WEBTERVEZÉS – GYAKORLATI JEGYZET

6. gyakorlat

Szerveroldali webprogramozás, PHP

Készítették:

Cservenák Bence

Farkas Anikó

Savanya Sándor

FIGYELEM!

A jegyzet folyamatosan készül, így előfordulhatnak benne apróbb hibák, hiányosságok, elírások. Ha valaki esetleg ilyet találna, kérem, írjon a `Cservenak.Bence@stud.u-szeged.hu` címre, hogy mihamarabb javíthassuk.

1. A webprogramozás motivációja

Az eddigi gyakorlatokon a HTML és CSS nyelvekkel foglalkoztunk.

Ezek nem tekinthetők programozási nyelveknek, hiszen hiányzik belőlük az alapvető programozási logika (nincsenek bennük például változók, operátorok, vezérlési szerkezetek stb.).

A webfejlesztés során (főként összetettebb weboldalak készítésekor) viszont gyakran felmerül az igény, hogy a weboldal működésébe programozási logikát vigyünk. Erre a célra szolgál a **webprogramozás**.

2. A web működése

A webes programok tipikusan két részből állnak: egy **kliens**ből és egy **szerver**ből.

A kliensprogram általában egy böngésző, ami a gépünkön fut, és adatokat (pl. HTML-t és CSS-t) jelenít meg. A szerver jellemzően egy távoli webszerver (pl. a Google szervere).

A kliens- és szerver a hálózaton (Internet) keresztül kommunikál egymással, a **HTTP** (HTTPS) hálózati protokoll segítségével. A kliens HTTP kérést küld a szervernek, amit a szerver fogad, feldolgoz, és HTTP válasszal reagálhat rá.

3. Kliens- és szerveroldali webprogramozás

Az általunk írt kódok futhatnak a kliensen, illetve a szerveren is. Ettől függően megkülönböztetünk **kliensoldali** és **szerveroldali webprogramozás**t.

Az alábbi táblázat tartalmazza a kétféle webprogramozás rövid összehasonlítását.

	Kliensoldali webprogramozás	Szerveroldali webprogramozás
Hol futnak a kódok?	a felhasználó gépén	egy webszerveren
Mire használják?	• látványelemek (pl. animációk) megvalósítása • űrlapadatok helyességének ellenőrzése (pl. e-mail cím formátuma)	• bizalmas adatok (pl. jelszavak) kezelése • adatbáziskezelés, fájlkezelés, jogosultságok kezelése
A programkód az oldal forráskódjában…	…látható	…nem látható
Biztonságos az adatok tárolása?	nem	igen
Programozási nyelvek	JavaScript	PHP, Python

A gyakorlatban a kliens- és szerveroldali programozás sokszor együtt használatos. Ennek oka, hogy bizonyos feladatokat célszerű kliensoldalon elvégezni, míg néhány feladatot csak szerveroldalon lehet biztonságosan végrehajtani.

Kliensoldali nyelvek használatakor mindig tartsuk észben, hogy az általunk írt kód bárki számára látható lesz a weboldal forráskódjában! Ez nem túl szerencsés, ha bizalmas adatokkal (pl. jelszavakkal, bankszámlaszámokkal, online vizsgateszt megoldásokkal) dolgozunk!

FONTOS!

Ha bizalmas adatokkal dolgozunk, **mindig** szerveroldali nyelvet használjunk!

4. PHP

A félév hátralévő részében a **PHP** (**P**HP: **H**ypertext **P**reprocessor) nyelvvel fogunk foglalkozni, ami a **szerveroldali programozási nyelv**ek közé tartozik.

