

Dr. Pál László, Sapientia EMTE, Csíkszereda

WEB PROGRAMOZÁS

3.ELŐADÁS



2020-2021
ősz

Automatikus osztálybetöltés (Autoloading), Névterek
Űrlapok kezelése

Fájlok beillesztése (include, require)

2

- Lehetőséget biztosítanak fájlok egymásba ágyazására és végrehajtására
- A `require()` és az `include()` megegyezik egymással a hibakezelését leszámítva
- Az `include()` nem ad végzetes hibát (fatal error), figyelmeztetést generál, a `require()` viszont végzetes hibát jelez
- A fájl beillesztése során a megadott fájl öröklí az `include()` helyén érvényes változó hatáskört.

require, require_once (vagy include)

3

□ *require*:

- ▣ Lehetőséget ad arra, hogy fájlt ágyazzunk be a PHP dokumentumunkba. A fájlban szereplő PHP kód úgy hajtódik végre, mintha a fődokumentum része lenne
- ▣ Használat: pld. `require("Member.php");`

□ *require_once*:

- ▣ Abban az esetben használjuk, ha egy fájlt többször kell beágyazni
 - ▣ Ha egy fájl már egyszer be lett ágyazva, többször már nem ágyazza be
- Ha a fájl nem létezik `E_COMPILE_ERROR` hibát dob és a szkript azonnal leáll

require, require_once

4

- Fejlécek, láblécek, menük beszúrásánál ajánlatos használni
- Példa:

```
function display_header($title){  
    ?>  
    <!DOCTYPE html>  
    <html>  
    <head>  
    <title><?php echo $title ?></title>  
    </head>  
    <body>  
    <?php  
}
```

```
function display_footer($content){  
    ?>  
    <footer>  
        <?php echo $content; ?>  
    </footer>  
    </body>  
    </html>  
    <?php  
}
```

```
function display_heading($tag,$content){  
    $valid_tags = array('h1','h2','h3','h4','h5','h6');  
    if(in_array($tag, $valid_tags))  
        echo sprintf("<%s>%s</%s>", $tag,$content,$tag);  
    else  
        die(sprintf("Invalid tag %s", $tag));  
}
```

require, require_once

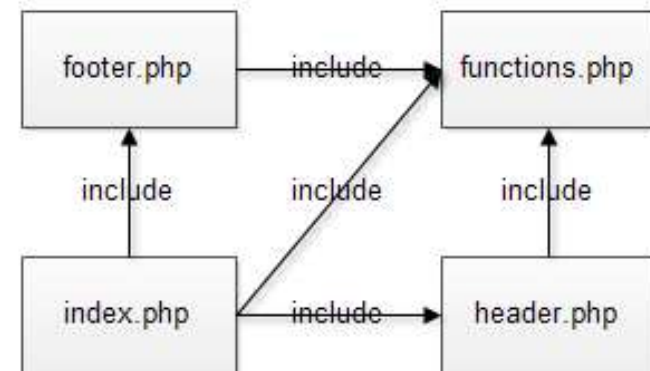
5

□ Példa:

```
include 'functions.php';  
display_header('PHP include file Demo');
```

```
include 'functions.php';  
  
$copyright = sprintf("Copyright %d © zentut.com. All Right Resevered",date("Y"))  
display_footer($copyright);
```

```
include('functions.php');  
  
include('header.php');  
  
display_heading('h1', 'This is the main heading');  
  
include('footer.php');
```



include, include_once

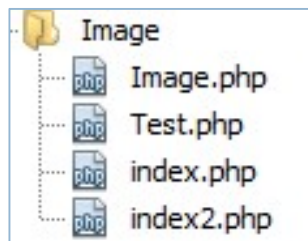
6

- Lehetőséget ad arra, hogy fájlt ágyazzunk be a PHP dokumentumunkba
- Szintaxis:
 - ▣ `include("path/to/filename");`
 - vagy
 - ▣ `include "path/to/filename";`
- Ha a fájl nem létezik `E_WARNING` hibát dob, de engedélyezi a script további futását

Autoloading

7

- Túl sok include használata rontja a program átláthatóságát, a csapatmunkát, stb.
- Autoloading:
 - ▣ Osztályok automatikus betöltése
 - ▣ Akkor töltődik be, ha egy olyan objektumot szeretnénk létrehozni, amelynek osztálya nem ismert
 - ▣ Szintaxis: `__autoload(classname)` – **elavult (deprecated)**
 - ▣ Példa:



```
class Image {  
    function __construct() {  
        echo 'Class Image loaded successfully <br />';  
    }  
}
```

```
class Test {  
    function __construct() {  
        echo 'Class Test working <br />';  
    }  
}
```

Autoloading

8

□ Példa (folytatás):

```
function __autoload($class_name) {  
    require_once $class_name . '.php';  
}  
  
$a = new Test();  
$b = new Image();
```

```
function __autoload($class_name) {  
    if (file_exists($class_name . '.php')) {  
        require_once($class_name . '.php');  
    } else {  
        throw new Exception("Unable to load $class_name.");  
    }  
}  
  
try {  
    $a = new Test();  
    $b = new Image();  
} catch (Exception $e) {  
    echo $e->getMessage(), "\n";  
}
```


