IV. RÉSZ

Összefoglaló példa

23. ÓRA

Teljes példa (első rész)

Ha figyelmesen követtük az egyes órák anyagát, jó alapokkal rendelkezünk a PHP programozáshoz. Ebben és a következő órában egy teljes, működő programot készítünk, amely az előző órákban tárgyalt eljárásokból építkezik.

Az órában a következőket tanuljuk meg:

- Hogyan készítsünk tervet?
- Hogyan használjuk az `include()` nyelvi szerkezetet, hogy függvénykönyvtárakat és újrahasznosítható navigációs elemeket készítsünk?
- Hogyan tartsuk nyilván az állapotokat `GET` típusú lekérdezésekkel, adatbázisokkal és munkamenet-függvényekkel?
- Hogyan válasszuk el a HTML és a PHP kódot, hogy a programozásban járatlan grafikus is dolgozhasson az oldalon?
- Hogyan védjük meg szolgáltatásunk oldalait felhasználó-azonosítással?

A feladat rövid leírása

Tegyük fel, hogy egy közösségi webhely tulajdonosai felkértek bennünket, hogy készítsünk el egy kis interaktív eseménynaptárt az általuk kiszolgált kisváros számára. Klubok és együttesek jegyeztethetik be magukat, hogy reklámozzák rendezvényeiket. A webhely felhasználói ezután különböző formákban kiírathatják majd az adatbázist, hogy lássák, milyen rendezvények lesznek a városban. A felhasználók képesek lesznek a lista szűkítésére a klubok típusa vagy akár a szervezett rendezvény helye szerint is.

Az oldalak felépítése

Mielőtt akár egy sor kódot is írnánk, el kell döntenünk, hogyan fog működni programunk. Milyen módon fogják elérni a felhasználók a rendszer különböző elemeit? Milyen oldalakra van szükségünk?

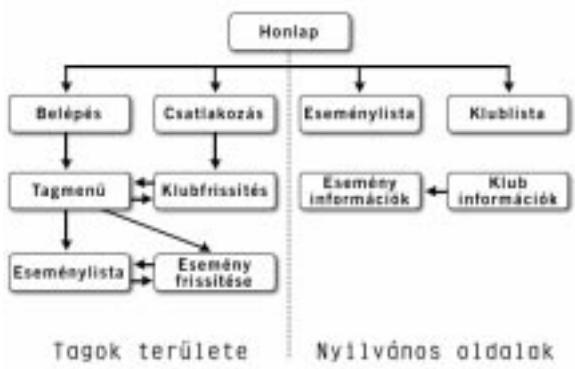
A program természetesen két részre tagolódik. Az első a tagok számára kialakított terület, amely a klubok információinak kezelésére, új események hozzáadására szolgál majd. A második a felhasználók területe, ahol az adatbázison lekérdezéseket kell majd végrehajtani.

A továbbiakban tagoknak nevezzük azokat a személyeket, akik klubjukat bejegyezve felügyeleti feladatokat látnak el, és felhasználóknak azokat az érdeklődőket, akik a listákat böngészve barangolnak a klubok és események között.

A 23.1. ábra az alkalmazás felépítését mutatja.

23.1. ábra

Az alkalmazás felépítése



Az új tagok a `csatlakozas.php` oldalon csatlakozhatnak a rendszerhez (itt jegyeztethetik be magukat a tagok közé), egy név–jelszó pár megadásával. Ha a választott név még nem foglalt, a leendő tag a `klubfrissites.php` oldalra kerül, ahol egy űrlapon meg kell adnia a klubról a szükséges információkat. Amíg ki nem tölti ezt az űrlapot, nem vihet be új rendezvényeket a rendszerbe. Ha a tag a hozzá tartozó klub adatait sikeresen bevitte, a tagok menüjéhez kerül (`tagmenu.php`), ahonnan a tagok részére készített valamennyi oldal elérhető.

A már bejegyzett tagok a belépő oldalról indulnak (`belepes.php`). Ha a megadott név és jelszó helyesnek bizonyult, egyenesen a `tagmenu.php` oldalra kerülnek.

A menü oldalról indulva a tagok új rendezvényeket adhatnak a rendszerhez (`esemenyfrissites.php`) és megtekinthetik az adatbázisban levő rendezvényeik listáját (`esemenylista.php`). A klub adatait a `klubfrissites.php` oldalon bármikor módosíthatják.

Minden felhasználó képes lesz az események hónapok, napok, típusok és területek alapján történő rendezett kiíratására, egyetlen PHP oldal segítségével (`esemenyekinfo.php`). Lehetőség lesz a klubok felsoroltatására is, terület vagy típus alapján (`klubokinfo.php`). Végül a felhasználók egy klubra vagy eseményre kattintva bővebb információkat tudhatnak meg (`esemenyinfo.php`, `klubinfo.php`).

Az adatbázis kialakítása

Az alkalmazás adatainak tárolásához létre kell hoznunk a `szervezo` adatbázist és ebben négy táblát: `klubok`, `esemenyek`, `teruletek`, `tipusok`. Nyilvánvaló, hogy a klubok és rendezvények adatait külön táblákban kell tárolnunk, hiszen egy klubhoz több rendezvény is tartozhat. A területek és típusok számára kialakított táblák jelentősen megkönnyítik a listázáshoz és adatbevitelhez szükséges lenyíló menük kialakítását. A táblákat SQL parancsokkal hozzuk létre, „kézi úton”. A `klubok` tábla a következőképpen hozható létre:

```
CREATE TABLE klubok (
      azonosito INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
      klubnev VARCHAR(50),
      tipus CHAR(3),
      terulet CHAR(3),
      email VARCHAR(50),
      ismerteto BLOB,
      nev VARCHAR(8),
      jelszo VARCHAR(8)
);
```

A tagok a `klubnev`, `email`, `ismerteto`, `nev` és `jelszo` mezők adatait fogják megadni. A `tipus` és `terulet` mezők értelemszerűen a `tipusok` és `teruletek` megfelelő soraira vonatkozó azonosítókat tartalmazzák majd.

Az `esemenyek` tábla az eseményekre vonatkozó valamennyi információt tartalmazza:

```
CREATE TABLE esemenyek (
      eazonosito INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
      etipus CHAR(3),
      eterulet CHAR(3),
      edatum INT,
      enev VARCHAR(100),
      ehelyszin VARCHAR(100),
      ecim VARCHAR(255),
      eirsz CHAR(4),
      eismerteto BLOB,
      eklub INT NOT NULL
);
```

Vegyük észre, hogy ez a tábla is tartalmaz `etipus` és `eterulet` mezőket, hiszen előfordulhat, hogy egy klub a város északi részén található, de valamelyik rendezvényét a déli városrészben tartja. Egy társadalmi csoport tarthat oktatási szemináriumot, de politikai találkozót is. Az `eklub` mező annak a klubnak az azonosítószámát tartalmazza, amelyik az adott esemény szervezője. A kapcsolat arra használható fel, hogy egy klub összes rendezvényét felsoroljuk, illetve hogy elérjük az eseményhez köthető klub adatait.

A `teruletek` és `tipusok` táblák igen egyszerűek:

```
CREATE TABLE teruletek ( azonosito CHAR(3),
 ➥ terulet VARCHAR(30));
CREATE TABLE tipusok ( azonosito CHAR(3),
 ➥ tipus VARCHAR(30));
```

A tagok számára nem adunk lehetőséget ezen táblák módosítására, inkább előre meghatározott csoportokat alakítunk ki, melyeket `INSERT` SQL parancsokkal adunk a táblákhoz. A tagok ezekből a listákból választhatnak majd.

```
INSERT INTO tipusok (azonosito, tipus) VALUES
 ➥   ("CSA", "Családi");
```

A 23.1. és 23.2. táblázatban az említett táblákba illesztendő adatok láthatók.

23.1. táblázat A teruletek táblához adott adatok

azonosito	terulet
ESZ	Észak
DEL	Dél
KEL	Kelet
NYU	Nyugat

23.2. táblázat A tipusok táblához adott adatok

azonosito	tipus
ZEN	Zenei
CSA	Családi
TRS	Társadalmi
KZS	Közösségi

Tervezési döntésünk

Az előző bekezdésekben már láttuk, milyen szerkezetű alkalmazást készítünk. Úgy döntöttünk, hogy a különböző szolgáltatásokhoz különböző PHP programokat készítünk, ahelyett, hogy egyetlen hatalmas programot építenénk fel, amely a körülményeknek megfelelően más-más oldalakat szolgáltat. Ennek a döntésnek természetesen vannak előnyei és hátrányai is.

Ha egy ilyen dinamikus környezetet több oldalból építünk fel, a kódok ismétlésének hibájába eshetünk és a program növekedésével megnehezíthetjük a fejlesztők dolgát. Másrészt viszont a befejezett prototípust átadhatjuk grafikusainknak, akik majdnem úgy kezelhetik azt, mintha pusztán HTML kódot tartalmazna.

