ALCYON
IT Services

LIBRERÍA PDO

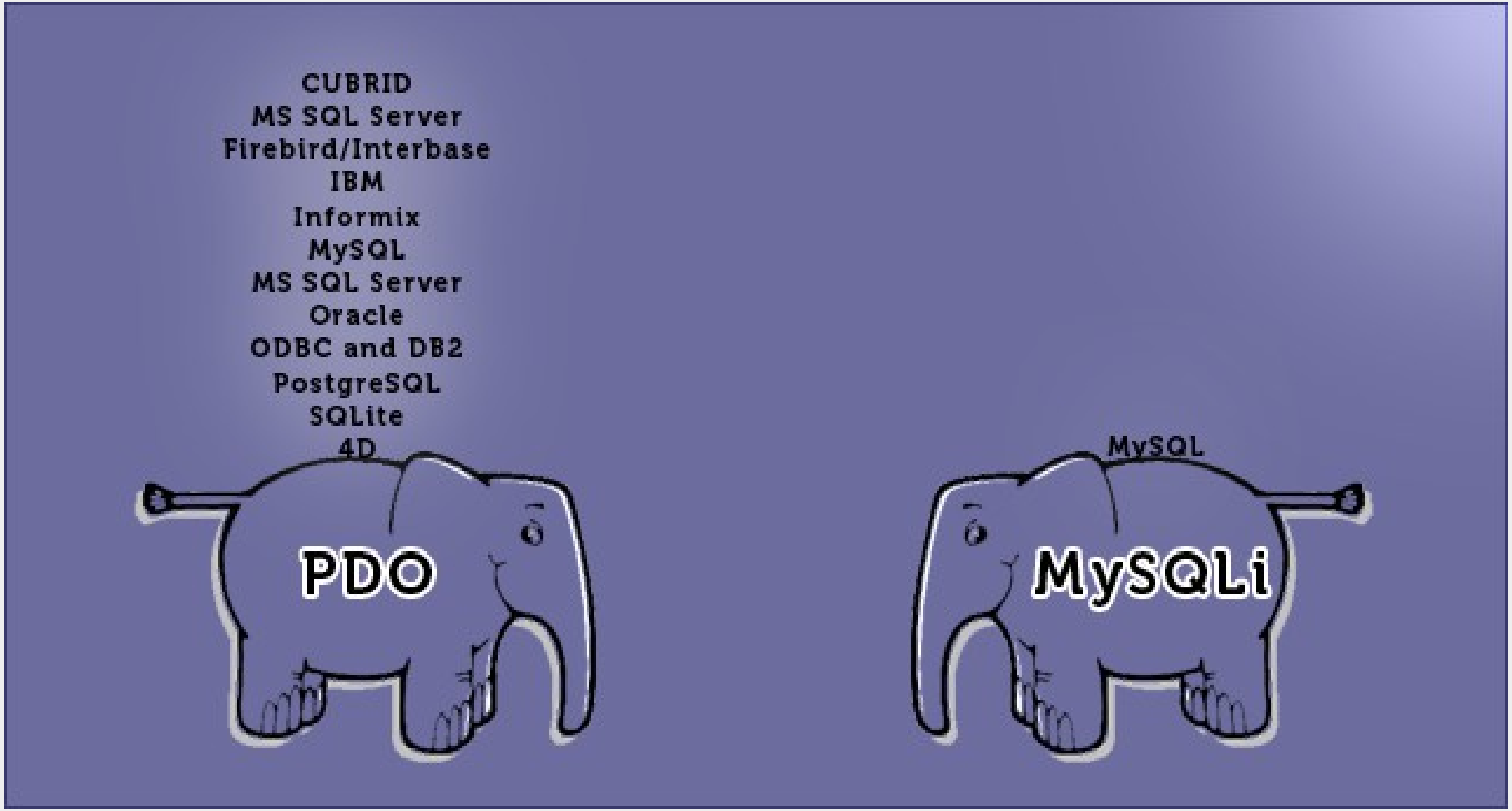
Fundamentos básicos

PDO (PHP Data Objects) es una extensión PHP para acceder a diferentes tipos de bases de datos (con el juego de instrucciones mysqli, tanto en su variante orientada a objetos como programación estructurada, solo podemos acceder a bases de datos basadas en mysql).

