Clase 06

PDO(PHP Data Object)

Volver

Temas:

- Introducción a PDO
- Conexiones
- Sentencias Preparadas
- PDOStatement

Introducción a PDO

PDO define una interfaz ligera para poder acceder a bases de datos en PHP.

PDO proporciona una capa de abstracción de acceso a datos, independientemente de la base de datos que se esté utilizando, se emplean las mismas funciones para realizar consultas y obtener datos.

PDO viene con PHP 5.1, y está disponible como una extensión PECL para PHP 5.0

PDO requiere las características nuevas de POO del núcleo de PHP 5, por lo que no se ejecutará con versiones anteriores de PHP.

Temas:

- Introducción a PDO
- Conexiones
- Sentencias Preparadas
- PDOStatement

Conexiones

Las conexiones se establecen creando instancias de la clase base PDO.

No importa el controlador que se utilice siempre se usará el nombre de la clase 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).

```
$conStr = "mysql:host=localhost;dbname=pruebaDB";
$pdo = new PDO($conStr, $user, $pass);
```

Conexiones

Si hubieran errores de conexión, se lanzará un objeto PDOException.

```
try {
        $conStr = "mysql:host=localhost; dbname=pruebaDB";
        $pdo = new PDO($conStr, $user, $pass);
}
catch(PDOException $e){
        echo "Error: " .$e->getMessage();
}
```

Conexiones

Una vez realizada con éxito una conexión a la base de datos, será devuelta 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 asegurándose de que todas las referencias a él existentes sean eliminadas (esto se puede hacer asignando NULL a la variable que contiene el objeto).

Si no se realiza explícitamente, PDO cerrará automáticamente la conexión cuando el script finalice.

DEMO

https://onlinegdb.com/SgLgRXbCW

Temas:

- Introducción a PDO
- Conexiones
- Sentencias Preparadas
- PDOStatement

Sentencias Preparadas

Muchas de las bases de datos más maduras admiten el concepto de sentencias preparadas.

Estas pueden definirse como un tipo de plantillas compiladas para SQL que las aplicaciones quieren ejecutar, pudiendo ser personalizadas utilizando parámetros.

Las sentencias preparadas ofrecen grandes beneficios:

- La consulta sólo necesita ser analizada (o preparada) una vez, pero puede ser ejecutada muchas veces con los mismos o diferentes parámetros.

Sentencias Preparadas

- Cuando la consulta se prepara, la base de datos analizará, compilará y optimizará su plan para ejecutarla.
- Mediante el empleo de una sentencia preparada, la aplicación evita repetir el ciclo de análisis/compilación/optimización. Esto significa que las sentencias preparadas utilizan menos recursos y se ejecutan más rápidamente.
- Los parámetros para las sentencias preparadas NO necesitan estar entrecomillados; el controlador automáticamente se encarga de esto.
- Si una aplicación usa exclusivamente sentencias preparadas, el desarrollador puede estar seguro de que no hay cabida para inyecciones de SQL.

Sentencias Preparadas

- Las sentencias preparadas son tan útiles que son la única característica que PDO emulará para los controladores que no las soporten. Esto asegura que una aplicación sea capaz de emplear el mismo paradigma de acceso a datos independientemente de las capacidades de la base de datos.

Las declaraciones preparadas básicamente funcionan así:

- Prepare() - Una plantilla de declaración de SQL se crea y se envía a la base de datos. Ciertos valores se dejan sin especificar (parámetros). (Retorna un objeto de tipo PDOStatement).

Sentencias Preparadas

- La base de datos analiza, compila y realiza la optimización de consultas en la plantilla y almacena el resultado sin ejecutarlo.
- Execute() - En un momento posterior, la aplicación enlaza ("bindea") los valores a los parámetros y la base de datos ejecuta la instrucción.

```php
// Sentencia preparada sin parámetros
$sentencia = $pdo->prepare('SELECT * FROM tabla');
$sentencia->execute();
// Sentencia preparada con parámetros
$sentencia = $pdo->prepare('SELECT * FROM tabla WHERE ID = :id');
$sentencia->execute(array(':id' => 3));
```

Temas:

- Introducción a PDO
- Conexiones
- Sentencias Preparadas
- PDOStatement

PDOStatement

Representa una sentencia preparada y, después de la ejecución de la instrucción, un conjuntos de resultados asociado.

Posee métodos para vincular (bindear) valores a parámetros.

Posee métodos para obtener los valores de un conjunto de resultados.

Temas:

- Introducción a PDO
- Conexiones
- Sentencias Preparadas
- PDOStatement
  - Métodos para vincular
  - Métodos para obtener valores

PDOStatement - Métodos para vincular

bindParam() - Vincula una variable de PHP a un parámetro de sustitución con nombre o de signo de interrogación correspondiente de la sentencia SQL que fue usada para preparar la sentencia. Sintaxis:

```
bindParam($param, &$variable ,$tipo? , $length? )
```

- $param - Identificador del parámetro.
- $variable - Nombre de la variable de PHP a vincular al parámetro de la sentencia SQL.
- $tipo - Tipo de dato explícito para el parámetro, usando las constantes PDO::PARAM_*.
- $length - Longitud del tipo de datos.

