

- El significado de PDO es PHP Data Objects
- La API PDO permitirá a PHP conectarse y trabajar con diferentes gestores de base de datos
- Para poder trabajar con PDO necesitamos crear una instancia de la clase PDO.
- La clase PDO contiene los métodos que permiten ordenar ejecución de mandatos en el gestor (p.e. prepare, execute, exec, beginTransaction, bindParam, commit, etc)
- Antes de empezar a trabajar con PDO, debemos verificar que tenemos dicha API disponible y activada en nuestra instalación.

- Crea el archivo phpinfo.php con el contenido que se muestra en la figura y ejecutalo

```
<?php
phpinfo();
?>
```

- Comprueba que muestra una información parecida a esta

PDO

PDO support	enabled
PDO drivers	mysql, oci, sqlite

pdo_mysql

PDO Driver for MySQL	enabled
Client API version	mysqlnd 5.0.8-dev - 20102224 - \$Id: 65fe78e70ce53d27a6cd578597722950e490b0d0 \$

- Si todo esta correcto entonces podemos interactuar con el gestor MySQL usando PDO.

Conexión a base de datos MySQL usando PDO

- Las conexiones se establecen creando instancias de la clase base PDO.
- El constructor acepta parámetros para especificar el origen de la base de datos (conocido como DSN) y, opcionalmente, el nombre de usuario y la contraseña (si la hubiera).

```
public PDO::__construct ( string $dsn [, string $username [, string $password [, array $options ]]] )
```

\$dsn	<p>Nombre de origen de datos compuesto por los siguientes elementos</p> <p>prefijo DSN Para MySQL es mysql:.</p> <p><i>host</i> nombre o IP del equipo donde reside el servidor de BD</p> <p><i>port</i> número de puerto por el que el servidor de BD está escuchando</p> <p><i>dbname</i> nombre de la base de datos</p> <p>Ejemplos: \$dsn1 = 'mysql:host=localhost;dbname=testdb'; \$dns2 = 'mysql:host=localhost;port=3307;dbname=empresa';</p> <p>Los DSN a otros gestores de BD cambian el prefijo, manteniendo el resto de la información \$dsn_sqlserver = "sqlsrv:Server=localhost,1521;Database=testdb" \$dsn_oracle = "oci:dbname=//localhost:1521/mydb"</p>
\$username	Nombre de usuario
\$password	Contraseña
\$options	Array asociativo con datos adicionales para hacer la conexión

- Si hay error al hacer la conexión, se lanza un objeto *PDOException*; Si la aplicación no captura la excepción lanzada por el constructor de PDO, la acción predeterminada es la de finalizar el script y mostrar información revelando detalles de la conexión a la BD incluyendo el nombre de usuario y la contraseña. Por ello se recomienda capturar esta excepción con una sentencia *catch*
- Si tiene éxito la conexión, se devuelve una instancia de la clase PDO al script. La conexión permanecerá activa durante el tiempo de vida del objeto PDO. Para cerrar la conexión, es necesario destruir el objeto (se puede hacer asignando **NULL** a la variable que contiene el objeto). Si no se realiza explícitamente, PHP cerrará la conexión cuando el script finalice.

```
<?php
class conectaBD
{ protected $db;
  function __construct()
  { $dsn = 'mysql:host=localhost;dbname=prueba;charset=utf8' ;
    $usuario='root';
    $pass = 'toor';
    try { $this->db = new PDO( $dsn, $usuario, $pass );
      } catch ( PDOException $e) {
        die( "¡Error!: " . $e->getMessage() . "<br/>" );
      }
    }
  public function getConBD() { return $this->db; }
}
$c1= new conectaBD(); $id_c1= $c1->getConBD();
foreach( $id_c1->query(' SELECT * from tabla1' ) as $fila)
{ print_r( $fila ); }
$c1=NULL;
?>
```

- En una aplicación web se puede también establecer conexiones persistentes a servidores de base de datos. Las conexiones persistentes no son cerradas al final del script, sino que son almacenadas en caché y reutilizadas cuando otro script solicite una conexión que use las mismas credenciales. Con ello se evita 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 usar conexiones persistentes, se deberá establecer **PDO::ATTR_PERSISTENT** en las opciones del array del controlador pasado al constructor de PDO.

```
$conexion1 = new PDO( $dsn, $usuario, $pass , array( PDO::ATTR_PERSISTENT => true ) );
```

