Kliensoldali programozás

Amikor kliensoldali programozást használsz, programkódot ágyazol a HTML-kódba, amelyet a szerver a HTML-kóddal együtt elküld a kliensböngészőnek. A böngészőnek tudnia kell észlelni és futtatni a beágyazott programkódot, vagy magában a böngészőben, vagy pedig különálló programként a böngészőn kívül. Ennek folyamatát az 1.2. ábra szemlélteti.

JavaScript

Manapság a legnépszerűbb kliensoldali programozási nyelv a JavaScript. A JavaScript egy szkriptnyelv, amelyet a weblapod normál HTML-kódjába kell ágyaznod. A kliensböngészőben fut, és ki tudja használni a böngésző olyan funkcióit, amelyek normális esetben nem érhetők el sima HTML-kódból. A JavaScript-kódot gyakran használják olyan előugró üzenetek és párbeszédablakok létrehozására, amelyeket az emberek a lap megtekintése közben használnak. Ezek olyan elemek, amelyeket HTML-kóddal nem lehet létrehozni.

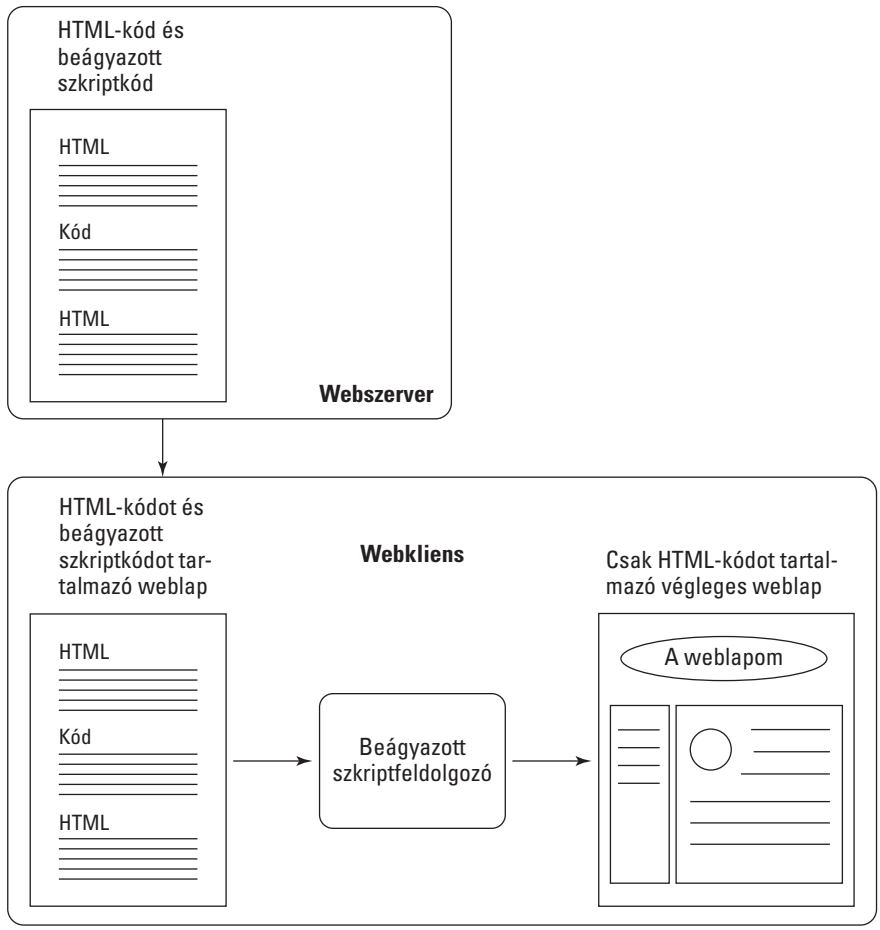
Ahogy az 1.2. ábrán láthatod, a kliensböngésző az egész weblapot letölti, a beágyazott JavaScript-kóddal együtt. A kliensböngésző észleli a beágyazott JavaScript-kódot, majd futtatja. Eközben feldolgozza a dokumentum HTML-címkéit, és alkalmazza a definiált CSS-stílusokat is, ha vannak. Szóval a böngészőnek jó sok mindent számon kell tartania!

