Extrémen brutál, gyors talpaló PHP nyelvhez (database). v2.1

Eljárásorientált MySqli

Írta: Girhiny Róbert a w3schools.com PHP MySQL leírása alapján.

http://www.w3schools.com/php/php_mysql_intro.asp
http://www.w3schools.com/php/php_ref_mysqli.asp

Előfeltétel:

Minimális PHP tudás: http://www.w3schools.com/php/default.asp
Minimális SQL tudás: http://www.w3schools.com/sql/default.asp

MySql adatbázis csatlakozás

Mielőtt az adatbázis adataihoz hozzáférnénk, előbb csatlakoznunk kell hozzá, PHP-ben erre szolgál a mysqli_connect() függvény.

Szintaktikája:

```
mysqli_connect(servername,username,password[,database_name,port]);
```

servername: A szerver nevét adhatjuk meg vele, amihez csatlakozni szeretnénk. Alapértelmezett értéke: localhost
username: A bejelentkezéshez a felhasználónév.
password: A bejelentkezéshez a jelszó.
database_name: Ha megadjuk, akkor alapértelmezetten kapcsolódik a kapott adatbázishoz.
port: Csatlakozási port

Visszatérési értéke:
Ha sikeres volt a csatlakozás akkor a csatlakozás azonosítójával, ha nem sikerült akkor FALSE értékkel tér vissza.

Forráskód:

```
1.  $con = mysqli_connect("localhost","root","");
2.  if (!$con) {
3.    die('Hiba: ' . mysqli_connect_error($con));
4.  }
5.  echo "Sikeresen csatlakoztunk az adatbázishoz<br />";
```

megj.: A mysqli_error() metódus a legutóbbi mysql hibaüzenettel tér vissza. Egyetlen paramétere a csatlakozási azonosító.

A die() metódus a paraméterül kapott sztringet kiírja, majd befejezi az adott fájl futtatását.

A mysqli_connect_error() metódus hasonló a mysqli_error() metódushoz, viszont ez a legutóbbi csatlakozás hibjával tér vissza. A mysqli_errno() metódus a legutóbbi hiba kódjával, míg mysqli_connect_errno() a legutóbbi csatlakozási hibakóddal tér vissza.

Adatbázis létrehozása

A CREATE DATABASE utasítás MySQL-ben arra szolgál hogy létrehozzunk egy adatbázist. Ezt a parancsot a PHP-vel is futtathatjuk, ehhez a mysqli_query() függvényt kell használni. Ez a függvény arra használható, hogy egy mysql lekérdezést futtasunk egy már korábban kapcsolódott adatbázison.

```
mysqli_query(connection_id,  query[,  resultmode]);
connection_id:  a csatlakozás azonosítója
query:          az sql utasítás maga
resultmode:     Opcionális. Nagy mennyiségű adatnál használjuk a
                MYSQLI_USE_RESULT konstnst.
```

Visszatérési értéke: SELECT típusú queryk esetén mysqli_result objektumot ad a lekérdezés eredményével. Ezt egy while ciklussal könnyen bejárhatjuk és feldolgozhatjuk a lekérdezett adatokat (lásd későbbi fejezet). Egyéb típusú query esetén TRUE vagy FALSE logikai értékekkel tér vissza.

megj.: Az összetett több soros sql utasításokat érdemes egy változóba írni, mert ha végrehajtásakor valami probléma adódik, akkor könnyen ki lehet echózni és rögtön látszik mit írtunk el.

A CREATE DATABASE utasításnál célszerű az adatbázis karakterkódolását is megadni, amit a CHARACTER SET = 'UTF8' utasítással tehetünk meg.

```php
$con = mysqli_connect("localhost","root",""); //újracsatlakozunk
// az alábbi példa létrehozza a tanulmany adatbázist
if (mysqli_query($con, "CREATE DATABASE tanulmany CHARACTER SET = 'UTF8'"))
    echo "Adatbázis létrehozva";
else
    echo "Hiba: " . mysqli_error($con);
```

Adatbázis kijelölése

Mielőtt bármilyen műveletet tudnánk végezni az adatbázison előbb ki kell választanunk, hogy melyik adatbázisal fogunk dolgozni. Erre szolgál a

`mysqli_select_db(connection_id, database_name) függvény.`

Első paramétere a csatlakozás azonosítója a második az adatbázis nevét határozza meg.

Visszatérési értéke: TRUE vagy FALSE, attól függően hogy sikerült e kijelölni az adatbázist.

Forráskód:

```php
mysqli_select_db($con, "tanulmany"); //adatbázis kijelölése csatlakozás azonosítóval
```

Tábla létrehozása

A CREATE TABLE utasítást használjuk tábla létrehozására.

Szintaxisa:

```
CREATE TABLE table_name
(
oszlop_név1 adat_típus,
oszlop_név2 adat_típus,
oszlop_név3 adat_típus,
....
)
```

Ahhoz hogy ezt az utasítás végrehajtsuk, át kell adnunk a mysqli_query() függvénynek.

