15. ÓRA

Dátumok kezelése

A dátumok annyira mindennapjaink részét képezik, hogy nem is gondolkodunk, amikor velük dolgozunk, naptárunk fortélyainak helyes kezelése azonban a programokat meglehetősen bonyolulttá teheti. Szerencsére a PHP hatékony eszközöket biztosít a dátumszámításhoz, amely megkönnyíti a feladatot.

Ebben a fejezetben a következőkről tanulunk:

- Hogyan kérdezhetjük le a pontos időt és dátumot?
- Hogyan szerezhetünk információt egy dátumról?
- Hogyan formázzunk meg egy dátumot?
- Hogyan ellenőrizzük a dátumok érvényességét?
- Hogyan állítsuk be a dátumot?
- Hogyan írjunk egyszerű naptárprogramot?

A dátum kiderítése a time() függvénnyel

A PHP `time()` függvénye mindent elárul, amire szükségünk lehet a pontos idővel és dátummal kapcsolatban. A függvénynek nincs bemenete (paramétere) és egy egész számot ad vissza, amely egy kicsit nehezen értelmezhető, viszont nagyon informatív.

A `time()` által visszaadott egész szám a greenwichi középidő szerinti 1970. január 1. éjfél óta eltelt másodpercek számát adja meg. Ez a dátum a „UNIX kor kezdete" néven ismeretes, az azóta eltelt másodpercek számát pedig időbélyegnek (néha időbélyegző, time stamp) nevezik. Az időbélyeget a PHP eszközeivel „emberi fogyasztásra alkalmasabb" formára hozhatjuk, de felmerül a kérdés: még ha ilyen nagyszerű eszközök is állnak rendelkezésünkre, az időbélyeg nem feleslegesen túlbonyolított módja a dátum tárolásának? Nem, éppen ellenkezőleg: egyetlen számból rengeteg információt szerezhetünk. És ami még ennél is jobb, az időbélyeggel sokkalta könnyebben lehet számításokat végezni, mint azt gondolnánk.

Képzeljünk csak el egy házi dátumrendszert, amelyben az éveket, hónapokat és a hónapok napjait jegyezzük fel. Most képzeljünk el egy olyan programot, amellyel egy napot kell hozzáadnunk egy adott dátumhoz. Ha ez a dátum éppenséggel 1999. december 31., akkor ahelyett, hogy 1-et adnánk a dátumhoz, olyan programot kellene írni, amely a napot 1-re írná át, a hónapot januárra, az évet pedig 2000-re. Az időbélyegben viszont csak egy napnyi másodpercet kell hozzáadni az aktuális dátumhoz és már készen is vagyunk. Ezt az új számot pedig kedvünk szerint alakíthatjuk át valamilyen barátságosabb formára.

Az időbélyeg átalakítása a getdate() függvénnyel

Most, hogy rendelkezésünkre áll az időbélyeg, át kell alakítanunk, mielőtt megmutatnánk a felhasználónak. A `getdate()` elhagyható bemenete egy időbélyeg, visszatérési értéke pedig egy asszociatív tömb. Ha nem adjuk meg az időbélyeget, a függvény az éppen aktuális időbélyeggel dolgozik, mintha azt a `time()` függvénytől kapta volna. A 15.1. táblázat a `getdate()` által visszaadott tömb elemeit részletezi.

15.1. táblázat A getdate() függvény által visszaadott asszociatív tömb

Kulcs	*Leírás*	*Példa*
`seconds`	A percből eltelt másodpercek száma (0-59)	`28`
`minutes`	Az órából eltelt percek száma (0-59)	`7`
`hours`	A nap órája (0-23)	`12`
`mday`	A hónap napja (1-31)	`20`

15.1. táblázat (folytatás)

wday	A hét napja (0-6)	4
mon	Az év hónapja (1-12)	1
year	Év (négy számjeggyel)	2000
yday	Az év napja (0-365)	19
weekday	A hét napja (névvel)	Thursday
month	Az év hónapja (névvel)	January
0	Az időbélyeg	948370048