A JavaScriptnek az a hátránya, hogy mivel a kliensböngészőben fut, ki vagy szolgáltatva annak, hogy az adott webböngésző hogyan értelmezi a kódot. A HTML nyelv szabványként indult, a JavaScript viszont egy kicsit más volt. Amikor a JavaScript még fiatal volt, a különböző böngészők más és más módszerekkel valósították meg a JavaScript különféle funkcióit. Nem ritkán találkozhattál olyan weblapokkal, amelyek egy adott típusú böngészőben teljesen jól működtek, egy másik típusú böngészőben viszont egyáltalán nem – mindez a JavaScript-feldolgozás következetlenségei miatt történt.

Végül befektették a munkát a JavaScript szabványosításába. A JavaScript nyelvet az Ecma nemzetközi szabványügyi szervezet karolta fel, és létrehozta az ECMAScript szabványt, amelyre ma a JavaScript épül. Ahogy az ECMAScript szabvány fejlődött, egyre több böngészőfejlesztő látta meg egy szabványos kliensoldali programozási nyelv használatának előnyeit, és ezeket beépítették a JavaScript-megvalósításukba. A könyv írásakor a szabvány nyolcadik verziója, az ECMAScript 2017 volt véglegesítve és megvalósítva a legtöbb böngészőben.

ELMÉLET

A JavaScript nevet azért választották, hogy kihasználják a webalkalmazások programozására használt Java programozási nyelv népszerűségét. Ugyanakkor egyáltalán nem hasonlít a Java programozási nyelvre, és semmilyen módon nem kapcsolódik hozzá.



1.2. ábra. Kliensoldali kód használata egy weblapon

jQuery

A JavaScript népszerű, de megvan az a hátránya, hogy kissé bonyolult lehet programozni benne. Mivel nagyon sokféle fejlesztő épített bele rengeteg különféle funkciót, a JavaScript-programok kódolása manapság gyorsan megterhelővé válhat.

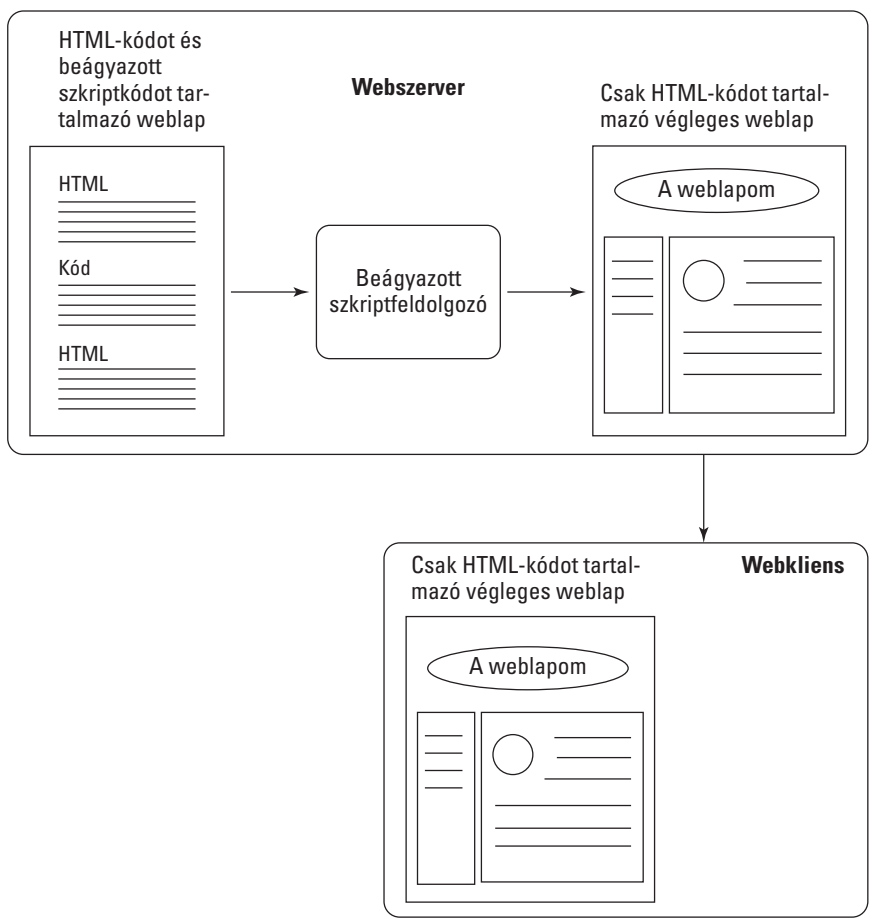
A probléma megoldására összefogott a fejlesztők egy csoportja, és könyvtárakat hoztak létre a JavaScripttel való kliensoldali programozás megkönnyítésére. Így született meg a jQuery.

