PHP alapok

Informatika 2 – Hallgatói segédlet

1. Szükséges előismeretek

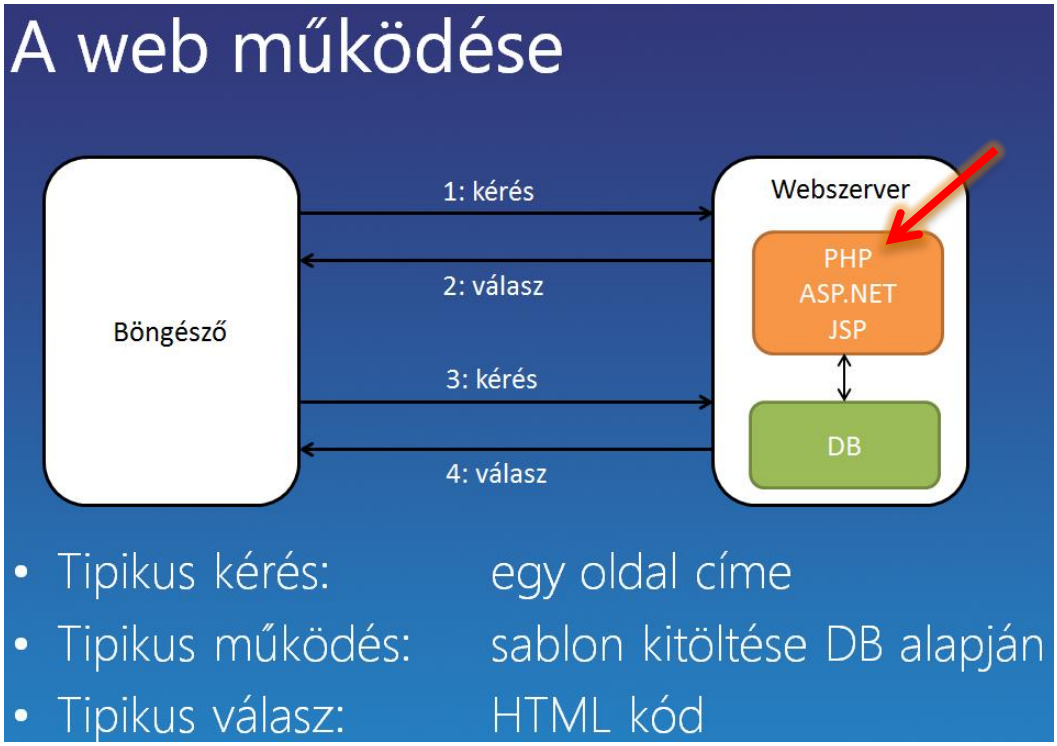
A gyakorlat célja a dinamikus weboldalak működésének bemutatása. A feladatok építenek a HTML és PHP nyelvek ismeretére. Ez a hallgatói segédlet a PHP nyelvbe ad rövid betekintést.

További hasznos segédanyagok:

- Matt Zandstra: Tanuljuk meg a PHP használatát 24 óra alatt
- PHP referencia és fejlesztői portál: http://php.net/
- Windows alá telepíthető webszerver és adatbázis szerver:
  - http://www.apachefriends.org/en/xampp.html
  - http://www.wampserver.com/en/
- Egy PHP tutorial: http://tutsplus.com/tutorials/?q=true&filter_topic=41

2. A PHP helye a weben

A PHP egy szerveroldali nyelv. Feladata, hogy a böngészőktől érkező webes kérésekre válaszul HTML kódot generáljon. Ez tipikusan valamilyen HTML sablon kitöltését jelenti egy adatbázisból felolvasott értékek alapján.



3. A PHP nyelv címszavakban

- Szerveroldali szkript nyelv – azaz a webszerveren történik a végrehajtása, nem pedig a böngészőben.
- Gyengén típusos – azaz a változóinak nincs explicit típusa (nem kell megadnunk, hogy pl. int vagy double típusú változóval szeretnénk dolgozni).
- Neve a "**P**HP **H**ypertext **P**reprocessor" rekurzív rövidítésből jön.
- Egy PHP forrásfájl tipikus kiterjesztése a *.php*, *.php3* vagy *.phtml*.
- A PHP forrásfájlokat a webszerver értelmezi és futtatja - a generált HTML kimenetet pedig visszaküldi a böngészőnek.
- Sokféle adatbázissal képes együttműködni (MySQL, MSSQL, PosgreSQL, Oracle, stb...)
- Ingyenes, nyílt forráskódú platform, emiatt szinte minden tárhelyszolgáltatónál elérhető.
- Windowson és Linuxon is használható.
- MySQL-el kombinálva teljes értékű webalkalmazás készíthető a segítségével.
- Viszonylag gyorsan tanulható.
- Egy PHP forrásfájl vegyesen tartalmazhat szöveget, HTML elemeket és forráskódot.