Forráskód:

```php
$sql = "CREATE TABLE diakok (
    id int NOT NULL AUTO_INCREMENT,
    PRIMARY KEY(id),
    nev varchar(50),
    osztaly varchar(5),
    eletkor int
    )";
if (mysqli_query($con, $sql)) // Lekérdezés futtatása
    echo "diakok tábla létrehozva";
else
    echo "Hiba: ". mysqli_error($con);
```

Kapcsolat karakterkódolásának beállítása

Mielőtt az adatbázisba beszúrnánk adatokat, célszerű beállítani a a kapcsolat karakterkódolását, hogy ne érjenek meglepetések ékezetes karakterek esetén. Erre szolgál a mysqli_set_charset() metódus, melynek első paramétere a csatlakozási azonosító, második a használni kívánt karakterkódolás.

Forráskód:

```php
mysqli_set_charset($con, 'utf8'); //beállítja adott kapcsolathoz az alapértelmezett karakterkódolást
```

Adatok beszúrása a táblába

Az insert into szintaktikája:

```
INSERT INTO table_name (column1, column2, column3,...)
VALUES (value1, value2, value3,...)
```

Forráskód:

```
1.  if (mysqli_query($con, "
2.      INSERT INTO diakok (nev, osztaly, eletkor)
3.      VALUES ('Aladár Péter', 'SZFX',21)"
4.      ))
5.      echo "az adatot sikeresen felvittük";
6.  else
7.      echo "Hiba: ". mysqli_error($con);
```

Legutóbb beszúrt sor ID-jának kinyerése

A mysqli_insert_id() függvény a legutóbb beszúrt sor id-jával (azzal az azonosítóval ami AUTO_INCREMENT-el lett generálva) tér vissza. Egyetlen paramétere a csatlakozás azonosítója.

Forráskód:

```
1.  echo "Legutóbb beszúrt sor id-ja: ".mysqli_insert_id($con);
```

megj.: A legutóbb beszúrt sor id-ja akkor lehet hasznos, ha egy kapcsolódó táblába szintén be kell szúrnunk egy másik rekordot és hivatkoznunk kell erre az id-ra.

Adatok beszúrása űrlapból

Adatokat felvihetjük űrlapból is. Az alábbi példa ugyanezt a fájlt hívja meg. Az űrlap elküldésekor a submit tulajdonság beállításra kerül. Az isset metódussal ezt figyeljük, ha van postolva submit tulajdonság, akkor fel kell vinni az adatbázisba az adatokat. (Tehát csak akkor hajtódik végre az insert parancs, ha a submit tulajdonság ott van a $_POSt tömbben). A $_POST['name'], $_POST['class'], $_POST['age'] elemek fogják tárolni az űrlapba beírt értékeket. Hogy miért name,class és age, azért mert ezeket a neveket adtuk a beviteli mezők name attribútum értékének. PHP-ben a $_POST tömbből lekérdezett értékeket hozzáfűzzük a futtatni kívánt INSERT INTO sql utasításhoz.

Forráskód (ügyeljünk hogy a kód 1-6-ig része sima HTML nem <?php ?> tagek közé írandó):

```
1.  <form method="post">
2.  név: <input type="text" name="name" />
3.  osztály: <input type="text" name="class" />
4.  életkor: <input type="text" name="age" />
5.  <input type="submit" name="submit">
6.  </form>
7.  <?php
```

```php
if (isset($_POST['submit'])) {
    if (mysqli_query($con, "
        INSERT INTO diakok (nev, osztaly, eletkor)
        VALUES ('".$_POST['name']."', '".$_POST['class']."',".$_POST['age'].")"
        ))
        echo "az adatot sikeresen felvittük";
    else
        echo "Hiba: ". mysqli_error($con);
}
?>
```

Egyszerű lekérdezés

MySQL-ben a SELECT használható adatok lekérésére az adatbázisból

Szintaxisa:

```
SELECT column_name(s) FROM table_name
```

column_name(s):	melyik oszlopokra van szükségünk
table_name:	melyik táblából

Szintén a mysqli_query() függvény használatával tudjuk végrehajtani a SELECT-et.

Visszatérési értéke ha sikeres volt, akkor a lekérdezett adatok (mysqli_result objektum típusként), ellenkező esetben FALSE.

Forráskód:

```php
$data = mysqli_query($con, "SELECT * FROM diakok"); //lekérdezés. eredménye a $data változóban.
if ($data)
    echo "a lekérdezés sikeres<_br />";
else
    echo "Hiba: ". mysqli_error($con);
//A lekérdezett adatok kiíratása. A $row asszociatív tömbben az indexek az oszlopok nevei.
//A mysqli_fetch_array() függvény az adathalmazból mindíg a következő sort veszi ki.

//Így a while ciklus addig fut, míg a $row változó fel nem veszi az összes sor értéket amit a SELECT visszaadott. Ha már nincs több sor NULL értékkel tér vissza.
while($row = mysqli_fetch_array($data)) {
    echo $row['nev']." ".$row['osztaly']." ".$row['eletkor']."<_br />";
}
```

Egyszerű lekérdezés táblázatos megjelenítéssel

A feladat ugyan az mint az előbb, a különbség hogy az adatokat táblázatos formában jelenítjük meg.