Proporciona una serie de métodos que siempre serán los mismos con independencia del SGDB que se esté utilizando

No estandariza SQL por lo que, dependiendo del sistema gestor se deberá seguir utilizando la sintaxis específica de las instrucciones de acceso a cada base de datos

ESQUEMA MYSQL/PDO



ALCYON
IT Services

HANDLERS

Manejadores o handlers

Para cada base de datos existe un manejador (driver) específico, que debe estar habilitado en el archivo de configuración de PHP (el archivo *php.ini*). Para mysql:

```
extension=php_pdo_mysql.dll
```

Si queremos saber cuantos drivers tiene instalada nuestra versión de php

```
var_dump( PDO::getAvailableDrivers() );
```

CLASES DE PDO

El sistema PDO se fundamente en tres clases:

PDO:

Encargada de mantener la conexión a la base de datos y manejo de transacciones

PDOStatement:

Encargada de manejar las sentencias SQL y devolver resultados

PDOException:

Encargada de manejar los errores.

ALCYON
IT Services

EXCEPCIONES I

PDO siempre trata los errores en forma de excepciones propias (clase PDOException) de forma que todas las sentencias de acceso a bbdd deberán ir encerradas en un bloque try/catch.

Si además necesitamos controlar nuestras propias excepciones, incluiremos un segundo bloque catch para capturarlas

```
try {
      …sentencias PDO…
      …sentencias propias…
} catch (PDOException $e){
      echo $e->getCode().' '.$e->getMessage();
} catch (Exception $e){
      echo $e->getCode().' '.$e->getMessage();
}
```

ALCYON
IT Services

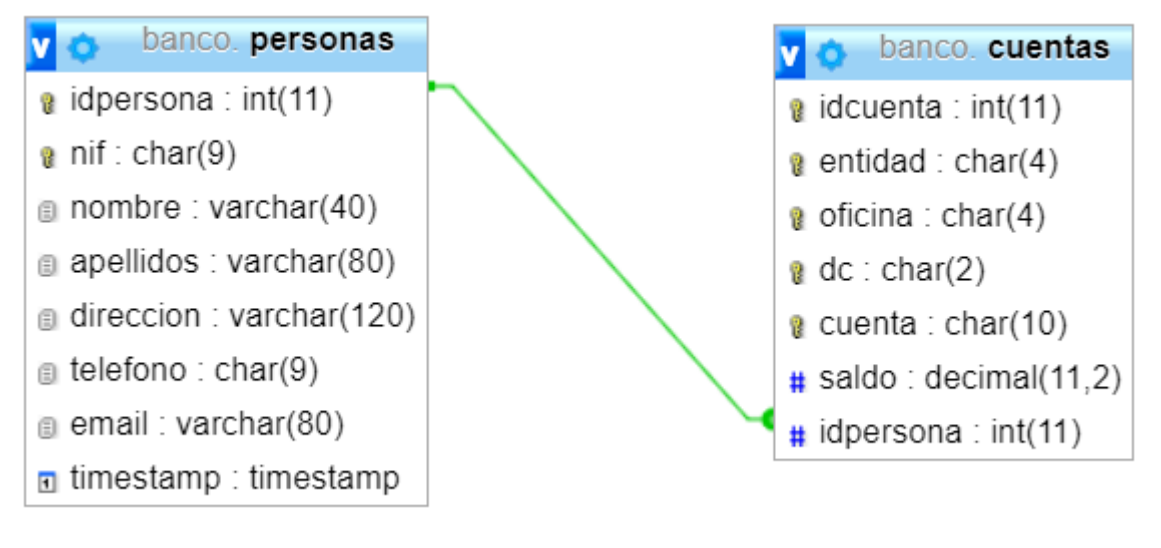
EXCEPCIONES II

Si necesitamos relanzar una excepción de la clase PDOException a una excepción de la clase nativa Exception tendremos que realizar un casting: Ejemplo

```
try {
      …sentencias PDO…
      …sentencias propias…
} catch (PDOException $e){
      throw new Exception((string)$e->getMessage(), (int)$e->getCode());
} catch (Exception $e){
      throw new Exception($e->getMessage(), $e->getCode());
}
```