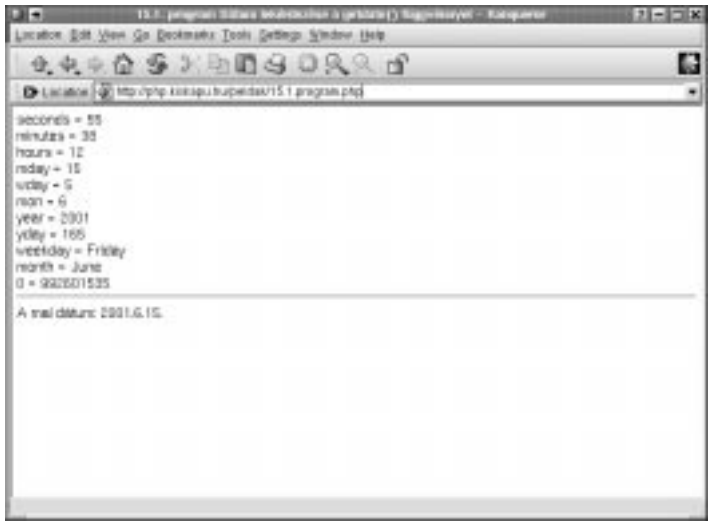
A 15.1. programban a `getdate()` függvénnyel szétbontjuk az időbélyeget, majd a `foreach` utasítással kiírjuk az egyes elemeket. A 15.1. ábrán a `getdate()` függvény egy lehetséges kimenetét láthatjuk. A `getdate()` a helyi időzóna szerinti dátumot adja vissza.

15.1. program Dátum lekérdezése a getdate() függvénnyel

```
 1: <html>
 2: <head>
 3: <title>15.1. program Dátum lekérdezése a getdate()
          függvénnyel</title>
 4: </head>
 5: <body>
 6: <?php
 7: $datum_tomb = getdate(); // nem adtunk meg bemenetet,
                  így a mai dátumot fogja visszaadni
 8: foreach ( $datum_tomb as $kulcs => $ertek )
 9:      {
10:      print "$kulcs = $ertek<br>";
11:      }
12: ?>
13: <hr>
14: <?
15: print "A mai dátum: " $datum_tomb["year"] "
           " $datum_tomb["mon"] "
           " $datum_tomb["mday"] <p>";
16: ?>
17: </body>
18: </html>
```

15.1. ábra

A getdate() függvény használata



Az időbélyeg átalakítása a date() függvénnyel

A `getdate()` függvényt akkor használjuk, amikor a kimenetét képező elemeket szeretnénk használni, a dátumot azonban általában szövegként szeretnénk megjeleníteni. A `date()` függvény a dátumot formázott karakterlánc formájában adja vissza. A `date()` függvény által visszaadott érték formátuma a paraméterként átadható formázó karakterlánccal nagymértékben befolyásolható. A formázó karakterláncon kívül a `date()` függvénynek átadhatjuk az időbélyeget is, bár ez nem kötelező. A 15.2. táblázatban láthatjuk azokat a kódokat, amelyeket a formázó karakterlánc tartalmazhat. A `date()` függvénynek átadott formázóban található minden egyéb szöveg a visszaadott értékben változatlanul meg fog jelenni.

15.2. táblázat A date() által használt formázási kódok.

Formátum	*Leírás*	*Példa*
`a`	`'am'` (délelőtt) vagy `'pm'` (délután), kisbetűvel	`pm`
`A`	`'AM'` (délelőtt) vagy `'PM'` (délután), nagybetűvel	`PM`
`d`	A hónap napja (bevezető nullákkal írt szám)	`05`
`D`	A hét napja (három betűvel)	`Thu`
`F`	A hónap neve	`January`
`h`	Óra (12 órás formátum, bevezető nullákkal)	`03`
`H`	Óra (24 órás formátum, bevezető nullákkal) 05	
`g`	Óra (12 órás formátum, bevezető nullák nélkül)	`3`
`G`	Óra (24 órás formátum, bevezető nullák nélkül)	`5`

15.2. táblázat (folytatás)

Formátum	*Leírás*	*Példa*
i	Perc	47
j	A hónap napja (bevezető nullák nélkül)	5
l	A hét napja (névvel)	Thursday
L	Szökőév ('1' ha igen, '0' ha nem)	1
m	Az év hónapja (számmal, bevezető nullákkal)	01
M	Az év hónapja (három betűvel)	Jan
n	Az év hónapja (számmal, bevezető nullák nélkül) 1	
s	Az óra percei	24
U	Időbélyeg	948372444
y	Év (két számjegy)	00
Y	Év (négy számjegy)	2000
z	Az év napja (0-365)	19
Z	A greenwichi középidőtől való eltérés másodpercben	0

A 15.2. program néhány formátumkódot mutat be.