Autoloading

9

- `spl_autoload_register()`
 - ▣ PHP 5.1.2-ben vezették be
 - ▣ Rugalmasabb, mint az előző változat: több betöltő függvényt (callback) tud definiálni
- Példa:

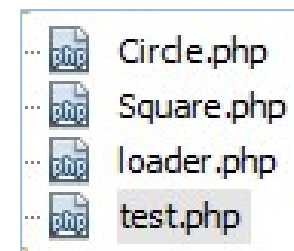
```
class Circle {  
    public function get() {  
        return 'Circle.php';  
    }  
}
```

```
class Square {  
    public function get() {  
        return 'Square.php';  
    }  
}
```

loader.php

```
spl_autoload_register(function($className) {  
    $file = $className . '.php';  
    if (file_exists($file)) {  
        include $file;  
    }  
});
```

```
include 'loader.php';  
  
$circle = new Circle;  
echo $circle->get() . "<br>";  
$square = new Square;  
echo $square->get() . "<br>";
```



Metódus láncolás (chaining)

10

- Több függvény láncszerű alkalmazása egy objektumpéldányon: az egyes függvények ugyanannak az objektumnak a referenciáját térítik vissza

- Példa:

```
class order {  
  
    public $order_status;  
  
    public function createOrder() {  
        //Apply logic to create order  
        $this->order_status = 'Order Created';  
        return $this;  
    }  
  
    public function sendOrderEmail() {  
        //Apply logic for sending email to order  
        $this->order_status = 'Email Sent';  
        return $this;  
    }  
  
    public function createShipment() {  
        //Apply logic for creating shipment  
        $this->order_status = 'Shipment Create';  
        return $this;  
    }  
}
```

```
$a = new Order();  
$a->CreateOrder()  
    ->sendOrderEmail()  
    ->createShipment();
```

Névterek (namespace)

11

- A PHP az 5.3-as verziótól támogatja
- A hosszú függvénynevek leváltására és a névütközések elkerülésére lettek bevezetve
- Névterek definiálása:
 - ▣ Minden PHP kód névterekbe szervezhető, viszont a névterek csak az alábbi 4 nyelvi elemet foglalják egységbe:
 - Osztályok
 - Interfészek
 - Függvények
 - Konstansok
 - ▣ a **namespace** kulcsszó használatával lehet definiálni

Névterek (namespace)

12

- Névterek definiálása:
 - ▣ Névteret csak a forrásfájl elején lehet meghatározni
 - ▣ Névteret több forrásfájl is leírhat, illetve egy fájlban belül több névtér is megadható
- Példa:

```
<?php
namespace MyProject1;

// PHP code for the MyProject1 namespace

namespace MyProject2;

// PHP code for the MyProject2 namespace

// Alternative syntax
namespace MyProject3 {
    // PHP code for the MyProject3 namespace
}
```

Névterek (namespace)

13

- Alnévterek definiálása:
 - ▣ A PHP-ban a névterek egymásba ágyazhatóak (ahogy a fájlrendszerben a könyvtárak), ezzel hierarchiát lehet kialakítani
- Példa: az App névtéren belül található a Lib1 névtér

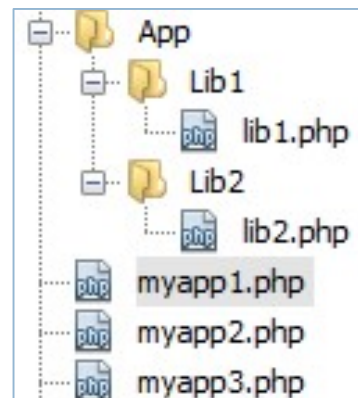
```
namespace App\Lib1;  
  
const MYCONST = 'App\Lib1\MYCONST';  
  
function MyFunction() {  
    return __FUNCTION__;  
}  
  
class MyClass {  
    static function WhoAmI() {  
        return __METHOD__;  
    }  
}
```

Névterek használata

14

- Általában a névterek a fájlrendszernek megfelelően vannak felépítve
- Névterek kiválasztási módjai:
 - ▣ Nem minősített névvel (unqualified name)
 - ▣ Minősített névvel (qualified name)
 - ▣ Teljesen minősített névvel (fully qualified name)

□ Példa:



Névterek használata

15

lib1.php

```
namespace App\Lib1;

const MYCONST = 'App\Lib1\MYCONST';

function MyFunction() {
    return __FUNCTION__;
}

class MyClass {
    static function WhoAmI() {
        return __METHOD__;
    }
}
```

lib2.php

```
namespace App\Lib2;

const MYCONST = 'App\Lib2\MYCONST';

function MyFunction() {
    return __FUNCTION__;
}

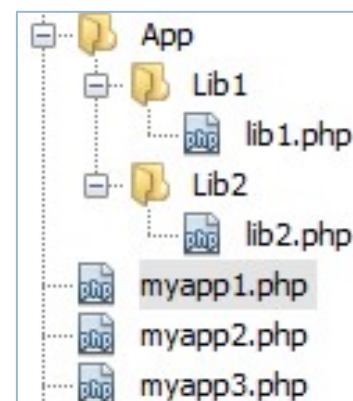
class MyClass {
    static function WhoAmI() {
        return __METHOD__;
    }
}
```