4. A PHP nyelv használata

Minden PHP forrásfájlban vegyesen fordulhat elő konstans szöveg és futtatandó PHP nyelvű forráskód. A forráskódot minden esetben <?php és ?> jelek közés kell zárni:

```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Első PHP oldalam</title>
    </head>
    <body>
        <?php
            echo "Ez az első PHP oldalam";
        ?>
    </body>
</html>
```

A PHP állományt a webszerver gyökérkönyvtárába vagy az alá kell menteni (például C:\xampp\htdocs\info2\index.php). Ezután a PHP szkript által generált eredményét egy böngészőben nézhetjük meg (például a http://localhost:8080/info2/index.php címen).

Ha változtattunk a PHP programunkon, akkor annak eredménye mindig csak azután lesz látható, hogy a forrásfájlt elmentettük és a böngészőt frissítettük (ebben a sorrendben).

4.1. Gyengén típusos nyelv.

PHP-ban a változókat bárhol deklarálhatjuk, a tartalmuknak megfelelő típussal jönnek létre, ami később dinamikusan változhat. Ha olyan változót próbálnuk használni ami még nem létezik, akkor magától létrejön. A változó nevek dollár jellel ($) kezdődnek, második karakterük nem lehet szám (csak betű vagy alulvonás karakter), és nem lehet benne szóköz. Például $welcome = "Első PHP oldalam";

Próbaképpen hozzunk létre három különböző típusú változót, majd a var_dump($valtozo) függvénnyel irassuk ki a tulajdonságaikat!

```
<body>
   <?php
        $szoveg = "Első PHP oldalam";
        $szam = 12;
        $logikai_ertek = false;

        var_dump($szoveg);
        var_dump($szam);
        var_dump($logikai_ertek);
    ?>
</body>
```

Eredmény:

```
string 'Első PHP oldalam' (length=17)

int 12

boolean false
```

Java-ban ez így nézne ki:

```
String szoveg = “Első PHP oldalam”;
int szam = 12;
boolean logikai ertek = false;
```

4.2. Műveletek Stringekkel

A PHP nyelv elsődleges célja szöveges (HTML nyelvű) kimenet generálása, ezért a nyelv gazdag sztringműveletekben. Ezek közül a legfontosabbak:

- Összefűzés pont karakterrel

```
$elso = "Hello";
$masodik = "World";
$szoveg = $elso . " " . $masodik . "!";
echo $szoveg; // → "Hello World!"
```

- Szöveg hossza: **strlen**($szoveg)

```
echo strlen($szoveg); // → 12
```

- Szövegrész keresés: **strpos**($miben, $mit). Ha tartalmazza, akkor az első találat indexét adja vissza, ellenkező esetben **false**-t

```
echo strpos($szoveg, "World"); // → 6
```

4.3. Operátorok

A PHP nyelvben használható operátorok hasonlítanak C nyelvű társaikhoz, de van néhány eltérés:

- Minden “szokásos” operátor használható (+, -, *, /, %)
- Gyengén típusos nyelv, így egy változó típusa változhat a futás során.

- Lehetőség van típus szerinti egyezést is ellenőrizni az === operátorral (párja: !==). Nem csak primitív típusokra, hanem pl. asszociatív tömbökre is használható:

```
$x == $y  // ← igaz, ha egyezik az érték, pl. 5==8 hamis, 5=="5" igaz
$x === $y // ← igaz, ha egyezik az érték és a típus
          // pl. 5==="5" hamis, 5===5 igaz
```

- PHP-ban megszokott dolog, hogy egy függvény különböző típusú visszatérési értékkel rendelkezik, például adatbázis táblában keresés a rekordot adja vissza találat esetén, egyébként null-t. Ilyenkor jól jön visszaadott érték típus ellenőrzése az === oprátorral.
- Logikai operátorok: vagy: ||, és: &&

