

UT5

**UTILIZACIÓN DE TÉCNICAS
DE ACCESO A DATOS**

EXTENSIÓN

PHP DATA OBJECTS

Acceso a datos desde PHP

- De las formas de almacenamiento permanente de datos se encuentran:
 - Archivos
 - Texto, binarios, XML, JSON, ...
 - **Bases de datos**
 - **Relacionales:** MySQL, Oracle, SQL Server,
 - NoSQL (*Not only SQL*: **MongoDB**, CouchDB, db4O, ...)

Acceso a datos desde PHP: ficheros

- ❑ La interacción automática entre nuestras páginas PHP y los **archivos** siguen siempre estos tres pasos consecutivos:
 - ❑ **Abrir** el archivo
 - ❑ Hacer alguna **operación** con el contenido de ese archivo
 - ❑ **Cerrar** el archivo.

Acceso a datos desde PHP: apertura ficheros

- ❑ La apertura de archivos se realiza con la función **fopen** que admite dos parámetros:
 - ❑ Ruta hacia el archivo que se abre
 - ❑ Modo de apertura
 - ❑ **r**: abre el archivo para **leer** sus datos.
 - ❑ **w**: abre el archivo para **escribir** en él. Sobrescribe todo lo que haya. Si no existiera el archivo se intentaría crear.
 - ❑ **a**: se abre para **añadir** datos al final del archivo. Si el archivo no existe, se crea.
 - ❑ **r+**: abre para **añadir** datos y además permite **leerlos**.
 - ❑ **w+**: abre para **escribir** y luego **leer**. Si el archivo existía, suprime todo lo que hubiera previamente. Si el archivo no existe, lo crea.
 - ❑ **a+**: abre para **añadir** y **leer**. Añade los datos al final del archivo, sin eliminar el contenido previo. Si el archivo no existe, lo crea.

Acceso a datos desde PHP: cierre ficheros

- ❑ Terminadas las operaciones a realizar en el archivo se procede a cerrarlo para que no ocupe memoria de forma innecesaria.
- ❑ El cierre de un archivo se realiza con la función **fclose** que admite como parámetro el identificador del recurso abierto previamente con *fopen*.

archivo.txt y apertura.php

Archivo de texto *archivo.txt*
Ejemplo para ser usado en PHP.

Se harán lecturas, modificaciones,
inserciones, etc.

```
<?php
$fp = fopen("archivo.txt", "r");
if ( $fp ) {

    // Realizar operaciones
    // ...

    fclose($fp);
}
else {
    echo "Error al abrir el archivo.";
}
```

Acceso a datos desde PHP: lectura archivos

- ❑ Un fichero de texto se puede leer de varias maneras:
 - ❑ Línea por línea
 - ❑ Un carácter o un conjunto de caracteres
 - ❑ Una única línea de texto
 - ❑ Una única línea de texto omitiendo etiquetas HTML y/o PHP
 - ❑ El archivo entero.

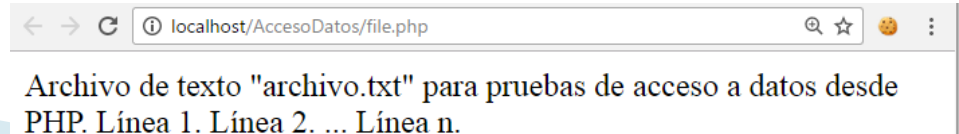
archivo.txt

file.php

```
1 Archivo de texto "archivo.txt" para
2 pruebas de acceso a datos desde PHP.
3 Línea 1.
4 Línea 2.
5 ...
6 Línea n.
```

```
<?php
// Lee archivo de texto línea por línea y
// las guarda en un array

$archivo = "archivo.txt";
$lineas = file($archivo);
for ($i=0; $i<count($lineas); $i++)
    echo $lineas[$i];
```



Acceso a datos desde PHP: lectura archivos

fread.php

```
<?php
if ( $fp = fopen("archivo.txt", "r") ) {
    $datos = fread($fp, 10); // Lee 10 caracteres
    fclose($fp);
    echo $datos;
}
else {
    echo "Error al abrir el archivo.";
}
```