Névterek használata

16

myapp1.php

```
namespace App\Lib1;  
  
require_once('App\Lib1\lib1.php');  
  
header('Content-type: text/plain');  
echo MYCONST . "\n";  
echo MyFunction() . "\n";  
echo MyClass::WhoAmI() . "\n";
```



Ugyanabban a névtérben, nincs szükség minősítésre

```
App\Lib1\MYCONST  
App\Lib1\MyFunction  
App\Lib1\MyClass::WhoAmI
```


Névterek használata

17

□ Névter elhagyása: hiba

```
//namespace App\Lib1;  
  
require_once('App\Lib1\lib1.php');  
//require_once('App\Lib2\lib2.php');  
  
header('Content-type: text/plain');  
  
// In the same namespace we may use unqualified names  
echo MYCONST . "\n";  
echo MyFunction() . "\n";  
echo MyClass::WhoAmI() . "\n";  
echo __NAMESPACE__;
```

```
<br />  
<b>Warning</b>: Use of undefined constant MYCONST - assumed 'MYCONST' (this  
<b>10</b><br />  
MYCONST  
<br />  
<b>Fatal error</b>: Uncaught Error: Call to undefined function MyFunction()  
Stack trace:  
#0 {main}  
    thrown in <b>D:\xampp\htdocs\Namespaces\myapp1.php</b> on line <b>11</b><br
```

Névterek használata

18

- Névter elhagyása: minősített névvel hivatkozunk

```
require_once('App\Lib1\lib1.php');

header('Content-type: text/plain');

// No namespace is defined for myapp0.php so the code exists in the global space
// Any direct reference to MYCONST, MyFunction() or MyClass will fail because the
// We must therefore add a prefix of \App\Lib1 to create a fully-qualified name

echo App\Lib1\MYCONST . "\n";
echo App\Lib1\MyFunction() . "\n";
echo App\Lib1\MyClass::WhoAmI() . "\n";
```

```
App\Lib1\MYCONST
App\Lib1\MyFunction
App\Lib1\MyClass::WhoAmI
```

Névtér importálás

19

- Névtereket az **use** kulcsszóval lehet importálni
- Példa:

```
use App\Lib2;

require_once('App\Lib1\lib1.php');
require_once('App\Lib2\lib2.php');

header('Content-type: text/plain');

// We have imported the AppLib2 namespace
// We still cannot refer directly to MYCONST, MyFunction or MyClass because our
// is in the global space and PHP will look for them there
// If we add a prefix of 'Lib2', they become qualified names;
// PHP will search through the imported namespaces until it finds a match

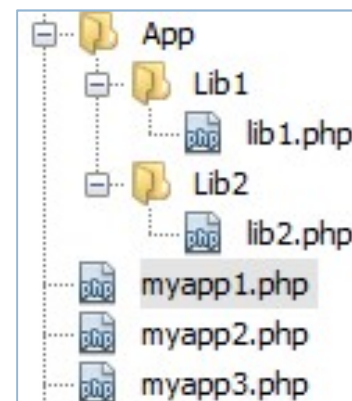
echo Lib2\MYCONST . "\n";
echo Lib2\MyFunction() . "\n";
echo Lib2\MyClass::WhoAmI() . "\n";
echo __NAMESPACE__;
```

Névtér álnevek (alias)

20

- A PHP lehetőséget nyújt a teljesen minősített névvel megadott névterek álnevesítésére
- Az álneveken keresztül könnyebben elérhetőek a kiválasztott névtér osztályai és interfészei
- Példa:

```
use App\Lib1 as L;  
use App\Lib2\MyClass as Obj;  
  
header('Content-type: text/plain');  
require_once('App\Lib1\lib1.php');  
require_once('App\Lib2\lib2.php');  
  
echo L\MYCONST . "\n";  
echo L\MyFunction() . "\n";  
echo L\MyClass::WhoAmI() . "\n";  
echo Obj::WhoAmI() . "\n";
```



Globális és környezeti változók

22

- A globális változók azok a változók, amelyeket a program legfelső szintjén, azaz a függvényeken kívül vezetünk be
- A függvényeken belül használt változók lokálisak
- A globális változókat **global** kulcsszóval kell deklarálni a függvényekben

```
$a = 1; /* globális hatáskör */

function Test() {
    echo $a; /* egy helyi változót vár */
}

Test();
```

```
$a = 1;
$b = 2;

function Osszead() {
    global $a, $b;
    $b = $a + $b;
}

Osszead();
echo $b;
```

Globális és környezeti változók

23

- Globális változók elérésének másik módja a PHP által definiált speciális `$GLOBALS` tömb használata
- Például, ha van egy **`$valtozo`** nevű változó a globális hatókörben, akkor a függvény törzsében erre a **`$GLOBALS["valtozo"]`** névvel hivatkozhatunk
- Példa:

```
$a = 1;
$b = 2;

function Osszead() {
    $GLOBALS['b'] = $GLOBALS['a'] + $GLOBALS['b'];
}

Osszead();
echo $b;
```