Métodos de la clase PDO

public __construct (string \$dsn [, string \$user [, string \$pass [, array \$options]]])	Crea instancia de PDO que representa una conexión a una BD
public int exec (string \$orden)	Ejecuta una orden SQL y devuelve nº de filas afectadas
public PDOStatement prepare (string \$orden [, array \$opciones = array()])	Prepara una orden para su ejecución y devuelve un objeto PDOStatement
public PDOStatement query (string \$orden)	Ejecuta una orden SQL, y devuelve conjunto de resultados como un objeto PDOStatement
public mixed errorCode (void)	Obtiene SQLSTATE de la última operación
public array errorInfo (void)	Obtiene mensaje del error asociado a la última operación
public mixed getAttribute (int \$atributo)	Devuelve valor de un atributo de la conexión a la BD
public bool setAttribute (int \$atributo , mixed \$valor)	Establece un atributo de la conexión a la BD
public bool beginTransaction (void)	Inicia una transacción
public bool commit (void)	Confirma transacción
public bool rollBack (void)	Deshace transacción
public bool inTransaction (void)	Comprueba si una transacción esta activa

<?php

```
$conn = new PDO('odbc:sample', 'db2inst1', 'ibmdb2');
$attributes = array( "AUTOCOMMIT", "ERRMODE", "CASE", "CLIENT_VERSION", "CONNECTION_STATUS",
    "ORACLE_NULLS", "PERSISTENT", "PREFETCH", "SERVER_INFO", "SERVER_VERSION", "TIMEOUT");
foreach ($attributes as $val) { echo "PDO::ATTR_$val: "; echo $conn->getAttribute(constant("PDO::ATTR_$val")) . "\n"; }
?>
```

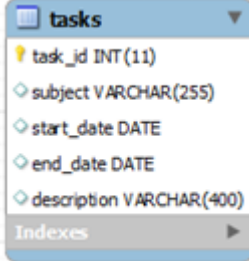
Métodos de la clase PDOStatement

| | |
|--|---|
| public int columnCount (void) | Devuelve el número de columnas de un conjunto de resultados |
| public string errorCode (void) | Obtiene el SQLSTATE asociado con la última operación del gestor de sentencia |
| public array errorInfo (void) | Obtiene información ampliada del error asociado con la última operación del gestor de sentencia |
| public bool execute ([array \$input_parameters]) | Ejecuta una sentencia preparada |
| public mixed fetch ([int \$fetch_style
[, int \$sentido = PDO::FETCH_ORI_NEXT
[, int \$desplaza = 0]]]) | Obtiene la siguiente fila de un conjunto de resultados |
| public array fetchAll ([int \$fetch_style
[, mixed \$fetch_argument
[, array \$ctor_args = array()]]]) | Devuelve un array que contiene todas las filas del conjunto de resultados |
| public mixed fetchColumn ([int \$column_number = 0]) | Devuelve una única columna de la siguiente fila de un conjunto de resultados |
| public int rowCount (void) | Devuelve el número de filas afectadas por la última orden SQL |
| public bool setFetchMode (int \$mode) | Establece el modo de obtención para esta sentencia |

Crear tabla MySQL usando PDO

El siguiente script PHP crea una tabla MySQL con la estructura de campos y restricciones que se muestra en la figura.

