where`

```php
$statement = $pdo -> prepare("UPDATE user SET password = :password WHERE id = :id");
$statement ->bindParam(':password', $password);
$statement ->bindParam(':id', $id);

// los parámetros pueden venir de un formulario
// en este caso debems comprobar siempre la validez
$id = 1;
$password = "YXDli_89%s";
// Ejecutamos
if ($statement ->execute()){
   echo "Clave actualizada";
} else {
   echo "SQL Error <br />";
   echo $statement->queryString."<br />";
   echo $statement->errorInfo()[2];
}
```

Borrar datos

Borrar se muy parecido a actualizar, ya que conviene indicar una condición a riesgo de borrar toda la tabla.

```php
// identificador del registro a borrar
// puede venir por código, por url o de un formulario

$publisher_id = 1;

// construimos la query
$sql = 'DELETE FROM publishers
        WHERE publisher_id = :publisher_id';

// preparamos y mapeamos los parámetros con las variables
$statement = $pdo->prepare($sql);
$statement->bindParam(':publisher_id', $publisher_id, PDO::PARAM_INT);

// ejecutamos
if ($statement->execute()) {
        echo 'publisher id ' . $publisher_id . ' ha sido borrado.';
}
```