Globális és környezeti változók

24

- A `$GLOBALS` asszociatív tömb, ahol a globális változó neve jelenti a kulcsot, és a változó értéke a tömbelem értéke
- A `$GLOBALS` tömb minden hatáskörben létezik, mivel a `$GLOBALS` egy *szuperglobális* változó

Globális és környezeti változók

25

- Ezeket a változókat környezeti változóknak nevezzük és a PHP program tetszőleges pontján használhatjuk a **global** kulcsszó használata nélkül
- Fontosabb változók:
 - ▣ \$GLOBALS: tartalmaz valamennyi globális hatókörrel definiált változót
 - ▣ \$_SERVER: A HTTP szerverről érkező változókkal feltöltött asszociatív tömb
 - ▣ \$_GET: GET metódus által szolgáltatott adatokat tartalmazó asszociatív tömb
 - ▣ \$_POST: POST metódus által szolgáltatott adatokat tartalmazó asszociatív tömb
 - ▣ \$_FILES: POST metódussal feltöltött fájlokról tartalmaz információkat
 - ▣ \$_SESSION: az aktuális szkripthez tartozó session változókat tartalmazó asszociatív tömb
 - ▣ \$_REQUEST: tartalmazza a \$_GET, \$_POST és \$_COOKIE tömböket

Globális változók kiírása

26

- Példa a \$GLOBALS kiírására:

```
<?php
    $user1 = "Judit";
    $user2 = "István";
    $user3 = "János";
    foreach ( $GLOBALS as $kulcs=>$ertek ){
        print "\$GLOBALS[\"$kulcs\"] == $ertek<br>";
    }
?>
```

- Eredmény:

```
$GLOBALS["GLOBALS"] = Array
$GLOBALS["_POST"] = Array
$GLOBALS["_GET"] = Array
$GLOBALS["_COOKIE"] = Array
$GLOBALS["_FILES"] = Array
$GLOBALS["user1"] = Judit
$GLOBALS["user2"] = István
$GLOBALS["user3"] = János
$GLOBALS["ertek"] = János
$GLOBALS["kulcs"] = ertek
```

Űrlap (Form)

27

- Különböző adatok elküldésére használjuk
- A űrlapból érkező adatokat a **\$_GET**/**\$_POST** és a **\$_FILES** tömbökön keresztül érjük el
- Egy egyszerű űrlap:

```
<form method="get" action="feldolgoz.php">  
  <input type="text" name="nev">  
  <input type="text" name="kor">  
  <input type="submit">  
</form>
```

Űrlap (Form)

28

- **Action:** azt közli a szerverrel, hogy melyik oldalra kell váltania akkor, amikor a felhasználó az űrlap küldő gombjára kattint
- **Method:** azt a módot szabályozza, ahogy az információk elküldésre kerülnek a szerver számára
 - ▣ **GET:** a felhasználó által az űrlapmezőkbe gépelt értékeket az URL-hez kapcsolja

Példa:

www.szerver.ro/feldolgoz.php?nev=Kis+Istvan&kor=23

- ▣ **POST:** az elküldött információk nem az URL útvonalban, hanem a HTTP kérés törzsében kerülnek továbbításra

Űrlapmezők és a PHP

29

- Az űrlap adatainak elküldése után azok a `$_GET` vagy `$_POST` asszociatív tömbbe kerülnek
- Az előbbi példa esetén a szövegbeviteli mezőbe írt karakterláncok a `$_GET["nev"]` valamint a `$_GET[„kor”]` tömbbelembe kerülnek
- Tehát az elküldött adatok a `$_GET` vagy `$_POST` globális tömbökön keresztül érhetők el

Űrlapok különféle feldolgozási módja

30

- A feldolgozó szkript külön fájlban van:

```
<html>
  <head><title>Űrlap</title></head>
  <body>
    <form method="post" action="feldolgoz.php">
      Név: <input type="text" name="nev" value="">
      <br><br>
      E-mail: <input type="text" name="email" value="">
      <br><br>
      <input type="submit" value="Elküldés">
    </form>
  </body>
</html>
```

feldolgoz.php

```
<?php
    print "Név: ".$_POST["nev"];
    print "<br><br>";
    print "E-mail: ".$_POST["email"];
?>
```

Űrlapok különféle feldolgozási módja

31

- A feldolgozást ugyanabban a fájlban végezzük el, ahol az űrlap kódja található:

feldolgoz2.php

```
<html>
<head><title>Űrlap és feldolgozó (2 az 1-ben!)</title></head>
<body>
<?php
    print "Név: " . $_POST["nev"];
    print "<br><br>";
    print "E-mail: " . $_POST["email"];
?>
<br><br>
<form method="post" action="feldolgoz2.php">
    Név: <input type="text" name="nev" value="">
    <br><br>
    E-mail: <input type="text" name="email" value="">
    <br><br>
    <input type="submit" value="Elküldés">
</form>
</body>
</html>
```