A jQuery szoftver nem egy különálló programozási nyelv, hanem JavaScript-kódot tartalmazó könyvárak egy készlete. A könyvtárak olyan önálló JavaScript-függvények, amelyekre hivatkozhatsz a saját JavaScript-programjaidban, és olyan gyakori feladatokat láthatsz el velük, mint például egy hely megkeresése egy weblapon szöveg megjelenítéséhez, vagy egy HTML-űrlapmezőben megadott érték beolvasása.

Ahelyett, hogy rengeteg sornyi JavaScript-kódot kellene megírnod, egyszerűen hivatkozhatsz egy-két jQuery-függvényre, amelyek elvégzik helyetted a munkát. Ezzel rengeteg időt takaríthatsz meg, és kiváló erőforrást is biztosít olyan fejlett funkciók megvalósításához, amelyeket magad sosem tudtál volna leprogramozni JavaScriptben.

Szerveroldali programozás

A webprogramozás másik lehetősége a szerveroldali programozás. A szerveroldali programozási nyelvek úgy oldják meg a különböző kliensoldali kódértelmezők problémáját, hogy a szerveren futtatják a kódot. A szerveroldali programozásban a webszerver azelőtt értelmezi a beágyazott programkódot, hogy elküldené a weblapot a kliens böngészőjének. A szerver fog minden HTML-kódot, amelyet a programkód generál, és közvetlenül beszúrja a weblapba, mielőtt elküldené azt a kliensnek. A szerver végzi el a szkriptkód futtatásával járó összes feladatot, így garantált, hogy minden weblap megfelelően fog futni. Ezt a folyamatot az 1.3. ábra szemlélteti.



1.3. ábra. Egy weblap létrehozása szerveroldali programozással

A kliensoldali programozással ellentétben számos manapság használt, népszerű szerveroldali programozási nyelv létezik, és mindnek megvannak a maga előnyei és hátrányai. Ebben a szakaszban néhány népszerűbb programozási nyelvre vetünk egy pillantást.

CGI-szkriptkezelés

A szerveroldali programozás támogatására tett egyik első kísérlet az Apache webszerver általános átjáró felülete (Common Gateway Interface, CGI) volt. A CGI csatlakozási felületet biztosított egy a webszerver és a szerver mögöttes operációs rendszere (OS-e) között, amely gyakran Unix-alapú volt.

Ez lehetővé tette a programozók számára, hogy a Unix platformon gyakran használt szkriptkódot ágyazzanak be, és így dinamikusan generáljanak HTML-kódot. A Unix világában leggyakrabban használt, és így a CGI-vel is széles körben alkalmazott szkriptnyelvek közül kettő a Perl és a Python.

Bár a CGI-programozás népszerűvé vált a net korai időszakában, nem sok időbe telt, mire kihasználták a biztonsági réseit. Egy kezdő rendszergazda egyszerűen túl könnyen tudott rossz jogosultságokat alkalmazni a CGI-szkriptekre, ami lehetővé tette, hogy egy leleményes támadó emelt szintű hozzáférést szerezzen a szerverhez. A szerveroldali programkód feldolgozásának más módszereit kellett kifejleszteni.

Java

A szabályozott szerveroldali programozási nyelvek kialakítására a Javával tették az egyik korai próbálkozást. Bár a Java programozási nyelv bármilyen számítógépes platformon futtatható önálló alkalmazások létrehozására szolgáló nyelvként vált népszerűvé, futtatható webalkalmazásokban, szerveroldali programozási nyelvként is. Amikor így használjuk, a neve JSP (Java Server Pages, Java-szerverlapok).

A JSP nyelvhez egy beágyazott Java-fordítóra van szükséged a webszerveren. A webszerver észleli a Java-kódot a HTML-kódban, majd elküldi a kódot a Java-fordítónak feldolgozásra. Ezután a Java-program minden kimenetét elküldi a kliensböngészőnek a HTML-dokumentum részeként. A leginkább elterjedt JSP-platform a nyílt forráskódú Apache Tomcat szerver.