A tagoknak szánt oldalak

Most már itt az ideje, hogy megkezdjük a kód írását. Az óra további részében az alkalmazás tagoknak szánt oldalait készítjük el. Jó ötlet ezekkel kezdeni, mivel így könnyebben vihetünk be próbaadatokat a rendszerbe. Mielőtt azonban az adatbevitelt megkezdhetnénk, képesnek kell lennünk a tagok felvételére.

csatlakozas.php és adatbazis.inc

A `csatlakozas.php` tartalmazza azt az űrlapot, amelyen keresztül az új tagok egy név és jelszó segítségével bejegyeztethetik klubjukat a rendszerbe. Ahhoz, hogy felvehessük az adatokat és kiszűrhessük az ismétlődéseket, először meg kell nyitnunk egy kapcsolatot az adatbázis felé. Ez az ilyen alkalmazásoknak annyira jellemző eleme, hogy célszerű külön függvényt készíteni erre a célra, melyet jelen esetben egy külső állományban fogunk tárolni. Ezt a külső fájlt később minden oldalon az `include()` nyelvi elemmel illesztjük a kódba. Tulajdonképpen minden adatbázissal kapcsolatos függvényt ebben tárolunk majd, ezért az `adatbazis.inc` nevet kapja, szerepe és a beillesztés módja után. Ezzel a fájllal a további PHP oldalakat mindennemű SQL parancstól mentesítjük.

Az adatbázissal kapcsolatot tartó kódok külön fájlba helyezése könnyebbé teszi a program későbbi módosítását vagy más adatbázismotorra való átültetését. Előfordulhat, hogy később újra kell írnunk a függvényeket, de a hívó kódok módosítására nem lesz szükség.

Hozzuk létre a függvényt, amely végrehajtja a csatlakozást:

23.1. program Részlet az adatbazis.inc fájlból

```
 1: $kapcsolat;
 2: dbCsatlakozas();
 3: function dbCsatlakozas()
 4:      {
 5:      global $kapcsolat;
 6:      $kapcsolat = mysql_connect( "localhost",
                     "felhasznalo", "jelszo" );
 7:      if ( ! $kapcsolat )
 8:          die( "Nem lehet csatlakozni a MySQL-hez" );
 9:      mysql_select_db( "szervezo", $kapcsolat )
10:          or die ( "Nem lehet megnyitni az adatbázist:
                      ".mysql_error() );
11:      }
```

A `dbCsatlakozas()` függvény a `$kapcsolat` globális változót használja arra, hogy a `mysql_connect()` által visszaadott kapcsolatazonosítót tárolja. Mivel a változó globális, a többi adatbázisfüggvény is elérheti. Nem csupán a MySQL kiszolgálóhoz csatlakozunk ebben a függvényben, hanem megpróbáljuk kiválasztani a `szervezo` adatbázist is. Mivel ezen műveletek sikere döntő fontosságú a teljes alkalmazás működése szempontjából, a `mysql_connect()` vagy `mysql_select_db()` végrehajtásának kudarca esetén befejezzük a program futását.

A munkamenetek követése és az azonosítással kapcsolatos feladatokat ellátó függvények számára egy újabb külső állományt használunk. A munkamenet-függvényeket arra használjuk, hogy egy `$munkamenet` nevű asszociatív tömböt őrizzünk meg kérésről kérésre. A `kozosfv.inc` állomány `session_start()` függvénye arra szolgál, hogy megkezdjünk vagy folytassunk egy munkamenetet, a változónkat pedig a `session_register()` függvénnyel rendeljük ehhez:

23.2. program Részlet a kozosfv.inc fájlból

```
1: session_start();
2: session_register( "munkamenet" );
```

Emlékezzünk rá, hogy minden PHP kód, amit ezen külső fájlokba írunk, PHP blokk kezdő (`<?php`) és záró (`?>`) elemek között kell, hogy legyen. Értelmetlen bonyolításnak tűnhet, hogy a különböző szolgáltatásokat külső állományokba helyezzük, de ez rengeteg kódismétléstől kímél majd meg bennünket, amikor oldalainkat elkészítjük. Most már készen állunk a `csatlakozas.php` oldal megírására.

23.3. program csatlakozas.php

```
 1: <?php
 2:      include("adatbazis.inc");
 3:      include("kozosfv.inc");
 4:
 5:      $uzenet="";
 6:
 7:      if ( isset( $mitkelltenni ) &&
              $mitkelltenni=="csatlakozas")
 8:           {
 9:           if ( empty( $urlap["nev"] ) ||
10:                empty( $urlap["jelszo"] ) ||
11:                empty( $urlap["jelszo2"] ) )
12:              $uzenet .= "Ki kell töltenie minden
                             mezőt!<br>\n";
13:
14:         if ( $urlap["jelszo"] != $urlap["jelszo2"] )
15:              $uzenet .= "A jelszavak nem
                             egyeznek!<br>\n";
16:
```

23.3. program (folytatás)

```
17:         if ( strlen( $urlap["jelszo"] ) > 8 )
18:              $uzenet .= "A jelszó hossza legfeljebb
19:                          8 karakter lehet!<br>\n";
20:
21:         if ( strlen( $urlap["nev"] ) > 8 )
22:              $uzenet .= "A tagsági név hossza
                            legfeljebb
23:                          8 karakter lehet!<br>\n";
24:
25:         if ( sorLekeres( "klubok", "nev",
                 $urlap["nev"] ) )
26:              $uzenet .= "A megadott tagsági néven.\""
                            $urlap["nev"] ."\"
27:                          már van bejegyzett tagunk.
28:                          Kérjük, adjon meg más
                            nevet!<br>\n";
29:
30:         if ( $uzenet == "" ) // nem találtunk hibát
31:              {
32:              $azon = ujTag( $urlap["nev"],
                        $urlap["jelszo"] );
33:              munkamenetFeltoltes( $azon, $urlap["nev"],
                                      $urlap["jelszo"] );
34:
35:              header( "Location:
                          klubfrissites.php?".SID );
36:              exit;
37:              }
38:         }
39:     ?>
40:
41:     <html>
42:     <head>
43:     <title>Csatlakozás</title>
44:     </head>
45:
46:     <body>
47:     <?php
48:     include("kozosnav.inc");
49:     ?>
50:     <p>
51:     <h1>Csatlakozás</h1>
```

23.3. program (folytatás)

```
52:
53:     <?php
54:     if ( $uzenet != "" )
55:          {
56:          print "<b>$uzenet</b><p>";
57:          }
58:     ?>
59:
60:     <p>
61:     <form action="<?php print $PHP_SELF;?>">
62:     <input type="hidden" name="mitkelltenni"
                    value="csatlakozas">
63:     <input type="hidden" name="<?php print
                    session_name() ?>"
64:            value="<?php print session_id() ?>">
65:     Tagsági név: <br>
66:     <input type="text" name="urlap[nev]"
67:            value="<?php print $urlap["nev"] ?>"
                     maxlength=8>
68:     </p>
69:     <p>
70:     Jelszó: <br>
71:     <input type="password" name="urlap[jelszo]"
                   value="" maxlength=8>
72:     </p>
73:     <p>
74:     Jelszó megerősítés: <br>
75:     <input type="password" name="urlap[jelszo2]"
                   value="" maxlength=8>
76:     </p>
77:     <p>
78:     <input type="submit" value="Rendben">
79:     </p>
80:     </form>
81:
82:     </body>
83:     </html>
```

Először az `include()` használatával beillesztjük a külső állományokat, tehát azonnal rendelkezésünkre áll egy adatbáziskapcsolat és egy aktív munkamenet. Létrehozunk egy `$uzenet` nevű változót. Ezzel a kettős célú változóval még

számos más oldalon találkozni fogunk. Az `$uzenet` célja egyrészt, hogy tartalmazza a hibaüzenetet, amit később kiírhatunk a böngésző számára, másrészt használhatjuk feltételes kifejezésekben, hogy ellenőrizzük, volt-e hibaüzenet vagy sem. Ha a változó üres marad, feltételezhetjük, hogy minden ellenőrzés sikeresen végrehajtódott.

Ezután a `$mitkelltenni` változó létezését és tartalmát vizsgáljuk. Ez is olyan elem, amely többször felbukkan majd az alkalmazásban. Minden általunk készített űrlapon beállítunk egy `mitkelltenni` nevű rejtett elemet és egy, az adott szolgáltatásra vonatkozó értéket adunk neki. Ha PHP programunkban a `$mitkelltenni` változó létezik és a várt érték található benne, biztosak lehetünk abban, hogy az űrlapot kitöltötte valaki, így folytathatjuk a programot az űrlap adatainak ellenőrzésével. Ha a változó nem létezik, tudhatjuk, hogy a látogató egy hivatkozáson keresztül vagy a böngészőjében lévő könyvjelző révén érkezett az oldalra és nem töltötte ki az űrlapot.