Eredmény:

Név: Kis Istvan

E-mail: kis@alma.ro

Név:

E-mail:

Elküldés

Űrlapok különféle feldolgozási módja

32

□ Változó létezésének vizsgálata

```
<?php
if (isset($_POST["nev"]) && isset($_POST["email"])){
    $adatok = "Név: " . $_POST["nev"] . "<br><br>E-mail: " . $_POST["email"];
}
else{
    $adatok = "";
}
print $adatok;
?>
```

- **isset(\$valtozo)** függvény: egy változó létezését vizsgálja, amely TRUE értéket ad vissza, ha a változó létezik, FALSE-t ha nem
- **\$PHP_SELF**: aktuális állományt jelöli

<form action="<?php print \$PHP_SELF?>" method="POST">

Beviteli mezők

33

- Szövegmező: egy ilyen elembe egysoros szöveget írhatunk

```
<input type="text" name="azonosító" value="szöveg">
```

- ▣ Elérése: `$_POST['azonosító']`
- ▣ Kezdőérték: `value` mező

- Jelszóbeviteli mező (password)

```
<input type="password" name="azonosító" value="szöveg">
```

- ▣ Paramétereinek funkciója ugyanaz, mint a szövegbeviteli mező esetén.

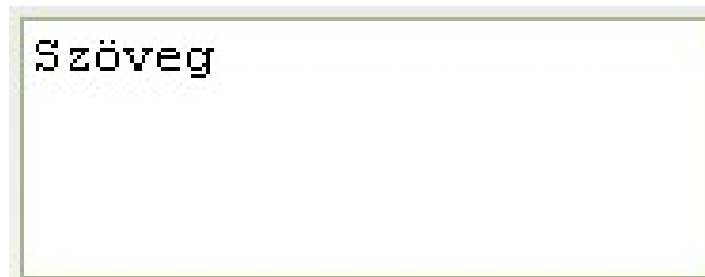
Beviteli mezők

34

□ Szövegterület:

```
<textarea name="azonosító" rows="5" cols="20">Szöveg</textarea>
```

- Elérése: `$_POST['azonosító']`
- Kezdőérték: tag-ek közé írt szöveg
- **rows**: sorok, **cols**: szélesség



Beviteli mezők

35

□ Szövegterület (Példa):

```
<html>
<head><title></title></head>
<body>
<form method="POST" action="szovegterulet.php">
Melyek a kedvenc webhelyeid?
<textarea name="WebHelyek" cols="50" rows="5">
http://
http://
http://
http://
</textarea>
<br><br><br>
<input type="submit" value="Elküld">
</form>
</body>
</html>
```

szovegterulet.php

```
<html>
<head><title></title></head>
<body>
A kedvenc webhelyeid:
<?php
echo $_POST['WebHelyek'];
?>
</body>
</html>
```

Beviteli mezők

36

□ Szövegterület (Példa):

Melyek a kedvenc webhelyeid?

```
http://honline.ro  
http://sapiencia.ro  
http://erdely.ma  
http://transindex.ro|
```

Elküld

Eredmény:

A kedvenc webhelyeid: `http://honline.ro http://sapiencia.ro http://erdely.ma http://transindex.ro`

Beviteli mezők

37

□ Rádiógombok (Radio):

```
<input type="radio" name="azonosító" value="1">
```

```
<input type="radio" name="azonosító" value="1" checked>
```

- Elérése: `isset($_POST['azonosító'])`
- Kezdőérték: ha **checked**, akkor ki van választva
- **value**: a csoportban való megkülönböztetésre használjuk
- Egy csoportba tartozó rádiógombok összekötése érdekében ugyanazt a nevet kell adni a csoport minden gombjának

□ Példa:

```
<input name="Kerdes1" type="radio" value="Portó">
```

```
<input name="Kerdes1" type="radio" value="Lisszabon">
```

```
<input name="Kerdes1" type="radio" value="Madrid">
```

Beviteli mezők

38

□ Rádiógombok (Radio):

- Ha az egyik elem ki van választva, akkor az űrlapelem name paraméterével megegyező kulcsú **\$_POST** tömbbelembe kerül a kiválasztott űrlapelem value paramétere

□ Példa:

```
<input name="Kerdes1" type="radio" value="Portó">
```

```
<input name="Kerdes1" type="radio" value="Lisszabon"  
checked="checked">
```

```
<input name="Kerdes1" type="radio" value="Madrid">
```

⇒ `$_POST["Kerdes1"] = "Lisszabon"`

- Ha nincs alapértelmezett érték beállítva, nem jön létre a megfelelő **\$_POST** változó (pld. `$_POST[„Kerdes1"] = null` lesz)

Beviteli mezők

39

□ Rádiógombok (Példa):

Mi Portugália fővárosa?

☐ Portó
☐ Lisszabon
☐ Madrid

Tipp elküldése

Tévedsz, nem Portó a helyes válasz!

Mi Portugália fővárosa?

☐ Portó
☐ Lisszabon
☐ Madrid

Tipp elküldése

Eltaláltad, Lisszabon a helyes válasz!