- Se usa el método **PDO::exec()** que devuelve el número de filas afectadas por la orden. Devuelve FALSE si no se puede llevar a cabo la orden.
- PDO::exec() **no devuelve resultados de una sentencia SELECT**.
- Para ejecutar SELECT se usaría los métodos PDO::query() ó PDO::prepare()
 sí la select se necesita hacer una sola vez en el programa → PDO::query()
 sí la select se necesita hacer varias veces en el programa → PDO::prepare()



| tasks | |
|-------------|--------------|
| task_id | INT(11) |
| subject | VARCHAR(255) |
| start_date | DATE |
| end_date | DATE |
| description | VARCHAR(400) |
| Indexes | |

```
<?php
class conectaBD
{
    private $conn = null ;
    public function __construct($database='test')
    {
        $dsn = "mysql:host=localhost;dbname=$database" ;
        try {
            $this->conn = new PDO( $dsn, 'root', 'toor' );
        } catch ( PDOException $e) { die( "¡Error!: " . $e->getMessage() . "<br/>" ); }
    }

    public function __destruct() // Cierra conexión asignándole valor null
    { $this->conn = null; }

    public function creaTablaTask() // Crea tabla; devuelve TRUE si tiene éxito y FALSE en caso contrario
    {
        $sql = <<<EOSQL
        CREATE TABLE IF NOT EXISTS tasks ( task_id int NOT NULL AUTO_INCREMENT,
                                            subject varchar(255) DEFAULT NULL,
                                            start_date date DEFAULT NULL,
                                            end_date date DEFAULT NULL,
                                            description varchar(400) DEFAULT NULL,
                                            PRIMARY KEY (task_id)
                                            );
        EOSQL;
        if( $this->conn->exec($sql) !== false) return true;
        return false;
    }
}

// ----- Proceso principal

$obj = new conectaBD( 'empresa' ); // crea conexión para usar bd empresa
if ( $obj->creaTablaTask() ) // invoca método de objeto para crear tabla
    echo 'tabla Tasks table creada';
else echo 'Error al crear table Task';

?>
```

Consultar datos

El siguiente script PHP realiza una consulta a una tabla MySQL y muestra los resultados

- Se usa el método **PDO::query()** que devuelve un objeto PDOStatement o FALSE si no se puede llevar a cabo la orden
- Establece que la extracción de los resultados será en array asociativo, poniendo para el método fetch() el modo PDO::FETCH_ASSOC, mediante el método setFetchMode()
- Se extrae cada fila del resultado con fetch() y se muestra

```
<?php
class conectaBD
{
    private $conn = null ;
    public function __construct($database='test')
    {
        $dsn = "mysql:host=localhost;dbname=$database" ;
        try {
            $this->conn = new PDO( $dsn, 'root', 'toor' );
        } catch ( PDOException $e) { die( "¡Error!: " . $e->getMessage() . "<br/>" ); }
    }

    public function __destruct() // Cierra conexión asignándole valor null
    { $this->conn = null; }

    public function consulta1($orden) // Ejecuta consulta y devuelve array de resultados o FALSE si falla ejecución
    {
        try {
            $q = $this->conn->query($orden);
            $filas=array();
            $q->setFetchMode(PDO::FETCH_ASSOC);
            while ( $r = $q->fetch() )
                $filas[]=$r;
            return $filas;
        } catch ( PDOException $e) {
            echo ( "¡Error! al ejecutar consulta: " . $e->getMessage() . "<br/>" );
            return false; }
    }

    public function consulta2($orden) // Ejecuta consulta y devuelve array de resultados o NULL si falla ejecución
    {
        $filas=array();
        $q = $this->conn->query($orden);
        if( $q !== false ) {
            $q->setFetchMode(PDO::FETCH_ASSOC);
            while ( $r = $q->fetch() )
                $filas[]=$r;
        }
        return $filas;
    }
}

// ----- Proceso principal

$obj = new conectaBD( 'empresa' ); // crea conexión para usar bd empresa
$sql = 'SELECT emp_no,apellido,oficio FROM empleados ORDER BY apellido';
$resultado = $obj->consulta2($sql) // invoca método de objeto para ejecutar consulta

if ( $resultado ) {
    foreach ( $resultado as $k=>$fila )
    {
        if ( $k==0 ) {
            echo "<table><tr><th>Codigo</th><th>Apellido</th><th>Oficio</th></tr>";
            echo "<tr><td>". $fila['emp_no']. "</td><td>". $fila['apellido']. "</td><td>". $fila['oficio']. "</td></tr>";
        }
        echo "</table>";
    }
}

?>
```

Consultar datos con sentencias preparadas

El siguiente script PHP permite realizar consulta en una tabla MySQL con una orden select preparada