EJEMPLO DE USO

Podemos utilizar la base de datos banco para ver los ejemplos de CRUD utilizando PDO



ALCYON
IT Services

CONEXIÓN

Para conectarnos a una base de datos necesitamos el DSN o Data Source Name (nombre y tipo de base de datos):

```
$dsn =
"mysql:host=localhost;dbname=nombre_bbdd;charset=UTF8";
```

Posteriormente definimos la variable con el manejador de la conexión y el modo de error

```
$dbh = new PDO($dsn, $usuario, $password);
```

```
$dbh->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
```

Toda la conexión ha de ir encerrada en un bloque try/catch ya que PDO maneja los errores en forma de excepciones

banco_conexión_bbdd

ALCYON
IT Services

INSERTAR DATOS CON PDO

ALCYON
IT Services

INSERTAR DATOS CON PDO I

PDO facilita el uso de sentencias preparadas en PHP, que mejoran el rendimiento y la seguridad de la aplicación.

Cuando se insertan datos, el esquema es: **PREPARE -> [BIND] -> EXECUTE**.

Se pueden indicar los parámetros en la sentencia con un interrogante "**?**" o mediante un **nombre específico**.

ALCYON
IT Services

INSERTAR DATOS CON PDO II

Utilizando parámetros para los valores:

```php
// Prepare
$stmt = $dbh->prepare("INSERT INTO Clientes (nombre, ciudad) VALUES (:nombre, :ciudad)");
// Bind
$nombre = "Charles";
$ciudad = "Valladolid";
$stmt->bindParam(':nombre', $nombre);
$stmt->bindParam(':ciudad', $ciudad);
// Excecute
$stmt->execute();
// Bind
$nombre = "Anne";
$ciudad = "Lugo";
$stmt->bindParam(':nombre', $nombre);
$stmt->bindParam(':ciudad', $ciudad);
// Execute
$stmt->execute();
```

banco_insert_personas

INSERTAR DATOS CON PDO IIb

Ejemplo:

```
//confeccionar la sentencia SQL
$sql = "INSERT INTO personas VALUES (NULL, :nif, :nombre,
:apellidos, :direccion, :telefono, :email, DEFAULT)";

//preparar la sentencia
$stmt = $dbh->prepare($sql);

//realizar el bind de los parámetros
$stmt->bindParam(':nif', $nif);
$stmt->bindParam(':nombre', $nombre);
…/...

//ejecutar la sentencia
$stmt->execute();
```

banco_insert_personas

ALCYON
IT Services

INSERTAR DATOS CON PDO III

Utilizando interrogantes para los valores:

```php
// Prepare
$stmt = $dbh->prepare("INSERT INTO Clientes (nombre, ciudad) VALUES (?, ?)");
// Bind
$nombre = "Peter";
$ciudad = "Madrid";
$stmt->bindParam(1, $nombre);
$stmt->bindParam(2, $ciudad);
// Excecute
$stmt->execute();
// Bind
$nombre = "Martha";
$ciudad = "Cáceres";
$stmt->bindParam(1, $nombre);
$stmt->bindParam(2, $ciudad);
// Execute
$stmt->execute();
```

INSERTAR DATOS CON PDO IIIb

Ejemplo:

```
//confeccionar la sentencia SQL
$sql = "INSERT INTO personas VALUES (NULL, ?, ?, ?, ?, ?, ?,
DEFAULT)";

//preparar la sentencia
$stmt = $dbh->prepare($sql);

//realizar el bind de los parámetros
$stmt->bindParam(1, $nif);
$stmt->bindParam(2, $nombre);
…/...

//ejecutar la sentencia
$stmt->execute();
```

banco_insert_personas

INSERTAR DATOS CON PDO IV

Si necesitamos recuperar el id asignado a la última clave primaria podemos utilizar la expresión siguiente después del execute():

```
$id = $dbh->lastInsertId();
```