← → ↻ 📄 localhost/AccesoDatos/fread.php

Archivo de

leerFich.php

```
<?php
    echo readFile('archivo.txt');
?>
```

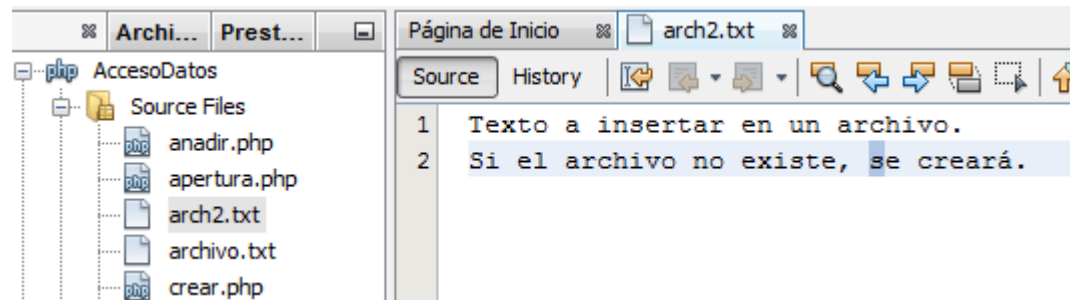
← → ↻ ⓘ localh... ⌵ ⌵ ≡

Archivo de texto "archivo.txt" para pruebas de acceso a PHP. Línea 1. Línea 2. ... Línea n. 95

Acceso a datos desde PHP: modificar archivos

crear.php

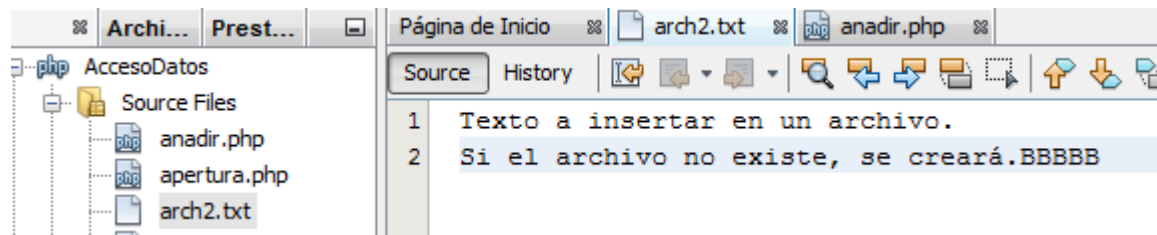
```
<?php
if ( $fp = fopen("arch2.txt", "w") ) {
    $texto = "Texto a insertar en un archivo.\n".
        "Si el archivo no existe, se creará.";
    fputs($fp, $texto);
    fclose($fp);
}
else {
    echo "Error al abrir el archivo.";
}
```



Acceso a datos desde PHP: añadir datos

anadir.php

```
<?php
    if ( $fp = fopen("arch2.txt", "a") ) {
        fwrite($fp, 'BBBBB');
        fclose($fp);
    }
    else {
        echo "Error al abrir el archivo.";
    }
}
```



Acceso a datos desde PHP: BBDD

- ❑ **SQL** es un lenguaje común para trabajar con todas las bases de datos.
- ❑ Las interfaces que los programadores usan para trabajar con Bases de Datos varían



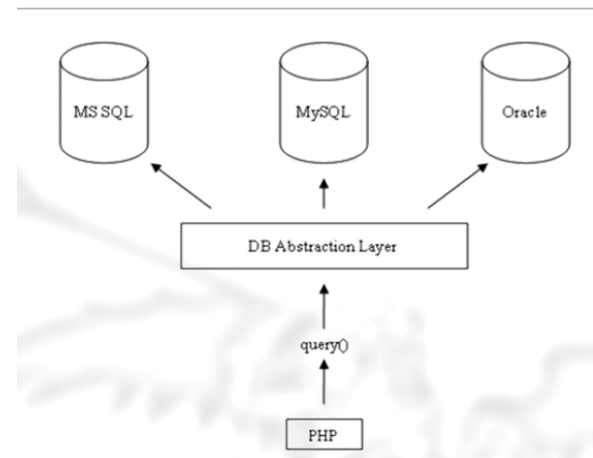
Dependencia de las aplicaciones
con respecto de la BD

- ❑ Para resolver estos problemas se necesitan capas de **abstracción** respecto de la BD, es decir, una **interfaz común**.

