PHP DATA OBJECTS – PDO

PDO es una interface de acceso a datos que nos permite, mediante varios drivers, conectarnos a diferentes bases de datos. Olvídate de esto, esto, esto e incluso de esto otro, ahora solo debes preocuparte por PDO. Esta librería escrita en C viene activada por defecto desde PHP 5.1 por lo cual la podrás utilizar en la mayoría de los servidores que actualmente soportan PHP5.

La conexión

Para todos los ejemplos utilizaré MySQL, pero también podria utilizar cualquier otra de las bases de datos soportadas adaptando un poco el código que sigue:

```
1 $db = new PDO('driver:host=servidor;dbname=bd', user, pass);
```

Y el ejemplo práctico:

```
1 $db = new PDO('mysql:host=localhost;dbname=pruebas', 'root', '');
```

Ahora en $db tenemos una instancia de PDO_MySQL

Primera consulta

Para la primer consulta haremos uso de prepare, execute y fetch.

```
1  require 'conexion.php';
2
3  //Nos conectamos
4  $db = new PDO('mysql:host=' . $servidor . ';dbname=' . $bd, $usuario, $contrasenia);
5
6  //Preparamos la consulta para dejarla lista para su ejecución
7  $consulta = $db->prepare('SELECT * FROM items');
8
9  //Ejecutamos la consulta
10 $consulta->execute();
11
12 //Recorremos el set de resultados mostrando la información
13 while($fila = $consulta->fetch())
14 {
15         echo $fila[0] . '  ' . $fila[1] . '<br />';
16 }
17
18 //Cerramos la conexión a la vez que destruimos nuestra instancia de PDO
19 $db = null;
```

Como verás no es nada complicado y es muy similar a lo que nos acostumbramos a hacer con las clásicas funciones mysql_.

Con las funciones MySQL también debíamos validar estrictamente los parámetros de entrada para evitar inyecciones SQL. En este caso, PDO lo hará por nosotros siempre y cuando utilicemos alguna de las varias formas que nos provee para realizar consultas parametrizadas. Este es un ejemplo:

```
1 //Preparamos la consulta marcando donde irán los parametros con ?
2 $consulta = $db->prepare('SELECT * FROM items WHERE id_item = ? OR id_item = ?');
3
4 //Ejecutamos la consulta incluyendo los parámetros en el mismo orden en el que deben
```

```
5 incluirse
  $consulta->execute(array(2, 4));
```

El ejemplo anterior generará una consulta de la siguiente manera:

```
1 SELECT * FROM items WHERE id_item = '2' OR id_item = '4'
```

Otra manera de hacer lo mismo:

```
1  $id = 6;
2
3  //Esta vez utilizamos un nombre-clave para cada parámetro
4  $consulta = $db->prepare('SELECT * FROM items WHERE id_item = :id');
5
6  //Con dicho nombre-clave, agregamos el valor del parámetro
7  $consulta->bindParam(':id', $id);
8
9  //Y ejecutamos la consulta
10 $consulta->execute();
```

```
1  SELECT * FROM items WHERE id_item =
   '6'
```

Ahora bien, si no confían, intenten inyectar SQL concatenando alguna sentencia en la variable $id y verán los resultados

Altas, Bajas y Modificaciones

El mecanismo sigue siendo el mismo que en las consultas anteriores, preparar la consulta, agregar los parámetros y ejecutar.

alta

```
1 $item = $_POST['item'];
2
3 $inserta = $db->prepare('INSERT INTO items (item) VALUES (:item)');
4 $inserta->bindParam(':item', $item);
5
6 $inserta->execute();
```

baja

```
1 $id = $_GET['id'];
2
3 $borra = $db->prepare('DELETE FROM items WHERE id_item = :id');
4 $borra->bindParam(':id', $id);
5
6 $borra->execute();
```

modificación

```
1 $item = $_POST['item'];
2 $id = $_POST['id'];
3
4 $actualiza = $db->prepare('UPDATE items SET item = :item WHERE id_item = :id');
5 $actualiza->bindParam(':item', $item);
6 $actualiza->bindParam(':id', $id);
```

```
7
8 $actualiza->execute();
```

Y esto es todo por el momento, solo un primer acercamiento a PDO. Podés bajarte todos estos ejemplos y varios mas desde aquí. Para que funcionen debes contar con un servidor que soporte PHP5 con las librerías PDO_MySQL instaladas, debes crear una base de datos, ejecutar el fichero items.sql y editar el archivo conexion.php con los datos que correspondan.