3. ÓRA

Első PHP oldalunk

A PHP telepítése és beállítása után eljött az ideje, hogy elkezdjünk vele dolgozni. Ebben az órában elkészítjük az első programunkat és elemezzük a kódot. Az óra végére képesek leszünk olyan oldalak készítésére, amelyek HTML és PHP kódot is tartalmaznak.

Ebben az órában a következőket tanuljuk meg:

- Hogyan hozzunk létre, töltsünk fel és futtassunk egy PHP programot?
- Hogyan vegyítsünk HTML és PHP kódot egy dokumentumon belül?
- Hogyan tehetjük a kódunkat olvashatóbbá megjegyzések használatával?

Első programunk

Vágjunk a közepébe és kezdjük egy kész PHP oldallal! A kezdéshez nyissuk meg kedvenc szövegfájl-szerkesztőnket! A HTML dokumentumokhoz hasonlóan a PHP állományok is formázás nélküliek, tisztán szövegfájlok. Ez azt jelenti, hogy bármilyen szövegfájl szerkesztésére íródott programmal készíthetünk PHP állományokat. Ilyenek például a Windows Jegyzettömbje, a Simple Text vagy a BBEdit MacOS alatt, vagy a VI és az Emacs UNIX esetében. A legnépszerűbb HTML-szerkesztők általában nyújtanak valamilyen különleges támogatást PHP-szerkesztéshez is. Ilyenek például a kódszínezés, a kifejezésszerkesztő vagy a tesztelő.

Gépeljük be a 3.1. programot, és mentsük `elso.php` néven.

3.1. program Az első PHP program

```
1: <?php
2: print ("Hello Web!");
3: ?>
```

A 3.1. ábra a 3.1. program kódját mutatja a KEdit programban.

3.1. ábra

Első PHP oldalunk begépelve a KEdit programban

A kiterjesztés igen fontos, mivel ez jelzi a kiszolgáló számára, hogy a fájl PHP kódot tartalmaz és futtatásához meg kell hívni a feldolgozót. Az alapbeállítású PHP kiterjesztés a `.php`, ez azonban a kiszolgáló beállításainak módosításával megváltoztatható. Ezzel az előző órában részletesen foglalkoztunk.

Ha nem közvetlenül a kiszolgálóprogramot futtató gépen dolgozunk, feltehetően az FTP nevű szolgáltatást kell használnunk arra, hogy PHP oldalunkat a kiszolgálóra juttassuk. A Windowsban többek között a WS-FTP használható erre a célra, MacOS használata esetén a Fetch lehet hasznos.

Ha sikerült elhelyeznünk a dokumentumot a kiszolgálón, elérhetjük egy webböngésző segítségével. Ha minden rendben ment, az ablakban a program kimenetét láthatjuk. A 3.2. ábra az `elso.php` kimenetét mutatja.

3.2. ábra

Siker: a 3.1. program kimenete

Ha a PHP-t nem sikerült telepíteni a kiszolgálóra vagy nem a megfelelő kiterjesztést használtuk, feltehetően nem az előző ábrán látható kimenetet kapjuk. Ebben az esetben általában a PHP program forráskódját látjuk viszont. Ezt mutatja a 3.3. ábra.

3.3. ábra

Kudarc: a kiterjesztés nem azonosítható

Ha ez történik, először ellenőrizzük a mentett fájl kiterjesztését. A 3.3. ábrán látható lapot véletlenül `elso.nphp` névvel mentettük. Ha a kiterjesztéssel nincs probléma, ellenőriznünk kell, hogy a PHP sikeresen felkerült-e a rendszerre és hogy a kiszolgálót beállítottuk-e a megfelelő kiterjesztés felismerésére. Ezekkel a kérdésekkel az előző órában foglalkoztunk részletesen.

Miután sikerült feltölteni és kipróbálni első programunkat, kielemezhetjük az imént begépelt kódot.

PHP blokkok kezdése és befejezése

Amikor PHP oldalakat írunk, tudatnunk kell a feldolgozóval, mely részeket hajtsa végre. Ha nem adjuk meg, hogy a fájl mely részei tartalmaznak PHP kódot, az értelmező mindent HTML-nek tekint és változtatás nélkül továbbküldi a böngésző számára. A 3.1. táblázat a PHP kód blokkjainak kijelölésére szolgáló elemeket sorolja fel:

3.1. táblázat PHP blokkok kezdő- és záróelemei

Elnevezés	*Kezdőelem*	*Záróelem*
Hagyományos	`<?php`	`?>`
Rövid	`<?`	`?>`
ASP stílusú	`<%`	`%>`
Script elem	`<SCRIPT LANGUAGE="PHP">`	`</SCRIPT>`

A 3.1. táblázatban látható lehetőségek közül csak az első és az utolsó áll minden beállítás esetén rendelkezésre. A rövid és ASP stílusú elemeket engedélyezni kell a `php.ini` fájlban. Az előző órában tárgyaltuk a beállítás módszerét.

