PHP - MySql

PDO (PHP Data Object) egy PHP kiterjesztés, amely adatbázis absztrakciós réteget valósít meg. Előnyei:

- Egységes interfész különböző adatbázis kezelőkre. A PDO megvalósítja a különböző adatbázisok kezelőit.
- A paraméterek kötésének (parameter binding) köszönhetően biztonságosabb a kód.
- A paraméterek kötésének köszönhetően gyorsabb lesz a végrehajtás, ha ugyanazt a lekérdezést kell végrehajtani a paraméterek különböző értékeivel.
- Konzisztens hibakezelés lehetősége.

A PDO alapértelmezés szerint engedélyezve van a PHP telepítésekor. Két kiterjesztésre lesz szükségünk, hogy tudjunk dolgozni php szkriptekben mysql adatbázissal PDO használatával: **pdo** és **pdo_mysql**.

PHP - MySql

Kapcsolódás az adatbázishoz:

```
<?php
    $db = new PDO(
    'mysql:host=localhost;dbname=testdb;charset=utf8',  // DSN
    'username',  // Adatbázis felhasználó neve
    'password', // Adatbázis felhasználó jelszava
    array(PDO::ATTR_EMULATE_PREPARES => false,
            PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION) // Attribútumok tömbje
);
```

A DSN (Data Source Name, Adatforrás neve) adja meg melyik a kezelőt, jelen esetben mysql, és a kapcsolat részleteit.

Az attribútumok külön is definiálhatók:

```
<?php
$db = new PDO(
'mysql:host=localhost;dbname=testdb;charset=utf8', 'username', 'password'
);
$db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
$db->setAttribute(PDO::ATTR_EMULATE_PREPARES, false);
```

PHP - MySql

Hibakezelés:

A PDO-ban van három hibakezelési mód:

- PDO::ERRMODE_SILENT - ellenőrizni kell az eredményeket, majd a $db->errorInfo() segítségével megtekinteni a hiba részleteit
- PDO::ERRMODE_WARNING dob PHP figyelmeztetéseket
- PDO::ERRMODE_EXCEPTION dob PDOException típusú kivételeket.

```
<?php
    try {
        $db = new PDO('mysql:host=localhost;dbname=testdb;charset=utf8', 'username',
                        'password');
        $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        $db->setAttribute(PDO::ATTR_EMULATE_PREPARES, false);
        $db->query('hi');          //Érvénytelen utasítás!
    } catch(PDOException $ex) {
        echo "Hiba történt!";    // Felhasználó barát üzenet
        some_logging_function($ex->getMessage()); // Naplózási funkció hívása
    }
```

PHP - MySql

Egyszerű lekérdezések (egyszerű SELECT utasítás):

```
<?php
    foreach($db->query('SELECT * FROM table') as $row) {  // $row – egy sort tartalmazó tömb
        echo $row['field1'].' '.$row['field2'];      //stb...
    }
```

vagy

```
<?php
    $stmt = $db->query('SELECT * FROM table');    // $stmt tartalmaz egy PDOStatement objektum
    while($row = $stmt->fetch(PDO::FETCH_ASSOC)) {   // $row – egy sort tartalmazó tömb
        echo $row['field1'].' '.$row['field2'];      //stb...
    }
```

vagy

```
<?php
    $stmt = $db->query('SELECT * FROM table');
     $results = $stmt->fetchAll(PDO::FETCH_ASSOC);
    //használjuk $results-t (az összes sort tartalmazó tömb, $results[0], $results[1], … az egyes sorok)
```

Fetch módok:

- PDO::FETCH_ASSOC – a sorok asszociatív tömbök, amelyekben a mezők nevei a tömb kulcsai
- PDO::FETCH_NUM – a sorok tömbök numerikus kulcsokkal
- PDO::FETCH_BOTH (alapértelmezett) – az adatok kétszer szerepelnek, egyszer asszociatív kulccsal és egyszer numerikus kulccsal

PHP - MySql

Eredmény sorok száma:

```
<?php
$stmt = $db->query('SELECT * FROM table');
$row_count = $stmt->rowCount();
echo $row_count.' sor az eredményben';
```

Egyszerű INSERT, UPDATE vagy DELETE utasítás:

```
<?php
$affected_rows = $db->exec("UPDATE table SET field='value'");
echo $affected_rows.' sor módosítva';
```

Megjegyzés: teljesen azonos az INSERT és a DELETE végrehajtása.