15.2. program Dátum formázása a date() függvénnyel

```
 1: <html>
 2: <head>
 3: <title>15.2. program Dátum formázása a date()
          függvénnyel</title>
 4: </head>
 5: <body>
 6: <?php
 7: print date("Y.m.d. H:i:s<br>", time());
 8: // 2000.12.10. 14:42:55
 9:
10: $de_du = Array ("am" => "délelőtt", "pm"
          => "délután");
11: $honapok = Array("január", "február", "március",
             "április",
12: "május", "június", "július", "augusztus",
    "szeptember",
13: "október", "november", "december");
```

15.2. program (folytatás)

```
14:
15: $napszak = $de_du[date("a")];
16: $honap = $honapok[date("n")-1];
17:
18: print " Ma " . date("Y ") . $honap .
         date(" j. \\n\ap\j\a v\a\\n, "  );
19: print $napszak . date(" g ór\a i perc");
20: // Ma 2000 december tizedike van,
    délután 2 óra 42 perc.
21: ?>
22: </body>
23: </html>
```

A magyar dátumok formázása kétségtelenül bonyolultabb, mint a belsőleg támogatott angol dátumformátumok, ezért létre kell hoznunk egy tömböt, amely a „délelőtt” és „délután” szavakat tartalmazza, valamint egy másik tömböt a hónapok nevei számára. A `$napszak` egyszerűen a `date()` függvény `"a"` formátumra adott kimenetéből származtatható, a `$honap`-hoz pedig a tömb `date("n")-1`-edik elemét kell vennünk, mivel a tömbök nullától számozódnak.

Bár a formázás bonyolultnak tűnik, nagyon könnyű elvégezni. Ha olyan szöveget szeretnénk a formázáshoz adni, amely formátumkód-betűket tartalmaz, a fordított perjel (`\`) karakterrel jelezhetjük, hogy az utána jövő karakter nem formátumkód. Azon karakterek esetében, amelyek a `\` után válnak vezérlőkarakterré, ezt a `\` elé tett újabb `\` segítségével kell jeleznünk. A `"\n"`-t például ennek megfelelően `"\\n"`-nek kell írnunk, ha egy “n”-t szeretnénk a formázási karaktersorozatba írni. A fenti példában a formátumkarakterek ütközését elkerülendő vettük külön a `$honap` és `$napszak` kiírását, hogy a `date()` függvény ne tekintse ezek karaktereit értelmezendő formátum-meghatározásoknak. A `date()` függvény a helyi időzóna szerint adja vissza a dátumot. Ha azt szeretnénk, hogy greenwichi középidő (Greenwich Mean Time, GMT) szerint kapjuk meg, használjuk a `gmdate()` függvényt, amely egyébként minden más tekintetben ugyanígy működik.

Időbélyeg készítése az mktime() függvénnyel

Már képesek vagyunk az aktuális dátumot használni, de tetszőleges dátumokat még nem tudunk kezelni. Az `mktime()` függvény által visszaadott időbélyeget a `date()` vagy a `getdate()` függvényekkel használhatjuk. Az `mktime()` függvény bemenete hat egész szám, a következő sorrendben:

óra
perc
másodperc
hónap
hónap napja
év

A 15.3. program az `mktime()` függvénnyel állít elő időbélyeget, amit aztán a `date()` függvénnyel használ.

15.3. program Időbélyeg készítése az mktime() függvénnyel

```
 1: <html>
 2: <head>
 3: <title>15.3. program Időbélyeg készítése az mktime()
           függvénnyel </title>
 4: </head>
 5: <body>
 6: <?php
 7: // időbélyeget készít 1999. 05. 01. 12 óra 30 perchez
 8: $ido = mktime( 12, 30, 0, 5, 1, 1999 );
 9: print date("Y.m.d. H:i:s<br>", $ido);
10: // 1999.05.01. 12:30:00
11: ?>
12: </body>
13: </html>
```

Az `mktime()` függvény néhány vagy akár az összes eleme elhagyható, ilyenkor az aktuális időnek megfelelő értékeket használja. Az `mktime()` emellett azokat az értékeket is kiigazítja, amelyek túl vannak a megfelelő értéktartományon, így ha 25-öt adunk meg óra paraméterként, akkor a meghatározott nap, hónap, év paraméterek utáni nap reggeli 1.00-ként fogja kezelni azt. Az `mktime()` függvény kimenete általában adatbázisban vagy fájlban való tároláshoz, illetve dátumszámításokhoz lehet hasznos.