A rövid kezdőelemek felismerését a `short_open_tag` beállítással engedélyezhetjük, ha „On” állapotba tesszük:

```
short_open_tag = On
```

A rövid kezdőelemek alapbeállításban engedélyezettek, ezért nem kell szerkesztenünk a `php.ini` fájlt, hacsak valaki előzőleg ki nem kapcsolta.

Az ASP stílusú elemek engedélyezéséhez az `asp_tags` beállítást kell bekapcsolni:

```
asp_tags = On
```

Ez akkor lehet hasznos, ha olyan termékkel készítünk PHP oldalakat, amely törli a PHP blokkokat, de az ASP részeket érintetlenül hagyja.

A `php.ini` fájl módosítása után lehetőségünk van választani a bekapcsolt típusok közül. Rajtunk múlik, hogy melyik elemet választjuk, de ha XML-lel is szeretnénk dolgozni, mindenképpen a hagyományos formát kell alkalmaznunk és le kell tiltanunk a rövid stílust. Mivel a hagyományos forma minden rendszeren rendelkezésre áll, a fejlesztők programjaikban általában ezt alkalmazzák, így a könyvben is ezzel fogunk találkozni.

Nézzük meg, hogyan fest egy PHP fájl, amely a fent említett formákat használja az előző program kibővítéseként. Ha engedélyeztük, bármely négy kezdő- és záróelemet használhatjuk a programban:

```
<?
print ("Hello Web!");
?>

<?php
print ("Hello Web!");
?>

<%
print ("Hello Web!");
%>

<SCRIPT LANGUAGE="PHP">
print ("Hello Web!");
</SCRIPT>
```

Ha PHP-ben egysoros kódot írunk, nem kell külön sorba tennünk a kezdő- és záróelemeket, illetve a programsort:

```
<?php print("Hello Web!"); ?>
```

Miután megtanultuk, hogyan határozhatunk meg egy PHP kódblokkot, nézzük meg még közelebbről a 3.1. programot.

A print() függvény

A `print()` függvény kiírja a kapott adatokat. A legtöbb esetben minden, ami a `print()` függvény kimenetén megjelenik, a böngészőhöz kerül. A függvények olyan parancsok, amelyek valamilyen műveletet végeznek, többnyire attól függően, hogy milyen adatokat kapnak. A függvényeknek átadott adatokat majdnem mindig zárójelbe kell tennünk a függvény neve után. Ebben az esetben a `print()` függvénynek egy karakterláncot adtunk át. A karakterláncokat mindig (egyes vagy kettős) idézőjelbe kell tenni.

A függvények általában megkövetelik, hogy zárójeleket helyezzünk a nevük után, akár küldünk adatokat számukra, akár nem. A `print()` ebből a szempontból kivételes, mivel hívása esetén a zárójelek elhagyhatók. Ez a `print()` függvény megszokott formája, ezért a továbbiakban ezt alkalmazzuk.

A 3.1. program egyetlen sorát pontosvesszővel fejeztük be. A pontosvessző tudatja a feldolgozóval, hogy az utasítás véget ért.

ÚJDONSÁG Az utasítás a feldolgozónak adott parancs. Általánosságban a PHP számára azt jelenti, mint az írott vagy beszélt nyelvben a mondat. Az utasításokat a legtöbb esetben pontosvesszővel kell lezárni, a mondatokat írásjellel. Ez alól kivételt képeznek azok az utasítások, amelyeket más utasítások vesznek körül, illetve azok, amelyek blokk végén állnak. A legtöbb esetben azonban az utasítás végéről lefelejtett pontosvessző megzavarja az elemzőt és hibát okoz.

Mivel a 3.1. programban az egyetlen megadott utasítás a blokk végén áll, a pontosvessző elhagyható.

HTML és PHP kód egy oldalon

3.2. program PHP program HTML tartalommal

```
 1: <html>
 2: <head>
 3: <title>3.2. program PHP program HTML
    tartalommal</title>
 4: </head>
 5: <body>
 6: <b>
 7: <?php
 8: print ("Hello Web!");
 9: ?>
10: </b>
11: </body>
12: </html>
```

Látható, hogy HTML kód beépítése a PHP oldalakba egyszerűen a HTML tartalom begépeléséből áll. A PHP feldolgozó figyelmen kívül hagy mindent a PHP nyitó- és záróelemeken kívül. Ha a 3.2. programot megnézzük böngészővel, a „Hello Web!” szöveget látjuk vastagon, mint ahogy ez a 3.4. ábrán megfigyelhető. Ha a dokumentum forrását is megnézzük, ahogy a 3.5. ábra mutatja, a kód hagyományos HTML dokumentumnak tűnik.

