OO PDO

PDO VS MYSQLi VS MYSQL

==========

A PHP mysql metódusai elavultak, helyette lehet hazsnálni a MYSQLi metódusokat, amelyek szinte azonos szignatúrával használhatók. A MYSQLi-n az "i" betű az improved továbbfejlesztett szóból ered.

Tehát a sima MYSQL metódusok használatát kerüljük.

PRO KONTRA

==========

A PDO 12 különböző adatbáziskezelőt támogat, míg a MYSQLi csak a mysqlt.

Tehát PDO használatával, könnyen átállhatunk egy másik adatbáziskezelőre, anélkül hogy a kódot teljes egészében újraírnánk.

Mind a kettő objektum orientált, de a MYSQLi-t lehet eljárásorientáltan is használni.

Mindkettő támogatja a prepared statmentet, amely véd az SQL injection támadások ellen.

A névvel ellátott paramétereket csak a PDO támogatja, melyet a prepared statmentnél kell használni. Hiánya a MYSQLi-nél nem egy nagy hátrány, megszokás kérdése.

Mindkettő támogatja az Object mappingot, amely segítségével nagyon szépen lehet kezelni a rekordokat.

Teljesítményben, mindkettő gyors bár a MYSQLi a tesztek alapján ~2.5%-al gyorsabb a prepared statementses lekérdezéseknél, míg a sima lekérdezések esetében ~6.5%-al gyorsabb.

CSATLAKOZÁS AZ ADATBÁZISHOZ

```
1.  //A "dbname" értéke adja meg azt az adatbázist amihez csatlakoznmi szeretnénk
2.  $servername = "localhost"; //szervernév
3.  $username = "root"; //felhasználónév
4.  $password = ""; //jelszó
5.  $dbName = "tanulmany"; //adatbázis név amihez csatlakozni akarunk
6.  try {
7.      $con = new PDO("mysql:host=$servername;dbname=$dbName;",
8.                      $username, $password);
9.      echo "Sikeres csatlakozás";
10. } catch(PDOException $e) { //a kivételkezelő catch ága, amely PDO kivételeket kezel

11.     echo "Hiba: ".$e->getMessage();
12. }
```

CSATLAKOZÁS LEZÁRÁSA

```
1.  $con = null;
```

ADATBÁZIS LÉTREHOZÁSA

Hibakezelés beállítása $con->setAttribute(PDO::ATTR_ERRMODE, …)

```
1.  try {
2.      //újra csatlakozunk, viszont itt nem adjuk meg, hogy melyik adatbázishoz, hiszen
     azt később hozzuk létre
3.      $con = new PDO("mysql:host=$servername;",
4.                      $username, $password);
5.      $con-
    >setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION); //Hiba esetén kivételt dob

6.      //$con-
    >setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING); //Hiba esetén warning üzenet

7.      //$con-
    >setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_SILENT); //Hiba esetén nem történik se
    mmi
8.
9.      //$dbName nevű adatbázis létrehozása
10.     $sql = "CREATE DATABASE ".$dbName." CHARACTER SET = 'UTF8'";
11.     $con->exec($sql); //sql parancs végrehajtása
12.     echo "Adatbázis létrehozva";
13. } catch(PDOException $e) { //a kivételkezelő catch ága, amely PDO kivételeket kezel

14.     echo "Hiba: ".$e->getMessage();
15. }
```

TÁBLA LÉTREHOZÁSA

```php
try {
    //adatbázis kijelölése
    $con->exec("USE ".$dbName);

    // sql utasítás ami létrehozza a diakok táblát
    $sql = "CREATE TABLE diakok (
        id INT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
        nev varchar(50),
        osztaly varchar(5),
        eletkor int
    )";
    $con->exec($sql); //sql utasítás futtatása
    echo "Tábla létrehozva";
} catch(PDOException $e) { //a kivételkezelő catch ága, amely PDO kivételeket kezel

    echo "Hiba: ".$e->getMessage();
}
```

EGY ADAT BESZÚRÁSA (a legutóbb beszúrt sor ID visszaadása)

```php
try {
     //karakterkódolás beállítása 1* kell egy csatlakozáshoz beállítani
     $con->exec("SET NAMES 'utf8'");

     $sql="INSERT INTO diakok (nev, osztaly, eletkor)
     VALUES ('Aladár Péter', 'SZFX',21)";
     $con->exec($sql);
     $id = $con->lastInsertId(); //legutóbbi beszúrt sor idja
     echo "Legutóbbi beszúrt sor idja: ".$id;
 } catch(PDOException $e) { //a kivételkezelő catch ága, amely PDO kivételeket kezel

     echo "Hiba: ".$e->getMessage();
 }
```

TÖBB ADAT BESZÚRÁSA (tranzakciókezelés)

```php
try {
    $con->beginTransaction(); //tranzakciókezelés kezdete

    $con->exec("INSERT INTO diakok (nev, osztaly, eletkor)
                VALUES ('Aladár Józsi', 'SZFX',25)");
    $con->exec("INSERT INTO diakok (nev, osztaly, eletkor)
                VALUES ('Példa Léna', 'SZFX',26)");

    $con->commit(); //ha minden rendben, mehet adatbázisba
    echo "Adatok felvitele sikeres";
} catch(PDOException $e) {
    $con->rollback(); //hiba esetén nem kerül semmi adatbázisba
    echo "Hiba: ".$e->getMessage();
}
```

PREPARED STATEMENT (SQL injection támadás)

A prepared statement az SQL injection támadások ellen nyújt védelmet.