A. A nyelv főbb jellemzői

- A PHP kódot a **PHP értelmező** (PHP interpreter) értelmezi szerveroldalon
  - felhasználó oldalon már csak a kód kimenetét, eredményét láthatjuk
- Főbb alkalmazási területei
  - dinamikus HTML tartalmak létrehozása
  - HTML űrlapok feldolgozása
  - adatbázis-kezelés
  - menetkövetés
  - sütik küldése, fogadása
  - fájlkezelés
  - jogosultságok kezelése
  - objektumorientált programozás
  - …
- Miért pont PHP?
  - rengeteg platform támogatja (Windows, Linux, Unix, Mac OS X, …)
  - szinte minden ma használt szerverrel kompatibilis (Apache, IIS, Tomcat, …)
  - hatékonyan fut szerveroldalon
  - könnyedén tanulható, akár kezdők számára is ideális
  - teljes mértékben ingyenes és open-source
  - PHP-n alapulnak olyan népszerű weboldalak, mint például a Facebook, Wikipédia, Wordpress és Tumblr

B. Mire lesz szükségünk?

A PHP-ban való programozáshoz kelleni fognak:

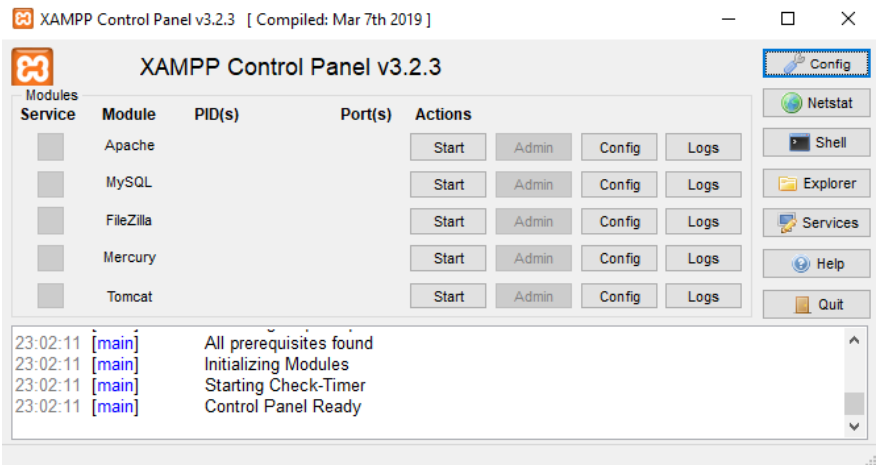
- Egy szerkesztőprogram, amibe a kódot írjuk
  - akár a legegyszerűbb szövegszerkesztő is megfelel
  - pl. Gedit, Notepad++, Sublime Text 3, PhpStorm, …
- Egy webszerver
- PHP telepítése
- Adatbázisszerver
  - ezen a kurzuson nem tárgyaljuk

C. A XAMPP szoftver

A gyakorlaton a **XAMPP** nevű, ingyenesen letölthető szoftvert fogjuk használni. Ez biztosít nekünk egy webszervert, egy adatbázisszervert, valamint a PHP-t is telepíti.

Töltsük le és telepítsük a XAMPP-ot!

1. Kattintsunk a következő linkre: https://www.apachefriends.org/hu/download.html
2. Válasszuk ki az operációs rendszerünknek megfelelő letöltési linket
3. A letöltés után telepítsük a szoftvert a telepítési utasítások követésével
4. A telepítés után nyissuk meg a programot – ekkor valami ehhez hasonlót kell látnunk:



D. PHP kód írása és futtatása

A PHP kódot tetszőleges szerkesztőprogramba írjuk, és **`.php`** kiterjesztéssel mentjük el, a **`[XAMPP telepítési helye]/xampp/htdocs`** mappán belül.

Nyissuk meg a XAMPP-ot, és az `Apache` felirat melletti `Start` gombra kattintva indítsuk el a szervert. Nyissunk egy böngészőt, majd az URL sávba írjuk be a következő címet: **`localhost/[fájl elérési útvonala a htdocs mappán belül]`**.

Ha minden jól működik, akkor már láthatjuk is az általunk megírt kód kimenetét.