Egyelőre ugorjunk a HTML rész tárgyalására. A `body` HTML elemen belül először egy újabb külső fájlt illesztünk be, amely a különböző oldalakra mutató hivatkozásokat tartalmazza. Célszerű már a kezdetektől beilleszteni a navigációs sávot, mivel ez megkönnyíti az alkalmazás ellenőrzését, ahogy folyamatosan fejlesztjük azt. A navigációs elemeket a `kozosnav.inc` nevű állományban helyezzük el. Egyelőre ez a fájl csak a látogatók számára is elérhető oldalakra mutató hivatkozásokat tartalmazza.

23.4. program Részlet a kozosnav.inc fájlból

```
1: <p>
2: <a href="klubokinfo.php?<?php print SID
    ?>">Klubinformációk</a> |
3: <a href="esemenyekinfo.php?<?php print SID
    ?>">Eseményinformációk</a> |
4: <a href="csatlakozas.php?<?php print SID
    ?>">Csatlakozás</a> |
5: <a href="belepes.php?<?php print SID ?>">Belépés</a> |
6: <a href="index.php?<?php print SID ?>">Honlap</a>
7: </p>
```

Azzal, hogy a navigációs elemeket külön állományba helyezzük, egyetlen mozdulattal módosíthatjuk a hivatkozásokat, illetve az elemek kinézetét a teljes alkalmazásra vonatkozóan.

Visszatérve a `csatlakozas.php` oldalra, miután kiírjuk az oldal címsorát, ellenőrizzük az `$uzenet` változót. Ha nem üres, kiírjuk a tartalmát a böngésző számára. Az összegyűjtött hibaüzenetek kiírásával jelezhetjük a tagnak, hogy a bevitt adatok nem dolgozhatók fel.

A HTML űrlap elemei között három látható mezőt hoztunk létre, az `urlap[nev]`, `urlap[jelszo]` és `urlap[jelszo2]` mezőket. Azért használjuk ezeket az elsőre esetleg furcsának látszó elnevezéseket, mert a PHP az így megadott nevekkel érkező értékeket egy asszociatív tömbbe rendezi, amit `$urlap` néven érhetünk majd el. Ez megóv bennünket attól, hogy a globális változók környezetét „beszennyezzük", azaz feleslegesen sok globális változót hozzunk létre. Programjainkban így minden munkamenet-érték a `$munkamenet` asszociatív tömbben, az űrlapértékek pedig az `$urlap` asszociatív tömbben érhetők el. Ez megvéd bennünket a változók ütközésétől. Célszerű a lehető legkevesebbre csökkenteni a globális változók számát egy ilyen méretű programban, hogy megelőzzük a kavarodást. A továbbiakban minden űrlapot tartalmazó oldalon az `$urlap` tömbbel fogunk találkozni.

Az űrlapban létrehozzuk a már korábban említett `mitkelltenni` rejtett elemet is, valamint egy másikat, amely a munkamenet nevét és értékét tartalmazza majd. Az alkalmazás elkészítése során mindig oldalról-oldalra fogjuk adni a munkamenet-azonosítót, mivel nem kívánjuk elveszteni azokat a látogatókat, akik nem tudnak vagy nem akarnak sütiket fogadni.

Ha munkameneteket használó programokat ellenőrzünk, célszerű kikapcsolni a böngészőben a sütik elfogadását. Így pontosan tudni fogjuk, hogy a munkamenet azonosítóját sikeresen át tudjuk-e adni oldalról oldalra ezen szolgáltatás nélkül is vagy sem.

Most, hogy megnéztük a HTML űrlapot, rátérhetünk arra a kódra, amely a bemenő adatok ellenőrzését végzi. Először meg kell bizonyosodnunk róla, hogy a leendő tag valóban kitöltötte-e az összes mezőt és hogy egyik mező hossza sem haladja meg a nyolc karaktert. Ezután meghívjuk az új `sorLekeres()` függvényt. Ez azon függvények egyike, amelyek az `adatbazis.inc` fájlban találhatók. Első paramétere egy táblanév, a második egy mezőnév, a harmadik egy érték. Ezen adatok alapján a függvény egy illeszkedő sort keres az adatbázisban, visszaadva azt egy tömbben.

23.5. program Részlet az adatbazis.inc fájlból

```
1: function sorLekeres( $tabla, $mezonev, $mezoertek )
2:      {
3:      global $kapcsolat;
4:      $eredmeny = mysql_query( "SELECT * FROM $tabla
                   WHERE $mezonev='$mezoertek'",
                   $kapcsolat );
5:      if ( ! $eredmeny )
6:          die ( "sorLekeres hiba: ".mysql_error() );
7:      return mysql_fetch_array( $eredmeny );
8:  }
```

A függvénynek a `klubok` táblanevet, a `nev` mezőnevet és a látogatók által bevitt `$urlap["nev"]` mezőértéket adjuk át. Ha a `mysql_fetch_array()` nem üres tömböt ad vissza, tudhatjuk, hogy egy ilyen belépési névvel rendelkező tag már van a rendszerben, ezért hibaüzenetet kell adnunk.

Ha az ellenőrzések után az `$uzenet` változó még mindig üres, létrehozhatjuk az új tagot a bevitt adatokkal. Ez két lépésből tevődik össze. Először is frissíteni kell az adatbázist. Ehhez egy új függvény tartozik az `adatbazis.inc` állományból, `ujTag()` néven:

23.6. program Részlet az adatbazis.inc fájlból

```
1: function ujTag( $nev, $jelszo )
2:      {
3:      global $kapcsolat;
4:      $eredmeny = mysql_query( "INSERT INTO klubok
                    (nev, jelszo)
5:                  VALUES('$nev', '$jelszo')",
                    $kapcsolat);
6:      return mysql_insert_id( $kapcsolat );
7:      }
```

A függvény egy név és egy jelszó paramétert vár, majd ezeket új sorként felveszi a klubok táblájába. A függvény a `mysql_insert_id()`-t használja, hogy megkapja a felvett új tag azonosítóját.

Mivel már rendelkezünk az új tag azonosítójával, meghívhatjuk a felvételéhez szükséges újabb függvényt, ami a `kozosfv.inc` fájlban található.
A `munkamenetFeltoltes()` függvény egy azonosítót, egy belépési nevet és egy jelszót vár paraméterül és hozzáadja azokat a `$munkamenet` tömbhöz.
Így ezek az értékek már elérhetőek lesznek minden munkamenetet kezelő oldalunk számára. Ezzel képesek leszünk a tag azonosítására a további oldalakon is.

23.7. program Részlet a kozosfv.inc fájlból

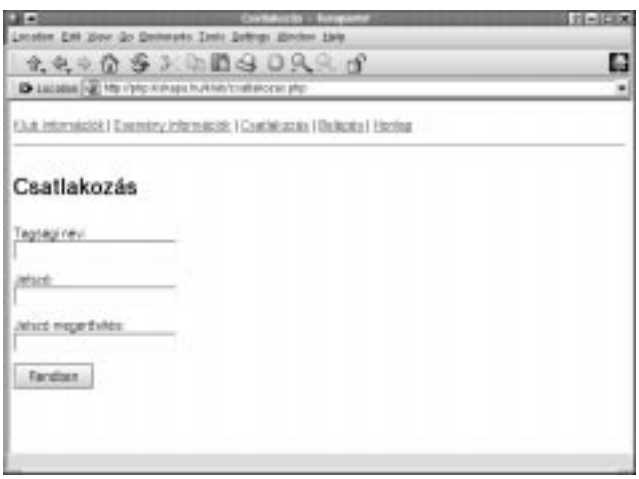
```
1: function munkamenetFeltoltes( $azonosito, $nev,
                                 $jelszo )
2:      {
3:      global $munkamenet;
4:      $munkamenet["azonosito"] = $azonosito;
5:      $munkamenet["nev"] = $nev;
6:      $munkamenet["jelszo"] = $jelszo;
7:      $munkamenet["belepett"] = true;
8:      }
```

Az azonosító, név és jelszó megadása mellett a belépési állapotot jelző értéket (`belepett`) is beállítjuk a munkameneteket kezelő oldalaink számára.
Végül, miután a `csatlakozas.php` oldal frissítette az adatbázist és a munkamenethez rendelt értékeket, útjára engedhetjük új tagunkat. Meghívjuk a `header()` függvényt, átirányítva a böngészőt a `klubfrissites.php` oldalra, ahol a tagnak be kell majd állítania az újonnan bejegyzett klub részletes adatait.

A `csatlakozas.php` kimenetét a 23.2. ábrán láthatjuk.

23.2. ábra

A csatlakozas.php kimenete



klubfrissites.php

Ez az oldal kettős célt szolgál. Egyrészt az új tagoknak itt kell megadniuk a klub részletes adatait, a bejegyzett tagok pedig itt módosíthatják ezeket.