SQL injection: Amikor a megírt sql sztringünkbe nem értéket szúrunk be, hanem egy rossz indulatú SQL utasítást.

PL.: Tegyük fel, hogy van egy hozzászólás űrlap az oldalunkon egyetlen beviteli mezővel (message).

Az sql injection esetén a felhasználó nem egy hozzászólást küld a szervernek, hanem egy rossz indulatú SQL utasítást:

"hozzászólás');DROP ('users"

Vizsgáljuk meg mi történik ha ezt beszúrjuk az SQL utasításunkba:

INSERT INTO uzenetek (uzenet) VALUES('$_POST["messages"]')

Ebben az esetben, az SQL utasítás alapja kerül csak megírásra az értékek, amelyek az elküldött űrlapból érkeznek azokat csak később helyettesíti be a rendszer az SQL utasításba.

Amikor a rendszer egyesével behelyettesíti az értékeket, mindegyiken végrehajt egy ellenőrzést.

Ha valamelyik ellenőrzés elbukik, akkor nem hajtja végre a rendszer az SQL utasítást.

```php
try {
    //1. felkészülünk az sql utasításra, az űrlapból érkező adatok helyére ? jelet r
akunk
    $stmt = $con->prepare("INSERT INTO diakok (nev, osztaly, eletkor)
                           VALUES(?,?,?)");
    //2. változók hozzákapcsolása az sql utasításhoz (sorrendre figyelni)
    //Az 1-es az első kérdőjelhez, a 2-es a másodikhoz és így tovább
    $stmt->bindParam(1, $nev);
    $stmt->bindParam(2, $osztaly);
    $stmt->bindParam(3, $eletkor);

    //3. Változók feltöltése értékkel (űrlapból) és az sql utasítás futtatása.
    $nev       = "Prepare Pista";
    $osztaly   = "STMT";
    $eletkor   = "22";
    $stmt->execute();

    //3. Újabb sor felvitele. Csak az értékek változnak az sql utasítás ugyan az.
    $nev       = "Prepare Béla";
    $osztaly   = "STMT";
    $eletkor   = "23";
    $stmt->execute();

} catch(PDOException $e) {
    echo "Hiba: ".$e->getMessage();
}
```

PREPARED STATEMENT (nevesített paraméterekkel)

Ugyan az, mint az előző, csak a kérdőjelek helyett nevesített paraméterlistát használunk. Előnye hogy nem kell a sorrendre figyelnünk.

```
try {
      $stmt = $con->prepare("INSERT INTO diakok (nev, osztaly, eletkor)
                             VALUES(:nev,:osztaly,:eletkor)");
      //az előzőhöz hasonlóan csak nevesített paraméterekel
      /*$stmt->bindParam(':nev', $nev);
      $stmt->bindParam(':osztaly', $osztaly);
      $stmt->bindParam(':eletkor', $eletkor);

      $nev       = "Neves Prepare";
      $osztaly   = "STMT";
      $eletkor   = "22";
      $stmt->execute();*/

      //a fentivel ekvivalens, csak rövidebb
      $stmt->execute([
          'nev'       => 'Neves Prepare',
          'osztaly'   => 'STMT',
          'eletkor'   => 222,
      ]);
  } catch(PDOException $e) {
      echo "Hiba: ".$e->getMessage();
  }
```

SELECT (+prepared statements)

```
try {
    $stmt = $con->prepare("SELECT * FROM diakok WHERE eletkor = ?");
    $stmt->bindParam(1, $eletkor);
    $eletkor = 22;
    $stmt->execute();
    $result = $stmt->setFetchMode(PDO::FETCH_ASSOC);
    while ($row = $stmt->fetch()) {
        print_r($row);
    }

} catch(PDOException $e) {
    echo "Hiba: ".$e->getMessage();
}
```

UPDATE (+prepared statements)

```
try {
    $stmt = $con-
>prepare("UPDATE diakok SET nev='Norbi Update' WHERE eletkor = ?");
    $stmt->bindParam(1, $eletkor);
    $eletkor = 22;
    $stmt->execute();
    echo "Sikers update";
} catch(PDOException $e) {
    echo "Hiba: ".$e->getMessage();
}
```

DELTE (+prepared statements)

```
try {
    $stmt = $con->prepare("DELETE FROM diakok WHERE eletkor = ?");
    $stmt->bindParam(1, $eletkor);
    $eletkor = 21;
    $stmt->execute();
    echo "Rekordok törlve";
} catch(PDOException $e) {
    echo "Hiba: ".$e->getMessage();
}
```

TÁBLA TÖRLÉSE

```
try {
    $con->exec("DROP TABLE IF EXISTS diakok");
    echo "Sikers tábla törlés";
} catch(PDOException $e) {
    echo "Hiba: ".$e->getMessage();
}
```

ADATBÁZIS TÖRLÉSE

```
try {
    $con->exec("DROP DATABASE IF EXISTS tanulmany");
    echo "Sikers adatbázis törlés";
} catch(PDOException $e) {
    echo "Hiba: ".$e->getMessage();
}
```