Forráskód:

```php
$str = "<table border='1'>";
$data = mysqli_query($con, "SELECT * FROM diakok"); //lekérdezés. eredménye a $data változóban.
while($row = mysqli_fetch_array($data)) {
    $str .= "<tr>";
    $str .= "<td>".$row['nev']."</td><td>".$row['osztaly']."</td><td>".$row['eletkor']."</td>";
    $str .= "</tr>";
}
$str .= "</table>";
echo $str;
```

Where záradék

A where záradék való arra, hogy csak azokat az adatokat kérdezzük le, amelyek eleget tesznek egy bizonyos feltételnek.

Szintaxis:

```
SELECT column_name(s) FROM table_name
WHERE column_name operator value
```

column_name:	oszlop név, amire a feltételt szeretnénk megadni
operator:	összehasonlító operátor
value:	érték, ezzel vizsgáljuk (pl.: hogy egynelő-e vele, nagyobb-e stb.)

Forráskód:

```php
$data = mysqli_query($con, "SELECT * FROM diakok WHERE osztaly='wp11'"); //csak azokat a diákokat kérdezzük le, akik a wp11 osztályba járnak

while($row = mysqli_fetch_array($data)) {
    echo $row['nev']." ".$row['osztaly']." ".$row['eletkor']."<br />";
}
```

Az ORDER BY kulcsszó

Az ORDER BY kulcsszót használhatjuk arra,

hogy a lekérdezés adatait rendezzük, növekvő vagy csökkenő sorrendbe.

Szintaxis:

```
SELECT column_name(s) FROM table_name
ORDER BY column_name(s)2 ASC|DESC
```

column_name(s)2:	Melyik oszlop szerint rendezzünk
ASC|DESC:	ASC esetén növekvő, DESC esetén csökkenő sorrendbe történik a rendezés. Ha elhagyjuk, akkor alapértelmezetten növekvő sorrendbe rendezi.

Forráskód:

```php
$data = mysqli_query($con, "SELECT * FROM diakok ORDER by eletkor DESC"); //csak azokat a diákokat kérdezzük le, akik a wp11 osztályba járnak

while($row = mysqli_fetch_array($data)) {
    echo $row['nev']." ".$row['osztaly']." ".$row['eletkor']."<br />";
}
```

UPDATE

Az UPDATE parancsot használhatjuk arra, hogy egy meglévő sort módosítsunk

Szintaxis:

```
UPDATE table_name
SET column1=value, column2=value2,...
WHERE some_column=some_value
```

A WHERE feltétel határozza meg hogy melyik sort/sorokat módosítsuk. Ha elhagyjuk akkor az összes sort módosítja. A SET kulcsszóval állíthatjuk be hogy melyik mezőnek mi legyen az új értéke.

Forráskód:

```php
$sql = "UPDATE diakok SET eletkor=100
        WHERE osztaly='wp11' AND eletkor<25";
if (mysqli_query($con, $sql))
    echo "A módosítás sikeresen megtörtént";
else
    echo "Hiba: ".mysqli_error($con);
```

Sorok törlése táblából

A DELETE FROM table_name parancsot használhatjuk arra, hogy sorokat töröljünk az adatbázisból.

Szintaxis:

```
DELETE FROM table_name
WHERE some_column = some_value
```

table_name: Melyik táblából
A WHERE feltétel határozza meg hogy melyik sort/sorokat töröljük.

Forráskód:

```
1. mysqli_query($con, "DELETE FROM diakok WHERE eletkor=100"); //azokat a diákokat töröljük, akkik 100 évesek
```

Tábla törlése

A DROP TABLE table_name parancsot használhatjuk táblák törlésére.

Szintaxis:

```
DROP TABLE [IF EXISTS] table_name [, table_name] ...
```

table_name: A törölni kívánt táblák neve, vesszővel több is megadható.

Forráskód:

```
1. mysqli_query($con, "DROP TABLE IF EXISTS diakok");//eldobjuk a táblát ha már létezik
```

Adatbázis törlése

A DROP DATABASE *db_name* parancsot használhatjuk arra, hogy töröljünk egy adatbázist a benne lévő táblákkal és adatokkal.

Szintaxis:

```
DROP DATABASE [IF EXISTS] db_name
    db_name:    A törölni kívánt adatbázis neve
```

Forráskód:

```
1. mysqli_query($con, "DROP DATABASE IF EXISTS tanulmany"); //eldobjuk az adatbázist
```

Csatlakozás lezárása, befejezése

A csatlakozás automatikusan bezárul, ha a szkript véget ér. Ha előbb be szeretnénk zárni, akkor a mysqli_close(*connection_id*) függvénnyel tehetjük meg. Egyetlen paramétere a csatlakozás azonosítója. Visszatérési értéke TRUE vagy FALSE, attól függően, hogy sikerült e lezárni a kapcsolatot.

Forráskód:

```
1.  if (mysqli_close($con))
2.      echo "Kapcsolat lezárva";
3.  else
4.      echo "Hiba: " . mysqli_error($con);
```