Esta interfaz común es en el caso de PHP, **PDO (PHP Data Objects)**.

En el caso de Java, **JDBC**.

Y existen otras como **ORM (Mapeo Objeto-Relacional)**



Acceso a datos desde PHP: PDO

- ❑ **PHP Data Objects (PDO)** es una de estas interfaces.
- ❑ Así pues, *PDO* define una interfaz **ligera** y consistente para acceder a bases de datos en PHP.
- ❑ Implementada con tecnología **orientada a objetos**.
- ❑ A partir de la versión PHP 5.1 se considera **estable y la interfaz viene incluida por defecto en el intérprete PHP**.

Acceso a datos desde PHP: PDO

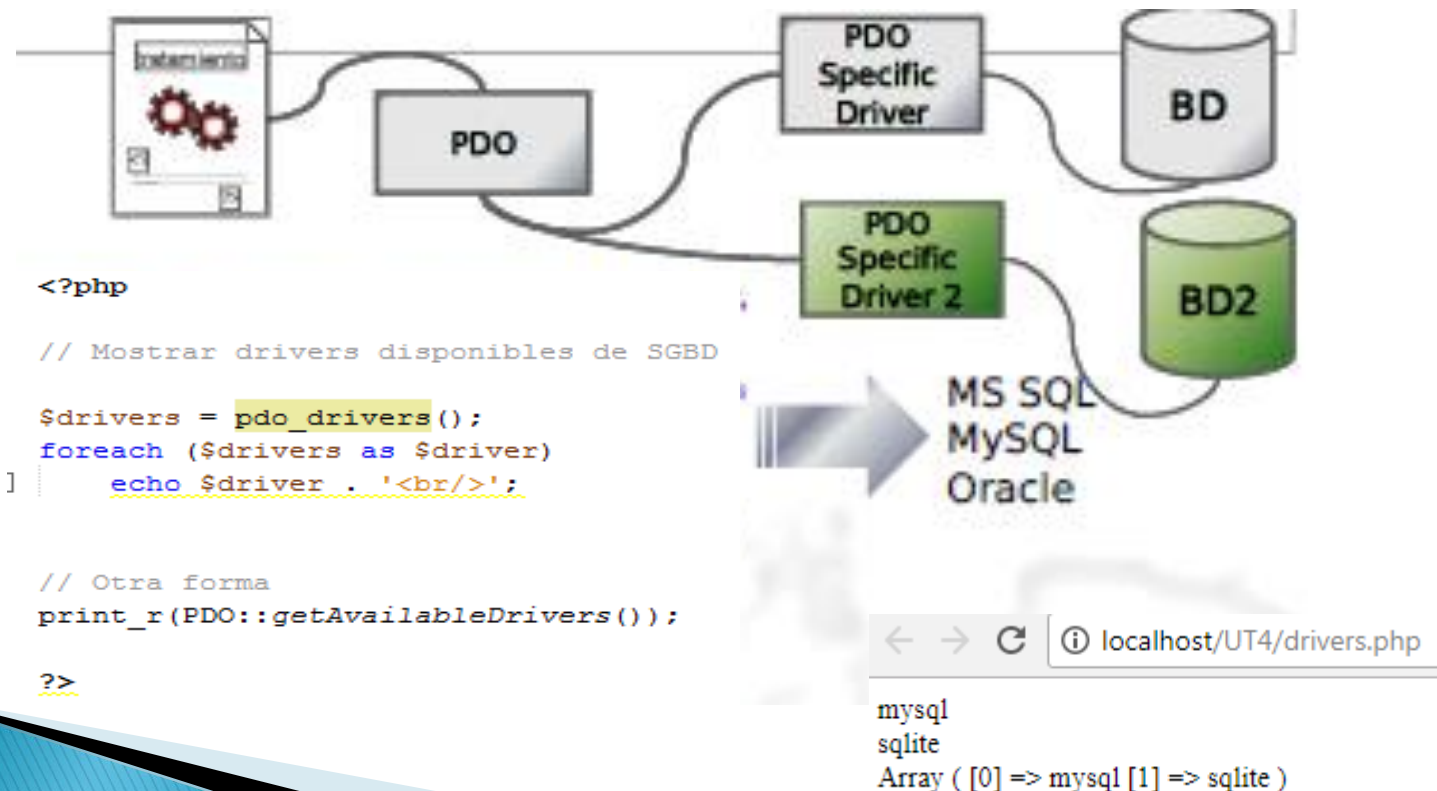
❑ Ventajas de PDO

- ❑ Permite **abstraernos del tipo de gestor de base de datos** al que conectar.
- ❑ Se encarga de **desinfectar** los datos automáticamente al utilizar ***consultas preparadas***.
 - ❑ Previene de ataques de inyección SQL.
- ❑ Proporciona el uso de **transacciones**.