23.8. program klubfrissites.php

```
 1: <?php
 2:      include("adatbazis.inc");
 3:      include("kozosfv.inc");
 4:
 5:      $klub_sor = azonositas();
 6:
 7:      $uzenet = "";
 8:
 9:      if ( isset( $mitkelltenni ) &&
              $mitkelltenni=="frissites" )
10:         {
11:         if ( empty( $urlap["klubnev"] ) )
12:              $uzenet .="A klubnév mező nincs
                           kitöltve!<br>\n";
13:
14:         if ( ! sorLekeres( “teruletek”, “azonosito”,
                $urlap["terulet"] ) )
15:              $uzenet .= "KRITIKUS HIBA: A terület kód-
                            ja nem található!<br>";
16:
17:         if ( ! sorLekeres( "tipusok", "azonosito",
                 $urlap["tipus"] ) )
18:              $uzenet .= "KRITIKUS HIBA: A típus kódja
                            nem található!<br>";
19:
20:         if ( $uzenet == "" )
21:              {
22:              klubFrissites( $munkamenet["azonosito"],
                                $urlap["klubnev"],
                                $urlap["terulet"],
23:                             $urlap["tipus"],
                                $urlap["email"],
                                $urlap["ismerteto"] );
24:              header("Location: tagmenu.php?".SID);
25:              exit;
26:              }
27:         }
```

23.8. program (folytatás)

```
28:     else
29:         {
30:         $urlap = $klub_sor;
31:         }
32:     ?>
33:
34:     <html>
35:     <head>
36:     <title>Klubinformációk frissítése</title>
37:     </head>
38:
39:     <body>
40:     <?php
41:     include("kozosnav.inc");
42:     ?>
43:     <h1>Klubinformációk frissítése</h1>
44:     <?php
45:     if ( $uzenet != "" )
46:         {
47:         print "<b>$uzenet</b><p>";
48:         }
49:     ?>
50:
51:     <form action="<?php print $PHP_SELF;?>">
52:     <input type="hidden" name="mitkelltenni"
                  value="frissites">
53:     <input type="hidden" name="<?php print
                  session_name() ?>"
54:             value="<?php print session_id() ?>">
55:
56:     <p>
57:     Klub neve: <br>
58:     <input type="text" name="urlap[klubnev]"
59:           value="<?php print
                    stripslashes($urlap["klubnev"]) ?>">
60:     </p>
61:     <p>
62:     Klub helye: <br>
63:     <select name="urlap[terulet]">
64:     <?php optionLista( "teruletek", $urlap["terulet"] ) ?>
```

23.8. program (folytatás)

```
    </select>
    </p>
    <p>
    Klub típusa: <br>
    <select name="urlap[tipus]">
    <?php optionLista( "tipusok", $urlap["tipus"])?>
    </select>
    </p>
    <p>
    Kapcsolattartó e-mail címe: <br>
    <input type="text" name="urlap[email]"
           value="<?php print
                  stripslashes($urlap["email"]) ?>">
    </p>
    <p>
    Klub ismertetője: <br>
    <textarea name="urlap[ismerteto]" rows=5 cols=30
                wrap="virtual">
    <?php print stripslashes($urlap["ismerteto"]) ?>
    </textarea>
    </p>
    <p>
    <input type="submit" value="Rendben">
    </p>
    </form>

    </body>
    </html>
```

Beillesztjük az `adatbazis.inc` és `kozosfv.inc` nevű külső állományainkat, így rögtön kész adatbáziskapcsolattal és munkamenettel kezdjük az oldalt. Ezután meghívjuk az új `azonositas()` nevű függvényt, ami a `kozosfv.inc` fájlban található. Ez összehasonlítja a munkamenet adatait az adatbázissal.

23.9. program Részlet a kozosfv.inc fájlból

```
function azonositas( )
    {
    global $munkamenet, $belepett;
    $munkamenet["belepett"] = false;
```

23.9. program (folytatás)

```
 5:      $klub_sor = sorLekeres( "klubok", "azonosito",
                     $munkamenet["azonosito"] );
 6:      if ( ! $klub_sor ||
 7:        $klub_sor["nev"] != $munkamenet["nev"] ||
 8:        $klub_sor["jelszo"] != $munkamenet["jelszo"] )
 9:        {
10:        header( "Location: belepes.php" );
11:        exit;
12:        }
13:      $munkamenet["belepett"] = true;
14:      return $klub_sor;
15:      }
```

Az `azonositas()` jól átlátható függvény. A `$munkamenet["azonosito"]` elemét használjuk arra, hogy a `sorLekeres()` függvény segítségével lekérjük a klubhoz tartozó sort az adatbázisból. Ezt a `$klub_sor` asszociatív tömbben tároljuk és ellenőrizzük a `nev` és `jelszo` elemek egyezését a `$munkamenet` megfelelő tömbelemeivel. Ha nem egyeznek, a `belepes.php` oldalra küldjük a tagot.

Miért vállaljuk fel az adatbázis lekérdezésének költségét az azonosításhoz? Miért nem egyszerűsítjük a problémát a `belepett` elem ellenőrzésére? Ez visszaéléseknek engedne teret, mivel egy rosszindulatú látogató egyszerűen hozzáadhatna az oldal címéhez egy ilyen részt:

```
munkamenet%5Bbelepett%5D=1&munkamenet%5Bazonosito%5D=1
```

így átvéve az `1`-es azonosítóval rendelkező tag fölött az irányítást, a PHP ugyanis ezt a `$munkamenet` tömbbé alakítja, amely egy felületesebb ellenőrzésen túljuthatna. A rosszindulatú látogató most is alkalmazhat egy hasonló trükköt, hogy belépést nyerjen, de mivel az adatbázisban lévő névvel és jelszóval történő egyezést vizsgáljuk, a „hamisított” `$munkamenet` változónak helyes tartalommal kell rendelkeznie, így pedig felesleges a csalás.

Az `azonositas()` függvény melléktermeke a visszaadott tömb, amely a kérdéses klubról az adatbázisban található összes adatot tartalmazza. Később az oldalak ezt az információt saját céljaikra használhatják fel. Miután az `azonositas()` függvény visszatér a klub adatait tartalmazó tömbbel, ezt a `$klub_sor` változónak adjuk értékül.

Ismételten ugorjunk a HTML részre. Itt először a `kozosnav.inc` állományt illesztjük be. Ebben a fájlban azonban továbblépünk az eddigieknél és szerepeltetjük a tagok menüjét is:

23.10. program kozosnav.inc

```
 1: <p>
 2: <a href="klubokinfo.php?<?php print SID ?>">
     Klubinformációk</a> |
 3: <a href="esemenyekinfo.php?<?php print SID ?>">
     Eseményinformációk</a> |
 4: <a href="csatlakozas.php?<?php print SID ?>">
     Csatlakozás</a> |
 5: <a href="belepes.php?<?php print SID ?>">Belépés</a> |
 6: <a href="index.php?<?php print SID ?>">Honlap</a>
 7: </p>
 8: <?php
 9: if ( $munkamenet["belepett"] )
10:      {
11:      ?>
12:      <p>
13:      <A HREF="klubfrissites.php?<?php print SID ?>”>
         Klub részletei</A> |
14:      <A HREF="esemenylista.php?<?php print SID ?>">
         Bejegyzett események</A> |
15:      <A HREF="esemenyfrissites.php?<?php print SID ?>">
         Új esemény</A> |
16:      <A HREF="tagmenu.php?<?php print SID ?>">
         Tagok honlapja</A>
17:      </p>
18:      <?
19:      }
20: ?>
21: <hr>
```

Ha a `$munkamenet["belepett"]` igazra van állítva, a csak tagok számára megjelenítendő hivatkozásokat is kiírjuk. Vegyük észre, hogy minden hivatkozásban megadjuk a `SID` állandót. Ez tartalmazza a munkamenet nevét és azonosítóját, így biztosítjuk, hogy az azonosító oldalról-oldalra átadódjon, még akkor is, ha a sütik nincsenek bekapcsolva.

A `klubfrissites.php` oldal űrlapja igazán figyelemreméltó. Az eddigieknek megfelelően a `mitkelltenni` és a munkamenetet azonosító rejtett mezőket is felvesszük az űrlapba. Szövegmezőket biztosítunk a klub neve, ismertetője és a kapcsolattartó elektronikus levélcíme számára. A klubtípusok és területek lenyíló menüinek előállítására egy új függvény, az `optionLista()` szolgál.

23.11. program Részlet az adatbazis.inc fájlból

```
 1: function optionLista( $tabla, $azon )
 2:      {
 3:      global $kapcsolat;
 4:      $eredmeny = mysql_query( "SELECT * FROM $tabla",
                                  $kapcsolat );
 5:      if ( ! $eredmeny )
 6:           {
 7:           print "Nem lehet megnyitni: $tabla<p>";
 8:           return false;
 9:           }
10:      while ( $egy_sor = mysql_fetch_row( $eredmeny ) ){
11:           print "<option value=\"$egy_sor[0]\"";
12:           if ( $azon == $egy_sor[0] )
13:                print "SELECTED";
14:           print ">$egy_sor[1]\n";
15:           }
16:      }
```

A függvény a `$tabla` paramétert használja, hogy kiválasszon minden sort a `teruletek` vagy `tipusok` táblából. Ezután végiglépked minden soron az eredménytáblában, kiírva egy HTML `option` elemet a böngésző számára. Ha a `mysql_fetch_array()` által visszaadott tömb első eleme egyezik az `$azon` paraméterben megadott azonosítóval, a `SELECTED` karakterláncot is kiírja. Ezzel biztosítjuk, hogy a megfelelő elem kerül alapbeállításban kiválasztásra az űrlapon.