E. Technikai problémák megoldása

Előfordulhat, hogy a webszervert nem sikerül elindítani. Ennek gyakori oka, hogy valami másik program (pl. Skype) már használja a portokat. Ez esetben át kell konfigurálni a portokat.

Az `Apache` felirat melletti `Config` gombra kattintva az első két menüpontra lesz szükségünk.

1. Kattintsunk az első menüpontra, és a megnyíló konfigurációs fájlban (`httpd.conf`) keressünk rá erre: `Listen 80`! Ez az a portszám, amit a webszerver figyel. Ezt írjuk át egy szabad portra, például `81`-re (ha ez sem szabad, próbálkozzunk tovább: `82`, `83`, …)!
2. Kattintsunk a második menüpontra, és a megnyíló `httpd-ssl.conf` fájlban keressünk rá erre: `Listen 443`! A portszámot itt is írjuk át, például `444`-re!

Az első fájlban a HTTP, míg a másodikban a HTTPS protokollok portszámát állítottuk be.

F. PHP kód beszúrása

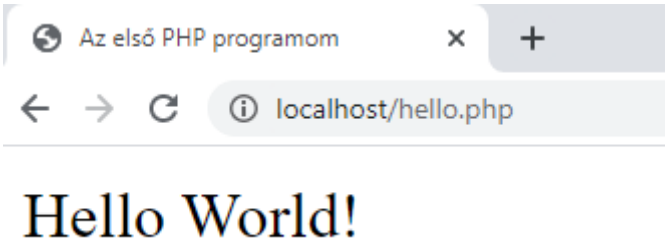
- A PHP fájlok kiterjesztése: **`.php`**
  - ezekbe nem csak PHP, hanem például HTML kód is írható, amit a szerver kiszolgál
- PHP kódot **`<?php`** és **`?>`** között tudunk írni
  - kiíratások esetén használható még: **`<?=`** `kiírandó kifejezés…` **`?>`**
- Külső fájl beágyazására az **`include`**`("…")`, illetve **`require`**`("…")` utasítások szolgálnak
  - zárójelek nélkül is használhatók, a beágyazandó fájl nevét várják paraméterül
  - ha a beágyazandó fájl nem létezik, mindkét utasítás hibajelzést ad
    - `require` esetén ekkor a program futása megszakad
    - `include` esetén a program fut tovább
  - a többszörös beágyazás elkerülésére használhatók az **`include_once`**`("…")` és **`require_once`**`("…")` utasítások
  - sok beágyazandó fájl esetén érdemes azokat egy külön könyvtárba szervezni, majd hozzáadni a könyvtárat az `include path`-hoz a következő utasítással: **`set_include_path`**`("könyvtár elérési útvonala");`

G. Az első PHP program: "Hello World"

Hozzunk létre egy `hello.php` fájlt a `xampp/htdocs` mappában, és írjuk bele az alábbi kódot:

```
<!DOCTYPE html>
<html>
<head>
   <title>Az első PHP programom</title>
   <meta charset="UTF-8">
</head>
<body>
   <?php
      echo "Hello World!";
   ?>
</body>
</html>
```

Mentsük el a fájlt, indítsuk el XAMPP-ban a szervert (ha még nem tettük), majd írjuk be a böngésző URL sávjában a `localhost/hello.php` útvonalat! Ha jól dolgoztunk, ezt az eredményt látjuk:



Nézzük meg az így előálló weboldal forráskódját (Jobb klikk + Vizsgálat)! Azt látjuk, hogy a kiadott PHP utasítások (jelen esetben: `echo`) nem jelennek meg a forráskódban, csupán azok kimenete. Ez különösen hasznos akkor, amikor bizalmas adatokat szeretnénk elrejteni a felhasználó elől.

H. Külső fájlok beágyazása

Példa: Ágyazzuk be a PHP fájlunkba az általunk elkészített `header.html` és `footer.html` fájlokat!

```
...
<body>
   <?php include "header.html"; ?>
   Ide jön valami tartalom...
   <?php include "footer.html"; ?>
</body>
```