Acceso a datos desde PHP: PDO

■ Actores en PDO

- ❑ Proveedor de datos (PDO Specific Driver).
- ❑ Conexión (Objeto PDO).
- ❑ Bases de Datos: MySQL, Oracle, SQLite, MS SQL Server, ...



Acceso a datos desde PHP: PDO

- Las clases implicadas directamente en el acceso a datos con la librería *PDO* son:
 - **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 una excepción generada por un **error** en PDO.

Extensión PDO: Conexión a BD

- ❑ La conexión a la BD la realizamos con el constructor de la clase PDO

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

donde `$con` es el *manejador* de la BD y `$cadenaDSN` (data source name) es de la forma:

“gestorBD:host=descrip_host; dbname=nombreBD”

y se suele especificar:

- Sistema gestor de base de datos
- Host donde se alberga la BD
- Nombre de la base de datos
- Se puede especificar también el puerto
- Se puede especificar el conjunto de caracteres de codificación
- Usuario y contraseña de conexión.

Extensión PDO: Conexión a BD

- Diferentes SGBD tienen diferentes métodos de conexión, que es lo que se denomina **drivers** o **controladores**.

MySql

```
$con = new PDO ('mysql:host=localhost; dbname=clientes;  
charset=utf8', $usuario, $pwd);
```

Oracle Express

```
$con = new PDO ('oci:dbname=xe: host=localhost;  
port=1521', $usuario, $pwd);
```

Sqlite

```
$con = new PDO ('sqlite:/ruta a archivo .sqlite');
```


Extensión PDO: Conexión a BD

```
function getConexion() {  
  
    try {  
  
        $this->conexion = new PDO('mysql:host='.$this->hostname.  
                                ' ;dbname='.$this->database . ' ;charset='.$this->charset,  
                                $this->user, $this->password);  
  
        $this->conexion->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);  
  
    } catch (PDOException $error) {  
  
        echo "¡ERROR: !". $error->getMessage();  
        die();  
    }  
  
    return $this->conexion;  
}
```

```
$stmt = $conn->prepare('SELECT * FROM proveedores');  
$stmt->execute();
```

Extensión PDO: Excepciones

- ❑ Para capturar los posibles errores producidos, **incluida la conexión:**

PDO::errorInfo()

PDO::errorCode()

```
print_r($con->errorInfo());
```

- ❑ Para ver todos los controladores de acceso a las diferentes bases de datos se puede emplear :

PDO::getAvailableDrivers()

```
print_r(PDO::getAvailableDrivers());
```

Extensión PDO: Excepciones

- ❑ *PDO* permite activar el uso de excepciones para capturar errores:

`PDO::setAttribute()`

`PDO::ATTR_ERRMODE`

`PDO::ERRMODE_SILENT`

`PDO::ERRMODE_EXCEPTION`

`PDO::ERRMODE_WARNING`

- ▶ **PDO::ERRMODE_SILENT** es el valor por defecto y, como hemos mencionado antes, no lanza ningún tipo de error ni excepción; *es tarea del programador comprobar si ha ocurrido algún error después de cada operación con la base de datos.*
- ▶ **PDO::ERRMODE_WARNING** genera un error `E_WARNING` de PHP si ocurre algún error. Este error es el mismo que se muestra usando la API de *mysql* presentando por pantalla una descripción del error que ha ocurrido.
- ▶ **PDO::ERRMODE_EXCEPTION** es el que acabamos de explicar que genera y lanza una excepción si ocurre algún tipo de error.

Extensión PDO: Excepciones

```
try {  
    $con = new PDO ('mysql:dbname=cliente;host=localhost;charset=utf8',  
    $usuario, $pwd);  
    $con->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);  
  
} catch (PDOException $e) {  
    echo 'error: '. $e->getMessage();  
}
```

!! Se debe indicar el modo de error de excepciones para que las lance !!