A PHP kódban, ahol ellenőrizzük a bevitt adatokat, első feltételünk az `urlap["klubnev"]` kitöltöttségének tényét vizsgálja. Ezután az `urlap["terulet"]` és `urlap["tipus"]` értékek helyességét ellenőrizzük. Ha az `$uzenet` változó ezeket követően üres marad, tudjuk, hogy az adatok a meghatározott feltételeknek eleget tesznek, ezért átadjuk azokat egy új függvénynek, a `klubFrissites()`-nek, amely végrehajtja a szükséges módosítást az adatbázisban. A függvény egy klubazonosítót, egy klubnevet, egy területet, egy típust, egy e-mail címet és egy ismertetőt vár a `klubok` tábla számára.

23.12. program Részlet az adatbazis.inc fájlból

```
1: function klubFrissites( $azonosito, $klubnev,
                           $terulet, $tipus, $email,
                           $ismerteto )
2:      {
3:      global $kapcsolat;
4:      $lekeres = "UPDATE klubok set klubnev='$klubnev',
        terulet='$terulet', tipus='$tipus',
5:      email='$email', ismerteto='$ismerteto' WHERE
        azonosito='$azonosito'";
6:      $eredmeny = mysql_query( $lekeres, $kapcsolat );
7:      if ( ! $eredmeny )
8:          die ( "klubFrissites hiba: ".mysql_error() );
9:      }
```

Miután a sort frissítettük, a tagot a `tagmenu.php` oldalra küldhetjük.

Ha a tag az oldalt egy hivatkozásra kattintva vagy a böngészőjében lévő könyvjelzővel érte el, a `$mitkelltenni` változó nem kerül beállításra és az ellenőrzést és frissítést végző kód nem hajtódik végre. Még mindig van azonban egy feladatunk. A `$klub_sor` asszociatív tömb tartalmazza a klub adatait, amit az `azonositas()` függvénnyel töltöttünk fel. Ebben találhatók az adott klubhoz tartozó mezőnevek és értékek az adatbázisból. Ezt a tömböt adjuk értékül az `$urlap` tömbnek, így biztosítva, hogy az űrlap alapbeállításban az adatbázisban található érvényes információkat tartalmazza.

A 23.3. ábra a `klubfrissites.php` kimenetét mutatja.

23.3. ábra

A klubfrissites.php kimenete

tagmenu.php

A `tagmenu.php` a tagok területének központja. Alapvetően hivatkozások listájából áll, de a tagok számára fontos hírek és ajánlatok is helyet foglalhatnak az oldalon.

23.13. program tagmenu.php

```
 1: <?php
 2:      include("adatbazis.inc");
 3:      include("kozosfv.inc");
 4:
 5:      $klub_sor = azonositas();
 6:      klubAdatEllenorzes( $klub_sor );
 7:      ?>
 8:
 9:     <html>
10:     <head>
11:     <title>Üdvözlet!</title>
12:     </head>
13:
14:     <body>
15:     <?php
16:     include("kozosnav.inc");
17:     ?>
18:
19:     <h1>Tagok honlapja</h1>
20:
21:     <a href="klubfrissites.php?<?php print SID ?>">
        Klub részletei</a><br>
22:     <a href="esemenylista.php?<?php print SID ?>">
        Bejegyzett események</a><br>
23:     <a href="esemenyfrissites.php?<?php print SID ?>">
        Új esemény</a><br>
24:
25:     </body>
26:     </html>
```

Az oldalon egyetlen újdonság található. Miután meghívjuk az `azonositas()` függvényt, hogy meggyőződjünk róla, hogy a látogatónk tag, az általa visszaadott tömböt egy új függvénynek adjuk át. Ez a függvény, a `klubAdatEllenorzes()`, a `kozosfv.inc` állományban található és azt ellenőrzi, hogy a tag megadta-e már a klub adatait. Amíg egy tag ki nem tölti legalább a klub nevét, nem adhat rendezvényeket a rendszerhez.

23.14. program Részlet a kozosfv.inc fájlból

```
function klubAdatEllenorzes( $klubtomb  )
     {
     if ( ! isset( $klubtomb["klubnev"] ) )
          {
          header( "Location: klubfrissites.php?".SID );
          exit;
          }
     }
```

belepes.php

Mielőtt továbblépünk a rendezvényeket kezelő oldalakra, el kell készítenünk a `belepes.php` programot. Ez az oldal ad lehetőséget a bejegyzett tagoknak, hogy a későbbiekben belépjenek a rendszerbe.

23.15. program belepes.php

```
<?php
    include("adatbazis.inc");
    include("kozosfv.inc");

     $uzenet="";

     if ( isset( $mitkelltenni ) && $mitkelltenni ==
          "belepes" )
          {
          if ( empty( $urlap["nev"] ) || empty(
               $urlap["jelszo"] )  )
             $uzenet .= "Ki kell töltenie minden
                         mezőt!<br>\n";

        if ( ! ( $sor_tomb =
                  jelszoEllenorzes( $urlap["nev"],
                  $urlap["jelszo"] ) ) )
             $uzenet .= "Hibás név vagy jelszó, próbál-
                         kozzon újra!<br>\n";

```

23.15. program (folytatás)

```
16:         if ( $uzenet == "" ) // nem találtunk hibát
17:              {
18:              munkamenetFeltoltes( $sor_tomb["azonosito"],
                 $sor_tomb["nev"],
19:                                   $sor_tomb["jelszo"] );
20:              header( "Location: tagmenu.php?".SID );
21:              }
22:         }
23:     ?>
24:
25:     <html>
26:     <head>
27:     <title>Belépés</title>
28:     </head>
29:
30:     <body>
31:
32:     <?php
33:     include("kozosnav.inc");
34:     ?>
35:
36:     <h1>Belépés</h1>
37:
38:     <?php
39:     if ( $uzenet != "" )
40:          {
41:          print "<p><b>$uzenet</b></P>";
42:          }
43:     ?>
44:
45:     <p>
46:     <form action="<?php print $PHP_SELF;?>">
47:     <input type="hidden" name="mitkelltenni"
               value="belepes">
48:     <input type="hidden" name="<?php
               print session_name() ?>"
49:              value="<?php print session_id() ?>">
50:     </p><p>
51:     Tagsági név: <br>
52:     <input type="text" name="urlap[nev]"
53:              value="<?php print $urlap["nev"] ?>">
```

23.15. program (folytatás)

```
54:     </p><p>
55:     Jelszó: <br>
56:     <input type="password" name="urlap[jelszo]"
         value="">
57:     </p><p>
58:     <input type="submit" value="Rendben">
59:     </form>
60:
61:     </body>
62:     </html>
```

Az oldal szerkezete már ismerős kell, hogy legyen. Az `adatbazis.inc` és `kozosfv.inc` állományokat használjuk, hogy adatbáziskapcsolatot létesítsünk és aktív munkamenettel rendelkezzünk.

Ha a `$mitkelltenni` változó be van állítva és a várt `"belepes"` értéket tartalmazza, ellenőrizzük az űrlapról érkezett adatokat. Az új `adatbazis.inc` függvényt használjuk arra, hogy az `urlap["nev"]` és `urlap["jelszo"]` értékeket vizsgáljuk.

23.16. program Részlet az adatbazis.inc fájlból

```
 1: function jelszoEllenorzes( $nev, $jelszo )
 2:      {
 3:      global $kapcsolat;
 4:      $eredmeny = mysql_query( "SELECT azonosito, nev,
                    jelszo FROM klubok
 5:                 WHERE nev='$nev' and
                    jelszo='$jelszo'",
 6:                 $kapcsolat );
 7:      if ( ! $eredmeny )
 8:           die ( "jelszoEllenorzes hiba: "
                   .mysql_error() );
 9:      if ( mysql_num_rows( $eredmeny ) )
10:           return mysql_fetch_array( $eredmeny );
11:      return false;
12:      }
```

A `jelszoEllenorzes()` egy belépési nevet és egy jelszót vár. Ezek felhasználásával egy egyszerű `SELECT` lekérdezést küld az adatbázisnak a klubok táblájára vonatkozóan.

Visszatérve a `belepes.php` oldalra, ha nem találunk hibát, meghívjuk a `munkamenetFeltoltes()` függvényt, amely beállítja az `azonosito`, `nev`, `jelszo` és `belepett` elemeket a `$munkamenet` tömbben. Ezután a tagot átirányítjuk a `tagmenu.php` oldalra.

esemenyfrissites.php

Most, hogy a tagok már képesek csatlakozni és belépni a rendszerbe, valamint módosítani is tudják az adataikat, lehetőséget kell adnunk a rendezvények bevitelére és módosítására. A teljes `esemenyfrissites.php` oldal a 23.17. programban látható.