A Microsoft ASP.NET-család

A Microsoft első belépője a szerveroldali programozás világába – az ASP (Active Server Pages, Aktív szerverlapok) – a JSP-éhez hasonló megjelenésű és működésű volt. Az ASP-programok ASP-szkriptkódot ágyazott be a normál HTML-kódba, és egy ASP-szervert kellett beépíteni a normál Microsoft Internet Information Services (IIS) webszerverbe a kód feldolgozásához.

A Microsoft fejlesztői azonban úgy döntöttek, hogy nem szükséges egy különálló programozási nyelvet fenntartaniuk a szerveroldali programozáshoz, szóval egy technológiában egyesítették a szerveroldali programozási és a Windows asztali programozási környezetet. A .NET-programozási nyelvek családjának megjelenésével a Microsoft a régi ASP-környezet frissítéseként kiadta az ASP.NET-et a webes környezethez.

Az ASP.NET-tel a Microsoft .NET-programkódok bármely típusát beágyazhatod a HTML-dokumentumaidba, hogy dinamikus tartalmat hozz létre. A .NET-programozási nyelvek családjába tartozik a Visual Basic .NET, a C#, a J#, és még a Delphi.NET is. Ez lehetővé teszi, hogy dinamikus weblapok létrehozásához is ugyanazt a kódot használd, mint amelylyel Windows asztali alkalmazásokat készítesz. Gyakran a webalkalmazásaidban is használhatod ugyanazokat a Windows-funkciókat – például a gombokat, a csúszkákat és a görgetősávokat –, mint amelyeket a Windows-alkalmazásokban is láthatsz.

JavaScript

Igen, jól olvastad. Ugyanaz a JavaScript nyelv, amely oly népszerű a kliensoldali programozás világában, manapság már kezd szerveroldali programozási nyelvként is előretörni. A Node.js könyvtár csatlakozási felületet biztosít a HTML-weblapokban szereplő JavaScript-kód szerveren való feldolgozásához.

A Node.js használatának az az előnye, hogy csak egy nyelvet kell megtanulnod mind a kliens-, mind a szerveroldali programozáshoz. Bár még viszonylag új játékosnak számít, a Node.js nyelv egyre népszerűbb.

PHP

Először csak a CGI-szkriptek egyszerű finomhangolási feladatának indult, aztán végül egy új szerveroldali programozási nyelv lett belőle, amely egy csapásra meghódította a világot. A PHP (Personal Home Page, Személyes honlap) programozási nyelvet Rasmus Lerdorf írta, hogy tökéletesítse a CGI-szkriptjei működését. Némi bátorítással és segítséggel a PHP önálló programozási nyelvvé nőtte ki magát, és az új neve PHP: Hypertext Preprocessor (PHP: hiperszöveg-feldolgozó) lett (igen, a nevében is szerepel a mozaikszó, szóval ez egy úgynevezett *rekurzív mozaikszó*).

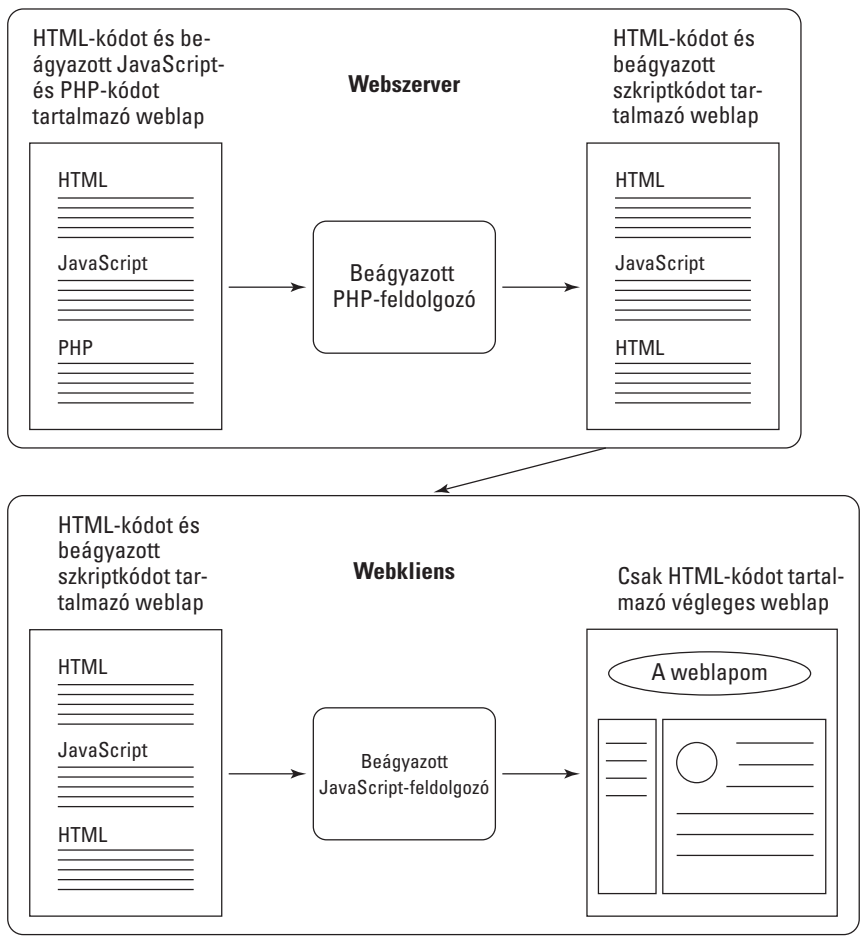
A PHP nyelv fejlesztői őszintén beismerik, hogy számos funkciót más népszerű nyelvekből emeltek át, például a Perlből, a Pythonból, a C-ből, és még a Unix shell szkriptekből is. A PHP-t viszont kifejezetten szerveroldali programozásra fejlesztették ki, és számos olyan beépített funkcióval rendelkezik, amelyek más szkriptnyelvekben nem állnak rendelkezésre. Nem kell fura beállításokkal és funkciókkal megküzdened ahhoz, hogy működésre bírd a PHP-t webes környezetben. Olyan fejlett funkciók teljes választéka fejlődött ki belőle, amelyek mindenre kiterjednek az adatbázisok elérésétől a grafikai elemek weblapodon való megrajzolásáig.