Si no capturamos la excepción, el servidor mostrará la traza.
!! Incluyendo el usuario y password de la BD !!

Fatal error: Uncaught PDOException: PDO::__construct(): getaddrinfo failed: Host desconocido. in C:\xampp\htdocs\UT5\claseConexionBD.php(20):
trace: #0 C:\xampp\htdocs\UT5\claseConexionBD.php(20):
PDO->construct('mysql:host=loca...', 'root', '') #1 C:\xampp\htdocs\UT5\Ejer2_bienhecho.php(19): ConectarBD->getConexion
SQLSTATE[HY000] [2002] php_network_getaddresses: get
in C:\xampp\htdocs\UT5\claseConexionBD.php:20 Stack tra
\\UT5\claseConexionBD.php(20): PDO->construct('mysql

Extensión PDO: Consultas no preparadas

- ❑ La librería *PDO* proporciona dos métodos para realizar consultas: **preparadas** y **no preparadas**.
- ❑ **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`

```
$stmt=$con->query('SELECT * nombre clientes');
```

Extensión PDO: Consultas preparadas

■ Consultas preparadas

- ❑ Escapa los parámetros de la consulta automáticamente
- ❑ Devuelve un objeto `PDOStatement` ó `False`
- ❑ La consulta se ejecuta en tres pasos:

1. Preparar la consulta

`PDO::prepare(string $consulta)`

2. Vincular cada valor a cada parámetro de sustitución

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

`PDOStatement::bindValue(1, $valor)` o

`PDOStatement::bindParam(1, $valor)`

3. Ejecutar la consulta

`PDOStatement::execute()`

`PDOStatement::execute(array(':parametro' => $valor))`



Extensión PDO: Consultas preparadas

❑ Ejemplos

```
$stmt = $con->prepare('SELECT * FROM articulos WHERE idArticulo=? and precio=?');
```

```
$stmt->bindValue(1, $articulo);
```

Hace referencia a la primera interrogación del prepare

```
$stmt->bindValue(2, $precio);
```

Hace referencia a la segunda interrogación del prepare

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

```
$stmt = $con->prepare('SELECT * FROM articulos WHERE idArticulo=:articulo and precio=:prec');
```

```
$stmt->bindValue(':articulo', $articulo);
```

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

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

Extensión PDO: Consultas preparadas

❑ Ejemplo 2

```
$stmt = $con->prepare('SELECT * FROM articulos WHERE  
idArticulo=:articulo and precio=:prec');  
  
$stmt->execute(array('articulo' => $articulo,  
                    ':prec' => $precio)  
);
```

Es preferible utilizar consultas preparadas ya que se minimiza el tiempo requerido en el servidor, dado que solo se envían los parámetros y no la consulta entera.

Además, se reduce el tiempo de análisis (“parsing”) de la consulta pues la preparación solamente se realiza una vez aunque la ejecución se realice múltiples veces.

Y de esta manera se evitan, entre otros, ataques de inyección SQL.

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)`

Modo: se verá a partir de la diapositiva nº 29

- ❑ 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: 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()
```

- ❑ Ejecutando una *query* con **exec** también se devuelve el número de filas afectadas en la inserción, actualización o borrado

```
public int PDOStatement::exec(string $query)
```

Extensión PDO: Otras funciones útiles

Ejemplo de modificación en la BD utilizando sentencias preparadas.

```
try {
    $con=new PDO
    ("mysql:host=$hostname;dbname=$db;charset=utf8",$user,$password);
    $con->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $stmt = $con->prepare("UPDATE table SET name=? WHERE id=?");
    $stmt->bindValue(1,$nombre);
    $stmt->bindValue(2,$id);
    $stmt->execute();
    $num_filas = $stmt->rowCount();
} catch (PDOException $e) {
    echo $e->getMessage();
}
```

Extensión PDO: Recogida de datos

Ejemplo de recogida de datos de modo asociativo. Se recogen en un array asociativo donde cada etiqueta del array es el nombre de la columna de la tabla.