A dátum ellenőrzése a checkdate() függvénnyel

Előfordulhat, hogy a felhasználó által bevitt dátummal kell dolgoznunk. Mielőtt nekikezdenénk a munkának vagy tárolnánk a dátumot, meg kell bizonyosodnunk róla, hogy az érvényes-e. A `checkdate()` függvény három egész számot, a hónapot, a napot, és az évet kéri paraméterként (ebben a sorrendben) és `true` (igaz) értéket ad vissza, ha a hónap 1 és 12 közt van, a nap elfogadható az adott

hónapban és évben (számításba véve a szökőéveket) és az év 0 és 32 767 közé esik. Vigyázzunk azonban, mert előfordulhat, hogy egy dátum ugyan érvényes, de más dátumfüggvények számára mégsem elfogadható. Például a következő sor `true` (igaz) értéket ad vissza:

```
checkdate( 4, 4, 1066 )
```

Viszont ha ezekkel az értékekkel egy dátumot szeretnénk létrehozni, az `mktime()` által visszaadott időbélyeg `-1` lenne. Általános szabályként fogadjuk el, hogy az 1902 előtti évekre nem használjuk az `mktime()` függvényt és az 1970 előttiekre is csak elővigyázatosan.

Egy példa

Próbáljunk meg ezen függvények közül egyetlen példán belül minél többet használni. Készítsünk naptárat, amely egy 1980 és 2010 közötti tetszőleges hónap napjait mutatja. A felhasználó lenyíló menüvel választhatja ki a hónapot és az évet. A program kimenetében az adott hónap dátumai a hét napjainak megfelelően kerülnek elrendezésre. Két globális változót használunk, a `$honap` és az `$ev` változókat, melyek adatait a felhasználó adja meg. A két változó segítségével elkészítjük a meghatározott hónap első napjának időbélyegét. Ha nem adtuk meg az évet vagy a hónapot, vagy megadtuk, de a bevitt adat nem érvényes, a program bemenetének alapbeállításként a folyó év aktuális hónapjának első napját tekintjük.

A felhasználó által bevitt adatok ellenőrzése

Amikor egy felhasználó először látogat el weboldalunkra, valószínűleg nem fogja megadni a bekért adatokat. Erre az `isset()` függvény segítségével készíthetjük fel programunkat. Ez a függvény a `false` (hamis) értéket adja vissza, ha a neki átadott változó nem meghatározott. (Használhatjuk helyette a `checkdate()` függvényt is.) A 15.4. példa azt a kódrészletet tartalmazza, amely ellenőrzi a `$honap` és az `$ev` változókat és egy időbélyeget hoz létre belőlük.

15.4. program A felhasználó által bevitt adatok ellenőrzése a naptárprogram számára

```
1: <?php
2: if ( ! checkdate( $honap, 1, $ev ) )
3:      {
4:      $mostTomb = getdate();
5:      $honap = $mostTomb["mon"];
6:      $ev = $mostTomb["year"];
7:      }
```

15.4. program (folytatás)

```
 8: $kezdet = mktime ( 0, 0, 0, $honap, 1, $ev );
 9: $elsoNapTombje = getdate($kezdet);
10: if ($elsoNapTombje["wday"] == 0)
    { $elsoNapTombje["wday"] = 6; }
11: else { $elsoNapTombje["wday"]--; }
12: ?>
```

A 15.4. program egy nagyobb program részlete, így önmagában nincs kimenete. Az `if` utasításban a `checkdate()` függvényt használjuk a `$honap` és az `$ev` változók ellenőrzésére. Ha ezek nem meghatározottak, a `checkdate()` `false` (hamis) értéket ad vissza, mivel nem hozhatunk létre érvényes dátumot meghatározatlan hónap és év paraméterekből. E megközelítés további haszna, hogy egyúttal a felhasználó által bevitt adat érvényességét is ellenőrizzük.