Mi Portugália fővárosa?

☐ Portó
☐ Lisszabon
☐ Madrid

Tipp elküldése

```
<?php
if (isset($_POST['elkuldott'])) {
    if ($_POST['kerdes1'] == "Lisszabon") {
        echo "Eltaláltad, $_POST[kerdes1] a helyes válasz!<hr>";
    }
    if ($_POST['kerdes1'] != "Lisszabon") {
        echo "Tévedsz, nem $_POST[kerdes1] a helyes válasz!<hr>";
    }
}
?>
```

Beviteli mezők

40

□ Jelölőnégyzet (checkbox):

`<input type="checkbox" name="azonosító" value="1">`

`<input type="checkbox" name="azonosító" value="1" checked>`

- Elérése: `isset($_POST['azonosító'])`
- Kezdőérték: ha **checked**, akkor ki van választva
- **value**: a csoportban való megkülönböztetésre használjuk

□ Példa

```
<form method="POST" action="jelolonegyzet.php">
  <p> Írtál már, PHP programot?
  <input type="checkbox" name="Valasz">
  </p>
  <input type="submit" value="Elküld">
</form>
```

Írtál már, PHP programot? ☐

Elküld

Notice: Undefined index: Valasz in

jelolonegyzet.php

```
<?php
echo $_POST['Valasz'];
?>
```

Írtál már, PHP programot? ☒

Elküld

on

Beviteli mezők

41

□ Példa (folytatás):

```
<form method="POST" action="jelolonegyzet.php">
  <p> Írtál már, PHP programot?
    <input type="checkbox" name="Valasz">
  </p>
  <input type="submit" value="Elküld">
</form>
```

Írtál már, PHP programot? ☐

Elküld

Érték:

Jelölőnégyzetet nem jelölted be!

```
if (empty($_POST['Valasz'])) {
    $_POST['Valasz'] = "";
}
echo "Érték: " . $_POST['Valasz'] . " <br /><br />";

if ($_POST['Valasz'] == 'on') {
    echo "<b>Jelölőnégyzetet bejelölted!</b><br />";
} else {
    echo "<b>Jelölőnégyzetet nem jelölted be!</b><br />";
}
```

Beviteli mezők

42

- **Jelölőnégyzet (checkbox):** csoportos használat esetén a name paraméterben jelezni kell, hogy tömbként kívánjuk továbbítani az értékeket
- **Példa:**

```
<form method="POST" action="jelolonegyzetek.php">
  <input type="checkbox" name="nyelv[]" value="html" /> HTML
  <input type="checkbox" name="nyelv[]" value="css" /> CSS
  <input type="checkbox" name="nyelv[]" value="php" /> PHP
  <br>
  <input type="submit" value="Elkuld">
</form>
```

```
<?php
foreach ($_POST["nyelv"] as $nyelv){
    print $nyelv . "<br>";
}
?>
```

```
html
css
php
```

Beviteli mezők

43

□ Listadobozok:

▣ Egy elem kiválasztása

- Elérés: `$_POST['Arkategoria']`

```
<select name="Arkategoria">
  <option>1 millió Ft alatt</option>
  <option>1-2 millió Ft</option>
  <option>2-5 millió Ft</option>
  <option>5 millió Ft felett</option>
</select>
```

▣ Több elem kiválasztása

- Elérés: `$_POST['MotorMeret'][index]`

```
<select name="MotorMeret[]" multiple>
  <option>1,0 L</option>
  <option>1,4 L</option>
  <option>1,6 L</option>
</select>
```

Beviteli mezők

44

- **Rejtett űrlapmezők:** információk rejtett átadását teszik lehetővé.

```
<input type="hidden" name="azonosító" value="szöveg">
```

- Nem jeleníthetők meg

- Példa:

```
<?php
$Uzenet1="Ez egy láthatatlan üzenet";
echo "<form method='GET' action='valami.php'>";
echo "<input type='hidden' name='Rejtett1' value=' $Uzenet1'>";
echo "<input type='submit' value='Elküld'>";
echo "</form>";
?>
```

Űrlapokból származó értékek felhasználása

45

□ Példa:

Pénzeszsák Kereskedelmi Bank Rt. hitelkérő űrlap

Keresztnév: Vezetéknév: Életkor:

Cím:

Mennyi a jelenlegi havi jövedelme?

Mennyi hitelre lenne szüksége?