```
$stmt = $con->prepare('SELECT * FROM articulos WHERE  
                        idArticulo=:articulo and precio=:prec');  
$stmt->bindValue(':articulo', $articulo);  
$stmt->bindValue(':prec', $precio);  
$stmt->execute();  
while ($fila = $stmt->fetch(PDO::FETCH_ASSOC)) {  
    echo $fila['idArticulo']. ' '.$fila['precio'];  
}
```

Extensión PDO: Recogida de datos

```
/* Ejecutar una sentencia preparada usando un array de valores
para una cláusula IN */
$ids = array(1,5,6,8);

/* Crear una cadena para los parámetros de sustitución rellenos
con el número de parámetros */
$place_holders = implode(',', array_fill(0, count($ids), '?'));

$stmt = $con->prepare('SELECT * FROM articulos WHERE
                        idArticulo IN ($place_holders));
$stmt->execute($ids);
while ($fila = $stmt->fetch(PDO::FETCH_ASSOC)) {
    echo $fila['idArticulo']. ' '. $fila['precio'];
}
```

Extensión PDO: Recogida de datos

- ❑ El método **fetch**, si no se le pasa nada como parámetro, devolverá un array.
- ❑ Pero también se puede definir para que devuelva un objeto o un instancia de clase.
 - ❑ **PDO::FETCH_NUM**: devuelve un array indexado numéricamente con índice.
 - ❑ **PDO::FETCH_ASSOC**: devuelve un array indexado por los nombres de las columnas del conjunto de resultados.
 - ❑ **PDO::FETCH_BOTH**: es el valor por defecto y devuelve un array indexado tanto por nombre de columna, como numéricamente con índice.

Extensión PDO: Recogida de datos

- ❑ **PDO::FETCH_CLASS**: devuelve una instancia de la clase solicitada, haciendo corresponder las columnas del conjunto de resultados con los nombres de los atributos de la clase. Si se quiere llamar al constructor de la clase antes de hacer el fetch, habrá que añadir **PDO::FETCH_PROPS_LATE**.
- ❑ **PDO::FETCH_BOUND**: devuelve **TRUE** y asigna los valores de las columnas del conjunto de resultados a las variables de PHP a las que fueron vinculadas con el método **PDOStatement::bindColumn()**.
- ❑ **PDO::FETCH_OBJ**: devuelve un objeto anónimo con nombres de propiedades que se corresponden a los nombres de las columnas devueltas en el conjunto de resultados.
- ❑ **PDO::FETCH_INT**: devuelve un conjunto de resultados sobre un objeto de una clase que ha sido previamente instanciado.

Extensión PDO: Recogida de datos

```
<?php
```

```
$gsent = $gbd->prepare("SELECT name, colour FROM fruit");  
$gsent->execute();
```

```
/* Prueba de tipos de PDOStatement::fetch */
```

```
print("PDO::FETCH_ASSOC: ");
```

```
print("Devolver la siguiente fila como un array indexado por nombre  
de columna\n");
```

```
$result = $gsent->fetch(PDO::FETCH_ASSOC);
```

```
print_r($result);
```

```
print("\n");
```

→ PDO::FETCH_ASSOC: Devolver la siguiente fila como un array indexado por nombre de columna

Array

(

[name] => apple

[colour] => red

)

Extensión PDO: Recogida de datos

```
print("PDO::FETCH_BOTH: ");  
print("Devolver la siguiente fila como un array indexado por nombre y  
    número de columna\n");  
$result = $gsent->fetch(PDO::FETCH_BOTH);  
print_r($result);  
print("\n");
```




PDO::FETCH_BOTH: Devolver la siguiente fila como un array indexado por nombre y número de columna

Array

```
(  
    [name] => banana  
    [0] => banana  
    [colour] => yellow  
    [1] => yellow  
)
```

Extensión PDO: Recogida de datos

```
print("PDO::FETCH_OBJ: ");  
print("Devolver la siguiente fila como un objeto anónimo con nombres  
de columna como propiedades\n");  
$result = $gsent->fetch(PDO::FETCH_OBJ);  
print $result->nombre;  
print("\n");
```



PDO::FETCH_OBJ: Devolver la siguiente fila como un objeto anónimo con nombres de columna como propiedades
kiwi

`$result` es un objeto de una clase llamada *fruta*

```
class fruta {  
    private $nombre;  
    private $color;  
}
```

Extensión PDO: Recogida de datos

```
class Cd
{
    private $idcd;
    private $titulo;
    private $interprete;
}