23.17. program esemenyfrissites.php

```
 1: <?php
 2:     include("adatbazis.inc");
 3:     include("kozosfv.inc");
 4:     include("datum.inc");
 5:
 6:     $klub_sor = azonositas();
 7:     klubAdatEllenorzes( $klub_sor );
 8:
 9:     $edatum = time();
10:     $uzenet = "";
11:
12:     if ( ! empty( $eazonosito ) )
13:         $esemeny_sor = sorLekeres( "esemenyek",
                        "eazonosito", $eazonosito );
14:     else
15:         $eazonosito = false;
16:
17:     if ( isset( $mitkelltenni ) &&
             $mitkelltenni=="esemenyFrissites" )
18:         {
19:         if ( empty( $urlap["enev"] ) )
20:             $uzenet .="Az eseménynek névvel kell
                        rendelkeznie!<br>\n";
21:
```

23.17. program (folytatás)

```
22:         if ( ! sorLekeres( "teruletek", "azonosito",
                 $urlap["eterulet"] ) )
23:             $uzenet .= "KRITIKUS HIBA: A terület
                            kódja nem található!<br>";
24:
25:         if ( ! sorLekeres( "tipusok", "azonosito",
                 $urlap["etipus"] ) )
26:             $uzenet .= "KRITIKUS HIBA: A típus
                            kódja nem található!<br>";
27:
28:         foreach ( array( "ehonap", "eev", "enap",
                             "eperc" )
29:                   as $datum_egyseg )
30:             {
31:             if ( ! isset( $urlap[$datum_egyseg] ) )
32:                 {
33:                 $uzenet .= "KRITIKUS HIBA: A dátum
                                nem értelmezhető!";
34:                 break;
35:                 }
36:             }
37:         $edatum =  mktime( $urlap["eora"],
                       $urlap["eperc"], 0,
                       $urlap["ehonap"],
38:                    $urlap["enap"], $urlap["eev"] );
39:
40:         if ( $edatum < time() )
41:             $uzenet .= "Múltbeli dátum nem
                            fogadható el!";
42:
43:         if ( $uzenet == "" )
44:             {
45:             esemenyModositas( $urlap["enev"],
                          $urlap["ehelyszin"],
                          $urlap["eterulet"],
46:                       $urlap["etipus"],
                          $urlap["ecim"], $urlap["eirsz"],
47:                       $urlap["eismerteto"],
                          $munkamenet["azonosito"],
                          $edatum,
48:                       $eazonosito );
49:             header( "Location:
                          esemenylista.php?".SID );
50:             }
```

23.17. program (folytatás)

```
51:         }
52:     elseif ( $eazonosito )
53:         {
54:         //foreach( $esemeny_sor as $kulcs=>$ertek )
55:         //     $urlap[$kulcs] = $ertek;
56:         $urlap = $esemeny_sor;
57:         $edatum =  $esemeny_sor["edatum"];
58:         }
59:     else
60:         {
61:         $urlap["eterulet"] = $klub_sor["terulet"];
62:         $urlap["etipus"] = $klub_sor["tipus"];
63:         }
64:     ?>
65:
66:     <html>
67:     <head>
68:     <title>Esemény hozzáadása/frissítése</title>
69:     </head>
70:     <body>
71:
72:     <?php
73:     include("kozosnav.inc");
74:     ?>
75:
76:     <h1>Esemény hozzáadása/frissítése</h1>
77:
78:     <?php
79:     if ( $uzenet != "" )
80:         {
81:         print "<b>$uzenet</b>";
82:         }
83:     ?>
84:     <p>
85:     <form action="<?php print $PHP_SELF;?>">
86:     <input type="hidden" name="mitkelltenni"
                            value="esemenyFrissites">
87:     <input type="hidden" name="<?php
            print session_name() ?>"
88:              value="<?php print session_id() ?>">
```

23.17. program (folytatás)

```
 89:     <input type="hidden" name="eazonosito"
 90:            value="<?php print $eazonosito ?>">
 91:     Esemény neve: <br>
 92:     <input type="text" name="urlap[enev]"
 93:            value="<?php print
                stripslashes($urlap["enev"]) ?>">
 94:     </p>
 95:     <p>
 96:     Dátum és idő: <br>
 97:     <select name="urlap[ehonap]">
 98:     <?php honapLehetosegek( $edatum ) ?>
 99:     </select>
100:
101:     <select name="urlap[enap]">
102:     <?php napLehetosegek( $edatum ) ?>
103:     </select>
104:
105:    <select name="urlap[eev]">
106:    <?php evLehetosegek( $edatum ) ?>
107:    </select>
108:
109:    <select name="urlap[eora]">
110:    <? oraLehetosegek( $edatum ) ?>
111:    </select>
112:
113:    <select name="urlap[eperc]">
114:    <? percLehetosegek( $edatum ) ?>
115:    </select>
116:    </p>
117:    <p>
118:    Esemény helye: <br>
119:    <select name="urlap[eterulet]">
120:    <?php optionLista( "teruletek",
        $urlap["eterulet"] ) ?>
121:    </select>
122:    </p>
123:    <p>
124:    Esemény típusa: <br>
125:    <select name="urlap[etipus]">
126:    <?php optionLista( "tipusok", $urlap["etipus"])?>
127:    </select>
```

23.17. program (folytatás)

```
128:    </p>
129:    <p>
130:    Esemény ismertetője: <br>
131:    <textarea name="urlap[eismerteto]" wrap="virtual"
                rows=5 cols=30>
132:    <?php print stripslashes($urlap["eismerteto"]) ?>
133:    </textarea>
134:    </p>
135:    <p>
136:    Helyszín: <br>
137:    <input type="text" name="urlap[ehelyszin]"
138:          value="<?php print
                    stripslashes($urlap["ehelyszin"])?>">
139:    </p>
140:    <p>
141:    Helyszín címe: <br>
142:    <textarea name="urlap[ecim]" wrap="virtual"
                rows=5 cols=30>
143:    <?php print stripslashes($urlap["ecim"]) ?>
144:    </textarea>
145:    </p>
146:    <p>
147:    Helyszín irányítószáma: <br>
148:    <input type="text" name="urlap[eirsz]"
149:           value="<?php print
                     stripslashes($urlap["eirsz"]) ?>">
150:    </p>
151:    <p>
152:    <input type="submit" value="Rendben">
153:    </p>
154:    </form>
155:
156:    </body>
157:    </html>
```


Ezen az oldalon az esemenyek táblának megfelelő elemekkel készítünk egy űrlapot. Szokás szerint ellenőriznünk kell a beérkező értékeket és frissítenünk kell az adatbázist. Ha azonban az oldal meghívása alapján úgy ítéljük, hogy nem esemény hozzáadása a cél, hanem módosítás, akkor le kell kérdeznünk az adatbázisból a módosítandó adatokat. Az oldal meghívása során az `$eazonosito` változót kapja,

ha egy meglévő esemény módosítását kell elvégezni. Ha a változó nem üres, a `sorLekeres()` függvényt alkalmazva kapjuk meg az `$esemeny_sor` tömbben a rendezvény adatait. Ha az `$eazonosito` nincs megadva vagy üres, akkor hamis értékre állítjuk.

Ha az oldal űrlapkitöltés hatására hívódott meg, ellenőriznünk kell a beérkezett adatokat. A dátummal kapcsolatban meg kell vizsgálnunk, hogy az nem egy múltbeli érték-e. Ennek érdekében beállítunk egy `$edatum` nevű globális változót a beadott adatok függvényében.

Ha az adatok megfelelnek az általunk támasztott követelményeknek, az `adatbazis.inc` egy új függvényét hívjuk meg, melynek neve `esemenyModositas()`. Ez a függvény minden olyan értéket vár, amit az `esemenyek` táblában el kell helyeznünk. Az utolsó paraméter a rendezvény azonosítószáma. Ezt az `esemenyModositas()` függvény arra használja, hogy megállapítsa, új eseményt kell felvennie vagy módosítania kell egy már létezőt. Figyeljük meg, hogy a klub azonosítószáma a `$munkamenet["azonosito"]` változóban található, így ezt helyezzük az `esemenyek` tábla `eklub` mezőjébe. A dátum adatbázisban történő tárolására időbélyeget használunk.