☒ 1 millió forintos hitelcsomagunk kamata 8,0 %
☐ 5 millió forintos hitelcsomagunk kamata 11,5 %
☐ 10 millió forintos hitelcsomagunk kamata 15,0 %

Űrlapokból származó értékek felhasználása

46

□ Példa:

Szöveg mezők

```
Keresztnév:  
<input name="Keresztnev" type="text">  
Vezetéknév:  
<input name="Vezeteknev" type="text">  
Életkor:  
<input name="Eletkor" type="text" size="3"><br><br>  
Cím:  
<textarea name="Cim" rows="4" cols="40">  
</textarea>
```

Lista

```
Mennyi a jelenlegi havi jövedelme?  
<select name="Jovedelem">  
<option value=0>100 000 forintnál kevesebb </option>  
<option value=100000>100 000 - 200 000 Ft</option>  
<option value=200000>200 000 - 300 000 Ft</option>  
<option value=300000>300 000 forintnál több</option>  
</select>
```

Rádiógombok

```
Mennyi hitelre lenne szüksége?<br><br>  
<input name="Hitel" type="radio" value="1000000">  
1 millió forintos hitelcsomagunk kamata 8,0 %  
<br>  
<input name="Hitel" type="radio" value="5000000">  
5 millió forintos hitelcsomagunk kamata 11,5 %  
<br>  
<input name="Hitel" type="radio" value="10000000">  
10 millió forintos hitelcsomagunk kamata 15,0 %
```

Űrlapokból származó értékek felhasználása

47

□ Példa:

PHP szkript

```
<?php
$JovedelemHatar = $_POST['Jovedelem'] * 12 / 5;
$KorHatar = ($_POST['Eletkor'] / 10 - ($_POST['Eletkor'] % 10) / 10) - 1;
$HitelHatar = $JovedelemHatar * $KorHatar;
echo "Igényelt hitelösszeg: $_POST[Hitel]<br>";
echo "Teljesíthető hitelösszeg: $HitelHatar<br><br>";
if ($_POST['Hitel'] <= $HitelHatar) echo "Kedves $_POST[Keresztnev] $_POST[Vezeteknev],";
if ($_POST['Hitel'] > $HitelHatar) echo "Kedves $_POST[Keresztnev] $_POST[Vezeteknev],";
?>
```

Űrlapokból származó értékek felhasználása

48

□ 2. Példa:

Roncsderbi autókölcsönző

Keresztnév: Vezetéknév: Életkor:

Cím:

qwewq

Rendelkezik érvényes jogosítvánnyal? ☒

Űrlapokból származó értékek felhasználása

49

□ 2. Példa:

```
<form method="post" action="auto.php">
<input type="hidden" name="elkuldott" value="true">
Keresztnév: <input name="keresztnev" type="text">
Vezetéknév: <input name="vezeteknev" type="text">
Életkor: <input name="eletkor" type="text"size="3"><br><br>
Cím:
<textarea name="cim" rows=4 cols=40>
</textarea>
<br><br>
Rendelkezik érvényes jogosítvánnyal?
<input name="jogositvany" type="checkbox">
<br><br>
<input type="submit" value="Kérelem elküldése">
</form>
```

Űrlapokból származó értékek felhasználása

50

□ 2. Példa:

```
<?php
if (isset($_POST['elkuldott'])) {
    if ($_POST['eletkor'] > 20 and $_POST['jogositvany'] == "on") {
        echo ("Kölcsönzési igényét elfogadtuk.<hr>");
    }
    if ($_POST['eletkor'] < 21 or $_POST['jogositvany'] == "") {
        echo ("Sajnos nem áll módunkban autót kölcsönözni Önnek.<hr>");
    }
}
?>
```

Dinamikus oldal készítése

51

□ Példa:

```
<form method="POST" action="dinamikus.php">
  <input type="hidden" name="elkuldott" value="true">
  Hány gyermeke van?
  <input name="szam" type="text">
  <br>
  <br>
  <input type="submit" value="Szám elküldése">
  <br>
</form>
```

Hány gyermeke van?

Gépelje be a(z) 1. gyermeke nevét:

Nyomja meg a gombot a továbblépéshez!

Dinamikus oldal készítése

52

```
<?php
if (isset($_POST['elkuldott'])) {
    echo "<form method='POST' action='dinamikus.php'>";
    for ($szamlalo = 0; $szamlalo < $_POST['szam']; $szamlalo++) {
        $eltolas = $szamlalo + 1;
        echo "<br>Gépelje be a(z) $eltolas. gyermeke nevét:<br>";
        echo "<input name='gyerek[]' type='text'>";
    }
    echo "<br>Nyomja meg a gombot a továbblépéshez!<br>";
    echo "<input type='submit' value='Következő'>";
    echo "<input type='hidden' name='elkuldott01' value='true'></form>";
} else {
    if (isset($_POST['elkuldott01'])) {
        $szaml = 0;
        echo "Gyermekei neve:";
        do {
            $gyerekek_neve = $_POST['gyerek'][$szaml];
            echo "<br><b>$gyerekek_neve</b>";
            $ures_ell = $gyerekek_neve;
            $szaml = $szaml + 1;
        } while ($ures_ell != "");
        if ($szaml == 1) {
            echo "Nem megfelelő";
        }
    }
}
?>
```


Űrlap beállítása

54

- Paraméterek:
 - ENCTYPE="multipart/form-data"

```
<form action="f0.php" method="post" enctype="multipart/form-data">
```

- Megjelenítés böngészőben (Firefox, Chrome):



Browse_ No file selected. Upload!