$stmt = $conn->prepare('SELECT id, titel, interpret FROM cds');
$cd = new Cd;
$stmt->setFetchMode(PDO::FETCH_INTOTO, $cd);
$stmt->execute();
while ( $cd = $stmt->fetch(PDO::FETCH_INTOTO) ) {
    echo "<td>$cd->idcd</td><td>$cd->titulo</td><td>$cd->
    >interprete</td>";
}
}
```

Conexión realizada con éxito !!!

Relación de cds

ID	Título	Intérprete
1	Beauty	Ryuichi Sakamoto
4	Goodbye Country (Hello Nightclub)	Groove Armada
5	Glee	Bran Van 3000

Extensión PDO: Recogida de datos

Ejemplo consulta con fetch asociativo

```
<?php
```

```
// Parámetros acceso a una BD
define('HOST', 'localhost');
define('BD', 'gesventa');
define('USER', 'root');
define('PASSWORD', '');
define('CHARSET', 'utf8');
```

```
?>
```

Config_BD.php

```
<?php
```

```
require_once 'config_BD.php';
```

```
class ConectarBD {
```

```
    private $hostname = HOST;
    private $database = BD;
    private $user = USER;
    private $password = PASSWORD;
    private $charset = CHARSET;
    private $conexion;
```

```
    function getConexion() {
```

```
        try {
```

```
            $this->conexion = new PDO('mysql:host='.$this->hostname.
                ';dbname='.$this->database . ';charset='.$this->charset,
                $this->user, $this->password);
```

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

```
        } catch (PDOException $error) {
```

```
            echo "¡ERROR: !".$error->getMessage();
            die();
```

```
        }
```

```
        return $this->conexion;
```

```
    }
```

```
    function cerrarConexion() {
```

```
        $this->conexion = null;
```

```
    }
```

```
}
```

```
?>
```

claseConexionBD.php

Extensión PDO: Recogida de datos

Ejemplo consulta con fetch asociativo (cont.)

```
<!DOCTYPE html>

<html>
  <head>
    <meta charset="UTF-8">
    <title>Consultas con PDO</title>
    <link rel='stylesheet' type='text/css' href='css/estilos_2.css' />
  </head>
  <body>
    <h3>Relación de Proveedores</h3>
    <table border=1>
      <tr><th>Cif</th><th>Nombre</th><th>Dirección</th>
        <th>Población</th></tr>

<?php

include_once 'claseConexionBD.php';

$BD = new ConectarBD();
$conn = $BD->getConexion();

$stmt = $conn->prepare('SELECT * FROM proveedores');
$stmt->execute();

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

while ( $proveedor = $stmt->fetch() ) {
  echo "<tr>";
  echo "<td>".$proveedor['cif']. "</td><td>".$proveedor['nom_emp'].
    "</td><td>".$proveedor['direccion'].
    "</td><td>".$proveedor['poblacion']. "</td>";
  echo "</tr>";
}
$BD->cerrarConexion();
?>
</table>

<?php echo 'Número de filas: ' . $stmt->rowCount(); ?>

</body>
</html>
```

Relación de Proveedores

Cif	Nombre	Dirección	Población
D7767763A	Ozone	C/ Miralrio, 61	Tres Cantos
J04131348	Logitech	C/ Faisán, 13	Leganés
P3941094I	Razer	C/ Vieja, 12	Madrid
R93200509	BenQ	C/ Velázquez, 80	Madrid
S9939407D	Acer	C/ Campillo 7	Alcobendas

Número de filas: 5

Extensión PDO: Inserción de datos

```
<?php
```

```
$cif = 'B0041567I';  
$nombre = 'Toshiba España';  
$dir = 'Carretera Fuencarral, 46';  
$pobl = 'Alcobendas';
```

```
include_once 'claseConexionBD.php';
```

```
$BD = new ConectarBD();  
$conn = $BD->getConexion();
```

```
$stmt = $conn->prepare('INSERT INTO proveedores  
    (cif, nom_emp, direccion, poblacion) '  
        . 'VALUES (:cif, :nom_emp,  
            :direccion, :poblacion)');
```

```
try {
```

```
    $stmt->execute( array( ':cif' => $cif,  
                          ':nom_emp' => $nombre,  
                          'direccion' => $dir,  
                          'poblacion' => $pobl));
```