Mivel a PHP fejlesztői elkötelezettek voltak amellett, hogy egy első osztályú szerveroldali programozási nyelvet hozzanak létre, és mivel ingyenes, nyílt forráskódú szoftver, a PHP pillanatok alatt az internetes világ kedvencévé vált. Számos webhelyszolgáltató cég alapvető szolgáltatási csomagjának része a PHP. Ha már rendelkezel tárhellyel egy webhelyszolgáltató szerverén, lehet, hogy már most is van PHP-hozzáférésed!

Kombináljuk a kliensoldali és a szerveroldali programozást

A kliensoldali és a szerveroldali programozásnak is megvannak a maga előnyei és hátrányai. Ahelyett, hogy egy módszert próbálnál választani a dinamikus weblapok létrehozásához, egyszerre mindkettőt is használhatod!

Ugyanabba a weblapba könnyedén beágyazhatsz egyszerre kliensoldali és a szerveren futtatott szerveroldali programkódot is, ahogy azt az 1.4. ábrán is láthatod.



1.4. ábra. A kliensoldali és a szerveroldali programozás kombinálása

Ebben a részben azt vesszük át, hogy hogyan végezheted el a lista első három pontját a MySQL három különböző felületén, amelyekkel a korábbi részében megismerkedtünk. Az adatok lekérdezése elég összetett téma, szóval azzal majd egy külön szakaszban foglalkozunk. Lássunk is hozzá, és nézzük meg, hogyan kezelheted a tábláidban található adatokat.

A MySQL parancssori felülete

A MySQL CLI-jében szabványos SQL-utasításokkal kommunikálhatsz a MySQL-szerverrel. Mindössze három alapvető SQL-utasítást kell ismerned:

- `INSERT` (beszúrás): Új adatrekordok beszúrása egy táblába
- `UPDATE` (frissítés): A meglévő adatrekordok módosítása egy táblában
- `DELETE` (törlés): A meglévő adatrekordok eltávolítása egy táblából

Vegyük sorra:

Szúrjunk be új adatokat

Az `INSERT` SQL-utasítással szúrhatsz be egy vagy több adatrekordot az adatbázis egy táblájába. Az adatrekordokban minden egyes adatmezőhöz az adatértékek egyetlen példánya található.


TIPP

Egyes MySQL-dokumentációkban gyakran találkozhatsz majd azzal, hogy az egyes adatmezőkre az *oszlop (column)* kifejezéssel, a teljes adatrekordokra pedig az *értéksor (tuple)* kifejezéssel hivatkoznak. Ebben a könyvben az általánosabb *adatmező* és *adatrekord* kifejezést használjuk.

