PIACTÉR

készítette:

Németh András

Jedlik Iskola

2014

.

Összefoglalás

Szakdolgozat egy Apród/Hardverapróhoz hasonló weboldalról, PHP HTML CSS alapokon. Ahol a felhasználók könnyen és gyorsan értékesíthetik használt/megunt tárgyaikat.

Bevezetés

Témaválasztás

A mai világban az internet széles körben elterjedt, az emberek életének a szerves részévé vált. A legtöbb felhasználó böngészőn keresztül éri el az internetet, amely ma már minden számítógépen megtalálható és a mobiltelefonok nagy részén is. Ez egy olyan multifunkcionális eszközzé fejlődik, amelyen keresztül a felhasználók információt gyűjthetnek, kommunikálhatnak, filmeket nézhetnek. A kis és nagy vállalatok számára a marketing mind az értékesítés területén elengedhetetlen eszköz. Szóval a jövőnek mindenképpen szerves része és egy olyan eszköz amire a fejlesztést mindenképpen érdemes megtanulni és hosszútávon jó befektetés.

Mivel teljesen lenyűgözött ez az eszköz, úgy döntöttem a nyáron hogy szeretnék én is foglalkozni vele, később beiratkoztam a Jedlikbe, és amikor a szakdolgozat témaválasztására került a sor a választás nem volt túl nehéz, valami olyat szerettem volna fejleszteni aminek a témája komolyabb mint egy játék. Később munkakeresésnél referenciának vagy egyetemen valamilyen formában újrahasznosítható. Nem utolsó sorban szakmailag kihívás és lehetőséged ad ezen a területen való fejlődésre.

Úgy gondoltam hogy a logikai rész megírására a PHP-t használom mivel elég gyorsan lehet vele fejleszteni (dinamikus változók, megengedő syntax). Az oldal megformázásához HTML, CSS-re esett a választás de erről majd késöbb.

Eredeti specifikáció

Szakdolgozatnak egy piacteret fogok készíteni (vatera, ebay).

Megjelenítés: html, css, javascript. Szerveroldalon: php, mysql adatbázissal.

A felhasználóknak regisztrálniuk kell (emailes megerősítés).

Főoldalon véletlenül kilistázott termékek, kategóriák.

A felhasználók termékeket(hirdetéseket) tudnak feltölteni címszavakkal, képpel, kategória, leírással.

Keresés: címszavakra rálehet keresni, kilistázza a találatokat.

Szűrés bizonyos opciókra (szállítás, ár…)

A termékeknek külön kigenerált oldala lesz:

* Termékleírás
* Kép
* Termék helye
* Hirdetés érvényessége
* Kommentek
* Hirdető profilja

Az üzlet lebonyolítása után lehet értékeli egymásnak a feleket.

Irodalomjegyzék:

* Tanuljuk meg a PHP5 használatát 24 óra alatt - Matt Zandstra
* stackoverflow.com
* caniuse.com
* Tiszta kód - Az agilis szoftverfejlesztés kézikönyve - Robert C. Martin
* Biztonságos webalkalmazások PHP nyelven - Tricia Ballad, William Ballad

Fejleszői környezet

Nyelvek kiválasztása

Egy piactér szerű dinamikus oldal fejlesztéséhez elengedhetetlen egy jó programnyelv választása nálam a PHP-ra esett a választás. A böngészőben való megjelenítéshez a leíró nyelvek közül nem ekkora a választék, ezért a két legelterjedtebbre esett a választás a HTML5 és a CSS3-ra.

PHP

A programlogika (buisness logic) megírása ezen a nyelven folyik. A legújabb változatot használom mivel ez már támogatja az objektum orientált programozást és az új függvények is hasznosak. A syntaxisa megengedő nem kis-nagy betű érzékeny és néhány parancs többször is szerepel különböző néven, ezenkívül az összes változó dinamikus. Ezek tulajdonságok lehetnek ugyanúgy előnyök mint hátrányok mivel nagy szabadságot ad a programozóknak könnyedén lehet csúnya és átláthatatlan kódot írni. Ezeket figyelembevéve próbáltam valami egyszerű konvenciót kialakítani hogy az eltérő időben megírt kódok egyformán nézzenek ki. A programot szinte teljes egészében objektumorientáltan írtam meg mivel szerintem ez átláthatóvá teszi a kódot és többszemélyes fejlesztés esetén az procedurális programkód szóba sem jöhet.

Framework-ök

Fat-free

A Programozásnak október körül álltam neki de rövidesen rájöttem hogy a programkód és a megjelenítés összevegyítése nem a legcélravezetőbb dolog hosszútávon. Ekkor a programozást egy kis időre abbahagytam és jött a kutatás rész, ekkoriban botlottam az „MVC” elnevezésbe ami a Modell-View-Controller rövidítése, ennek az a lényege hogy az oldalat 3 részre bontjuk:

* **Modell**: Itt történnek a számítások, a vezérlőtöl megkapott adatok feldolgozása az adatbázissal való komunikáció, sajnos nálam ez néhol egybeolvadt a vezérlővel a gyorsabb fejlesztés érdekében.
* **Nézet**: ezen a felületen történik a felhasználói interakció, jellemzően egy felhasználói felület. Különböző célokra különböző nézetek létezhetnek ugyanahhoz a modellhez.
* **Vezérlő**: Ez a réteg köti össze a modell-t a nézettel. A bemeneti adatok ezen keresztül jutnak el a modellhez amely azt feldolgozza. A feldolgozott adatot ez adja át a nézetnek.