Ha a dátum nem érvényes, a `getdate()` függvénnyel létrehozunk egy, az aktuális dátumon alapuló asszociatív tömböt. Ezután a tömb `mon` és `year` elemeivel magunk állítjuk be a `$honap` és az `$ev` változók értékeit.

Most hogy megbizonyosodtunk, hogy a program bemenetét képező két változó érvényes adatot tartalmaz, az `mktime()` függvényt használjuk a hónap első napja időbélyegének létrehozására. A későbbiek során még szükségünk lesz ennek az időbélyegnek a napjára és hónapjára, ezért létrehozunk egy `$elsoNapTombje` nevű változót, amely a `getdate()` függvény által visszaadott és ezen az időbélyegen alapuló asszociatív tömb lesz. Végül a magyar héthasználatnak megfelelően át kell alakítanunk a hét napjára vonatkozó értéket. A `getdate()` és más dátumfüggvények a hét első napjának a vasárnapot tekintik. Ezért ha a hét első napján vagyunk (ez kapja a nullás számot), akkor hetedikre írjuk át, egyébként pedig a nap számát eggyel csökkentjük. Így a vasárnap a hét végére kerül, a többi nap pedig eggyel előrébb, azaz a hétfő az első helyen szerepel.

A HTML űrlap létrehozása

Olyan kezelőfelületet kell létrehoznunk, amellyel a felhasználók lekérdezhetik egy hónap és év adatait. Ennek érdekében `SELECT` elemeket fogunk használni. Bár ezeket módosíthatatlan elemekként, a HTML kódban is megadhatnánk, azt is biztosítanunk kell, hogy a lenyíló menük alapbeállítása az adott hónap legyen, ezért a menüket dinamikusan hozzuk létre és csak ott adjuk hozzá a `SELECTED` tulajdonságot az `OPTION` elemhez, ahol szükséges. A 15.5. programban a létrehozott űrlapot láthatjuk.

15.5. program A naptárprogram HTML űrlapja

```
 1: <?php
 2: if ( ! checkdate( $honap, 1, $ev ) )
 3:      {
 4:      $mostTomb = getdate();
 5:      $honap = $mostTomb["mon"];
 6:      $ev = $mostTomb["year"];
 7:      }
 8: $kezdet = mktime ( 0, 0, 0, $honap, 1, $ev );
 9: $elsoNapTombje = getdate($kezdet);
10: if ($elsoNapTombje["wday"] == 0)
       { $elsoNapTombje["wday"] = 6; }
11: else { $elsoNapTombje["wday"]--; }
12: ?>
13: <html>
14: <head>
15: <title><?php print "Naptár:" $elsoNapTombje["year"]
16:           $elsoNapTombje["month"] ?></title>
17: <head>
18: <body>
19: <form method="post">
20: <select name="honap">
21: <?php
22: $honapok = Array ("január", "február", "március",
                     "április",
23: "május", "június", "július", "augusztus",
    "szeptember",
24: "október", "november", "december");
25: for ( $x=1; $x <= count( $honapok ); $x++ )
26:      {
27:      print "\t<option value=\"$x\"";
28:      print ($x == $honap)?" SELECTED":"";
29:      print ">".$honapok[$x-1]."\n";
30:      }
31: ?>
32: </select>
33: <select name="ev">
34: <?php
35: for ( $x=1980; $x<2010; $x++ )
36:      {
37:      print "\t<option";
38:      print ($x == $ev)?" SELECTED":"";
```

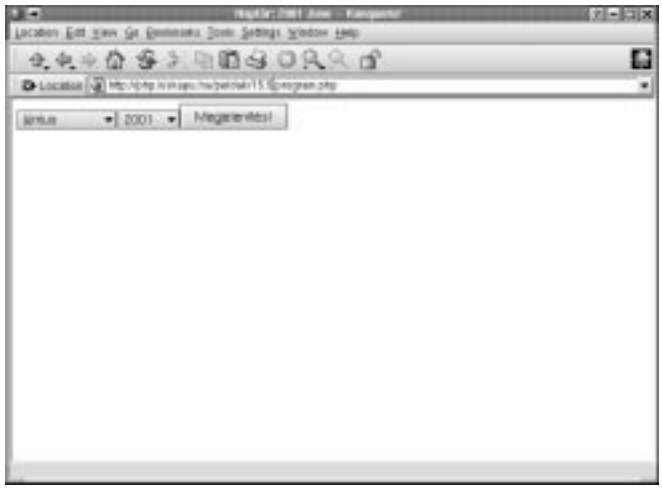
15.5. program (folytatás)

```
39:      print ">$x\n";
40:      }
41: ?>
42: </select>
43: <input type="submit" value="Megjelenítés!">
44: </form>
45: </body>
46: </html>
```