Si queremos controlar el error de registro duplicado (código 1062):

```
try {
      …sentencias PDO…
} catch (PDOException $e) {
      if ($e->errorInfo[1]==1062) {
            echo “persona ya existe en la base de datos";
      } else {
            echo $e->getMessage();
      }
 }
```

banco_insert_personas

ALCYON
IT Services

BINDPARAM Y BINDVALUE I

Existen dos métodos para enlazar valores:

Con bindParam() la variable es enlazada como referencia y sol será evaluada cuando se llame a execute()

```php
// Prepare:
$stmt = $dbh->prepare("INSERT INTO Clientes (nombre) VALUES (:nombre)");
$nombre = "Morgan";
// Bind
$stmt->bindParam(':nombre', $nombre); // Se enlaza a la variable $nombre
// Si ahora cambiamos el valor de $nombre:
$nombre = "John";
$stmt->execute(); // Se insertará el cliente con el nombre John
```

ALCYON
IT Services

BINDPARAM Y BINDVALUE I

Existen dos métodos para enlazar valores:

Con bindValue() se enlaza directamente el valor de la variable y permanece hasta execute()

```php
// Prepare:
$stmt = $dbh->prepare("INSERT INTO Clientes (nombre) VALUES (:nombre)");
$nombre = "Morgan";
// Bind
$stmt->bindValue(':nombre', $nombre); // Se enlaza al valor Morgan
// Si ahora cambiamos el valor de $nombre:
$nombre = "John";
$stmt->execute(); // Se insertará el cliente con el nombre Morgan
```

ALCYON
IT Services

QUERY Y EXECUTE

El método execute() exige que la sentencia SQL se haya confeccionado previamente utilizando prepare(), tal como hemos visto en los ejemplos anteriores.

El método prepare() tiene la ventaja que realiza el escape de los valores de forma automática además de ser mucho más eficiente.

El método query(). ejecuta la sentencia directamente y necesita que se escapen los datos adecuadamente para evitar inyección SQL y otros problemas.

ALCYON
IT Services

EJEMPLO USO DE QUERY

Un ejemplo de uso de query() podría ser el siguiente

1. confeccionar la sentencia

```
$sql = "INSERT INTO personas VALUES (NULL, '$nif', '$nombre',
'$apellidos', '$direccion', '$telefono', '$email', DEFAULT)";
```

2. ejecutar la sentencia

```
$dbh->query($sql);
```

3. Para controlar un error de registro duplicado en el catch de la clase PDOException:

```
if ($e->errorInfo[1] == 1062) {
      echo 'Nif ya existe';
}
```

ALCYON
IT Services

CONSULTAR DATOS CON PDO

ALCYON
IT Services

CONSULTA I

Para consultar datos se utiliza la sentencia fetch que devuelve una fila de un conjunto de resultados.

Previamente a la ejecución de execute() hay que:

- Construir la sentencia prepare()

```
$stmt = $dbh->prepare("SELECT * FROM personas WHERE nif=:nif);
```

- Bind de los parámetros (en caso de utilizar filtros con WHERE)

```
$stmt->bindParam(':nif', $nif);
```

- Especificar como se quieren devolver los datos:

```
$stmt->setFetchMode(PDO::FETCH_ASSOC);
```

banco_select_personas

ALCYON
IT Services

CONSULTA II

Podemos prescindir de la instrucción anterior en cada SELECT y especificar en el fichero de conexión la estructura del array que necesitamos para los datos:

```
$dbh->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE,
PDO::FETCH_ASSOC);
```

ALCYON
IT Services

CONSULTA III

Los parámetros más comunes son:

PDO::FETCH_ASSOC: devuelve un array indexado cuyos keys son el **nombre de las columnas**.
PDO::FETCH_NUM: devuelve un array indexado cuyos keys son **números**.
PDO::FETCH_BOTH: valor por defecto. Devuelve un array indexado cuyos keys son tanto el **nombre de las columnas** como **números**.