4.4. Vezérlési szerkezetek

- **if, if..else** ugyanúgy, mint más nyelvekben

```
$nap = "péntek"; // hét aktuális napjának lekérése
if($nap == "szombat" || $nap == "vasárnap"){
     echo "Jó hétvégét!";
}
else{
     echo "Jó napot!";
}
```

- Többszörös ág esetén: **if..elseif..else**

```
$nap = "péntek"; // hét aktuális napjának lekérése
if($nap == "szombat" || $nap == "vasárnap"){
   echo "Jó hétvégét!";
}
elseif($nap == "péntek"){
   echo "Mindjárt hétvége!";
}
else{
   echo "Jó napot!";
}
```

- **switch** működik Stringekre is!

```
$nap = "péntek"; // hét aktuális napjának lekérése
switch($nap){
   case "hétfő": // ha hétfő van
      echo "A hét első napja";
   break;
   case "szombat": // ha szombat
   case "vasárnap": // vagy vasárnap
      echo "Hétvége! ";
   break;
   default: // minden más esetben
      echo "Sima hétköznap";
   break;
}
```

- **for**: ugyanúgy mint a többi nyelvben

```
for($i=0 ; $i<15 ; $i++){
     echo $i." | ";
}
```

Kimenet:

```
0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
```

- **foreach** – lásd tömbök

4.5. Tömbök

A PHP tömbkezelése sokkal kifinomultabb a C nyelvénél: a PHP tömbök nem csak pozicióval hanem tetszőleges típusú értékekkel címezhetők. Ebben az esetben asszociatív tömbökről beszélünk. Az asszociatív tömbök jól használhatók kulcs-érték párok tárolására:

```
$config[ "email" ] = "webmaster@aut.bme.hu";
$config[ "logfile" ] = "C:\log.txt";
```

Gyakran használunk többdimenziós asszociatív tömböket struktúrált adatok tárolására.

Tömb előnyei:

- Egy változóban több érték
- Kulcs-érték párkét tárolható
- Tetszőlegesen sok dimenziós lehet
- Asszociatív (szöveges kulcsú) is engedélyezett

Jellemzői PHP-ban:

- Indexelés szögletes zárójelben, pl. echo $autok[2];
- Az index 0-ról indul
- Ha nem írunk indexet, akkor a kövezkező elemre mutat a pointer, lásd numerikus tömbök
- Nem kötelező deklarálni, az első értékadáskor létrejön, lásd numerikus tömbök

Tömbök típusai

- **Numerikus** – a kulcsok a tömb elemek indexei

```
$targyak = array(„Info 2","Webfejlesztés","Digitális technika");
```

VAGY

```
$targyak[0] ="Info 2";
// NEM deklaráltuk, futásidőben létrejön ha elkezdjük használni
$targyak[1] = $targyak = array(„Info 2","Webfejlesztés","Digitális
     technika");
```

VAGY

```
$targyak[0] ="Info 2";
// NEM deklaráltuk, futásidőben létrejön ha elkezdjük használni
$targyak[1] ="Webfejlesztés";
$targyak[2] ="Digitális technika";
```

VAGY

```
$targyak[] ="Info 2"; // nem írtunk indexet, ez lesz a 0. elem
$targyak[] ="Webfejlesztés"; // 1. elem
$targyak[] ="Digitális technika"; // 2. elem
Webfejlesztés";
$targyak[2] ="Digitális technika";
```

VAGY

```
$targyak[] ="Info 2"; // nem írtunk indexet, ez lesz a 0. elem
$targyak[] ="Webfejlesztés"; // 1. elem
$targyak[] ="Digitális technika"; // 2. elem
```

- **Asszociatív** – a kulcsok nem indexek, hanem Stringek

```
$targyak = array(
     "BMEVIAUA203" => "Info 2",
     "BMEVIAUAV08" => "Webfejlesztés",
     "BMEVIMIA102" => "Digitális technika"
);
```

VAGY

```
$targyak["BMEVIAUA203"] = "Info 2";
$targyak["BMEVIAUAV08"] = "Webfejlesztés";
$targyak["BMEVIMIA102"] = "Digitális technika";
```

- **Többdimenziós** – olyan tömb ami további tömböket tartalmaz

```
$bme_targyak  =  array(
     "BMEVIAUA203" => "Info 2",
     "BMEVIAUAV08" => "Webfejlesztés",
     "BMEVIMIA102" => "Digitális technika"
);

$elte_targyak = array(
     "ELTE123" => "Analízis 1",
     "ELTE345" => "Analízis 2",
     "ELTE2553443" => "Diszkrét matematika"
);

$ossz_targyak = array(
     "BME" => $bme_targyak,
     "ELTE" => $elte_targyak
);

print_r($ossz_targyak);
```

Kimenet:

```
Array
(
    [BME] => Array
        (
            [BMEVIAUA203] => Info 2
            [BMEVIAUV08] => Webfejlesztés
            [BMEVIMIA102] => Digitális technika
        )

    [ELTE] => Array
        (
            [ELTE123] => Analízis 1
            [ELTE345] => Analízis 2
            [ELTE2553443] => Diszkrét matematika
        )

)
```

Tömb elemeinek végigolvasása foreach ciklussal:

```
foreach($bme_targyak as $targykod => $targynev){
     echo "<b>".$targykod."</b>: " . $targynev . "<br/>";
}
```

Kimenet:

BMEVIAUA203: Info 2
BMEVIAUV08: Webfejlesztés
BMEVIMIA102: Digitális technika

Önálló feladat: Listázza ki az **`$ossz_targyak`** változó tartalmát **foreach** ciklus segítségével! Az elvárt kimenet böngészőben:

Egyetem: BME

BMEVIAUA203: Info 2
BMEVIAUV08: Webfejlesztés
BMEVIMIA102: Digitális technika

Egyetem: ELTE

ELTE123: Analízis 1
ELTE345: Analízis 2
ELTE2553443: Diszkrét matematika

4.5.1. Hasznos tömbökkel kapcsolatos függvények:

- **print_r**($tomb): tömb tartalmának kiírása formázott szövegként
- **explode**($elvalaszto, $string): egy stringet bont darabokra, a részeket tömbben adja vissza.

```
$string_elemei = explode(", ","Első, második, harmadik");
print_r($string_elemei);
```

Kimenet:

```
Array
(
    [0] => Első
    [1] => második
    [2] => harmadik
)
```

- **implode**($ragaszto, $reszek): az explode() párja, tömb elemei ragasztja össze egy String-é

```
$keywords = array("webfejlesztés", "alapoktól", "CMS", "PHP");
echo implode(", ", $keywords);
```

Kimenet: “webfejlesztés, alapoktól, CMS, PHP”

- **array_keys**($tomb): visszaadja az asszociatív tömb kulcsait egy tömbben

```
$kulcsok = array_keys($bme_targyak);
print_r($kulcsok);
```

Kimenet:

```
Array
(
    [0] => BMEVIAUA203
    [1] => BMEVIAUV08
    [2] => BMEVIMIA102
)
```

- **in_array**($mit, $tomb): Megvizsgálja, hogy a $mit szerepel-e értékként a $tomb-ben. Visszatérési értéke true vagy false.

```
$mit = "dió";
$tomb = array("körte", "alma", "dió");
if(in_array($mit, $tomb))
     echo "Benne van a " . $mit;
else
     echo "Nincs benne a " . $mit;
```

Kimenet: “Benne van a dió”

4.6. Függvények

PHP-ban nagyon egyszerűen hozhatunk létre és hívhatunk meg függvényeket. Egy függvénynek tetszőleges mennyiségű bemenő paramétere lehet és lehet visszatérési értéke is, de ez nem kötelező. A függvények használatát leggyorsabban egy példán keresztül érhetjük meg:

```
<!DOCTYPE html>
<html>
     <head></head>
     <body>
     <?php
       /* függvények elkészítése */
       function print_hello() {
          print( "Szia sztok! <br />" );
       }

       function return_hello( $name ) {
          return "Hello " . $name . "!";
```

```
        }

        /* függvények használata */
        print_hello();
        print( return_hello( "világ" ) );
    ?>
    </body>
</html>
```

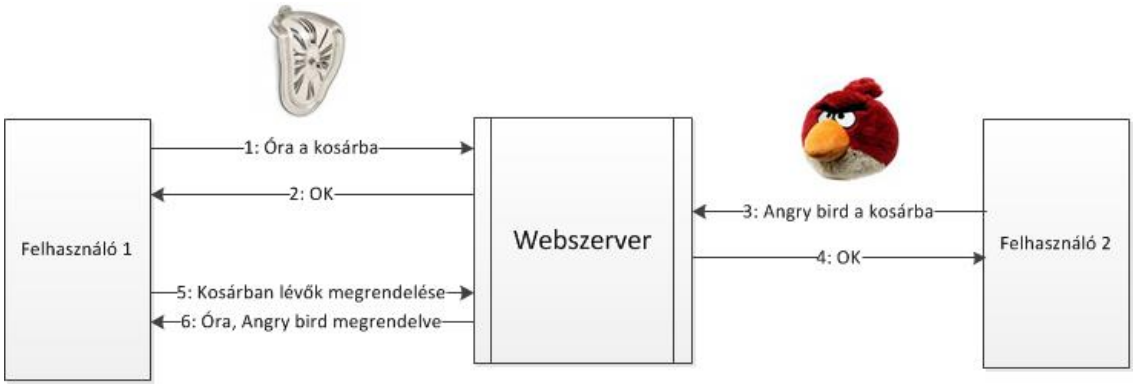
Kimenet:

Sziasztok!

Hello világ!

4.7. Session kezelés

A HTTP alapvetően kapcsolatmentes protokoll, azaz a kérések kiszolgálása egymástól teljesen függetlenül történik – például nem tudjuk hogy az előző kérés melyik felhasználótól származik.



1. ábra: **Mi a hiba?**

Ezt érzékeli a webszerver kapcsolatmentes protokoll esetén:

- Óra a kosárba
- Angry bird a kosárba
- Megrendelés → Óra és Angry bird megrendelése 

Ekkor hibásan a megrendelésbe belerakja a madarat is, pedig annak a másik felhasználóhoz kéne tartoznia! Megoldás: felhasználói munkamenet (session) azonosítása és követése.

Használata:

termeklista.php:

```
session_start();
$_SESSION["userId"] = 1; // speciális változó
$_SESSION["kosar"][] = "Óra";
```

megrendeles.php:

```
session_start();
$kosar = $_SESSION["kosar"]; // = array("Óra")
```

5. Adatok fogadása és küldése

5.1. Űrlap adatok kezelése

Kétféle módon juthat el egy HTML űrlap mezőinek tartalma a webszerverre (akár PHP, akár más platformot futtat):

- GET: Az URL-ben generálódnak paraméterek, pl.

  `login.php?email=tivadar@gmail.com&jelszo=alma`

- POST: URL-től függetlenül egy asszociatív tömbben utaznak az adatok

HTML oldali beállítása:

```
<form action="login.php" method="get|post">
   E-mail cím: <input type="text" name="email" /><br/>
   Jelszó: <input type="password" name="jelszo" /><br/>
   <input type="submit" value="Bejelentkezés!" />
</form>
```

A kapott adat feldolgozására PHP oldalon két különleges tömböt használhatunk. Amennyiben az űrlapon *method="get"*-et állítottunk be, akkor szerver oldalon a *$_GET* asszociatív tömbből olvashatjuk ki az adatokat. A tömb kulcsai az űrlapon lévő input mezők *name* attribúrumainak értékei, például a fenti HTML form esetén az e-mail címet a *$_GET["email"]*, a jelszót pedig a *$_GET["jelszo"]* tartalmazza.

Ugyanez *method="post"* esetén: *$_POST["email"]* és *$_POST["jelszo"]*.

GET-ben, azaz URL-be belefűzve nem csak űrlap elküldés során adhatunk értéket egy szerver oldali kódnak, hanem például linkek megnyitásával is, így működik a Google kereső is.

```
$email = $_GET["email"];
$jelszo = $_GET["jelszo"];

Ha az űrlapon POST-ra állítottuk a method attribútumot:
$email = $_POST["email"];
$jelszo = $_POST["jelszo"];

A $_GET és a $_POST tömböket egyesítve tartalmazza a $_REQUEST:
$email = $_REQUEST["email"];
$jelszo = $_REQUEST["jelszo"];
```

6. MySQL

A szerver oldali fejlesztés szinte mindig együtt jár adatbázis használatával, melyek közül a legnépszerűbb a MySQL. A PHP nyelv lehetőséget teremt az ilyen adatbázisok teljeskörű elérésére és manipulálására. Az adatbázissal való együttműködés során az alábbi parancsokat (függvényeket) használjuk a leggyakrabban:

- mysql_connect: kapcsolatot épít fel a PHP program és egy adott szerveren futó MySQL adatbázisszerver között.
- mysql_select_db: beállítja, hogy a további SQL parancsok mely logikai adatbázison fussanak le. Működése az SQL nyelv USE kulcsszavához hasonló.

- mysql_query: segítségével egy SQL utasítást futtathatunk.
- mysql_real_escape_string: a paraméterül kapott változót olyan formátumúra hozza, hogy azt biztonsággal lehessen használni SQL parancsokban változóként. Feladata az sql injection típusú támadások elleni védelem, **használata létfontosságú**!

```
$query = sprintf("INSERT INTO telefonkonyv( nev ) VALUES( '%s' )",
     mysql_real_escape_string($nev) );
```

- mysql_fetch_row: egy SQL parancs eredményén(például a lekérdezett könyvek listáján) haladhatunk végig a segítségével. Az egyes rekordokban található adatmezőkre a mező (adatbázis oszlop) SQL lekérdezésbeli sorszámával hivatkozhatunk.
- mysql_fetch_assoc: egy SQL parancs eredményén haladhatunk végig a segítségével. Az egyes rekordokban található adatmezőkre a mező nevével hivatkozhatunk.
- mysql_fetch_array: egy SQL parancs eredményén haladhatunk végig a segítségével. Az egyes rekordokban található adatmezőkre a mező sorzsámával és nevével is hivatkozhatunk.
- adatmezőkre a mező (adatbázis oszlop) sorszámával hivatkozhatunk.
- myslq_free_result: felszabadítja egy SQL lekérdezés eredménye által lefoglalt memóriát. Szerepe a C nyelvből ismert *free*-hez hasonló, de meghívása nem kötelező (általában nem kritikus): a lefoglat memóriaterület a PHP szkript lefutása után automatikusan felszabadul.
- mysql_close: bontja az adatbázsikapcsolatot.

Az alábbi példában a *localhost* gépen futó MySQL adatbázis *konyvtar* nevű logikai adatbázisának a *konyv* táblájából kérdezzük le a nyilvántartott könyvek azonosítóját és címét:

```
<?php
   mysql_connect("localhost", "mysql_user", "mysql_password")
        or die("Kapcsolódási hiba: " . mysql_error());
   mysql_select_db("konyvtar");

   $result = mysql_query("SELECT id, cim FROM konyv");

   while ($row = mysql_fetch_array($result)) {
       printf("ID: %s  Cím: %s", $row["id"], $row["cim"]);
   }

   mysql_free_result($result);
   mysql_close();
?>
```

Az alábbi példa név-telefonszám párosokat ment el a telefonkonyv adatbázis phonebook táblájába:

```
<!DOCTYPE html>
<html>
    <head></head>
    <body>
        <form method="post">
            Név: <input type="text" name="nev" />
            <br />
            Telefonszám: <input type="text" name="szam" />
            <br />
            <input type="submit" name="ok" value="ok" />
        </form>
```

```
    <?php
        if( isset( $_POST[ "ok" ] ) ) {
            $name = $_POST[ "nev" ];
            $number = $_POST[ "szam" ];

            mysql_connect( "localhost", "mysql_user", "mysql_password" );
            mysql_select_db( "telefonkonyv" );
            $cmd = "INSERT INTO phonebook ( name, number ) VALUES ( '"
                      . mysql_real_escape_string($name) . "', '"
                      . mysql_real_escape_string($number) . "' );";
            mysql_query( $cmd );
            mysql_close();
        }
    ?>
    </body>
</html>
```

7. MySQLi

A mysql_connect függvény egy globális változóba menti el a MySQL kapcsolat adatait. Innentől kezdve minden további mysql_ prefixű függvény ezt az alapértelmezett kapcsolatot fogja használni. Ez a módszer nem túl közkedvelt, ezért a fenti MySQL könyvtárat és annak függvényeit valójában nem szokták használni, helyette a MySQLi könyvtárat (http://php.net/manual/en/book.mysqli.php) támogatják. Ennek függvényei megegyeznek a korábban ismertetettel néhány fontos különbségtől eltekintve:

- A függvények prefixe nem mysql_, hanem mysqli_. Tehát a függvénynevek helyesen: mysqli_connect, mysqli_select_db, mysqli_query stb.
- A mysqli_connect nem egy alapértelmezett kapcsolatot kezel, hanem létrehoz egy új kapcsolat (vagy link) objektumot és visszatérési értékként visszaadja. Például:
  $link = mysqi_connect(("localhost", "mysql_user", "mysql_password");
- A mysqli_connect kivételével minden további függvény első paramétere a korábban létrehozott kapcsolat objektum lesz. Például:
  mysqli_select_db($link, ”konyvtar”);
  $result = mysqli_query($link, "SELECT id, cim FROM konyv");

A fenti különbségektől eltekintve a mysqli_ prefixű függvények azonos paraméterezésűek és azonosan működnek, mint a korábbiakban ismertetett függvények.

8. Több PHP fájl használata, hivatkozás más fájlokra

Azokat a programrészeket, amiket gyakran használunk – például az adatbázishoz történő kapcsolódást – érdemes csak egyszer elkészíteni és utána mindig ezt az implementációt használni. Ennek a legegyszerűbb módja, ha ezekre a részekre egy függvényt készítünk és ezt egy önálló PHP fájlba (pl. *seged.php*) tesszük. Később ha szükség van erre a függvényre, akkor csak betöltjük az őt tartalmazó fájlt és használjuk a függvényt. Ezt a betöltést többféleképpen is megtehetjük, de a legegyszerűbb és legbiztonságosabb módszer a *require_once("seged.php")* utasítás használata.

Az alábbi példa az adatbázsiműveleteket szervezi ki egy újrahasznosítható segédfájlba:

db-helper.php:

```
<?php
  $db_server   = "localhost";
  $db_user     = "mysql_user";
  $db_password = "mysql_password";
  $db_default  = "telefonkonyv";

  function my_connect() {
    global $db_server, $db_user, $db_password, $db_default;
    mysql_connect( $db_server, $db_user, $db_password );
    mysql_select_db( $db_default );
  }

  function my_insert( $nev, $szam ) {
    $cmd = "INSERT INTO phonebook ( name, number ) VALUES ( '"
           . mysql_real_escape_string($name) . "', '"
           . mysql_real_escape_string($number) . "' );";
    mysql_query( $cmd );
  }

  function my_close() { mysql_close(); }
?>
```

adatbevitel.php:

```
<!DOCTYPE html>
<html>
  <head></head>
  <body>
    <form method="post">
      Név: <input type="text" name="nev" />
      <br />
      Telefonszám: <input type="text" name="szam" />
      <br />
      <input type="submit" name="ok" value="ok" />
    </form>
  <?php
    require_once( "db-helper.php" );

    if( isset( $_POST[ "ok" ] ) ) {
      my_connect();
      my_insert( $_POST["nev"], $_POST["szam"] );
      my_close();
    }
  ?>
  </body>
</html>
```

A másik gyakori eset, hogy az oldalaink közös részeit (például a menüt) helyezzük egy külső fájlba és utána az egyes oldalainkon a menüt mindig ebből a fájlból töltjük be. A legegyszerűbb ilyenkor az include utasítás használata. Erre mutat példát az alábbi kódrészlet:

header.php:

```
<!DOCTYPE html>
<html>
  <head></head>
  <body>
    <a href="home.php">Kezdőoldal</a> |
    <a href="blog.php">Blog</a> |
    <a href="contact.php">Kapcsolat</a>
```

footer.php:

```
  </body>
</html>
```

home.php:

```
<?php
     include("header.php");
     <br />

     // ide jöhet az oldal tartalma

     include("footer.php");
?>
```

9. Objektumorientált PHP

A PHP támogatja az objektum orientált fejlesztést. Fontos tudni, hogy a láthatósági attribútumok (private, protected, public) csak a nyelv 5. főverziójától kezdve elérhetők, PHP4 esetén még nem használhatók. A nyelv támogatja az öröklést (csak egyszerest, többszöröst nem) és interfészek (akár több is) definiálását, megvalósítását.

Példa:

```
class User{
    var $userId, $email, $password, $userName;

    function __construct($userId, $email, $password) {
        $this->userId = $userId;
        $this->email = $email;
        $this->password = $password;
    }
}
```

Ebben a tárgyban nem használjuk ki a PHP objektumorientált lehetőségeit.

10. Ismétlő kérdések

- Milyen különbségek vannak a PHP és egy erősen típusos nyelv (pl. Java) típuskezelése között?
- Mire jó a *$_REQUEST* tömb?
- Írjon egy PHP függvényt, mely kiszámolja a paraméterül kapott szám faktoriálisát és az eredményt kiírja.
- Írjon kódot, mely kapcsolódik a localhost gépen található MySQL adatbázsszerverhez és utána bontja a kapcsolatot.
- Készítsen el egy PHP oldalt, mely listázza az *autok(id, rendszam, szin)* adatbázistábla minden sorát.