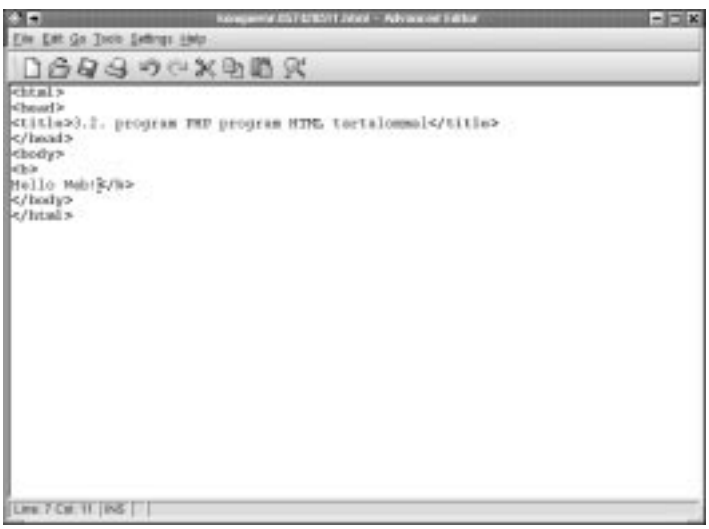
Egy dokumentumban a HTML elemek közé tetszőleges számú PHP kódblokk írható. Bár több kódblokkot helyezhetünk el egy dokumentumon belül, ezek együttesen alkotnak egy programot. Bármi, amit egy megelőző blokkban határoztunk meg, (változók, függvények vagy osztályok) a dokumentumon belül elérhető lesz a későbbi blokkokban is. A több együttműködő, összességében egy nagyobb programot megvalósító PHP fájlt nevezzük PHP alkalmazásnak.

3.4. ábra

Ha a 3.2 programot megnézzük böngészővel, a „Hello Web!" szöveget látjuk vastagon

3.5. ábra

Ha a dokumentum forrását nézzük, a kód hagyományos HTML dokumentumnak tűnik



Megjegyzések a PHP kódokban

Az íráskor átlátható programkód hat hónappal később reménytelenül kuszának tűnhet. Ha azonban megjegyzéseket fűzünk a kódhoz, miközben írjuk, a későbbiekben sok időt takaríthatunk meg magunknak és más programozóknak, akik az adott kóddal fognak dolgozni.

ÚJDONSÁG A megjegyzések a kódban található olyan szövegrészek, amelyeket a feldolgozó figyelmen kívül hagy. Segítségükkel olvashatóbbá tehetjük a programot és jegyzeteket fűzhetünk hozzá.

A PHP egysoros megjegyzései két perjellel (`//`) vagy egy kettőskereszt karakterrel (`#`) kezdődhetnek. Az ezen jelzések után található szöveget a PHP-értelmező mindig figyelmen kívül hagyja, a sor vagy a PHP blokk végéig.

```
// Ez megjegyzés
# Ez is megjegyzés
```

A többsoros megjegyzések egy perjelet követő csillag karakterrel kezdődnek (`/*`) és egy csillagot követő perjellel (`*/`) fejeződnek be.

```
/*
Ez egy megjegyzés.
A PHP-értelmező
ezen sorok egyikét
sem fogja feldolgozni.
*/
```

Összefoglalás

Mostanra rendelkezésünkre állnak azok az eszközök, melyekkel képesek vagyunk egy egyszerű PHP program futtatására egy megfelelően beállított kiszolgálón.

Ebben az órában elkészítettük első PHP programunkat. Láttuk, hogyan használható egy szöveges szerkesztő, hogy létrehozzunk és mentsünk egy PHP dokumentumot. Elemeztük a négy rendelkezésre álló blokk kezdő- és záróelemet. Megtanultuk, miként kell használnunk a `print()` függvényt, hogy a böngésző számára kimenetet küldjünk, majd összeállítottunk egy HTML és PHP kódot is tartalmazó állományt. Végül tanultunk a megjegyzésekről és arról, hogyan építhetjük be ezeket a PHP dokumentumokba.

Kérdések és válaszok

Melyik a legjobb kezdő- és záróelem?
Ez nagyrészt saját döntésünkön múlik. A hordozhatóság szem előtt tartásával a hagyományos `<?php ?>` megoldás a legjobb döntés. A rövid kezdőelemek alapesetben engedélyezettek és rendelkeznek a tömörség előnyös tulajdonságával.

Milyen szerkesztőprogramokat kell elkerülni a PHP kódok készítésekor?
Ne használjunk olyan szövegszerkesztőket, amelyek saját formátummal rendelkeznek, mint a Microsoft Word. Ha ilyen típusú szerkesztővel mentjük szöveges fájlként az elkészült dokumentumot, a kódban megbújó rejtett karakterek biztos gondot fognak okozni.

Mikor kell megjegyzéseket fűzni a kódokhoz?
Ez is nagyrészt a programozó döntésén múlik. A rövid programokhoz nem érdemes magyarázatot fűzni, mivel ezek még hosszú idő után is érthetőek maradnak. Ha a program azonban akár csak egy kicsit is bonyolult, már javasolt megjegyzésekkel bővíteni. Ez hosszú távon időt takarít meg számunkra és kíméli idegeinket.

Műhely

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

Kvíz

1. A felhasználó böngészőjével elolvashatja-e a PHP kódot?
2. Adjuk meg a PHP hagyományos kezdő- és záróelemeit!
3. Adjuk meg az ASP stílusú kezdő- és záróelemeket!
4. Adjuk meg a Script kezdő- és záróelemeket!
5. Mely függvény használatos a böngésző számára küldött adatok kiírásához?

Feladatok

1. Gyakoroljuk a PHP oldalak elkészítését, feltöltését és futtatását!