- Ejecutar la sentencia

```
$stmt->execute();
```

- Posteriormente ejecutamos el método fetch dentro de un while para tratar cada una de las filas devueltas en la consulta:

```
while ($fila = $stmt->fetch()){
    echo "$fila[nif] $fila[nombre] $fila[apellidos]<br>";
}
```

banco_select_personas

ALCYON
IT Services

CONSULTA IV

Otra opción para recuperr los datos es utilizar fetchAll que nos extrae todo el array de filas y columnas de una sola vez:

```
$filas = $stmt->fetchAll()
```

Si necesitamos conocer el número de filas devueltas en la consulta:

```
echo $stmt->rowCount();
```

banco_select_personas

ALCYON
IT Services

CONSULTA V

Otros parámetros interesantes de setFetchMode son:

PDO::FETCH_BOUND: asigna los valores de las columnas a las variables establecidas con el método PDOStatement::bindColumn
PDO::FETCH_CLASS: asigna los valores de las columnas a propiedades de una clase. Creará las propiedades si éstas no existen.
PDO::FETCH_INTO: actualiza una instancia existente de una clase.
PDO::FETCH_OBJ: devuelve un objeto anónimo con nombres de propiedades que corresponden a las columnas.
PDO::FETCH_LAZY: combina **PDO::FETCH_BOTH** y **PDO::FETCH_OBJ**, creando los nombres de las propiedades del objeto tal como se accedieron.

ALCYON
IT Services

CONSULTA VI

Si necesitamos realizar una consulta sin bind de parámetros:

- Confeccionamos la sentencia

```
$sql = "SELECT * FROM personas”
```

- Ejecutamos la sentancia

```
$stmt = $this->dbh->query($sql);
```

- Para recuperar las filas

```
$filas = $stmt->fetch()
$filas = $stmt->fetchAll()
```

- Para comprobar si hemos recuperado alguna fila

```
if (!$filas) {echo 'No se han recuperado datos'}
```

banco_select_personas

ALCYON
IT Services

MODIFICAR DATOS CON PDO

ALCYON
IT Services

MODIFICACIÓN I

Para modificar datos, al igual que con insert, debemos preparar la sentencia previamente con prepare()

```
$stmt = $dbh->prepare("UPDATE personas SET nombre=:nombre WHERE idpersona=:id")
```

Hacer el bind de los parametros a modificar y los utilizados en el WHERE

```
$stmt->bindParam(':id', $id);
$stmt->bindParam(':nombre', $nombre);
```

Y ejecutar la sentencia con execute():

banco_update_personas

ALCYON
IT Services

MODIFICACIÓN II

Si necesitamos conocer el número de filas afectadas en la modificación:

```
echo $stmt->rowCount();
```

Recordad comprobar en el control de errores el código 1062 que indicaría que hemos modificado una clave primaria asignandole un valor ya existente en otro registro de la base de datos

banco_update_personas

ALCYON
IT Services

BAJA DE DATOS CON PDO

BAJA I

Para dar de baja datos:

```
$stmt = $dbh->prepare("DELETE FROM personas WHERE idpersona=:id ")
```

Hacer el bind de los valores a modificar

```
$stmt->bindParam(':id', $id);
```

Y ejecutar la sentencia con execute():

```
$stmt->execute();
```

Si queremos saber el número de filas borradas:

```
$numfilas = $stmt->rowCount();
```

banco_delete_personas

BAJA II

Si queremos controlar el error de restricción de clave foránea (código 1451):

```
try {
     …sentencias PDO…
} catch (PDOException $e) {
     if ($e->errorInfo[1]==1451) {
          echo “persona con cuentas no puede darse de baja";
     } else {
          echo $e->getMessage();
     }
 }
```

banco_delete_personas

BAJA III

Si no necesitramos realizar el bind:

```
$sql = "DELETE FROM personas WHERE idpersona=$id "
```

Y ejecutar la sentencia con query():

```
$stmt = $dbh->query($sql);
```

Aviso Legal

David Alcolea
david-alcolea@alcyon-it.com
www.alcyon-it.com