- El uso de ordenes preparadas evita inyección de SQL y ofrecen mayor rendimiento
- Los parámetros a sustituir en la orden tendrán un nombre precedido de :
- Se usa el método **PDO::prepare()** que devuelve un objeto PDOStatement o **FALSE** **sí no se puede llevar a cabo la orden**
- Las sentencias preparadas se ejecutan con el método execute() debiéndole pasar como argumento un array con los valores para los parámetros a sustituir.
- Ejecutada la orden se extrae cada fila del resultado con fetch() y se muestra

```
<?php
class conectaBD
{
    private $conn = null ;
    public function __construct($database='test')
    {
        $dsn = "mysql:host=localhost;dbname=$database" ;
        try {
            $this->conn = new PDO( $dsn, 'root', 'toor' );
        } catch ( PDOException $e) { die( "¡Error!: " . $e->getMessage() . "<br/>"); }
    }

    public function __destruct() // Cierra conexión asignándole valor null
    { $this->conn = null; }

    public function consultaPreparada() // Prepara y ejecuta consulta
    {
        try
        {
            $sql = 'SELECT emp_no, apellido, oficio FROM empleados
                    WHERE apellido LIKE :par1 OR oficio LIKE :par2';
            $q = $this->conn->prepare($sql);
            $q->execute(array(':par1' => 'A%', ':par2' => '%R'));
            $q->setFetchMode(PDO::FETCH_ASSOC);
            return $q;
        } catch (PDOException $pe) { die("Error al ejecutar orden select : " . $pe->getMessage()); }
    }
}

// ----- Proceso principal

$obj = new conectaBD( 'empresa' ); // crea conexión para usar bd empresa
$consulta=$obj->consultaPreparada();
while ($fila = $consulta->fetch())
    echo sprintf('%s -- %s -- %s <br/>', $fila['emp_no'], $fila['apellido'], $fila['oficio'] );

?>
```

Insertar , actualizar y borrar datos con sentencias preparadas

El siguiente script PHP permite insertar fila, cambiar datos en filas y eliminar filas con ordenes preparadas

- Crea array con la asociación de parámetros de la orden y sus valores

```
<?php
class conectaBD
{ private $conn = null ;
  public function __construct($database='test')
  { $dsn ="mysql:host=localhost;dbname=$database" ;
    try {
      $this->conn = new PDO( $dsn, 'root', 'toor' );
    } catch ( PDOException $e) { die( "¡Error!: " . $e->getMessage() . "<br/>" ); }
  }

  public function __destruct() // Cierra conexión asignándole valor null
  { $this->conn = null; }

  public function insertaFila($num,$apel,$job,$sal,$dep=30) // Prepara y ejecuta consulta
  {
    $datos=array(':par1'=>$num, ':par2'=>$apel, ':par3'=>$job, ':par4'=>$sal, ':par5'=>$dep);
    $sql = ' INSERT INTO  empleados (emp_no,apellido,oficio,salario,dep_no)
            VALUES ( :par1 , :par2 , :par3, :par4 , :par5) ' ;
    $q = $this->conn->prepare($sql);
    return $q->execute($datos);
  }

  public function cambiaOficio($num,$job) // Prepara y ejecuta consulta
  {
    $datos=array(':par1'=>$num, ':par2'=>$job);
    $sql = ' UPDATE  empleados SET  oficio= :par2 WHERE emp_no=:par1 ' ;
    $q = $this->conn->prepare($sql);
    return $q->execute($datos);
  }

  public function borraEmpleado($num) // Prepara y ejecuta consulta
  {
    $datos=array(':par1'=>$num);
    $sql = ' DELETE FROM  empleados  WHERE emp_no=:par1 ' ;
    $q = $this->conn->prepare($sql);
    return $q->execute($datos);
  }
}

// ----- Proceso principal

$obj = new conectaBD( 'empresa' ); // crea conexión para usar bd empresa

if ($obj->insertaFila(1234,'Ruiz','Director',2500.67,10) !== false)
{echo ' se inserto con exito';
  $obj->cambiaOficio(1234,'vendedor');}
else
  echo ' fallo al insertar';

?>
```