Utolsó beszúrt sor azonosítójának lekérdezése:

```
<?php
$result = $db->exec("INSERT INTO table(firstname, lastname) VAULES('József', 'Kovács')");
$insertId = $db->lastInsertId();
```

PHP - MySql

Paraméteres utasítások:

```
<?php
$stmt = $db->prepare("SELECT * FROM table WHERE id=? AND name=?");
$stmt->execute(array($id, $name));
$rows = $stmt->fetchAll(PDO::FETCH_ASSOC);
```

vagy

```
<?php
$stmt = $db->prepare("SELECT * FROM table WHERE id=? AND name=?");
$stmt->bindValue(1, $id, PDO::PARAM_INT);
$stmt->bindValue(2, $name, PDO::PARAM_STR);
$stmt->execute();
$rows = $stmt->fetchAll(PDO::FETCH_ASSOC);
```

FONTOS MEGJEGYZÉS:
A **prepare** utasítás küldi az sql utasítást az adatbázis szervernek, amely lefordítja az sql utasítást és várja a paraméterek értékeit. Az **execute** utasítás küldi a paraméterek értékeit és az sql utasítás kerül végrehajtásra, de újabb fordítás nem történik, így a paraméter értékeiben nem mehet végrehajtandó sql utasítás, azaz az sql injekció nem lehetséges.

PHP - MySql

Névvel ellátott paraméterek (helyörzők):

```
<?php
$stmt = $db->prepare("SELECT * FROM table WHERE id=:id AND name=:name");
$stmt->bindValue(':id', $id, PDO::PARAM_INT);
$stmt->bindValue(':name', $name, PDO::PARAM_STR);
$stmt->execute();
$rows = $stmt->fetchAll(PDO::FETCH_ASSOC);
```

vagy

```
<?php
$stmt = $db->prepare("SELECT * FROM table WHERE id=:id AND name=:name");
$stmt->execute(array(':name' => $name, ':id' => $id));
$rows = $stmt->fetchAll(PDO::FETCH_ASSOC);
```

PHP - MySql

Előkészített INSERT:

```
<?php
$stmt = $db->prepare("INSERT INTO table(field1,field2,field3,field4,field5)
                                VALUES(:field1,:field2,:field3,:field4,:field5)");
$stmt->execute(array(':field1' => $field1, ':field2' => $field2, ':field3' => $field3, ':field4' => $field4,
                                ':field5' => $field5));
$affected_rows = $stmt->rowCount();
```

Előkészített DELETE:

```
<?php
$stmt = $db->prepare("DELETE FROM table WHERE id=:id");
 $stmt->bindValue(':id', $id, PDO::PARAM_STR);
$stmt->execute();
$affected_rows = $stmt->rowCount();
```

Előkészített UPDATE:

```
<?php
$stmt = $db->prepare("UPDATE table SET name=? WHERE id=?");
$stmt->execute(array($name, $id));
$affected_rows = $stmt->rowCount();
```

PHP - MySql

Előkészített utasítások SQL függvényekkel:

```
<?php
$name = 'BOB';
$stmt = $db->prepare("INSERT INTO table(`time`, `name`) VALUES(NOW(), ?)");
$stmt->execute(array($name));
```

```
<?php
$name = 'BOB';
$password = 'badpass';
$stmt = $db->prepare("INSERT INTO table(`hexvalue`, `password`) VALUES(HEX(?), PASSWORD(?))");
$stmt->execute(array($name, $password));
```

Megjegyzés: A paraméter szerepelhet függvény attribútumaként, de függvény nem lehet paraméter értéke.

Előkészített utasítások LIKE kulcsszóval:

```
<?php
$stmt = $db->prepare("SELECT field FROM table WHERE field LIKE ?");
$stmt->bindValue(1, "%$search%", PDO::PARAM_STR);
$stmt->execute();
```

PHP - MySql

Előkészített SQL utasítások cikluson belül:

```
<?php
$values = array('bob', 'alice', 'lisa', 'john');
$name = '';
$stmt = $db->prepare("INSERT INTO table(`name`) VALUES(:name)");
$stmt->bindParam(':name', $name, PDO::PARAM_STR);
foreach($values as $name) {
    $stmt->execute();
}
```

A bindValue és a bindParam hasonló, de amíg a bindValue értéket köt a paraméterhez, a bindParam változót köt a paraméterhez. Ez a különbség nagyon fontos:

- A bindValue köti a második paramétere aktuális értékét az SQL utasítás paraméteréhez.
- A bindParam köti a második paramétere helyén szereplő változót (csak változó lehet!) az SQL utasítás paraméteréhez, amely csak akkor kapja a változó értékét, amikor az execute kerül végrehajtásra. Ha a kötés után változott a változó értéke, akkor az execute a változó megváltozott (az execute végrehajtásának pillanatában aktuális) értékével kerül végrehajtásra.

PHP - MySql

Transzakciók:

```
<?php
try {
	$db->beginTransaction();
	$db->exec("EGY SQL UTASÍTÁS");
	$stmt = $db->prepare("MÁSIK SQL UTASÍTÁS");
	$stmt->execute(array($value));
	$stmt = $db->prepare("MÉG EGY SQL UTASÍTÁS");
	$stmt->execute(array($value2, $value3));
	$db->commit();
}
catch(PDOException $ex) {
	// Hiba volt valahol!
	$db->rollBack();
	echo $ex->getMessage();
}
```