Az `INSERT` utasítás alapvető formátuma a következő:

```
INSERT INTO tábla [(mezőlista)] VALUES (értéklista)
```

A *`mezőlista`* paramétert nem kötelező megadni. Az `INSERT` utasítás alapértelmezés szerint megpróbálja a tábla minden egyes adatmezőjébe betölteni az *`értéklista`* vesszőkkel tagolt értékeit, abban a sorrendben, amelyben az adatmezők a tábla definíciójában szerepelnek. Ennek a minikönyvnek a 3. fejezetében olvashatsz arról, hogy hogyan listázhatod ki a tábla adatmezőit a `SHOW CREATE TABLE` utasítással. Egy másik módszerként a `DESCRIBE` (leírás) SQL-utasítást is használhatod:

```
MariaDB [dbteszt1]> DESCRIBE alkalmazottak;
+--------------------+-------------+------+-----+---------+-------+
| Field              | Type        | Null | Key | Default | Extra |
+--------------------+-------------+------+-----+---------+-------+
| alkalmazottazon    | int(11)     | NO   | PRI | NULL    |       |
| vezetéknév         | varchar(50) | NO   |     | NULL    |       |
| keresztnév         | varchar(50) | NO   |     | NULL    |       |
| részlegkód         | varchar(5)  | NO   |     | NULL    |       |
| kezdésdátuma       | date        | YES  |     | NULL    |       |
| fizetés            | date        | YES  |     | NULL    |       |
| születésidátum     | date        | YES  |     | NULL    |       |
+--------------------+-------------+------+-----+---------+-------+
7 rows in set (0.01 sec)

MariaDB [dbteszt1]>
```

Nem jeleníti meg, hogy pontosan milyen SQL-utasítással lett létrehozva a tábla, de megjeleníti a táblában található adatmezők rövid összegzését. Csak erre van szükséged ahhoz, hogy az `INSERT` utasításhoz tudd, milyen sorrendben szerepelnek az adatmezők a táblában.

A következő lépésekkel szúrhatsz be egy adatrekordot az `alkalmazottak` táblába, amelyet még ennek a minikönyvnek a 3. fejezetében hoztunk létre. (Ha kihagytad ezt a részt, vagy még nem olvastad el, akkor ehhez csak lapozz vissza oda és futtasd a `CREATE` utasításokat. Addig megvárlak.)

1. **Győződj meg róla, hogy a MySQL-szervered el van indítva, majd nyisd meg a MySQL CLI-programját.**
2. **Jelentkezz be a `root` felhasználói fiókkal.**
3. **Add meg a `dbteszt1` adatbázist a 3. fejezetből alapértelmezett adatbázisként a `USE` paranccsal:**

   ```
   MariaDB [(none)]> USE dbteszt1;
   Database changed
   MariaDB [dbteszt1]>
   ```

4. **Szúrj be egy új adatrekordot a következő INSERT utasítás beírásával:**

   ```
   MariaDB [dbteszt1]> INSERT INTO alkalmazottak VALUES
       -> (123, 'Nagy', 'Antal', 5, '2020-01-01',
   250000, '2000-05-01');
   Query OK, 1 row affected (0.12 sec)

   MariaDB [dbteszt1]>
   ```

TIPP

Az INSERT utasításban a szöveges és a dátumértékeket idézőjelek között kell megadnod, hogy jelezd a szöveges értékek elejét és végét. A számértékekhez nem kell idézőjeleket használnod. Figyeld meg, hogy az INSERT utasítás itt két részre van osztva. Ez nem kötelező, de hasznos lehet, ha nem szeretnéd, hogy túl hosszú legyen az INSERT utasítás sora.

Az INSERT utasítás egy állapotüzenetet ad vissza, amely jelzi, hogy hány adatrekord lett sikeresen beszúrva a táblába. (Ha kell, az adatértékek egynél több csoportját is megadhatod az *értéklista* részeként, mindegyiket zárójelek közé téve.)

5. **A SELECT utasítással ellenőrizd le, hogy az adatok beszúrása megfelelő-e:**

```
MariaDB [dbteszt1]> SELECT * FROM alkalmazottak;
+---------+-----------+-----------+------+---------
-----+---------+----------------+
| alkazon | vezetéknév | keresztnév | rkód |
kezdésdátuma | fizetés | születésidátum |
+---------+-----------+-----------+------+---------
-----+---------+----------------+
|     123 | Nagy       | Antal      | 5    | 2020-01-01
   |  250000 | 2000-05-01     |
+---------+-----------+-----------+------+---------
-----+---------+----------------+
1 row in set (0.00 sec)

MariaDB [dbteszt1]>
```

A SELECT utasítás a tábla adatmezőit (az alkalmazott-azonosító és a részlegkód adatmező nevét kicsit megvágtuk ebben a kimenetben, hogy ne legyen szélesebb, mint a könyv oldala) , és aztán a táblában található adatrekordokat jeleníti meg.