Miután létrehoztuk a `$kezdet` időbélyeget és az `$elsoNapTombje` változót, megírjuk az oldal HTML kódját. Vegyük észre, hogy az `$elsoNapTombje` változót arra használjuk, hogy az évet és a hónapot a `TITLE` (cím) elemhez adhassuk. Az előző példákban a `FORM` elemen belüli `$PHP_SELF` globális változót használtuk arra, hogy biztosítsuk, az űrlap az elküldés után újra megjeleníti magát, e programban viszont azt a tényt használtuk ki, hogy ha egy `FORM` kódból kihagyjuk az `ACTION` paramétert, alapbeállítás szerint ugyanez történik. A lenyíló menük `SELECT` elemének létrehozásához visszalépünk PHP módba, hogy létrehozzuk az egyes `OPTION` címkéket. Először létrehozunk egy `$honapok` nevű tömböt, amely a 12 hónap nevét tartalmazza. Ezen aztán végiglépkedünk egy ciklussal, mindegyikből készítve egy `OPTION` címkét. Ez az egyszerű `SELECT` elem létrehozásának túlbonyolított módja lenne, ha nem kellene közben figyelnünk az `$x` változót (a `for` ciklusváltozóját). Ha az `$x` és a `$honap` megegyezik, a `SELECTED` kifejezést hozzáadjuk az `OPTION` címkéhez, biztosítván ezzel, hogy a megfelelő hónap automatikusan kijelölődik a lap betöltésekor. Hasonló módszerrel írjuk meg az év menüjét is, végül HTML módba visszatérve létrehozzuk a Megjelenítés gombot.

Így egy olyan űrlapot kapunk, amely saját magának elküldi a hónap és év paramétereket, és amelynek vagy az aktuális év és hónap, vagy pedig az előzőleg megadott év és hónap az alapbeállítása. A kész űrlapot a 15.2. ábrán láthatjuk.

15.2. ábra
A naptár űrlapja



A naptár táblázatának létrehozása

Most egy táblázatot kell létrehoznunk, amit a kiválasztott hónap napjaival kell feltöltenünk. Ezt a 15.6. példában követhetjük nyomon, amely a teljes naptárprogramot tartalmazza.

15.6. program A teljes naptárprogram

```
 1: <?php
 2: define("EGYNAP", (60*60*24) );
 3: if ( ! checkdate( $honap, 1, $ev ) )
 4:      {
 5:      $mostTomb = getdate();
 6:      $honap = $mostTomb["mon"];
 7:      $ev = $mostTomb["year"];
 8:      }
 9: $kezdet = mktime ( 0, 0, 0, $honap, 1, $ev );
10: $elsoNapTombje = getdate($kezdet);
11: if ($elsoNapTombje["wday"] == 0)
       { $elsoNapTombje["wday"] = 6; }
12: else { $elsoNapTombje["wday"]--; }
13: ?>
14: <html>
15: <head>
16: <title><?php print "Naptár:" $elsoNapTombje["year"]
17:          $elsoNapTombje["month"] ?></title>
18: <head>
19: <body>
```

15.6. program (folytatás)

```
20: <form method="post">
21: <select name="honap">
22: <?php
23: $honapok = Array ("január", "február", "március",
                      "április",
24: "május", "június", "július", "augusztus",
    "szeptember",
25: "október", "november", "december");
26: for ( $x=1; $x <= count( $honapok ); $x++ )
27:     {
28:     print "\t<option value=\"$x\"";
29:     print ($x == $honap)?" SELECTED":"";
30:     print ">".$honapok[$x-1]."\n";
31:     }
32: ?>
33: </select>
34: <select name="ev">
35: <?php
36: for ( $x=1980; $x<2010; $x++ )
37:     {
38:     print "\t<option";
39:     print ($x == $ev)?" SELECTED":"";
40:     print ">$x\n";
41:     }
42: ?>
43: </select>
44: <input type="submit" value="Megjelenítés!">
45: </form>
46: <p>
47: <?php
48: $napok = Array ("hétfő", "kedd", "szerda",
                    "csütörtök",
49: "péntek", "szombat", "vasárnap");
50: print "<TABLE BORDER=1 CELLPADDING=5>\n";
51: foreach ( $napok as $nap )
52:     print "\t<td><b>$nap</b></td>\n";
53: $kiirando = $kezdet;
54: for ( $szamlalo=0; $szamlalo < (6*7); $szamlalo++ )
55:     {
56:     $napTomb = getdate( $kiirando );
```

