UD3 : Acceso a Bases de Datos
Extensión PDO.
[PHP Data Objects]

Acceso a datos desde PHP: PDO

- PHP Data Objects (PDO) define una interfaz ligera y consistente para acceder a bases de datos en PHP.
- Implementada con tecnología orientada a objetos.
- A partir de PHP 5.1 se considera estable y la interfaz viene incluida por defecto.

Acceso a datos desde PHP: PDO

- Ventajas
  - Permite abstraernos del tipo de gestor al que conectar.
  - Se encarga de desinfectar los datos automáticamente al utilizar consultas preparadas.
  - Proporciona el uso de transacciones.

Acceso a datos desde PHP: PDO

- Las clases implicadas directamente en el acceso a datos con esta librería:
  - PDO
    - Crea una instancia de PDO que representa una conexión a una base de datos
  - PDOStatement
    - Representa una sentencia preparada y, después de la ejecución de la consulta, un conjunto de resultados asociado.
  - PDOException
    - Representa un error generado por PDO

Extensión PDO: Conexión a bd

- La realizamos con el constructor de la clase PDO

$con = new PDO($cadenaDSN, $usuario, $password,$opciones)

$cadenaDSN (data source name) es de la forma:

"gestorBD:host=descrip_host; dbname=nombreBD"

```
$con = new PDO ('mysql:host=localhost; dbname=pizzeria', $usuario, $pwd);
```

Conexiones persistentes

- Las conexiones persistentes no se cierran al final del script, son reutilizadas cuando otro script solicite una conexión que use las mismas credenciales
- Permite evitar la carga adicional de establecer una nueva conexión cada vez que un script necesite comunicarse con la base de datos, dando como resultado una aplicación web más rápida
- Para establecer una conexión persistente: utilizaremos la siguiente opción al realizar la conexión:

PDO::*ATTR_PERSISTENT* => true

Otras opciones de la conexión

- En el array de opciones que le pasamos al crear la conexión podemos utilizar, entre otras, las siguientes:
  - Para que extraiga los datos de la base de datos en utf8:
    - PDO::*MYSQL_ATTR_INIT_COMMAND* => "SET NAMES utf8"
  - Para que devuelva los errores en forma de excepciones:
    - PDO::*ATTR_ERRMODE* => PDO::*ERRMODE_EXCEPTION*

Extensión PDO: Conexión a bd

```
$opciones = array(
  PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8",
  PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
  PDO::ATTR_PERSISTENT => true
);
$con = new PDO(
  'mysql:host=localhost;dbname=agenda;charset=utf8',
  'root',
  '',
  $opciones
);
```

Extensión PDO: Consultas

- La librería PDO proporciona dos métodos para realizar consultas:
- Consultas no preparadas

  PDO::query(string $consulta)

  - Utilizado cuando las consultas no contienen parámetros externos
  - No se escapan automáticamente los parámetros
  - Devuelve un objeto PDOStatement o False

```
$result=$con->query('SELECT nombre FROM ingredientes');
```

Extensión PDO: Consultas

- Consultas preparadas
  - Escapa los parámetros de la consulta automáticamente.
  - Devuelve un objeto PDOStatement ó False
  - La consulta se ejecuta en tres pasos:
    - Preparar la consulta

      PDO::prepare(string $consulta)
    - Vincular un valor a los parámetros de sustitución

      PDOStatement::bindValue(':parametro', $valor)

      PDOStatement::bindParam(1, $valor)
    - Ejecutar la consulta

      PDOStatement::execute()

Extensión PDO: Consultas preparadas

```
$result = $con->preprare('SELECT * FROM masas WHERE idMasa=:masa and precio=:prec');

$result->bindValue(':masa', $masa);

$result->bindValue(':prec', $precio);

$result->execute();
```

Extensión PDO: Recogida de datos

- Existen tres métodos principales para acceder al resultado de una ejecución:
  - Obtener la siguiente fila de un conjunto de resultados

    PDOStatement::fetch($modo)
  - Obtener un array bidimensional con todas las filas del resultado

    PDOStatement::fetchAll($modo)
  - Obtener la siguiente fila y devolverla como un objeto

    PDOStatement::fetchObject(string nombreClase)

Extensión PDO: Recogida de datos

```
$result = $con->preprare('SELECT * FROM masas WHERE idMasa=:masa and
precio=:prec');
$result->bindValue(':masa', $masa);
$result->bindValue(':prec', $precio);
$result->execute();
while ($fila = $result->fetch(PDO::FETCH_ASSOC)) {
   echo $fila['idMasa']. " ".$fila['precio']."<br>";
}
```

PDO: Otras funciones útiles

- Obtener el id de la última fila insertada en la bd:

  public string PDO::lastInsertId([string $name])

- Obtener el número de filas afectadas por la última inserción, actualización o borrado

  public int PDOStatement::rowCount()

PDO: Resumen

```
try{
    $opciones = array(PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES
     utf8", PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
     PDO::ATTR_PERSISTENT => true);
    $con = new PDO('mysql:host=localhost;dbname=virtualmarket', 'root', '',
$opciones);

    $result = $con->prepare('SELECT * FROM productos WHERE origen=:origen
and precio=:prec');
    $result->bindParam(':origen', $origen);
    $result->bindParam(':prec', $precio);
    $origen='Italia';
    $precio=1;
    $result->execute();

    while ($fila = $result->fetch(PDO::FETCH_ASSOC)) {
        echo $fila['idMasa']. " ".$fila['precio']."<br>";
     }
}catch (PDOException $e){
            echo $e->getMessage();
}
```

Transacciones

- Por defecto PDO trabaja en modo autocommit
- Confirma de forma automática cada sentencia que ejecuta el servidor
- Para trabajar con transacciones, PDO incorpora tres métodos:
  - beginTransaction. Deshabilita el modo autocommit y comienza una nueva transacción, que finalizará cuando ejecutes uno de los dos métodos siguientes
  - commit. Confirma la transacción actual
  - rollback. Revierte los cambios llevados a cabo en la transacción actual
- Una vez ejecutado un commit o un rollback, se volverá al modo de confirmación automática

Ejemplo

```
try{
        $pdo->beginTransaction();
        $pdo->exec('DELETE ...');
        $pdo->exec('UPDATE ...');
        $pdo->commit();
}
catch(PDOException $exception) {
        $pdo->rollback();
}
```

Separar Modelo

Clase base de datos

```
class Bd
{
        private $link;
        function __construct(){
          try{
              $opciones = array(PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8",
              PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,PDO::ATTR_PERSISTENT =>
              true);
              $this->link= new PDO("mysql:host=localhost;dbname=virtualmarket", "root", "",
              $opciones);
          }catch(PDOException $e){
              $dato= "¡Error!: " . $e->getMessage() . "<br/>";
              require "vistas/mostrar.php";
              die();
           }
        }
        function __get($var){
            return $this->$var;
        }
}
```

Separar Modelo

Clase producto

```
Class Producto{
        private $origen;
        private $precio;

        function __construct($origen,$precio){
                $this→origen=$origen;
                $this→precio=$precio;
        }
        static function buscar($con){
                try{
                        $result = $con->prepare('SELECT * FROM productos WHERE origen=:origen and
                       precio=:prec');
                        $result->bindParam(':origen', $this->origen);
                        $result->bindParam(':prec', $this->precio);
                        $result->execute();
                        return $result;
               }catch (PDOException $e){
                        echo $e->getMessage();
                 }
        }
}
```