A fájlválasztó mező

55

- A fájl kiválasztásához fájlválasztó mezőt (type="file") használunk:

```
<input type="file" name="kep" />
```

- Ebben az esetben csak egy állomány választható ki

Az űrlap elküldése

56

- Az űrlap elküldés után a feltöltött fájl egy ideiglenes mappába kerül, az állomány jellemzői pedig a `$_FILES` szuperglobális tömb változóján keresztül érhetők el

□ **Példa:** `print_r($_FILES["kep"])`

```
Array (
    [name] => smiley.jpeg (a fájl neve)
    [type] => image/jpeg (a fájl típusa)
    [tmp_name] => G:\xampp\tmp\phpCF80.tmp (temp. mappa)
    [error] => 0 (hiba kódja)
    [size] => 6602 (fájl méret bájtban)
)
```


A `$_FILES` globális tömb

57

- A `$_FILES` globális változó tartalma:
 - ▣ `$_FILES["file"]["name"]` – a feltöltött fájl neve
 - ▣ `$_FILES["file"]["type"]` - a feltöltött fájl típusa
 - ▣ `$_FILES["file"]["size"]` - a feltöltött fájl mérete
 - ▣ `$_FILES["file"]["tmp_name"]` - a feltöltött fájl ideiglenes neve
 - ▣ `$_FILES["file"]["error"]` - a feltöltés folyamán jelentkező hiba

A feltöltött fájl mentése

58

- Általában a feltöltött fájlt áthelyezzük az ideiglenes helyéről egy előre megadott mappába
- Példa: fájl áthelyezése az **uploads** mappába a **tmp_name** nevű mappából

```
move_uploaded_file($_FILES["kep"]["tmp_name"],  
    "uploads/" . $_FILES["kep"]["name"]);
```

A feltöltéssel kapcsolatos ellenőrzések

59

□ Ellenőrizni szoktuk:

- ▣ A feltöltéskor keletkező hibákat
- ▣ A feltöltött fájl típusát
- ▣ A feltöltött fájl méretét

A feltöltés sikeressége

60

- A `$_FILES` tömbben található **error** mező lehetséges értékei
 - ▣ 0: sikeres feltöltés
 - ▣ nem nulla, ha valami gond volt (pld. 1,2: maximális fájlméret meghaladása, 3: parciális feltöltés, stb.)
- Példa:

```
if ($_FILES["kep"]["error"] != 0) {  
    die("Hiba a feltöltés során");  
}
```

A feltöltött fájl mérete

61

- Fontos a fájl méret ellenőrzése, különben tetszőleges méretű fájlokkal is lehetne próbálkozni
- Példa: 1 MB-nál nagyobb fájlok tiltása

```
if($_FILES["kep"]["size"] > 1024*1024) {  
    die("Túl nagy méretű fájl");  
}
```

A feltöltött fájl típusa

62

□ Példa:

```
$valid_formats = array("image/jpg", "image/png", "image/bmp");  
if(!in_array($_FILES["kep"]["type"], $valid_formats)) {  
    .....  
    die("Csak JPG, PNG vagy BMP!");  
}
```

Több fájl feltöltése

63

□ Példa:

```
<form action="f3.php" method="post" enctype="multipart/form-data">
  Send these files:<br />
  <input name="userfile[]" type="file" /><br />
  <input name="userfile[]" type="file" /><br />
  <input type="submit" value="Upload!" />
</form>
```

Array (

[name] => Array ([0] => smiley1.jpeg [1] => smiley2.jpeg)

[type] => Array ([0] => image/jpeg [1] => image/jpeg)

[tmp_name] => Array ([0] => G:\xampp\tmp\php3AA5.tmp [1] =>
G:\xampp\tmp\php3AA6.tmp)

[error] => Array ([0] => 0 [1] => 0)

[size] => Array ([0] => 6602 [1] => 6602)

)

Több fájl feltöltése

64

□ Vagy:

```
<form action="f4.php" method="post" enctype="multipart/form-data">
    <input type="file" id="file" name="files[]" multiple="multiple"/>
    <input type="submit" value="Upload!" />
</form>
```

□ Feldolgozás:

```
foreach ($_FILES['files']['name'] as $f => $name) {
    if ($_FILES['files']['error'][$f] == 4) {
        continue; // Skip file if any error found
    }
    if ($_FILES['files']['error'][$f] == 0) {
        if ($_FILES['files']['size'][$f] > $max_file_size) {
            $message[] = "$name is too large!.";
            continue; // Skip large files
        }
        elseif( ! in_array(pathinfo($name, PATHINFO_EXTENSION), $valid_formats) ){
            $message[] = "$name is not a valid format";
            continue; // Skip invalid file formats
        }
    }
}
```


Kérdések

65

- Mi a szerepe az include és require függvényeknek?
- Mi a különbség az előző kettő között?
- Mi a szerepe az autoloading-nak?
- Milyen problémákat old meg a névterek használata?
- Mi jellemzi a globális változókat?
- Milyen űrlap feldolgozási lehetőségek vannak?
- Hogyan dolgozzuk fel a radio és checkbox űrlap elemeket?
- Mi a szerepük a rejtett űrlapmezőknek?
- Fájl feltöltésének menete?

Könyvészet

66

- <http://nyelvek.inf.elte.hu>
- <http://hu.wikipedia.org>
- <https://www.w3schools.com/php>