15.6. program (folytatás)

```
57:      if ( (($szamlalo) % 7) == 0 )
58:           {
59:           if ( $napTomb["mon"] != $honap )
60:                break;
61:           print "</tr><tr>\n";
62:           }
63:      if ( $szamlalo < $elsoNapTombje["wday"] ||
              $napTomb["mon"] != $honap )
64:           {
65:           print "\t<td><br></td>\n";
66:           }
67:      else
68:           {
69:           print "\t<td>" . $honapok[$napTomb["mon"]-1]
                    " $napTomb["mday"] "</td>\n";
70:           $kiirando += EGYNAP;
71:           }
72:      }
73: print "</tr></table>";
74: ?>
75: </body>
76: </html>
77:
```

Mivel a táblázat fejlécébe a hét napjai kell, hogy kerüljenek, a programot végiglép-tetjük a napok neveinek tömbjén és mindegyiket saját cellájába írjuk. A lényeg a program utolsó `for` ciklusában történik.

Megadjuk a `$szamlalo` változó kezdeti értékét, majd biztosítjuk, hogy a ciklus 42 ismétlés után leáll. Erre azért van szükség, hogy a dátumoknak elegendő cellát hozzunk létre. A ciklusban a `$kiirando` változót a `getdate()` függvénnyel egy dátumtömbbé alakítjuk és az eredményt a `$napTomb` változóba tesszük. Bár a ciklus indításakor a `$kiirando` változó értéke a hónap első napja lesz, az időbélyeget a ciklus minden lefutásakor 24 órával növeljük.

A maradékos osztás segítségével megnézzük, hogy a `$szamlalo` változó értéke osztható-e 7-tel. Az ehhez az if utasításhoz tartozó kódrész csak akkor fut le, ha a `$szamlalo` 0 vagy a 7 többszöröse. Így eldönthetjük, hogy befejezzük-e a ciklust vagy új sort kezdjünk.

Ha látjuk, hogy még a ciklus első lefutásában vagy egy cellasor végénél vagyunk, elvégzünk még egy ellenőrzést: megvizsgáljuk, hogy a `$napTomb` `mon` (hónapszám) eleme nem egyenlő-e a `$honap` változóval. Ha nem egyenlő, a ciklust befejezhetjük. Emlékezzünk vissza, hogy a `$napTomb`-ben ugyanaz az időpont van, mint ami a `$kezdet`-ben, ami viszont a hónapnak pontosan az a része, amelyet megjelenítünk. Ha a `$kezdet` túljut az aktuális hónapon, a `$napTomb[mon]` más számot fog tartalmazni, mint a felhasználó által megadott `$honap` szám. A maradékosztás igazolta, hogy éppen egy sor végén állunk: ha új hónapban járunk, elhagyhatjuk a ciklust, ha viszont a sor végére értünk és még mindig ugyanabban a hónapban vagyunk, befejezzük a sort és újat kezdünk.