23.18. program Részlet az adatbazis.inc fájlból

```
 1: function esemenyModositas( $enev, $ehelyszin,
            $eterulet, $etipus, $ecim, $eirsz,
            $eismerteto, $eklub, $edatum, $eazonosito )
 2:     {
 3:     global $kapcsolat;
 4:     if ( ! $eazonosito )
 5:         {
 6:         $lekeres = "INSERT INTO esemenyek (enev,
                ehelyszin, eterulet, etipus,
 7:                ecim, eirsz, eismerteto, eklub,
                edatum )
 8:                  VALUES( '$enev', '$ehelyszin',
                         '$eterulet', '$etipus',
                         '$ecim',
 9:                '$eirsz', '$eismerteto', '$eklub',
                '$edatum')";
10:         }
11:     else
12:         {
```

23.18. program (folytatás)

```
13:          $lekeres = "UPDATE esemenyek SET enev='$enev',
                              ehelyszin='$ehelyszin',
14:                           eterulet='$eterulet',
                                       etipus='$etipus',
                                       ecim='$ecim',
                                       eirsz='$eirsz',
15:
eismerteto='$eismerteto', eklub='$eklub', edatum='$edatum'
16:                  WHERE eazonosito='$eazonosito'";
17:          }
18:      $eredmeny = mysql_query( $lekeres, $kapcsolat );
19:      if ( ! $eredmeny )
20:          die ( "esemenyModositas hiba: "
                   .mysql_error() );
21:      }
```

Látható, hogy a függvény az `$eazonosito` paraméter értékétől függően működik. Ha hamis értékű, egy `INSERT` SQL parancs hajtódik végre a megadott adatokkal. Egyéb esetben az `$eazonosito` egy `UPDATE` kérés feltételében szerepel. Miután az adatbázist frissítettük, a tagot az `esemenylista.php` oldalra irányítjuk, ahol a teljes rendezvénynaptárt láthatja.

Ha a tag még nem töltötte ki az űrlapot, átugorjuk az ellenőrző és adatbázis-frissítő kódokat. Ha azonban rendelkezünk az `$eazonosito` változóval, már lekértük az esemény információit az adatbázisból és az űrlapra írhatjuk azokat. Ezt úgy tehetjük meg, hogy az `$esemeny_sor` változót értékül adjuk az `$urlap` változónak, az `$edatum` változót pedig az `$esemeny_sor["edatum"]` értékével töltjük fel.

Ha nem kaptunk űrlapértékeket és `$eazonosito` változónk sincs, az `$urlap["terulet"]` és `$urlap["tipus"]` elemeknek a klubnak megfelelő adatokat adjuk értékül a `$klub_sor` tömbből. Ezzel a megfelelő lenyíló menükben a klubokhoz tartozó értékek lesznek alapbeállításban kiválasztva.

A HTML űrlapon az egyetlen említésre méltó kód a dátum és idő számára készített lenyíló menüket állítja elő. Ezeket a menüket elég egyszerűen beépíthettük volna a PHP kódba, de szükségünk van arra, hogy alapbeállításban az aktuális időt, a tag által korábban választott időt, vagy az `esemenyek` táblában tárolt időt jelezzék. Az alapbeállításban kiválasztandó értéket már elhelyeztük az `$edatum` változóban. Ezt a program elején az aktuális időre állítottuk be. Ha beérkező adatokat észlelünk, annak megfelelően állítjuk be ezt az értéket. Egyéb esetben, ha már meglévő eseményt módosítunk, az `esemenyek` tábla `edatum` mezőjének értékét adjuk az `$edatum` változónak.

Minden lenyíló menühöz az `$edatum` változóban lévő időbélyeget adjuk át a megfelelő függvénynek. Ezek az új `datum.inc` nevű külső állományban találhatók.

23.19. program Részlet a datum.inc fájlból

```
function honapLehetosegek( $datum )
      {
      $datum_tomb = getDate( $datum );
      $honapok = array( "Jan","Feb","Már",
                        "Ápr","Máj","Jún",
                        "Júl","Aug","Szep",
                        "Okt","Nov","Dec" );
      foreach ( $honapok as $kulcs=>$ertek )
          {
          print "<OPTION
                  VALUE=\"".($kulcs+1)."\"";
          print ( ( $datum_tomb["mon"] ==
                    ($kulcs+1) )?"SELECTED":"" );
          print ">$ertek\n";
         }
    }
 function napLehetosegek( $datum )
    {
    $datum_tomb = getDate( $datum );
    for ( $x = 1; $x<=31; $x++ )
        {
        print "<OPTION VALUE=\"$x\"";
        print ( ( $datum_tomb["mday"] ==
                  $x )?"SELECTED":"" );
        print ">$x\n";
        }
    }

 function evLehetosegek( $datum )
    {
    $datum_tomb = getDate( $datum );
    $most_tomb = getDate(time());
    for ( $x = $most_tomb["year"];
          $x <= ($most_tomb["year"]+5); $x++ )
        {
        print "<OPTION VALUE=\"$x\"";
```

23.19. program (folytatás)

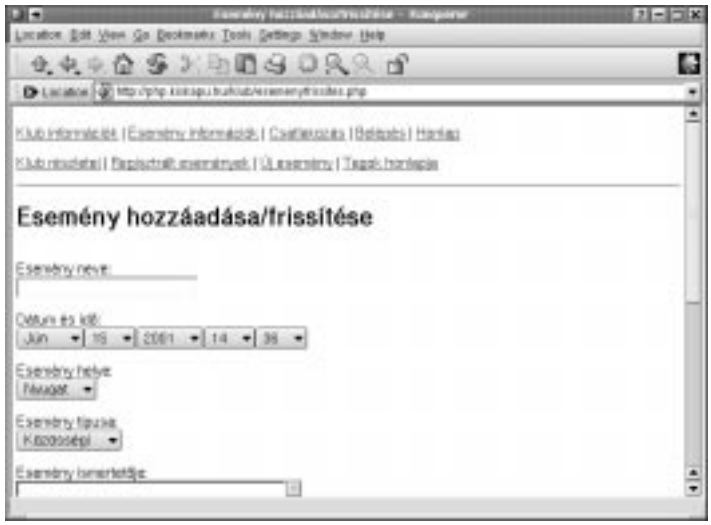
```
31:             print ( ( $datum_tomb["year"] ==
                          $x )?"SELECTED":"" );
32:             print ">$x\n";
33:             }
34:         }
35:
36:     function oraLehetosegek( $datum )
37:         {
38:         $datum_tomb = getDate( $datum );
39:         for ( $x = 0; $x< 24; $x++ )
40:             {
41:             print "<OPTION VALUE=\"$x\";
42:             print ( ( $datum_tomb["hours"] ==
                          $x)?"SELECTED”:"" );
43:             print ">".sprintf("%'02d",$x)."\n";
44:             }
45:         }
46:
47:     function percLehetosegek( $datum )
48:         {
49:         $datum_tomb = getDate( $datum );
50:         for ( $x = 0; $x<= 59; $x++ )
51:             {
52:             print "<OPTION VALUE=\"$x\"";
53:             print ( ( $datum_tomb["minutes"] ==
                          $x )?"SELECTED":"" );
54:             print ">".sprintf("%'02d",$x)."\n";
55:             }
56:         }
```

Minden függvény egy időbélyeget vár és HTML `OPTION` elemek listáját írja ki. A `getDate()` PHP függvényt használják arra, hogy az időbélyeg megfelelő elemének indexét megállapítsák (év, hónap, hónap napja, óra, perc). Ezután a kapott számot összehasonlítják az alkalmazott tartomány értékeivel (a tartomány a hónap napjainál 1-től 31-ig, a perceknél 0-tól 59-ig terjed). Ha egyezést észlelnek, a `SELECTED` jelzőt adják az `OPTION` elemhez, így választják ki a megfelelő sort.

A 23.4. ábra az `esemenyfrissites.php` kimenetét mutatja.

23.4. ábra

Az esemenyfrissites.php kimenete



esemenylista.php

Végül biztosítanunk kell a tagoknak egy oldalt, ahol minden általuk bevitt eseményt végigtekinthetnek. Ennek az oldalnak lehetőséget kell adnia arra is, hogy a tagok az eseményeket szerkeszthessék vagy törölhessenek egyet.
Az `esemenylista.php` az eddigiek ismeretében egyszerű program.

23.20. program esemenylista.php

```
 1: <?php
 2:      include("adatbazis.inc");
 3:      include("kozosfv.inc");
 4:      $klub_sor = azonositas();
 5:      klubAdatEllenorzes( $klub_sor );
 6:
 7:      function esemenyKiiras()
 8:          {
 9:         global $klub_sor;
10:         $esemenyek = esemenyekLekeres
                          ( $klub_sor["azonosito"] );
11:         if ( ! $esemenyek )
12:              {
13:              print "Nincs esemény a listában<P>";
14:              return;
15:              }
```

23.20. program (folytatás)

```
16:         print "<table border=1>\n";
17:         print "<td><b>Dátum</b></td>\n<td>
                   <b>Név</b></td>\n
18:                <td><b> </b></td>\n";
19:         foreach ( $esemenyek as $sor )
20:             {
21:           print "<tr>\n";
22:           print "<td>".date("j M Y H.i",
                    $sor["edatum"])."</td>\n";
23:           print "<td><a href=
                    \"esemenyfrissites.php?eazonosito=".
                    $sor["eazonosito"]."&".SID."\">".
24:                    html($sor["enev"])."</a></td>\n";
25:           print "<td><a href=\"".
                    $GLOBALS["PHP_SELF"]."?eazonosito=".
                    $sor["eazonosito"];
26:           print "&mitkelltenni=
                    esemenyTorles&".SID."\" ";
27:           print "onClick=\"return
                    window.confirm('Biztos benne, hogy
                    törli ezt az eseményt?')\">";
28:           print "törlés</a><br></td>\n";
29:           print "</tr>\n";
30:           }
31:         print "</table>\n";
32:         }
33:     $uzenet="";
34:
35:     if ( isset( $mitkelltenni ) &&
36:           $mitkelltenni == "esemenyTorles" &&
                             isset( $eazonosito ) )
37:         {
38:         esemenyTorles( $eazonosito );
39:         $uzenet .= "Az esemény törlése
                      megtörtént!<br>";
40:         }
41:
42:     ?>
43:     <html>
44:     <head>
45:     <title>Események listája</title>
46:     </head>
```