PDOStatement - Métodos para vincular

Ejemplo:

```php
$variableId = 2432; // ID a buscar

// Sentencia con parámetros nombrados
$sentencia = $pdo->prepare('SELECT * FROM tabla WHERE ID = :id');
$sentencia->bindParam(':id', $variableId, PDO::PARAM_INT);
$sentencia->execute();

// Sentencia con Parámetros posicionales
$sentencia = $pdo->prepare('SELECT * FROM tabla WHERE ID = ?');
$sentencia->bindParam(1, $variableId, PDO::PARAM_INT);
$sentencia->execute();
```

PDOStatement - Métodos para vincular

bindValue() - Vincula un valor al parámetro de sustitución con nombre o de signo de interrogación correspondiente de la sentencia SQL que fue usada para preparar la sentencia. Sintaxis:

```
bindParam($param, $valor ,$tipo?)
```

- $param - Identificador del parámetro.
- $valor - Valor a vincular al parámetro de la sentencia SQL.
- $tipo - Tipo de dato explícito para el parámetro, usando las constantes PDO::PARAM_*.

PDOStatement - Métodos para vincular

Ejemplo:

```php
$variableId = 2432; // ID a buscar

// Sentencia con parámetros nombrados
$sentencia = $pdo->prepare('SELECT * FROM tabla WHERE ID = :id');
$sentencia->bindValue(':id', 2432, PDO::PARAM_INT);
$sentencia->execute();

// Sentencia con Parámetros posicionales
$sentencia = $pdo->prepare('SELECT * FROM tabla WHERE ID = ?');
$sentencia->bindValue(1, $variableId, PDO::PARAM_INT);
$sentencia->execute();
```

PDOStatement - Métodos para vincular

bindColumn() - Vincula una columna a una variable de PHP. Cada llamada a PDOStatement::fetch() o a PDOStatement::fetchAll() actualizará todas las variables que estén vinculadas a columnas.

Sintaxis:

```
bindColumn($column, $variable ,$tipo?, $maxLen?)
```

- $column - Número (base 1) o nombre de la columna del conjunto de resultados.
- $variable - Nombre de la variable de PHP a la que vincular la columna.
- $tipo - Tipo de dato explícito para el parámetro, usando las constantes PDO::PARAM_*.
- $maxLen - Longitud máxima sugerida para la pre asignación.

PDOStatement - Métodos para vincular

Ejemplo:

```php
$sentencia = $pdo->prepare('SELECT col1, col2, col3 FROM tabla ');
$sentencia->execute();

/* Vincular por número de columna */
$sentencia->bindColumn(1, $var1, PDO::PARAM_INT);
$sentencia->bindColumn(2, $var2, PDO::PARAM_STR);

/* Vincular por nombre de columna */
$sentencia->bindColumn('col3', $var3, PDO::PARAM_BOOL);

while ($fila = $sentencia->fetch(PDO::FETCH_BOUND)) {
        $datos = $var1 . "\t" . $var2 . "\t" . $var3 . "\n";
        print $datos;
}
```

Temas:

- Introducción a PDO
- Conexiones
- Sentencias Preparadas
- PDOStatement
  - Métodos para vincular
  - Métodos para obtener valores

PDOStatement - Métodos para obtener valores

fetch() - Obtiene una fila de un conjunto de resultados asociado al objeto PDOStatement. El parámetro fetch_style determina cómo PDO devuelve la fila. Sintaxis:

```
fetch($fetch_style?)
```

- $fetch_style - Este valor debe ser una de las constantes PDO::FETCH_*.
- El valor de retorno de esta función en caso de éxito depende del tipo de obtención. En todos los casos, se devuelve FALSE en caso de error.

PDOStatement - Métodos para obtener valores

- PDO::FETCH_ASSOC - Devuelve un array indexado por los nombres de las columnas del conjunto de resultados.
- PDO::FETCH_NUM - Devuelve un array indexado por el número de columna tal como fue devuelto en el conjunto de resultados, comenzando por la columna 0.
- PDO::FETCH_BOTH - Devuelve un array indexado tanto por nombre de columna, como numéricamente con índice de base 0 tal como fue devuelto en el conjunto de resultados.

PDOStatement - Métodos para obtener valores

- 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_CLASS - Devuelve una nueva instancia de la clase solicitada, haciendo corresponder las columnas del conjunto de resultados con los nombres de las propiedades de la clase.
- 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().

PDOStatement - Métodos para obtener valores

fetchAll() - Devuelve un array que contiene todas las filas de un conjunto de resultados. El parámetro fetch_style determina cómo PDO devuelve la fila. Sintaxis:

```
fetchAll($fetch_style?)
```

- $fetch_style - Este valor debe ser una de las constantes PDO::FETCH_*.
- El valor de retorno de esta función en caso de éxito depende del tipo de obtención. En todos los casos, se devuelve FALSE en caso de error.

PDOStatement - Métodos para obtener valores

fetchObject() - Obtiene la siguiente fila y la devuelve como un objeto.

Esta función es una alternativa para PDOStatement::fetch() con el estilo PDO::FETCH_CLASS o PDO::FETCH_OBJ. Sintaxis:

```
fetchObject($className, $args?)
```

- $className - Nombre de la clase creada.
- $args - Los elementos de este array son pasados al constructor.

Dudas ó consultas