```
    if ($stmt->rowCount() > 0) // Se ha realizado el  
        borrado
```

```
        echo 'Insertadas ' . $stmt->rowCount() . ' '  
        filas';
```

```
}
```

```
catch (PDOException $ex) {  
    print "¡Error!: " . $ex->getMessage() . "<br/>";  
    die();  
}
```

```
$BD->cerrarConexion();
```

```
?>
```

Extensión PDO: Modificación de datos

```
<?php
```

```
$nombre = 'Ozone';  
$pobl = 'Tres Cantos';
```

```
include_once 'claseConexionBD.php';
```

```
$BD = new ConectarBD();  
$conn = $BD->getConexion();
```

```
$stmt = $conn->prepare('UPDATE proveedores SET poblacion = :poblacion WHERE  
nom_emp = :nom');  
$stmt->execute(array(':poblacion' => $pobl, ':nom' => $nombre));
```

```
if ($stmt->rowCount() > 0) // Se ha realizado la modificación  
    echo 'Modificadas ' . $stmt->rowCount() . ' filas';
```

```
$BD->cerrarConexion();
```

```
?>
```

Extensión PDO: Eliminación de datos

```
<?php
```

```
$pobl = 'Madrid';
```

```
include_once 'claseConexionBD.php';
```

```
$BD = new ConectarBD();
```

```
$conn = $BD->getConexion();
```

```
$stmt = $conn->prepare('DELETE FROM proveedores WHERE poblacion =  
:poblacion');
```

```
$stmt->execute(array(':poblacion' => $pobl));
```

```
if ($stmt->rowCount() > 0) // Se ha realizado el borrado  
    echo 'Eliminadas ' . $stmt->rowCount() . ' filas';
```

```
$BD->cerrarConexion();
```

```
?>
```


Extensión PDO: Transacciones

```
<?php
```

```
$cif = 'B0041567I';  
$nombre = 'Toshiba España';  
$dir = 'Carretera Fuencarral, 46';  
$pobl = 'Alcobendas';
```

```
include_once 'claseConexionBD.php';
```

```
$BD = new ConectarBD();  
$conn = $BD->getConexion();
```

```
$stmt = $conn->prepare('INSERT INTO proveedores (cif, nom_emp, direccion, poblacion) '  
    . 'VALUES (:cif, :nom_emp, :direccion, :poblacion)');
```

```
$conn->beginTransaction();
```

```
try {  
    $stmt->execute( array( ':cif' => $cif,  
        ':nom_emp' => $nombre,  
        ':direccion' => $dir,  
        ':poblacion' => $pobl));
```

Extensión PDO: Transacciones

```
$conn->commit();
```

```
$conn->beginTransaction();
```

```
$stmt->execute( array( ':cif' => "B00415672",  
                      ':nom_emp' => 'Toshiba España',  
                      'direccion' => 'Carretera Fuencarral, 46',  
                      'poblacion' => 'Alcobendas')));
```

```
$stmt->execute( array( ':cif' => "B00415672",  
                      ':nom_emp' => 'Toshiba España',  
                      'direccion' => 'Carretera Fuencarral, 46',  
                      'poblacion' => 'Alcobendas')));
```

```
if ($stmt->rowCount() > 0) // Se ha realizado el borrado  
    echo 'Insertadas ' . $stmt->rowCount() . ' filas';
```

```
}
```

```
catch (PDOException $ex) {
```

```
    $conn->rollBack();
```

```
    print "¡Error!: " . $ex->getMessage() . "<br/>";
```

```
    die();
```

```
}
```

```
$BD->cerrarConexion();
```

```
?>
```

Extensión PDO: Tratamiento de errores

- ❑ Todas las operaciones con objetos *PDO* pueden generar excepciones que deben capturarse: bloques **try/catch**
- ❑ El objeto que lanza las excepciones es de tipo **PDOException** que es una subclase de *Exception*, la cual tiene los siguientes métodos:
 - ❑ **getMessage()**: descripción de la excepción
 - ❑ **getFile()**: fichero en el que ocurrió la excepción
 - ❑ **getLine()**: línea desde la que se originó la excepción
 - ❑ **getTrace()** y **getTraceAsString()**: devuelven la traza de ejecución en el momento del error.