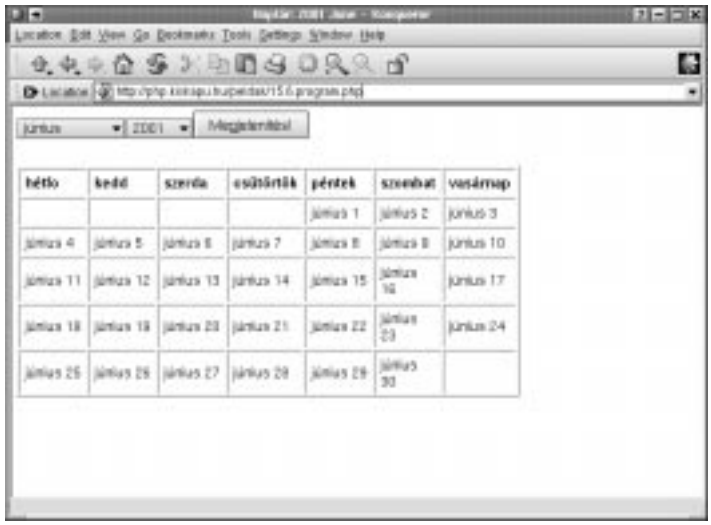
A következő `if` utasításban azt határozzuk meg, hogy írjunk-e szöveget a cellába. Nem minden hónap kezdődik hétfővel, így könnyen meglehet, hogy egy-két üres cellával indítunk. Emellett kevés hónap végződik sor végén, így az is valószínű, hogy mielőtt lezárnánk a táblázatot, egy pár üres cellát is kell írnunk. A hónap első napjának adatait az `$elsoNapTombje` változóban találjuk. Ebből az `$elsoNapTombje["wday"]` kifejezés segítségével kideríthető, hogy az adott dátum hányadik nap a héten. Ha a `$szamlalo` kisebb, mint ez a szám, akkor tudjuk, hogy még nem értük el azt a cellát, amelybe már írhatnánk. Ugyanezt felhasználva ha a `$honap` változó már nem egyenlő a `$napTomb["mon"]` változóval, tudhatjuk, hogy elértük a hónap végét (de a sor végét még nem). Mindkét esetben egy üres cellát írunk ki a böngészőbe.

A végső `else` utasításnál jön a feladat érdekes része. Azzal már tisztában vagyunk, hogy a kiírandó hónapban vagyunk és az aktuális nap oszlopa megegyezik az `$elsoNapTombje` változóban tárolt számmal. Most kerül felhasználásra a ciklus korábbi részében létrehozott `$napTomb` asszociatív tömb, valamint a hónapok magyar neveit tartalmazó `$honapok` tömb, hogy kiírjuk a hónap nevét és napját egy cellába.

Végül növeljük a `$kiirando` változót, amely az időbélyeget tartalmazza. Egyszerűen hozzáadjuk egy nap másodperceinek számát (ezt az értéket a program elején határoztuk meg) és már újra is kezdhetjük a ciklust a `$kiirando` változó egy új értékével. A 15.3. ábrán a program egy lehetséges eredményét láthatjuk.

15.3. ábra

A naptárprogram



Összefoglalás

Ezen az órán megtanultuk, hogyan használjuk a `time()` függvényt az aktuális időbélyeg megszerzésére. Megtanultuk, hogyan használjuk az időbélyegből a dátum részeinek kinyerésére a `getdate()` függvényt és az időbélyeget formázott szöveggé alakító `date()` függvényt. Arról is tanultunk, hogyan hozzunk létre időbélyeget az `mktime()` függvénnyel. Megtanultuk ellenőrizni a dátum érvényességét a `checkdate()` függvénnyel, magyar elvárások szerint formáztunk dátumokat, végül tanulmányoztunk egy példaprogramot, amely az említett függvényeket alkalmazta.

Kérdések és válaszok

Vannak olyan függvények, amelyekkel a különböző naptárak között váltogatni lehet?

Igen, a PHP naptárak teljes sorozatát biztosítja. Ezekről a hivatalos PHP kézikönyvben olvashatunk, a `http://www.php.net/manual/ref.calendar.php` címen.

Műhely

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

Kvíz

1. Hogyan kapunk meg egy UNIX időbélyeget, amely a pontos dátumot és időt jelöli?

2. Melyik az a függvény, amelynek bemenete egy időbélyeg és egy, az adott dátumot jelképező asszociatív tömböt ad vissza?

3. Mely függvényt használnánk a dátum formázására?

4. Hogyan kapjuk meg egy tetszőleges dátum időbélyegét?

5. Melyik függvényt használjuk egy dátum érvényességének ellenőrzésére?

Feladatok

1. Készítsünk egy születésnapig visszaszámláló programot. Űrlappal lehessen megadni az év, hó és nap értékeit, kimenete pedig egy üzenet legyen, amely közli a felhasználóval, hogy hány nap, óra, perc, másodperc van még hátra a nagy napig.