Ha az adatrekordban nem szeretnél minden adatmezőhöz értéket rendelni, akkor meg kell adnod a *mezőlista* paramétert. Ebben azokat a mezőket (és azok sorrendjét) adhatod meg, amelyekben el szeretnéd helyezni az adatértékeket:

```
MariaDB [dbteszt1]> INSERT INTO alkalmazottak
(alkalmazottazonosító, vezetéknév, keresztnév)
    -> VALUES (124, 'Nagy', 'Barbara');
Query OK, 1 row affected (0.10 sec)

MariaDB [dbteszt1]>
```

VIGYÁZAT

Légy óvatos, amikor új adatrekord beszúrásakor kihagysz adatmezőket. Ha egy NOT NULL adatmegszorítású adatmezőhöz nem rendelsz adatértéket, lehet, hogy a szerver elutasítja az INSERT utasítást. Azért mondom azt, hogy „lehet”, mert ez a MySQL-szerver konfigurációjától függ. Hogy a MySQL a régebbi verzióival való visszamenőlegesen kompatibilis maradjon, a MySQL alapértelmezés szerint nem alkalmaz néhány adatmegszorítást, például a NOT NULL megszorítást. Ennek alkalmazásához módosítanod kell az sql_mode beállítást, vagy a MySQL-szerver konfigurációjában, vagy úgy, hogy beállítod a MySQL-kapcsolati munkamenetben. Az sql_mode beállítás STRICT_ALL_TABLES (Szigorú_minden_tábla) értéke arra utasítja a MySQL-t, hogy minden táblára alkalmazzon minden adatmegszorítást. Ha így teszel, akkor hibaüzenetet fogsz kapni, ha nem adsz meg értéket bármelyik olyan adatmezőhöz, amelynek NOT NULL megszorítása van:

```
MariaDB [dbteszt1]> set sql_mode=STRICT_ALL_TABLES;
Query OK, 0 rows affected (0.00 sec)

MariaDB [dbteszt1]> INSERT INTO alkalmazottak
(alkalmazottazonosító, vezetéknév) VALUES
    -> (126, 'Kati');
ERROR 1364 (HY000): Field 'keresztnév' doesn't have a default value
MariaDB [dbteszt1]>
```

Módosítsunk meglévő adatokat

Ha már a táblában lévő adatokat kell módosítanod, akkor sem kell aggódnod – nincs minden veszve. A táblában meglévő bármelyik adatrekordot módosíthatod, feltéve hogy a MySQL-felhasználói fiókodhoz rendelt jogosultságok között szerepel az UPDATE jogosultság.

Az UPDATE SQL-utasítással frissíthetsz egy vagy több táblában található adatrekordot. Az UPDATE utasítás egy újabb olyan SQL-utasítás, amelynek bár az elve egyszerű, de könnyen bonyolulttá válhat. Az UPDATE utasítás alapvető formátuma a következő:

```
UPDATE tábla SET adatmező = érték [WHERE feltétel]
```

Az utasítás alapvető formátumával meghatározzuk a *tábla* módosítandó *adatmezője*-t, amely a megadott *érték* paraméterrel módosítja ennek az adatmezőnek az adatértékét. A WHERE (ahol) záradék azt a feltételt adja meg, amelynek az adatrekordoknak meg kell felelniük ahhoz, hogy azokra a rendszer alkalmazza a módosítást. Figyeld meg, hogy ezt nem kötelező megadni, és ez rengeteg problémát okozhat.