Ez után tovább keresgéltem a megoldás után mivel ezt nem igazán tudtam elképzelni hogy hogyan programozzam le, ekkor az egyik ismerősöm ajánlotta hogy használjak egy PHP frameworkot, így esett a választás a Fat-Free frameworkre (késöbbiekben csak F3). Ez egy kisebb framework ami támogatja az MVC fejlesztési mintát. Legfontosabb funkciói:

**URL routing**: Ez az összes url-t az index.php ra irányítja egy HTACCESS file segítségével ahol aztán defeniálva van ennek egy funkció ami annak a kérésnek megjelenít egy oldalat vagy elvégez egy feladatot. az ilyen útvolnalak lehetnek: staikusak és dinamikusak.

1-1 példa mindkettőre:

* **Statikus**:

$f3->route('GET /Register', 'RegisterController->display');

Amennyiben a felhasználó beírja az oldal nevét/register (ez alap esetben GET (request method)) meghívja a RegisterController osztály display funkcióját amely megjeleníti a regsztrációs oldalat.

* **Dinamikus**:

$f3->route('GET /Ad/@adid','AdController->specificad');

Amikor a felhasználó oldalneve/Ad/tetszőleges számot ad meg, meghívódik az AdController osztály specificad funkciója amely eldönti hogy létezik e ilyen ’id’-vel rendelkező hirdetés és ennek függvényében megjeleníti a hirdetést vagy a hibaüzenetet.

Amennyiben a routing listában nem definiált oldalat hívunk egy 404 oldalra dob.

**Template kezelő rendszer:** Számomra ez a funkció is nagyon hasznosnak bizonyult ez különíti el a nézetet a logikai résztől. Az előző példák közül az első:

function display($f3) {

NAVBARController::buttons($f3);

echo Template::instance()->render('main.tpl');

echo Template::instance()->render('register.tpl');

echo Template::instance()->render('endofmain.tpl');

}

Ez egy elég egyszerű funkció, az első sora (NAVBARController::buttons($f3);) meghív egy button nevű funkciót ami beállítja az oldal tetején lévő gombok értékeit a attól függöen hogy a felhasználó bevan-e jelentkezve. Ebből egy részlet:

if (isset($\_SESSION['username'])) {

$f3->set('signinstatus', 'Hirdetésfeladás');

$f3->set('signinstatuslink', '/adupload');

} else {

$f3->set('signinstatus', 'Rólunk');

$f3->set('signinstatuslink', '/aboutus');

}

A set nevű funkció beállít framework változókat bizonyos értékekre.

A display funkció következő sora:

echo Template::instance()->render('main.tpl');

kirendereli a main nevezetű templatet:

<!DOCTYPE html>

<html>

<head>

<title>Piactér</title>

<include href="style.tpl" />

</head>

<body>

<include href="navbar.tpl" />

<div class="wrapperSJ" id="content">

<div class="row">

<!-- END OD MAIN TPL -->

Ez a rövid részlet lesz az oldal forrásának az eleje, egy hozzáértő ember rögtön észreveszi hogy ebben van kettő darab nem ide illő tag: <include href="style.tpl" /> és <include href="navbar.tpl" />. Az include tageket a template kezelő rendszer értelmezi és a megadott templateket beilleszti a helyére. Az első include a css-eket és javascripteket illeszti be. A második pedig a navbart, ebben találhatóak framework változók, pl:

* {{ @signinstatus }}
* {{ @signinstatuslink }}

ezeknek az értékét állította be a NAVBARController::buttons($f3) úgyhogy itt a bejelentkezett felhasználók, a hirdetésfeladás gombot kapják „/adupload” linkkel és nem bejelenkezettek pedig egy Rólunk feliratú gombot kapnak „/about us” linkkel.

KÉP JÖN IDE A NAVBARRÓL

**Autoloader** egy framework által kezelt tömb amiben mappanevek vannak eltárolva ennek köszönhetően a funkciókban nem kell require/incude-olni a az osztályokat. Amikor osztályt példányosítunk vagy staikusan hívunk függvényt automatikusan megnézi a megadott mappákat és amennyiben ugyanolyan néven vannak elmentve a file-ok mint a bennük tárolt osztály (csak egyosztály lehet egy fileban) az autoloader megtalálja őket és meghívja őket helyettünk.

**Hibakezelő rendszer** az F3 hibakező rendszere kigenerálja

Fejlesztőkörnyezet

A programkódok nagy részét a Netbeans IDE-ban írtam meg. (PHP, CSS, HTML)

Az adatbázis kialakításához és menedzseléséhez a Navicat-ot használtam.

Helyi hálózaton való teszteléshez az Easy PHP-t csomagját használtam:

* Apache
* MySQL

Teszteléshez a google chrome-t használtam a megjelenés erre van letesztelve, optimalizálva.

Az elkészült program

# Az interface

A megjelenítés mindig 3 részből tevődik össze:

* Navbar, az oldal tetején helyezkedik el,ez tartalmazza a navigációhoz szükséges gombokat, logót, közösségi oldalra linkeket.
* Tartalom, ez tartalmazza mindig az éppen aktulás dolgokat amire a felhasználó kiváncsi, itt történik az interakció, adatcsere nagy része.
* Footer, a lap alján található, az oldalról alapvető információk…

Fejezetek:

* Bevezetés (1)
* Eredeti specifikáció (1.1)
* Témaválasztás (1.2)
* Iroalomjegyzék (1.3)
* Halsznált eszközök/technológiák
* programnyelv
* fejlesztőkörnyezet
* Az elkészült program
* Megvalósult dolgok
* Megvalósulatlan funkciók
* Plussz funkciók
* Felhasználói dokumentáció
* Tesztek, problémák, megoldások
* Összefoglalás, konklúzió(vége)