23.20. program (folytatás)

```
47:     <body>
48:     <?php
49:     include("kozosnav.inc");
50:     ?>
51:     <h1>Események listája</h1>
52:     <?php
53:     if ( $uzenet != "" )
54:          {
55:          print "<b>$uzenet</b>";
56:          }
57:     ?>
58:
59:     <?php
60:     esemenyKiiras();
61:     ?>
62:
63:     </body>
64:     </html>
```

Szokás szerint az `adatbazis.inc` fájlt használjuk, hogy kész adatbáziskapcsolattal kezdjük az oldalt és a `kozosfv.inc` fájlt, hogy aktív munkamenetünk legyen. Úgyszintén a megszokott módon ellenőrizzük, hogy érvényes tag kérte-e le az oldalt. Ezután létrehozunk egy új `esemenyKiiras()` nevű függvényt, amely közvetlenül a böngészőbe írja az eseményről elérhető információkat. Emlékezzünk rá, hogy az `azonositas()` függvény által visszaadott `$klub_sor` tömb tartalmazza a klubadatokat. A `$klub_sor["azonosito"]` értéket használhatjuk arra, hogy a klubhoz tartozó összes rendezvényt listába írjuk. Ehhez az `adatbazis.inc` `esemenyekLekeres()` függvényét használjuk. Ezt a függvényt alaposabban a következő órában tárgyaljuk, egyelőre elegendő annyit tudnunk, hogy egy klubazonosítót vár első paraméteréül, így egy kétdimenziós tömbben adja vissza a klubhoz tartozó összes esemény minden adatát.

A függvény visszatérési értékét az `$esemenyek` tömbben tároljuk. Valójában több adatot kérünk le, mint amennyire szükségünk van, de az `esemenyekLekeres()` függvény olyan rugalmas, hogy nyugodtan használhatjuk, hacsak nem érzékelünk teljesítménybeli problémákat a program futása során. Ha az `$esemenyek` hamis értékű, figyelmeztető üzenetet küldünk a kimenetre és befejezzük a függvény futását. Egyéb esetben egy HTML táblázatot építünk fel: végiglépkedve az `$esemenyek` tömbön kiírunk minden eseményre vonatkozóan néhány információt.

A `date()` PHP függvénynek átadva minden tömb `edatum` elemét, formázott dátumokat írunk ki. A dátum mellett minden esemény számára egy hivatkozást hozunk létre, felhasználva az esemény azonosítóját és az `enev` értéket. Az átadandó `eazonosito` változó mellett szerepeltetjük a `SID` állandót, így biztosítva, hogy a munkamenet érvényes marad az új oldalon is. A hivatkozás az `esemenyfrissites.php` oldalra mutat, amely így meg tudja jeleníteni az esemény adatait a szerkesztés számára. Végül minden rendezvény sorában egy erre az oldalra visszamutató hivatkozást is szerepeltetünk. Ehhez az `eazonosito` mellett egy `esemenyTorles` értékű `mitkelltenni` paramétert is meghatározunk. Ez az adott esemény törlésére szolgál majd, ezért egy JavaScript eseménykezelőt is adunk a hivatkozáshoz, hogy ne lehessen véletlenül törölni egy rendezvényt sem.

Miután elkészítettük az `esemenyKiiras()` függvényt, kezelnünk kell azt az esetet is, amikor a tag törölni kíván egy eseményt. Ezt a `$mitkelltenni` és az `$eazonosito` változók ellenőrzésével tehetjük meg. Ha rendben megkaptuk ezeket, egy új `adatbazis.inc` függvényt, az `esemenyTorles()`-t hívjuk meg, átadva az `$eazonosito` értéket.

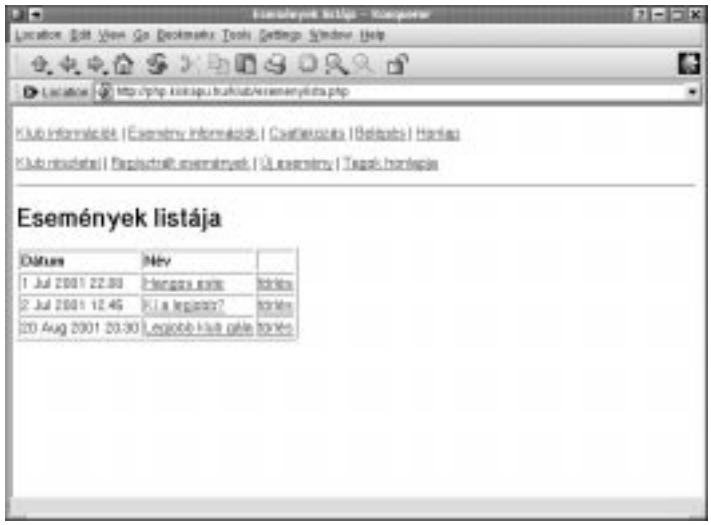
23.21. program Részlet az adatbazis.inc fájlból

```
1: function esemenyTorles( $eazonosito )
2:      {
3:      global $kapcsolat;
4:      $lekeres = "DELETE FROM esemenyek WHERE
                    eazonosito='$eazonosito'";
5:      $eredmeny = mysql_query( $lekeres, $kapcsolat );
6:      if ( ! $eredmeny )
7:          die ( "esemenyTorles hiba: ".mysql_error() );
8:      return ( mysql_affected_rows($kapcsolat) );
9:      }
```

Az `esemenylista.php` egy lehetséges kimenetét a 23.5. ábra mutatja.

23.5. ábra

Az esemenylista.php kimenete



Összefoglalás

Az óra végére elkészültünk a teljesen működő, tagok számára készült környezettel a rendezvényeket nyilvántartó rendszerhez. Lehetőséget teremtettünk grafikusainknak az oldalak jó kialakítására, azzal, hogy a lehető legtöbb kódot külső állományokban helyeztük el. Gyakran használtuk a `header()` függvényt, hogy új oldalra irányítsuk a tagot, ha adatokat várunk tőle. A belépési információk tárolásához munkamenet-kezelő függvényeket, a tagok azonosításához adatbázis-lekérdezéseket használtunk minden oldalon.

A következő órában hasonló megoldásokat fogunk használni a program nyilvános felületének kialakításához.

Kérdések és válaszok

A több oldalból álló alkalmazások megtervezése és karbantartása bonyolultnak tűnik. Van valami titok emögött?

A példaalkalmazás elkészítése során majdnem minden oldalon egyszerű mintát követtünk. Ahogy alkalmazásaink nagyobbá válnak, a teljes környezetről egyetlen programként kell gondolkodnunk. A különálló oldalak csak a látogató csatlakozási pontjai az egész alkalmazáshoz. Tartsuk az összes egynél több oldalon ismétlődő kódot külön állományban!

Ha esetleg úgy döntünk, hogy egyetlen központi programot készítünk a teljes alkalmazás számára, egy egyszerű, alapvető HTML elemeket tartalmazó oldalt kell készítenünk. Minden más tartalom dinamikusan áll majd elő a program futása során. Szükségünk lesz egy `mitkelltenni` típusú változóra, amely mindig a megfelelő szolgáltatást választja ki a teljes alkalmazásból. Ez sokkal formásabbá teheti a kódot és segít elkerülni a nem túl elegáns `Location` fejléctrükköt. Másrészt ez azzal jár, hogy sokkal több HTML-t kell beépítenünk a PHP kódok közé.

Műhely

A műhelyben kvízkérdések találhatók, melyek segítenek megszilárdítani az órában szerzett tudást. A válaszokat az A függelékben helyeztük el.

Kvíz

1. Melyik PHP függvényt használjuk arra, hogy MySQL adatbázishoz csatlakozzunk?

2. Melyik függvényt alkalmazzuk a munkamenet elkezdésére vagy folytatására?

3. Melyik függvény alkalmas arra, hogy külső állományokat illesszünk a PHP kódba?

4. Melyik PHP függvénnyel küldünk SQL kéréseket egy MySQL adatbázis számára?

5. Melyik állandót alkalmazhatjuk arra, hogy a munkafolyamat-azonosítót eljuttassuk egyik oldalról a másikra egy HTML hivatkozásban?

6. Hogyan küldhetjük tovább a látogatót egy másik oldalra?

7. Melyik függvény alkalmas dátumok formázására?

Feladatok

1. Tekintsünk végig az órában tárgyalt forráskódokon. Találunk olyan megoldásokat, amelyeket saját programjainkban is tudunk majd hasznosítani?