Gyakran a következő forgatókönyv valósul meg: Tegyük fel, hogy vissza kell térned az Alkalmazottak táblába, hogy módosítsd Barbara NULL értékű kezdésdátuma adatmezőjét, amelyet nem adtál meg a rekord lét-

rehozásakor. Ha csak az `UPDATE` utasítás alapvető formátumát használod, meglepetésben lesz részed:

```
MariaDB [dbteszt1]> UPDATE alkalmazottak SET kezdésdátuma =
'2020-01-02';
Query OK, 2 row affected (0.10 sec)
Rows matched: 2  Changed: 2  Warnings: 0

MariaDB [dbteszt1]>
```

Először a MySQL-szerver által visszaadott kimenetből már sejteni fogod, hogy valami rossz történt. A `Rows matched` (Illesztett sorok) és a `Changed` (Módosítva) mező azt jelzi, hogy két adatrekord lett frissítve – te viszont csak egy adatrekordot szerettél volna módosítani. Ha futtatsz egy `SELECT` utasítást, azzal igazolhatod a hibádat:

```
MariaDB [dbteszt1]> SELECT * FROM alkalmazottak;
+---------+------------+------------+------+--------------+-
--------+----------------+
| alkazon | vezetéknév | keresztnév | rkód | kezdésdátuma |
fizetés | születésidátum |
+---------+------------+------------+------+--------------+-
--------+----------------+
|     123 | Nagy       | Antal      | 5    | 2020-01-02   |
250000 | 2000-05-01     |
|     124 | Nagy       | Barbara    | NULL | 2020-01-02   |
NULL | NULL           |
+---------+------------+------------+------+--------------+-
--------+----------------+
2 rows in set (0.00 sec)

MariaDB [dbteszt1]>
```

A kiinduló `UPDATE` utasítás a tábla összes adatrekordjában módosította a `kezdésdátum` adatmező értékét! Ez egy nagyon gyakori hiba, amelyet még a tapasztalt adatbázis-adminisztrátorok és -programozók is elkövetnek. A MySQL alapértelmezés szerint a tábla összes adatrekordjára alkalmazza a frissítést.

Ennek a problémának a megoldásához meg kell adnod a `WHERE` záradékot, hogy pontosan meghatározd, hogy mely rekord(ok)ra szeretnéd, ha vonatkozna a módosítás:

```
UPDATE alkalmazottak SET kezdésdátuma = '2020-01-01' WHERE
alkalmazottazonosító = 123;
```

Az egy jó megoldás, ha megszokod, hogy mindig megadod a `WHERE` záradékot az `UPDATE` utasításaidban, még akkor is, ha tényleg azt szeret-

TIPP

néd, hogy a frissítés az összes adatrekordra vonatkozzon. Így tudod, hogy a frissítés mindig a megfelelő adatrekordokra lesz alkalmazva, és elkerülheted a költséges hibákat.

Töröljünk meglévő adatokat

A `DELETE` utasítással adatokat távolíthatsz el egy táblából úgy, hogy közben sértetlenül hagyod magát a táblát (a `DROP` utasítással ellentétben, amely a táblát és az adatokat is eltávolítja). A `DELETE` utasítás formátuma a következő:

```
DELETE FROM tábla [WHERE feltétel]
```

Ez az utasítás is az `UPDATE` utasításhoz hasonlóan működik. A rendszer minden olyan adatrekordot töröl, amelyre igaz a *`feltétel`*, amely a `WHERE` záradékban meg lett adva. És pontosan úgy, mint az `UPDATE` utasításnál, ha elhagyod a `WHERE` záradékot, a `DELETE` függvény a tábla összes adatára érvényes lesz. Mielőtt használnád, győződj meg róla, hogy tényleg ezt akarod!

Lássunk néhány példát a `DELETE` utasítás használatára:

```
MariaDB [dbteszt1]> DELETE FROM alkalmazottak WHERE
alkalmazottazonosító = 124;
Query OK, 1 row affected (0.08 sec)

MariaDB [dbteszt1]> DELETE FROM alkalmazottak WHERE
alkalmazottazonosító = 124;
Query OK, 0 rows affected (0.00 sec)

MariaDB [dbteszt1]>
```

A második példában olyan adatrekordot próbáltunk meg törölni, amelyet már korábban töröltünk. Figyeld meg, hogy ha a `DELETE` utasítás nem talál olyan adatrekordot, amelyet törölhet, nem ad hibaüzenetet; ehelyett csak azt jelzi a visszaadott állapotban, hogy a törölt adatrekordok száma nulla.

A MySQL Workbench eszköz

A grafikus felületnek köszönhetően könnyűszerrel kezelhetsz táblákat a MySQL Workbench-ben. Semmilyen SQL-utasítást nem kell megjegyezned – csak egy űrlapot kell kitöltened, és alkalmaznod kell azt az adatbázisra. Mintha csak egy pizzát rendelnél!

A következő lépésekkel kísérletezhetsz a Workbench adatkezelési funkcióival: