

A PHP nyelv (webes alkalmazásokra)



Rigó Ernő
rigo@sztaki.hu
+36-1-2796222





Licensz

Jelen dokumentum a CC by-nc-sa licenc feltételeinek megfelelően szabadon felhasználható.

<http://creativecommons.org/licenses/by-nc-sa/2.5/hu>





Mivel fogunk megismerkedni?

- a PHP nyelv alapjai, helye a világban
- a PHP alkalmazása weboldalak kialakításában
- dinamikus megjelenítésű weblapok készítése
- webes adatbeviteli felületek tervezése
- egyszerű webes alkalmazások elkészítése
 - hitelesítés
 - adatbázisok használata
 - munkamenet-követés
 - stb...



Tematika

- Webes architektúrák és a PHP (N1, T1)
- Ismerkedés a munkakörnyezettel (N1, T1)
- A PHP nyelv alapjai (N1, T1)
- A PHP API áttekintése (N2, T1)
- PHP tervezési minták (N3-4, T2)



Számonkérés

- Egyik lehetőség: Vizsga
 - 10 kérdéses beugró (15 perc, min. 5 helyes válasz)
 - rövid PHP program írása (45 perc) és bemutatása
- Másik lehetőség: Önálló feladat
 - 10 kérdéses beugró (15 perc, min. 5 helyes válasz)
 - hosszabb PHP program írása és bemutatása
 - szándékot legkésőbb az utolsó gyakorlaton bejelenteni!



Segédanyagok, kapcsolat

- a gyakorlatokon a kivetített anyag alapján haladunk
- a letölthető példafájlok ismerete fontos
 - <http://www.tricon.hu/~mcree/php>
- a legfontosabb referencia a hivatalos PHP kézikönyv
 - <http://hu.php.net>
- kérek mindenkit, küldje el az email címét:
 - rigo@sztaki.hu
 - subject/téma: „PHP GYAK” (fontos!)





Webes architektúrák és a PHP

„Bev1” blokk

nappalisok: 1. alkalom

távoktatás: 1. alkalom



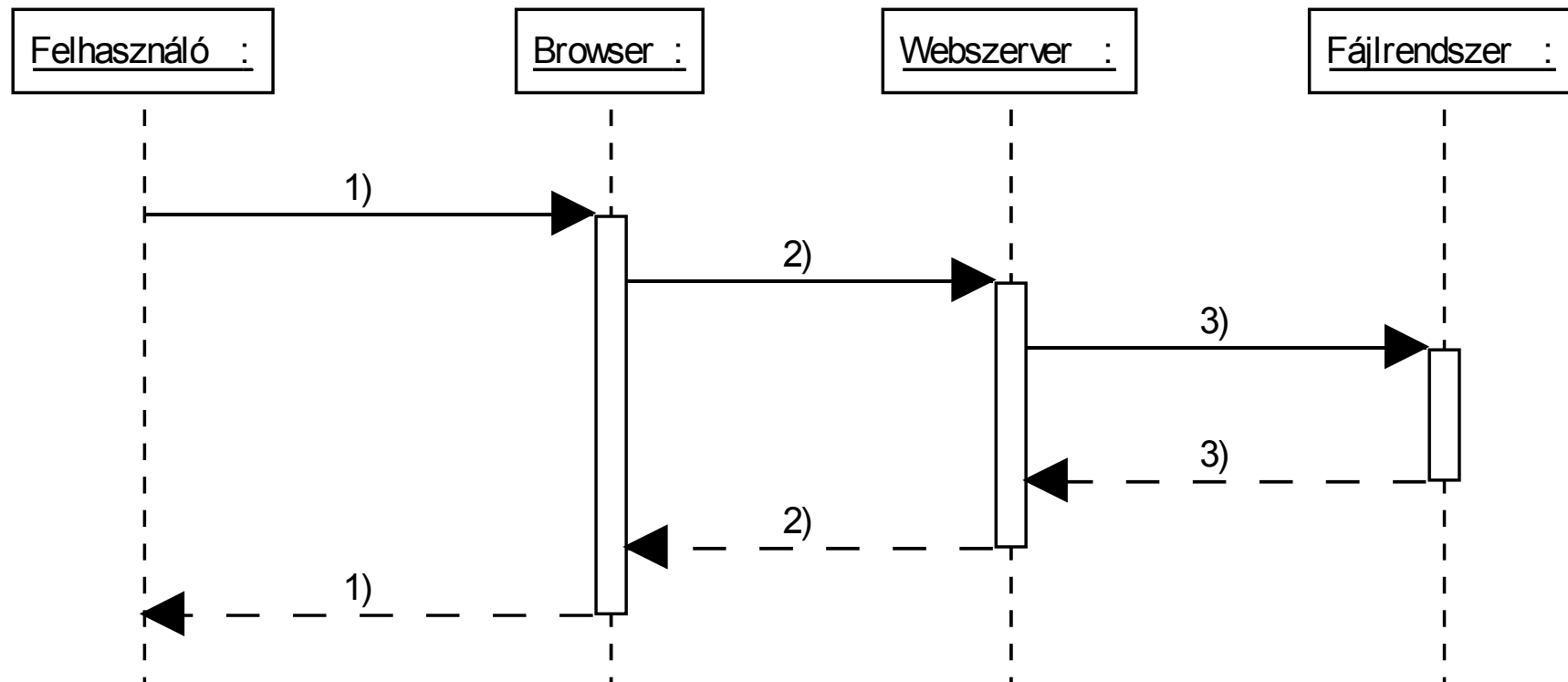


Bev1 – Miről lesz szó?

- webes architektúrák tegnap és ma
- a PHP interpreter
 - futtatása
 - bemenete
 - kimenete
- a safe mode

Bev1 – klasszikus webkiszolgálás

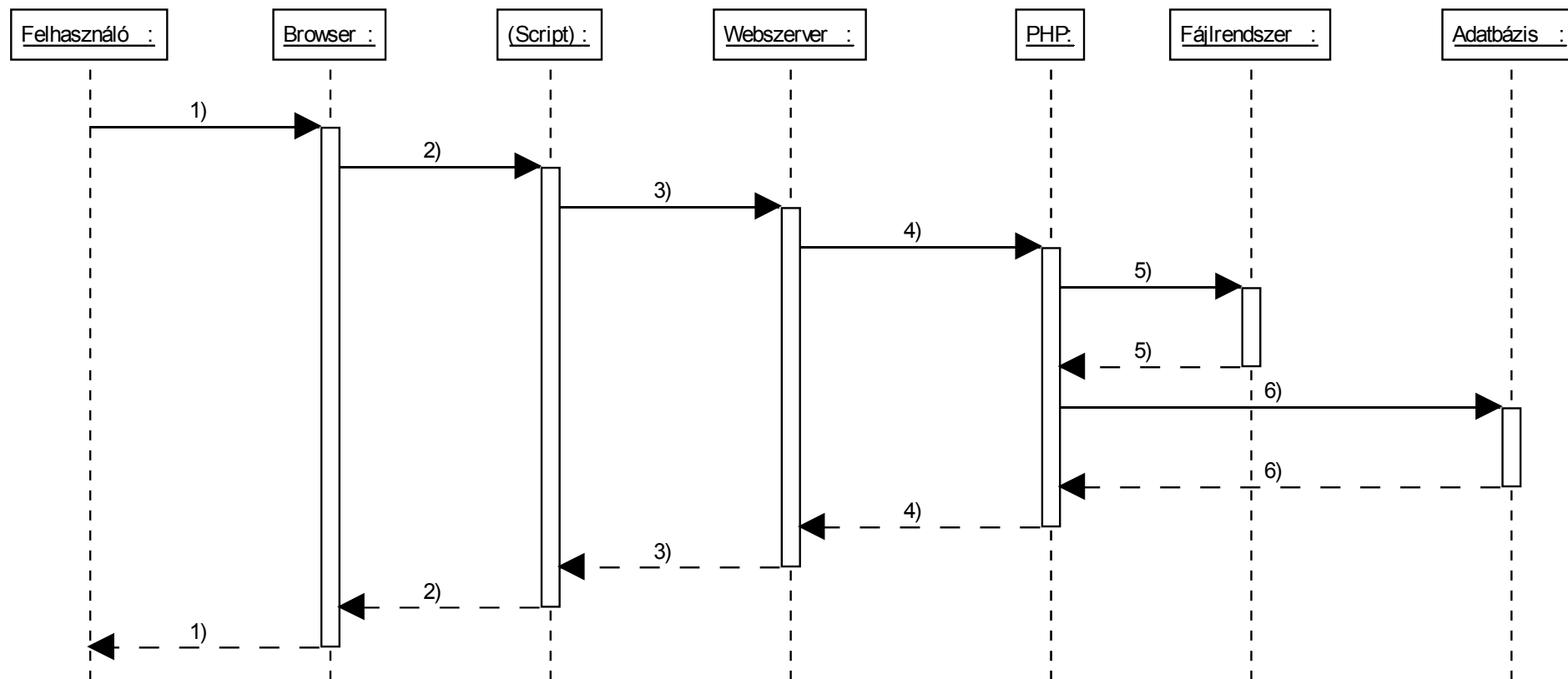
sd: Klasszikus HTTP





Bev1 – dinamikus kiszolgálás (PHP)

sd: Web2





Bev1 – A PHP interpreter

- Futásának módja
- Bemenete
- Kimenete
- Előnyök, hátrányok
- A PHP Safe Mode



Bev1 – A PHP futtatása (1)

- CLI
 - Command Line Interface
 - PHP, mint általános scriptnyelv
 - tipikusan nem webes célokra
- CGI
 - Common Gateway Interface
 - OS szempontjából azonos a CLI-vel
 - de facto szabvány: <http://www.w3.org/CGI>
 - 1995-ös a legutóbbi stabil verzió (1.1)



Bev1 – A PHP futtatása (2)

- MOD_PHP
 - Apache API beépülő modul, bővítmény:
<http://httpd.apache.org/docs/1.3/misc/API.html>
 - jelenleg a PHP futtatásának legelterjedtebb módja
 - az interpreter folyamatosan fut az Apache részeként
(az interpretált scriptek nem!)
 - Apache direktívákkal konfigurálható
 - előnyös és hátrányos is egyben (lásd később...)



Bev1 – A PHP futtatása (3)

- A PHP scriptek sosem folyamatosan futnak!
(kivéve CLI)
 - A HTTP állapotmentes protokoll, vagyis:
 - kérés érkezik: a script elindul
 - a script nem ismeri a múltbeli kéréseket
 - a script nem ismeri a párhuzamosan futó scripteket
 - kérés kiszolgálva: a script kilép
 - A PHP válasza: KISS (Keep It Simple, Stupid)
 - A Java válasza: application container
-
-



Bev1 – A PHP bemenete (1)

- minden esetben lehet:
 - fájlrendszer
 - adatbázis
 - egyszerű kliensként hálózati kapcsolat
 - CLI esetben lehet:
 - standard OS parancssor
 - standard futtatási környezeti változók (environment)
 - CGI esetben lehet:
 - CGI futtatási környezeti változók (environment)
 - MOD_PHP esetén lehet:
 - Apache szerver környezeti változók
-
-



Bev1 – A PHP bemenete (2)

- CGI és MOD_PHP esetében a környezet része:
 - a script neve, útvonala
 - a kérést kibocsátó gép címe
 - a kérés által használt csatorna (HTTP/HTTPS)
 - a webszerver által esetleg azonosított felhasználó neve, jelszava
 - a kéréshez tartozó protokoll-fejlécek:
 - POST paraméterek
 - GET paraméterek
 - COOKIE paraméterek



Bev1 – A PHP bemenete (3) - GET

- a GET paramétereket az URI hordozza
 - `http://example.com/script.php?par1=ert1&par2=ert2`
 - `http` protokoll azonosító
 - `://` protokoll elválasztó
 - `example.com` szerver azonosító (host)
 - `/` erőforrás azonosító elérési útvonal (path)
 - `script.php` erőforrás név
 - `?` elválasztó az erőforrás és paraméterek közt
 - `par1` az első paraméter neve, azonosítója
 - `=` elválasztó a paraméter és értéke közt
 - `ert1` az első paraméter értéke
 - `&` elválasztó a paraméterek között
-
-



Bev1 – A PHP bemenete (4)

- a POST adatok
 - a GET adatokhoz hasonlóan a konkrét kéréshez tartoznak
 - ugyanúgy név-érték párokból állnak
 - viszont nem „látszanak” a felhasználó számára, vagyis:
 - oldalújrátöltéskor elveszhetnek (a browser hatásköre)
 - nem tárolhatók könyvjelzőként
- a COOKIE adatok
 - nem „látszanak” a felhasználó számára
 - a böngészőprogramhoz (browser) vannak rendelve
 - lejáratási idővel rendelkeznek (ez lehet végtelen is)
 - nem kötelező őket elfogadni



Bev1 – A PHP kimenete

- minden esetben lehet:
 - fájlrendszer
 - adatbázis
 - egyszerű kliensként hálózati kapcsolat
(pl: más PHP programok felé: POST/GET/COOKIE)
- CLI esetben lehet:
 - OS standard kimenet (stdout)
- CGI és MOD_PHP esetén lehet:
 - a webservert bemeneti csatornája
 - vagyis indirekt módon a webböngésző



Bev1 – Előnyök, felhasználási terület

- CLI:
 - általános célú parancsnyelv, batch-feldolgozás, stb...
 - weblapok karbantartási munkáihoz
 - ütemezett futtatáshoz (cron)
 - CGI:
 - nem Apache webserverekhez is használható
 - nem csak a webservert nevében futtat (suphp, suexec)
 - elkülönülten fagy le, nem rántja a webservert magával
 - MOD_PHP:
 - leggyorsabb üzemmód, mert az interpreter mindig ott van
 - Apache virtual hostok, location-ök egyesével hangolhatók
-
-



Bev1 – Hátrányok

- CLI:
 - önmagában nehezen alkalmas webkiszolgálásra
 - tipikusan parancssori elérést igényel
 - pár web-specifikus szolgáltatás hiányzik (pl. session)
 - CGI:
 - lassabb, mint a MOD_PHP
 - elavult technológia, nem hangolható annyira
 - MOD_PHP
 - az egész webservert magával ránthatja
 - a webservert jogosultságaival fut
(nehéz pl. az egy gépen futó virtuális hosztok elválasztása)
-
-



Bev1 – A PHP Safe Mode

- Egy hibás/elkeseredett próbálkozás a szerver-oldali biztonság növelésére
- Lekorlátozza:
 - a fájműveleteket (csak saját fájlokra és könyvtárakra)
 - a hálózatelérést
 - az erőforráskorlátokat feloldó parancsokat
- A PHP6-ban már nem lesz benne



A gyakorlat munkakörnyezete

„Bev2” blokk

nappalisok: 1. alkalom

távoktatás: 1. alkalom



Bev2 – Miről lesz szó?

- Alkalmazások
 - WAMP5
 - EasyEclipse for PHP
- Szoftver architektúra
- Projektkezelés
- „Hello World!”





Bev2 – WAMP5 (1)

- <http://www.wampserver.com>
- Windows Apache MySQL PHP5
 - Apache 2.2.4
 - PHP 5.2.1 + PECL
 - SQLitemanager
 - MySQL 5.0.27
 - PHPMyAdmin





Bev2 – WAMP5 (2)

- webserverver PHP bővítménnyel
- adatbázis szerver
- konfigurációs és management WinXP applet
- több előretelepített PHP alkalmazás
- integrált telepítőkörnyezet





Bev2 – EasyEclipse for PHP

- <http://www.easyeclipse.org>
- PHP fejlesztéshez előrecsomagolt Eclipse
 - Eclipse integrált IDE (<http://www.eclipse.org>)
 - PHPEclipse plugin (<http://phpeclipse.sourceforge.net>)
- szintaxis-kiemelés
- kódkiegészítés
- hibakeresés
- verziókövetés





Bev2 – Szoftver architektúra

- Windows XP (OS), de lehetne Unix, MacOSX is
 - Apache (OS service, daemon)
 - PHP (Apache modul)
 - MySQL (OS service, daemon)
 - EasyEclipse for PHP (IDE)





Bev2 – Projektkezelés

- Eclipse oldalról:
 - integrált PHP projekt
 - névtér kezelés
 - webszervertől független
- PHP oldalról:
 - független szkriptfájlok halmaza
 - saját strukturálási lehetőségek
 - függhet a documentroot helyétől
- nem ragaszkodunk egyik oldalhoz sem



Bev2 – Hello World!

- lásd: <http://www.tricon.hu/~mcree/php/telepites>
- `<? echo 'hello world!' ?>`
- amire figyeljünk:
 - a fájlt a PHP értelmező csak az Apache-on keresztül futtatja le, egyébként a forráskódját fogjuk látni
 - az Apache csak a documentroot alatti területeket szolgáltatja, az Eclipse és a (futó) PHP kód viszont az egész gépet látja
 - a fájlban csak egy nyitó `<?` és egy záró `?>` jel legyen



A PHP nyelv alapjai (1. rész)

„Alap1” blokk

nappalisok: 1. alkalom

távoktatás: 1. alkalom



Alap1 – Miről lesz szó?

- alapvető szintaxis
- típusok
- változók és állandók
- kifejezések
- operátorok



Alap1 – Alapvető szintaxis (1)

- A PHP nyelvet a HTML fájlok kibővítésére tervezték
- HTML kompatibilis (kötelező) nyitó és záró tagek
 - `<script language="php"> ... </script>`
 - `<?php ... ?>`
 - `<? ... ?>` (rövidített stílus, opcionális)
 - `<% ... %>` (ASP stílus, opcionális)
- implicit echo tagek
 - `<?= kifejezés ?>`
 - `<%= kifejezés ?>`



Alap1 – Alapvető szintaxis (2)

- a tagek váltanak a HTML és PHP üzemmód közt
- ezt kihasználhatjuk a tömeges echo (print) utasítások kiváltására
- a végső záró tag opcionális

```
<?php
    if ($kifejezes) {
?>
        <h1>Ez igaz.</h1>
<?php
    } else {
?>
        <h1>Ez hamis.</h1>
<?php
    }
?>
```



Alap1 – Alapvető szintaxis (3)

- utasítások elválasztása (kötelező):
 - ;
 - a záró tag magába foglal egy pontosvesszőt is
 - (mint Java, C és Perl)
 - egysoros kommentezés:
 - // ...
 - # ...
 - sor végéig, vagy záró tagig érvényesek
 - (mint Java, C++, Perl, sh)
 - többsoros kommentezés
 - /* ... */
 - (mint C)
-
-



Alap1 – Típusok (1)

- négy skalár, egyértékű típus
 - boolean (logikai)
 - integer (egész szám)
 - float, double (lebegőpontos szám)
 - string (karakter sorozat)
 - két kompozit, összetett típus
 - array (tömb)
 - object (objektum)
 - két speciális típus
 - resource (erőforrás, pl. adatbázis kapcsolat)
 - NULL (az üres típus)
-
-



Alap1 – Típusok (2)

- a PHP lazán tipizált (loosely typed) nyelv!
- nincs szükség változók deklarálására
- implicit típuskonverziók a környezet függvényében
- létezik explicit típuskonverzió, de nem szükséges
- lásd a PHP kézikönyv P függelékét!
(típuskonverziós táblázat)



Alap1 – Típusok (3)

- boolean (logikai adattípus)
 - értéke: „true” vagy „false” (kisbetű/nagybetű érzéketlen)
- integer (egész szám)
 - megadható
 - oktális alakban (0-val kezdődik)
 - decimális alakban (nem 0-val kezdődik)
 - hexadecimális alakban (0x-el kezdődik)
 - mindig előjeles
 - architektúrafüggő pontosság (pl.: 32bit vagy 64bit)
 - a nem ábrázolható egész számok automatikusan lebegőpontossá alakulnak!



Alap1 – Típusok (4)

- float (lebegőpontos, valós számok)
 - megadható
 - tizedesponntal (12.34)
 - exponenciális alakban (2E-10)
 - véges (platformfüggő, 64bites IEEE formátum)
 - ne bízz meg a törtszámok értékében az utolsó jegyig!
 - ne vizsgálj pontos egyenlőségre két lebegőpontos számot!



Alap1 – Típusok (5)

- string (karakter sorozat)
 - aposztrófos formában
 - nincs külön értelmezés
 - idézőjeles és heredoc formában
 - változók értelmezve
 - \ escape szekvenciák
 - összevonása . jelekkel
 - részkepek [] jelekkel
 - lásd: PHP kézikönyv és később!

```
$s = "az a erteke: $a";  
$s = '150 $dollarom van';  
$s = <<<VALAMI  
ez több soros heredoc  
formatumu megadas  
VALAMI;  
$c = $s[10].' és '.$s[11];
```



Alap1 – Típusok (6)

- array (tömb)
 - array() az üres tömb
 - minden tömb asszociatív
 - hivatkozás [] jelekkel
 - kulcs megadása => jel előtt
 - érték akármilyen lehet
 - kulcs csak int és string
 - lásd: PHP kézikönyv és később!

```
$a = array();  
$b = array("k" => "ert");  
$c = $b["k"];  
$b["k2"] = "ert2";
```



Alap1 – Típusok (7)

- object (objektum)
 - OO paradigma
 - egy osztály példánya
 - létrehozása: new utasítás
 - hivatkozás -> jellel
 - lásd később!
 - resource (erőforrás)
 - beépített függvényekkel hozható létre és azok igénylik
 - NULL (az üres típus)
 - definiálatlan változók értéke
 - unset() függvény vagy NULL értékadás eredménye
-
-



Alap1 – Változók (1)

- jelölése: \$ jellel (mint Perl, sh)
 - „Egy érvényes változónév betűvel vagy aláhúzással kezdődik, amit tetszőleges számú betű, szám vagy aláhúzás követ” (PHP kézikönyv)
 - a név kis/nagybetű érzékeny
 - referencia változó: & jellel (mint C)
 - változó változók léteznek, pl: \$\$a
 - kétértelműségek feloldása: {} jelekkel
 - deklaráció nem kötelező, de lehetséges (var kulcsszó)
-
-



Alap1 – Változók (2)

- a változók hatásköre (scope) nem megszokott!
 - három scope létezik:
 - lokális (függvényben)
 - globális (függvényeken kívül)
 - szuperglobális (lokális + globális)
 - a globális változók alapértelmezésben nem látszanak a függvényekben!
 - `globals` kulcsszó
 - `$GLOBALS` szuperglobális asszociatív tömb
-
-



Alap1 – Változók (3)

- static kulcsszó: statikus változó függvényekben
- fontosabb szuperglobális asszociatív tömbök:
 - \$GLOBALS (globális változók)
 - \$_GET és \$_POST (lásd: PHP bemenete rész)
 - \$_COOKIE (lásd: PHP bemenete rész)
 - \$_SESSION (munkamenet, lásd: később)
- konstansok (csak skalár értékkel!)
 - definiálása: define(nev, érték) függvény
 - elérése: névvel
 - PHP-ben minden nem definiált konstans értéke a neve!



Alap1 – Kifejezések (1)

- kifejezés: „valami, aminek értéke van”
- a PHP számára majdnem minden kifejezés
- fontosabb példák:
 - függvények (function kulcsszó)
 - növelő és csökkentő operátorok (++ , --)
 - összehasonlító kifejezések (==, !=, <, >, stb...)
 - értékadás (\$a=\$b=42, \$a+=2, stb...)
 - a háromoperandusú feltételes operátor (bool ? val1 : val2)



Alap1 – Operátorok (1)

- operátor: kifejezésből egy másik kifejezést állít elő
 - operandusok száma: unáris, bináris, trenáris
 - asszociativitás
 - jobbról balra, pl.: $\$a=\$b=42$
 - balról jobbra, pl.: $\$a=\$b+\$c-42$
 - nem köthető, pl. ilyen nincs: $1<\$a<5$
 - precedencia
 - mint általában más prog. nyelveknél (C, Perl, Java)
 - ha bizonytalan vagy, zárójelezz!
 - lásd: PHP kézikönyv!
-
-



Alap1 – Operátorok (2)

- aritmetikai (+, -, *, /, %)
 - hozzárendelő (=)
 - bitorientált (&, |, ^, ~, <<, >>)
 - összehasonlító (==, ===, !=, <>, !==, <, >, <=, >=)
 - hibakezelő (@)
 - végrehajtó (`)
-
-



Alap1 – Operátorok (3)

- növelő, csökkentő (++ , --)
- logikai (and, or, xor, !, &&, ||)
 - az and és or szöveges változatának precendenciája kisebb!
- string (. , .=)
- tömb (+)
- típus (instanceof)



A PHP nyelv alapjai (2. rész)

„Alap2” blokk

nappalisok: 1. alkalom

távoktatás: 1. alkalom



Alap2 – Miről lesz szó?

- vezérlési szerkezetek
 - függvények
 - osztályok és objektumok
 - kivételek
 - referenciák
 - biztonság
-
-



Alap2 – Vezérlési szerkezetek (1)

- csoportosítás {} jelekkel
 - feltételes végrehajtás
 - if, else, elseif, switch
 - ciklusok
 - while, do-while, for, foreach, break, continue
 - declare (futási paraméterek beállítása)
 - jelenleg csak időmérésre
 - return (visszatérés függvényből)
 - külső fájlok beillesztése
 - require, include, require_once, include_once
-
-



Alap2 – Vezérlési szerkezetek (2)

```
// komplex if példa
if ($a==1) {
    echo "az a 1";
} elseif ($a==2) {
    echo "az a 2";
} else {
    echo "az a más";
}

// alternatív szintaxis
if ($a==1):
    echo "a egy";
endif;
```

```
// komplex switch példa
switch ($s) {
case "egy":
case "egyed":
    echo "egyedem";
case "begyed":
    echo "begyedem";
    break;
default:
    echo "tengertánc";
}
```



Alap2 - Vezérlési szerkezetek (3)

```
// while
while($a<2) {
    echo $a++." while<BR>";
}
```

```
// do-while
do {
    echo $a++." do<BR>";
} while($a<2);
```

```
// for
for ($a=0; $a<2; $a++) {
    echo "$a for<BR>";
}
```

```
// foreach
foreach (array(4,2) as $a)
{
    echo "$a foreach<BR>";
}
```

```
// break, continue
for ($a=0; $a<2; $a++) {
    for ($b=0; $b<2; $b++) {
        if ($a==$b) continue;
        echo "$a:$b<BR>";
        break;
    }
}
```



Alap2 – Függvények (1)

- function kulcsszóval (sőt: utasítással!) definiáljuk
- feltételes deklaráció lehetséges (lásd példák!)
- definiálásuktól fogva globálisan elérhetőek
- nincs polimorfizmus
- nem lehet függvényt újradefiniálni, megszüntetni
- mély rekurzió (100-200 szint felett) nem ajánlott!
- függvényargumentumok
 - csak névvel kell megadni (nem erősen típusos nyelv)
 - alapértékük megadható (csak konstans lehet)
 - változó számú argumentum is lehetséges



Alap2 – Függvények (2)

```
// egyszerű függvény
function összead($a, $b=2
) {
    return $a + $b;
    echo "sose írja ki";
}
echo összead(40). "<BR>";
// függvényváltozó
function alma() {
    echo "alma<BR>";
}
$korke="alma";
$korke();
```

```
// változó argumentumszám
function kiir() {
    echo func_num_args().
        " arg.<BR>";
    foreach(func_get_args()
        as $arg) {
        echo $arg."<BR>";
    }
}
kiir(4,2,"barack");
```



Alap2 – Függvények (3)

```
// feltételes deklaráció
if(1<2) {
    function foo() {
        function bar() {
            echo "hejj<BR>";
        }
    }
}
foo();
bar();
```

```
// referencia argumentum
function novel(&$a) {
    $a++;
}
$c=1;
novel($c);
echo "$c<BR>";
```



Alap2 – PHP5 OO alapok (1)

- hibrid OO paradigma (strukturált és OO egyszerre)
 - osztály deklaráció: `class` kulcsszó
 - osztály példányosítás (objektum létrehozás): `new`
 - egyszeres öröklés: `extends` kulcsszó
 - az őosztály azonosítója: `parent`
 - `__construct` és `__destruct`
 - nincs automatikus ősmetódus-hívás
 - hívó objektumpéldányra hivatkozás: `$this`
 - aktív osztályra hivatkozás: `self`
 - dinamikus hivatkozás: `->` operátor
 - statikus hivatkozás: `::` operátor
-
-



Alap2 – PHP5 OO alapok (2)

- mezők definíciója: public/private/protected kulcsszó
- metódusok definíciója: function kulcsszó
- metódusok és mezők láthatósága, attribútumai:
 - public: bárhonnan elérhető
 - private: csak a definiáló osztályból
 - protected: csak öröklő- és szülőosztályokból
 - final: öröklőosztályokban nem felülírható
 - static: csak statikusan érhető el (nincs ->)
 - const: statikus konstans érték
 - abstract: elvont, nem definiált függvénytorzs
 - a tartalmazó osztály nem példányosítható
 - ez a kulcsszó osztályra is alkalmazható



Alap2 – PHP5 OO alapok (3)

- interfészek definiálása: interface kulcsszó
 - csak publikus metódusokat tartalmazhat
 - implementálása: implements kulcsszó
- mező és metódus overloading
 - `__set()`, `__get()`, `__isset()`, `__unset()`, `__call()` metódusok
- `foreach()` iteráció lehetséges a publikus mezőkön
 - lásd még: Iterator beépített interfész, SPL bővítmény
- egyéb mágikus metódusok
 - pl.: `__toString`, `__sleep`, `__wakeup`, `__clone`, `__autoload`
 - mind `__` jelekkel kezdődnek, ez fenntartott név
 - bővebben lásd: PHP kézikönyv és később



Alap2 – PHP5 OO alapok (4)

- objektumok klónozása: clone kulcsszó
 - alapértelmezés: shallow copy (a referenciákat meghagyja)
 - lásd még: `__clone` mágikus metódus
 - objektumok tartalmi összehasonlítása: `==`
 - objektumpéldányok azonosságának vizsgálata: `===`
 - kivételkezelés (mint Java):
 - `try { ... } catch (Exception $e) { ... }`
 - a beépített függvények egyelőre nem dobnak exception-t
 - van reflection API is (mint Java)
 - type hinting: metódusok argumentumának típuskényszerítése (csak osztály és tömb lehet)
-
-



Alap2 – Referenciák (1)

- a PHP „szeret” mindent mindig másolni
 - értékadáskor
 - hivatkozáskor
 - paraméterátadáskor
 - stb...
- PHP referencia: „egy tartalom több néven érhető el”
- nem olyan, mint C-ben a pointer!
- inkább a UNIX hardlinkekhez hasonlít
- a referenciákat & jellel adjuk meg



Alap2 – Referenciák (2)

- referenciát használhatunk
 - függvényparaméterek megadásánál, pl.: `novel($a)`
 - referencia visszaadására, pl.: `$modositando=&keres()`
 - példányosításnál, pl.: `$obj = &new osztaly()`
 - memória- és sebességoptimalizáláshoz
- ne használjunk referenciát
 - összetett tömbökre való hivatkozásra (inkább másolódnak)
 - referenciák másolására (a referencia csak egy név!)
- alapértelmezett referenciák PHP-ban:
 - global kulcsszó: `& $GLOBALS('kulcsszó')`
 - `$this` mindig az adott objektumon dolgozik



Alap2 – Biztonság (1)

- a PHP egy sokszínű, funkciógazdag, elterjedt nyelv
 - megfelelő korlátozásokkal kell élnünk, hogy ne használhassák ellenünk
 - minimális jogosultságok az éles rendszerben
 - ne maradjanak debug információk, error reporting off
 - sose adjuk ki a script forrását a webre
 - sose bízunk meg felhasználótól érkező adatokban
 - register globals (már a múlté)
 - sql injection (speciálisan szerkesztett escapelt bemenetek)
 - magic quotes (egy újabb gyenge próbálkozás)
 - lásd: PHP kézikönyv, biztonság fejezete
-
-



Alap2 - Feladat

Készítsen egyszerű PHP programot, mely a kimeneten tetszőleges lépésben képes megjeleníteni az ún. fibonacci számsorozatot, melyben a soronkövetkező elem mindig az azt megelőző két elem összegével egyenlő. Egy példa a helyes kimenetre:

```
0 1 1 2 3 5 8 13 21 ... stb...
```

Pluszfeladat: oldja meg, hogy a program minden soronkövetkező elemre jelenítse meg új sorban az adott elemig kiszámított teljes sort. Egy példa a helyes kimenetre:

```
0
0 1
0 1 1
stb...
```



A PHP API áttekintése (1. rész)

„Api1” blokk

nappalisok: 2. alkalom

távoktatás: 1. alkalom





Api1 – Miről lesz szó?

- Tömbök kezelése
- Dátumkezelés
- Fájl- és könyvtár I/O
- Levelezés





Api1 – Tömbök

- Teljes referencia: PHP kézikönyv V. fejezet
- ismétlésként, a PHP tömbök mindig:
 - asszociatívak (kulcs-érték párokkal dolgoznak)
 - praktikusán „korlátlan”, változó méretűek
 - definiálnak egy alapértelmezett végigjárási sorrendet
- lásd: mellékelt forráskódok!



Api1 – Tömbök - feladat

Készítsen PHP programot, mely előállít egy tömböt, melyben a pozitív egész számok szerepelnek 1-től 100-ig. Készítsen függvényt, mely egy megadott tömb egy adott egész paraméterrel maradék nélkül osztható elemeit NEM tartalmazó tömbbel tér vissza. Az eredményeket jelenítse meg! Példa futási eredmény:

```
A tömb: 1 2 3 4 5 6 7 8 9 ... stb...
```

```
2-vel osztható elemek kiszűrése után: 1 3 5 7 9 ...  
stb...
```

```
3-al osztható elemek kiszűrése után: 1 5 7 11 13 17  
... stb...
```

Pluszfeladat: használja fel az előállított szűrőfüggvényt az 1 és 1000 közti prímszámok megkeresésére és megjelenítésére! Példa futási eredmény:

```
1 3 5 7 11 13 ... stb...
```



Api1 – Dátum és idő

- PHP kézikönyv X. és XXII. fejezet
- A PHP időszámítási alapja:
 - Unix Epoch (1970. január 1. 00:00:00 GMT)
 - A dátumok egy másodperc és egy időzóna információt tartalmaznak
- Lásd: a mellékelt forráskódok!



Api1 – Dátum és idő feladat

Feladat: készítsen programot mely meghatározza, a hét melyik napjára esik az Ön születésnapja! Egy lehetséges futási eredmény az 1979-09-10 dátummal:

hétfő

Pluszfeladat: a program azt is adja meg, a megadott évben mely napra esett húsvét! Egy lehetséges futási eredmény:

április 8.



Api1 – Fájlok és könyvtárak

- PHP kézikönyv XXIX. és XL. fejezet
- gyakorlatilag felület az O/S funkciókhoz
- sok függvény URL-eken is működik!
- lásd: a mellékelt forráskódok!





Api1 – Fájlok és könyvtárak feladat

Készítsen PHP programot, mely egy adott könyvtárban található összes fájl méretét összeadja és a felhasználó által könnyen érthető formában megjeleníti. Egy lehetséges megoldás:

12.34 kilobyte

Pluszfeladat: a program ne csak az adott könyvtárban, hanem az alkönyvtáraiban is számolja össze a fájlok méretét! A program számolja össze a könyvtárban található fájlok sorainak számát is! Egy lehetséges megoldás:

1.23 megabyte, 15252 sor



Api1 – SMTP

- PHP kézikönyv LXXV. és LXIII. fejezet
- A PHP külső programmal és saját erőből is képes levelek küldésére
- A levelek MIME törzsét a programozónak kell megadnia
- lásd: a mellékelt forráskódok!



A PHP API áttekintése (2. rész)

„Api2” blokk

nappalisok: 2. alkalom

távoktatás: 1. alkalom





Api2 – Miről lesz szó?

- SQL
- String függvények
- Reguláris kifejezések





Api2 – SQL gyorsfalpaló (1)

- Structured Query Language programozási nyelv
- adatok központi strukturált tárolására és elérésére
- alapvető strukturális elemek:
 - adatbázis (táblákat tartalmaz)
 - tábla (rekordokat/sorokat tartalmaz)
 - rekord/sor (típusos adatmezőket tartalmaz)
 - adatmező (jellemzően egy skaláris adatot tartalmaz)





Api2 – SQL gyorstalpaló (2)

- tábla létrehozása:
- CREATE TABLE név (mező1 típus1,)
- a leggyakoribb típusok:
 - TEXT, VARCHAR(bájthossz) – szöveg
 - INT(bájthossz) – szám
 - TIMESTAMP – dátum, időbélyeggel

```
-- SQL táblalétrehozás példa
CREATE TABLE emberek (
    nev TEXT,
    gyermekek INT(2),
    szuletett TIMESTAMP
)
```



Api2 - SQL gyorstalpaló (3)

- adatok beszúrása:
- INSERT INTO táblanév (mező1, mező2, ...)
VALUES (érték1, érték2 ...)
- az értékekre biztonsági szempontból figyelni kell!
- a típusok közti különbségek néha gondot jelentenek!

```
-- adatbeszúrás példa
```

```
INSERT INTO emberek (nev,gyerekek,szuletett)  
VALUES ('Teszt Elek',3,'1959-11-23')
```



Api2 - SQL gyorstalpaló (4)

- adatok lekérdezése:
- `SELECT mező1, mező2, ...`
`FROM tábla1, tábla2,`
`[WHERE feltétel1, feltétel2, ...]`
`[ORDER BY mező3, mező4, ... ASC/DESC]`
`[GROUP BY mező5, mező6, ...]`
`[LIMIT sorok száma]`
`[OFFSET sorok száma]`
- egy select több táblát kapcsolhat össze



Api2 - SQL gyorstalpaló (5)

```
-- lekérdezés példa
SELECT
    nev,gyerekek,szuletett
FROM
    emberek
WHERE
    szuletett > '2000-01-01'
    AND
    gyerekek > 2
ORDER BY
    nev DESC
LIMIT
    100
OFFSET
    50
```



Api2 – SQL

- PHP kézikönyv CXVIII. és sok más fejezet
- A PHP rengeteg adatbázisrendszert támogat
- kezdetben: minden adatbázishoz külön API
- PHP 5.1 óta: PDO (PHP Data Objects)
 - uniform objektumorientált API
 - számos adatbázis driver támogatása (ODBC is)
- lásd a mellékelt forráskódokat!



Api2 - SQL feladat

Készítsen adatbázisalapú programot, mely minden látogató számára megjeleníti, hogy az általa használt IP címről eddig hány alkalommal tekintették meg a weblapot! Példa a kimenetre:

```
az ön IP címe: 127.0.0.1, erről a címről eddig 15  
alkalommal néztek meg.
```

Pluszfeladat: jelenítse meg a 10 leggyakoribb IP címet sorrendben, valamint tárolja és jelenítse meg a legutóbbi látogatások dátumát is! Példa a kimenetre:

```
az ön IP címe: 127.0.0.1, erről a címről eddig 15  
alkalommal néztek meg,  
legutóbb 1956 január 2. 19 óra 33 perckor.
```

A 10 leggyakoribb látogató:

```
127.0.0.1 (15 látogatás)  
127.0.0.2 (13 látogatás)  
... stb...
```



Api2 - Stringkezelés

- PHP kézikönyv CLIV., XCII., CLX., CLXI. fejezet
- a stringek önálló skalárnak számítanak
- felemás multibyte támogatás!
- lásd a mellékelt forráskódokat!





Api2 – Stringkezelés feladat

Készítsen programot, mely egy hosszabb szövegben megszámolja a szavak számát, majd a 3 karakternél nagyobb szavakból új szöveget alkot melyben a szavakat szóközök választják el. Szövegforrásnak használhatja a www.lipsum.org webhelyet. Példa futási eredmény az "ez egy string, lehetne!" bemenetre:

```
4 szó  
string lehetne
```

Pluszfeladat: a program számolja és őrizze meg a pontuációs karaktereket. Példa futási eredmény az "ez egy string, lehetne!" bemenetre:

```
4 szó, 2 pontuációs karakter  
string, lehetne
```



Api2 – Reguláris kifejezések

- PHP kézikönyv CXVI. és CXXI. fejezet
- a PHP Perl és Posix stílusú kifejezéseket támogat
- a Perl stílusúak (PCRE) általában „ügyesebbek”
- számtalan felhasználási terület
- lásd a mellékelt forráskódokat!



Api2 – PCRE gyorsalpaló

- általában minden karakter önmagát jelenti, de vannak speciális karakterek:
 - . bármilyen karaktert helyettesít
 - * a megelőző minta nulla vagy több előfordulása
 - + a megelőző minta egy vagy több előfordulása
 - ? mohóság (greediness) kikapcsolása
 - [] karakterosztályok megadása
 - () rész minta megadása
 - / mintahatár
 - \ escape
 - ^ sor eleje
 - \$ sor vége
-
-



PHP tervezési minták (1. rész)

„Pat1” blokk

nappalisok: 3. alkalom

távoktatás: 2. alkalom





Pat1 – Miről lesz szó?

- Formkezelés
 - Input feldolgozása, ellenőrzése
 - Formok újramegjelenítése
 - Többnyelvűség
 - Fájlfeltöltés kezelése



Pat1 – HTML FORM (1)

- POST és GET metódus (lásd: bevezető!)
- <FORM> és </FORM> tagek zárják közre
- számos beviteli elem közül választhatunk:
 - text: sima szöveg
 - password: rejtett megjelenítésű szöveg
 - checkbox: jelölőnégyzet, választómező
 - radio: rádiógomb, választócsoport
 - submit: formot beküldő gomb
 - textarea: többsoros szöveg
 - select: legördülő menü és többes opcióválasztás
 - hidden: webes felhasználó előtt rejtett adat



Pat1 – HTML FORM (2)

- `<INPUT>` beviteli elemek attribútumai:
 - type – típus
 - text, password, checkbox, radio, submit, hidden
 - name – beviteli mező azonosítására szolgáló név
 - radio esetén a csoportot is azonosítja
 - value – alapértelmezett szöveg
 - submit esetén a gomb címkéje
 - radio és checkbox esetén nincs hatása
 - checked – alapértelmezett állapot
 - nem kell értéket megadni, ha szerepel azt jelenti, be van jelölve
 - csak checkbox és radio
 - disabled – felhasználói bevitel engedélyezett-e
 - ha szerepel, a mező letiltásra kerül
 - nem bízhatunk abban, hogy a böngésző ezt betartja!



Pat1 – HTML FORM (3)

- `<TEXTAREA>` tag attribútumai:
 - cols, rows – oszlopok és sorok száma
 - name – beviteli mezőt azonosító név
 - disabled – bevitel engedélyezése a felhasználó számára
 - a textarea alapértéke: a bezáró `</TEXTAREA>` tagig
 - `<SELECT>` tag attribútumai
 - name – beviteli mezőt azonosító név
 - disabled – bevitel engedélyezése a felhasználó számára
 - multiple – többes választási lehetőség bekapcsolása
 - nem kell értéket megadni, ha szerepel érvénybe lép
 - a select értékei: a bezáró `</SELECT>` tagig megadott `<OPTION` tagekben
-
-



Pat1 – HTML FORM (4)

- `<OPTION>` tag attribútumai:
 - value – az opció értéke
 - selected – az opció alapértelmezett állapota
 - nem kell értéket megadni, ha szerepel az opció kijelölésre kerül
 - az opció nevének megadása: `</OPTION>` tagig
- lásd a mellékelt forráskódokat!

```
<!-- SELECT példa -->  
  
<SELECT name="valasztas">  
  <OPTION value="1">első</OPTION>  
  <OPTION value="2" selected>második</OPTION>  
</SELECT>
```



Pat1 – Formkezelés és ellenőrzés

- a beküldött formok adatai: `$_POST` és `$_GET` szuperglobális asszociatív tömbök
- a tömbök kulcsai: a formelemek `name` attribútumai
- a `name` attribútumok speciális használata:
 - ha a név `[]`-re végződik, automatikusan tömb készül belőle
 - ha a név `[kulcs]`-ra végződik a kulcs lesz a tömbindex
- lásd a mellékelt forráskódokat!



Pat1 – Formok (újra)feltöltése

- mikor van erre szükség?
 - dinamikusan alapértelmezett értékeket akarunk megadni
 - felhasználó által „elrontott” form újrabekérésekor
- minden ismeretett formmezőnél megadható alapérték
- az alapértékek kitöltésekor ügyeljünk:
 - a speciális karakterekre
 - újrakitöltéskor a PHP magic quotes szolgáltatásra
- lásd a mellékelt forráskódokat!



Pat1 – Többnyelvűség

- mikor van erre szükség?
 - ugyanazt a formot akarjuk többször más nyelven megadni
 - ez a minta nem csak formoknál lehet hasznos
- jellemzően GET paraméterrel befolyásoljuk a nyelvet
- lásd a mellékelt forráskódokat!





Pat1 – Fájlfeltöltés kezelése

- csak POST adatként, speciális multipart formban
- saját <INPUT> type attribútum: file
- a feltöltés eredménye a \$_FILES szuperglobálisban
- a korlátokról lásd a PHP kézikönyv 38. fejezetét!
- lásd a mellékelt forráskódokat!

```
<!-- fájlfeltöltő FORM példa -->
<FORM enctype="multipart/form-data" method="POST">
  Fájl: <INPUT NAME="file" TYPE="file">
  <INPUT TYPE="submit" VALUE="Fájl feltöltése">
</FORM>
```



Pat1 – Feladat

Készítsen egyszerű adatlekérdező formot, melyen a felhasználó megadhatja nevét és nemét. A form feldolgozása során ellenőrizze, hogy a név és nem megfelelően kitöltésre került-e, ha nem, jelenítse meg a formot a felhasználó által már kitöltött adatok alapértelmezett értéként való megadásával és a hiányosságok megjelölésével. Ha minden mező megfelelő, jelenítsen meg rövid üdvözlő üzenetet, melyben a felhasználót nevének és nemének megfelelően köszönti! Ha a form még nem került elküldésre, ne jelenítsen meg hibaüzenetet!

Egy lehetséges futási eredmény helyes formkitöltés esetén:

Tisztelt Rigó Ernő úr! Köszönjük, hogy kitöltötte a formot!

Egy lehetséges futási eredmény helytelen kitöltés esetén:

Kérem válassza ki a nemét!

Név: Rigó Ernő Nem: () férfi () nő [küldés]

Pluszfeladat: Hiba esetén emelje ki pirossal azokat a mezőket amelyek nem megfelelően lettek megadva! Bővítse a formot, hogy e-mail címet és születési dátumot is meg lehessen adni, végezze el az email cím és a dátum formális ellenőrzését is! A dátum beviteléhez ajánlatos legördülő menüket felvenni külön az év, hónap és nap értékekre, így szűkíthető a hibalehetőségek sora. Egészítse ki a formot többnyelvű megjelenítési és fájlfeltöltési lehetőséggel!



PHP tervezési minták (2. rész)

„Pat2” blokk

nappalisok: 3. alkalom

távoktatás: 2. alkalom





Pat2 – Miről lesz szó?

- Munkamenet-kezelés (Sessionkezelés)
 - Felhasználók követése
 - Egyszerű „bevásárlókosár” tervezési minta
 - Cookie-kezelés („sütik”)



Pat2 – Munkamenetek

- lásd: PHP kézikönyv, CXLII. fejezet!
 - a HTTP állapotnélküliségének megszüntetése a cél
 - középtávú adattárolásra alkalmas
 - alapértelmezésben merevlemezen tárolódik
 - `$_SESSION` szuperglobális asszociatív tömb
 - a PHP script indítása után feltöltjük
 - `session_start()`
 - a script futása után elmentjük
 - automatikusan megtörténik, vagy `session_write_close()`
 - munkamenet törlése: `session_destroy()`
 - lásd a mellékelt forráskódokat!
-
-



Pat2 – Sessionkezelés feladat

Készítsen négy alpműveletes számológép funkciót megvalósító PHP programot, mely a műveletet, valamint két operandusát egy FORMon keresztül kéri be! A program használjon munkamenetet az összes bevitt számítás és eredményeik megjelenítésére! Egy lehetséges futási eredmény (az első sor a formot hivatott ábrázolni):

```
Adja meg a műveletet: _____ [+] _____ [=]
```

```
Eddigi műveletek:
```

```
23452345 + 23452345 = 46904690
```

```
134 - 234235 = -234101
```

```
42352345 * 32452345 = 1.3744329115E+15
```

Pluszfeladat: az eddigi műveleteket ne felsorolásként, hanem egy legördülő menüben jelenítse meg és tegye lehetővé a felhasználó számára, hogy bármelyik korábbi számításhoz visszatérve módosítsa azt! A számításhoz való visszatérést úgy valósítsa meg, hogy a legördülőben kiválasztott számítás adatai az alpműveletes számológép formjába töltsenek vissza alapértelmezett értéként! Tegye lehetővé az eddigi összes művelet törlését! Egy lehetséges futási eredmény:

```
Adja meg a műveletet: _____ [+] _____ [=]
```

```
Eddigi műveletek:
```

```
[23452345 + 23452345 = 46904690] [visszatér] [mindet töröl]
```



Pat2 – Cookie

- lásd: PHP kézikönyv XCVIII. fejezet!
- a böngésző számára adatokat küldhetünk ki
- a kiküldött adatokat ezután minden kérésnél megadja
- lejáratási idővel rendelkező kulcs-érték párok
- a munkamenet-kezelés is ezt használja ha elérhető
- nem kényes adatok hosszú távú tárolására alkalmas
- lásd a mellékelt forráskódokat!



Pat2 – Cookie feladat

Készítsen cookie alapú PHP programot, mely nevén nevezi és köszönti az oldalt meglátogató felhasználót! Ha a felhasználó még nem ismert, kérdezze meg a nevét, ha a felhasználó már ismert, tegye lehetővé, hogy az oldal elfelejtse! Az oldal 30 napig emlékezzen a felhasználókra!

Példa futási eredmény ismeretlen felhasználó esetén:

Szia! Add meg neved: _____ [emlékezz rám!]

Példa futási eredmény ismert felhasználó esetén:

Szia Balamka Sámuel! [felejts el!]

Pluszfeladat: a felhasználó a neve mellett adhasson meg egy jelszót is, melynek megadása legyen szükséges az elfelejtő funkció használatához!

Példa futási eredmény ismeretlen felhasználó esetén:

**Szia! Add meg neved: _____ jelszavad: _____
[emlékezz rám!]**

Példa futási eredmény ismert felhasználó esetén:

Szia Balamka Sámuel! Jelszó: _____ [felejts el!]



PHP tervezési minták (3. rész)

„Pat3” blokk

nappalisok: 4. alkalom

távoktatás: 2. alkalom





Pat3 – Miről lesz szó?

- Hitelesítés
 - HTTP alapú
 - Cookie alapú
- Template rendszerek
 - Kódszeparáció és -újrafelhasználás
 - MVC (Model-View-Controller)





Pat3 – HTTP Hitelesítés

- az azonosítást a webböngésző végzi
 - a cookie-hoz hasonlóan mindig küldi az adatokat
 - a realm-be tartozó felhasználónév és jelszó
 - azonosítási információk a `$_SERVER` szup.glob.-ban
 - azonosítás kezdeményezése: `header()` függvénnnyel
 - `WWW-Authenticate: Basic realm="Azonosítsd magad!"`
 - `HTTP/1.0 401 Unauthorized`
 - `realm`: az azonosítási terület
 - `Basic`: az azonosítás típusa (lehetne még: `Digest`)
 - lásd: PHP kézikönyv 34. fejezet!
 - lásd a mellékelt forráskódot!
-
-



Pat3 – Cookie alapú hitelesítés

- az azonosítást a PHP program végzi FORM-al
- siker esetén cookie-ba tárolja az információkat
- a „realm” a cookie domainje
- több felelősség a programozón
- lásd: korábbi cookie példák
- lásd a mellékelt forráskódot!



Pat3 – Template rendszerek (1)

- A PHP programok jellemzően vegyesek
 - megjelenítési utasítások, *nézet*
 - műveletek, „business logic”, logikai *modell*
 - felhasználói input feldolgozása, felhasználói *kontroll*
- A felsorolt három terület jobbára elkülöníthető
- A nézet (view)
 - a felhasználó ezt „látja”, tehát gyakran testre kell szabni
 - jellemzően sok HTML kódot tartalmaz
 - a PHP változókból leginkább adatokat nyer ki
 - gyakran hasonló adatokat másként kell megjeleníteni
 - gyakran egy megjelenítéshez más-más adatok tartoznak
 - tehát: az adatok logikai modelljével nem annyira törődik



Pat3 – Template rendszerek (2)

- Az adatok és rajtuk végzett műveletek (model)
 - az adatok gyakran adatbázisból származnak
 - a műveleteket gyakran adatbázison kell végrehajtani
 - a műveletek gyakran jellemző mintákat mutatnak
 - célszerű a kód újrafelhasználása (OO paradigma)
 - az adatok és műveletek a megjelenítéstől függetlenek
- Felhasználói input és műveletvégzés (controller)
 - jellemzően az adatokon akar műveleteket indítani
 - PHP esetén itt történik az előfeldolgozás (pl. stripslashes)
 - a HTTP protokoll állapotmentességét itt kell kompenzálni



Pat3 – Template rendszerek (3)

- a model-view-controller (MVC): ismert paradigma
- minden GUI alkalmazásnál, így a PHP-nál is hasznos
- az MVC vezérlése ciklikus, négy szereplős:
 - a felhasználó kérést bocsát ki
 - a controller feldolgozza a kérést és értesíti a modellt
 - a model műveleteket végez és értesíti a view-t
 - a view a kért formában megjeleníti a modellt
 - a felhasználó megkapja kérésének eredményét



Pat3 – Template rendszerek (4)

- MVC megvalósítási kérdések PHP esetén
 - a felhasználó egy browserrel dolgozik
 - a használt protokoll állapotmentes
 - a browser oldalon használt technológiák
 - Cookie, HTTP autentikáció, formok, JavaScript
 - controller technológiák
 - session kezelés, `$_POST`, `$_GET`, stb... feldolgozása
 - model technológiák
 - strukturált programozás, OOP, adatbáziskezelés, sessionök
 - view technológiák
 - XSLT, skin/template frameworkök (pl.: smarty)
 - lásd a mellékelt forráskódot!
-
-



Pat3 – feladat

- Feladat: bővítse a gyakorlaton bemutatott mini template forráskódot több adatbázismezővel (pl.: kiadás éve, kiadó neve, stb...) és egy táblázatos megjelenítési lehetőséggel is, melyben az összes adatrekord látható!
- Pluszfeladat: sessionkezelés segítségével tegye lehetővé a controlleren keresztül a view megfelelő módosításával a model adatainak bővítését (új könyvek felvitelét)!



PHP tervezési minták (4. rész)

„Pat4” blokk

nappalisok: 4. alkalom
távoktatás: 2. alkalom





Pat4 – Miről lesz szó?

- Adatbázis alapú funkciók
 - Scriptek közti kommunikáció (chat)
- Dinamikus képek és más erőforrások (GD)





Pat4 – Adatbázis alapú funkciók

- A PHP scriptek egymás közti kommunikációja
 - nem triviális, mert a scriptek nem folyamatosan futnak
 - az interakció nehézkes (állapotnélküli környezet)
 - legegyszerűbb megoldás: közös adatbázison keresztül
- Lásd a korábbi adatbázis részt!
- Lásd a mellékelt forráskódot!



Pat4 – Dinamikus képek

- A PHP program kimenete nem csak HTML lehet!
- Lásd: PHP kézikönyv LXII. fejezet!
- A dinamikus képek felhasználási területei
 - on-line grafikonok
 - captcha (robotok kiszűrésére)
 - dinamikus design célokra
- Lásd: a mellékelt forráskódot!



Pat4 – Feladat

- Feladat: egészítse ki a gyakorlaton bemutatott chat programot úgy, hogy egy (cookie, session alapú vagy HTTP) azonosítási lépés után ne csak a felhasználók IP címét, hanem az általuk választott nevet is jelenítse meg a csevegőablakban!
- Pluszfeladat: használjon dinamikus képeket a felhasználói nevek színes, grafikus megjelenítésére!

A PHP nyelv (webes alkalmazásokra)



Köszönöm a figyelmet!

