Debreceni Egyetem
Informatikai Kar

Webes áruház fejlesztése a Zend keretrendszer használatával

Témavezető:
Kósa Márk Szabolcs
egyetemi tanársegéd

Készítette:
Liszkai Zsófia
programozó matematikus

DEBRECEN
2010

Tartalomjegyzék

1. Bevezetés

1.1 Webes áruházak

Az internet mára már mindenki számára könnyen hozzáférhető, és információszerzés szempontjából az egyik legközpontibb médium, ezért – legyen szó egy egyszerű honlapról, blogról, vagy webáruházról – kiemelt szerepet kap a megjelentetett dokumentumok tartalmi és formai minősége. A világháló vonatkozásában a minőség leginkább a gyors és könnyű kezelhetőséget valamint az áttekinthetőséget jelenti, hiszen a honlapok és webes áruházak elsődleges a célja a hirdetés mellett, hogy leegyszerűsítse és felgyorsítsa az tájékozódást, különböző szolgáltatások igénybevételét, ügyintézést és vásárlást.

A webes áruház vagy webshop az interneten zajló online kereskedelem eszköze, azaz egy speciális honlap, melyen keresztül az eladó fél valamilyen terméket vagy szolgáltatást hirdet, értékesít, a vevő pedig lebonyolíthatja a vásárlást. A cél programozóként természetesen az, hogy mindezt mindkét fél számára a lehető legkényelmesebbé tegyük.

A webes áruházak közös jellemzője, hogy a vásárlók egy könnyen kezelhető felületen keresztül böngészhetnek az adatbázisban, majd egy virtuális bevásárlókosárba pakolhatják az általunk kiválasztott termékeket, a kívánt fizetési mód megadása után pedig véglegesíthetik a megrendelést.

A különböző termékeket árusító boltok és áruházak forgalma nagyságrendekkel nőhet, ha nem csak egy információkat közlő egyszerű honlappal rendelkeznek, hanem webes áruházat is létrehoznak. Ez megkönnyíti azon vásárlók lehetőségeit, akik szívesebben keresgélnek és vásárolnak otthonról, a számítógépük előtt ülve. Arról nem is beszélve, hogy hatékonyabban tudják a tulajdonosok reklámozni az új termékeket, esetleges változásokról információt közölni és a kapcsolattartás is – annak ellenére, hogy e-mailben zajlik – esetenként intenzívebb lehet, mint a személyes.

A webes áruházak arra is lehetőséget biztosítanak, hogy a kereskedő pontosan és erőfeszítések nélkül nyomon kövesse az áruház forgalmát, hiszen egy jól felépített rendszerben minden esemény és tevékenység rögzítésre kerül. Ez a későbbiekben sokat segíthet statisztikai elemzéseknél, melyek a bolt forgalma alapján a vásárlók elégedettségéről adnak információt, ezáltal hozzájárulva a változtatásokhoz és fejlesztésekhez.

Mindezen előnyök mellet, könnyű belátni azt is, hogy az online kereskedelem mennyivel költséghatékonyabb, mint a hagyományos. Nem véletlen tehát, hogy az online üzletek egyre növekvő számban elérhetőek, és szinte már minden piaci területet felölelnek.

1.2 Célkitűzés

A növekvő webes igényeknek eleget téve egyre több programozó fordul a webes alkalmazásfejlesztés irányába. Szakdolgozatom megírásának egyik célkitűzése az volt, hogy megismerkedjek azzal a folyamattal, ahogyan azok a honlapok elkészülnek, amiket nap mint nap én magam is látogatok. Az alkalmazásomhoz hasonlóak már évek óta forgalomban vannak, és folyamatosan fejlődnek. A szoftverem megírásakor természetesen merítettem ötleteket más lapokról, mégis igyekeztem a saját meglátásom szerint kidolgozni azt.

Szakdolgozatom témájául tehát egy webes áruház megvalósítását, és a megvalósítás különböző lépéseinek elemzését választottam a tervezéstől a felhasználói dokumentáció készítéséig. A programom tartalmazza a webes alkalmazások általános komponenseit, és az online áruházak alapvetően szükséges funkcióit; így vásárlói oldalról a termékek listázását, keresését, kosárba helyezését és a megrendelést; könyvtáros és adminisztrátori oldalról pedig a felhasználói adatok és tevékenységek nyomon követését, a terméklista és az ahhoz kapcsolódó adatok frissítését, valamint a rendszer teljes körű karbantartását.

A program felépítéséhez elsőként szükség volt egy adatbázisra, illetve egy, a tervezést és a megvalósítást segítő szoftverre. Az adatok kezeléséhez és a vezérléshez megfelelő nyelvet és fejlesztői környezetet kellett választanom, melyek az objektumorientált PHP nyelv, és az ahhoz készült MVC tervezési minta alapú Zend keretrendszer lett.

Lépések:

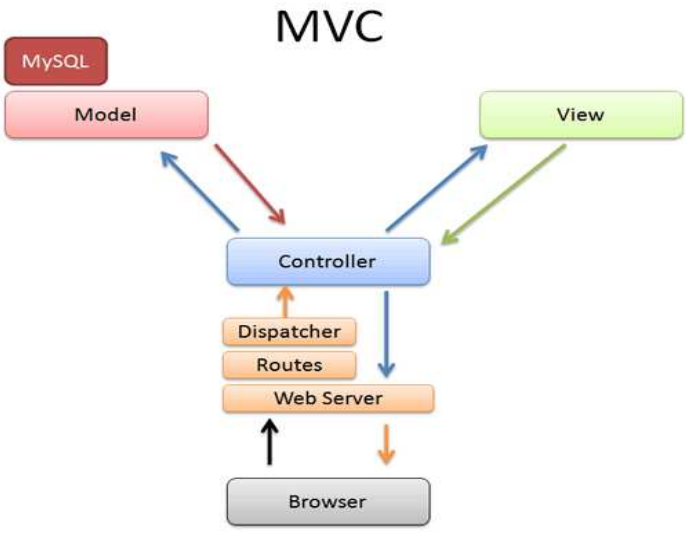
- tervezés: adatbázisterv, keretrendszer-választás, alkalmazás céljának, funkcióinak meghatározása;
- adatbázis és alkalmazás implementálása;
- adatbázis feltöltése adatokkal;
- program tesztelése;
- felhasználói és programozói dokumentáció készítése.

2. Technológiai háttér

2.1 MVC tervezési minta

Az MVC (Model-Controller-View) tervezési minta a komplex webes alkalmazások fejlesztésében használt módszer arra, hogy különválasszuk az adatbázist, az üzleti logikát és a megjelenítést. Arra törekszünk, hogy minden fájlban csak a ténylegesen odatartozó forráskód szerepeljen, ezáltal logikus, könnyebben átlátható kódot kapunk.

A modellek az adatbázisban szereplő táblákhoz készülnek, a sorobjektumhoz és a táblához implementálva egy-egy osztályt. Az osztályokban a táblák közötti kapcsolatokat is feltüntetjük. Ha elkészültünk a szükséges osztályokkal, és azok helyesen vannak implementálva, akkor innentől kezdve már nem kell közvetlenül az adatbázisunkkal foglalkoznunk; a megfelelő modelleket használva kommunikálunk, ezeken keresztül tudunk az adatbázishoz hozzáférni, onnan adatokat kiolvasni, vagy az adatbázisba új adatokat írni.



Az MVC tervezési minta

A *controllerek* vagy vezérlők arra szolgálnak, hogy az adatok kezelését megvalósítsák, az utasításokat végrehajtsák. Itt történik az adatbázisból lekérdezett adatok feldolgozása, vagy éppen a *view* irányából érkezők adatbázisba rögzítése. Így az adatokkal végzett műveletek minden mástól elkülönítve zajlanak, parancsok, függvényhívások és vezérlési szerkezetek segítségével. A controller réteg tehát a vezérlést a további két réteg között elhelyezkedve mindkét irányába kommunikálva valósítja meg. Ha például az adatbázisból kérdezünk le

adatokat, akkor a megfelelő információt kinyerése és esetleges módosítása után csak annyi dolgunk van, hogy értesítsük a viewt a szükséges adatok megjelenítéséről.

Az IndexControllernek kitüntetett szerepe van, ugyanis hacsak nem adunk meg más beállítást, alapértelmezés szerint az `index/index`, azaz az IndexControllerben található indexAction fog lefutni. A másik ilyen controller, az ErrorController, mely a hibakezelésért felelős. Bármilyen futás közben felmerülő hiba esetén ide adja át a böngésző a vezérlést, és a megfelelő hibához tartozó *actiont* – amit a továbbiakban akciónak vagy műveletnek fogok nevezni – próbálja futtatni. Ha nem írunk saját hibakezelőt, akkor az alapértelmezett és általános errorAction fogja a hibákat kezelni.

A megjelenítés vagy view feladata, hogy a controllertől kapott adatokat a szükséges formába öntse és létrehozza a felhasználói felületet anélkül, hogy a logikai felépítésben módosításokat végezne, vagy bonyolultabb vezérlési szerkezeteket tartalmazna. Szabványos HTML oldalon űrlapokkal, linkekkel, táblázatokkal jelenhetnek meg a kívánt információk.

2.2 Zend Framework

A Zend egy PHP nyelven írt MVC alapú objektumorientált keretrendszer, mely nagyszerűen támogatja a webes alkalmazások fejlesztését. 2005-ben való megjelenése óta teret hódított a PHP programozók körében. Az MVC architektúra használatával jobban átláthatóvá és könnyebben újrahasznosíthatóvá válnak kódjaink, így bonyolultabb alkalmazások fejlesztéséhez is ideális. A másik hatalmas előnye, hogy úgynevezett önálló komponensei vannak, amelyek bárhol használhatóak anélkül, hogy egyéb összetevőkkel kellene foglalkoznunk, akár arra is lehetőséget biztosít, hogy komponenseket más PHP-s keretrendszerrel együtt használjunk. Nem kötelez tehát minket programozókat arra, hogy az általa definiált eszközökkel dolgozzunk, mindig választhatjuk a már megszokott utat, ha nem ismerjük, vagy nem kívánjuk használni a keretrendszer által biztosított komponenseket.

Mivel a Zend keretrendszer az MVC fejlesztési paradigmán alapul, így végig szem előtt kell azonban tartanunk a törekvést, hogy az adatbázist, a logikát és a megjelenést jól elszeparáltan kódoljuk le. A relációs adatbázisokban tárolt adatok objektumorientált környezetben való elérésre is egyszerű megoldást kínál.

A keretrendszer egy projekt indulásánál is nagy segítséget nyújt, hiszen az alapvető könyvtárszerkezetet és az „alapprojektet” kigenerálja nekünk, ha szeretnénk. A szükséges könyvtárak a faszerkezetben az application mappa, mely tartalmazza a controllereket viewscripteket, modelleket, és az egyéb lényeges alkalmazás-összetevőket, mint addonok, pluginek és a konfigurációs fájl; a library mappa, mely magát a keretrendszert tartalmazza; és a public vagy web mappa, mely minden, a kliensoldali hibátlan megjelenítésért felelős fájlt magába foglal.

Zend_Acl

Könnyen megvalósítható és rugalmas utat biztosít a hozzáférés-vezérlő lista készítéséhez és a jogosultságok kezeléséhez, hiszen beépítetten tartalmazza a *role* (szerepkör) és a *resource* (erőforrás) alapvető implementációját. A Zend_Acl fastruktúra szerűen tárolja le a szerepeket, és támogatja a többszörös öröklődést, azaz egyszerre egy szerepkör több szülőtől is örökölhet. A hozzáférés-vezérlő lista létrehozása egy egyszerű objektum példányosításából áll, majd az erőforrások és a szerepek definiálása után megadhatjuk, hogy egyes szerepek milyen jogosultságokat kapjanak.

```
$acl = new Zend_Acl;
$acl->addRole(new Zend_Acl_Role('guest'));
$acl->add(new Zend_Acl_Resource("index/index"));
$acl->allow('guest', "index/index");
```

Zend_Auth

Az autentikáció vagy hitelesítés alapvetően szükséges funkció webes alkalmazások fejlesztésénél, mely egy „egyedet” azonosít, internetes alkalmazásoknál általában felhasználói név és jelszó alapján. A Zend_Auth ennek megvalósítására elkészített komponens, melyet a felhasználói bejelentkeztetés implementálásánál használtam. A Zend_Auth a Singleton mintát valósítja meg, azaz az azonosított adatai belépés után bárhol lekérdezhetőek, melyet a következő sorral tehetünk meg.

```
$user = Zend_Auth::getInstance()->getIdentity();
```

Az én alkalmazásomban az autentikációt szabványszerűen az AuthController valósítja meg, méghozzá a bejelentkező űrlap feldolgozását végző identifyAction segítségével.

```
$dbAdapter = Zend_Registry::get('db');
$authAdapter = new Zend_Auth_Adapter_DbTable($dbAdapter);
$authAdapter->setTableName('users');
$authAdapter->setIdentityColumn('username');
$authAdapter->setCredentialColumn('password');
$authAdapter->setIdentity($formData['username']);
$authAdapter->setCredential(md5($formData['password']));
$auth = Zend_Auth::getInstance();
$result = $auth->authenticate($authAdapter);
```

Zend_Db

A Zend_Db_Table osztályt használtam a tábláim modelljeinek megírásakor. Ez egy objektumorientált interfész adatbázistáblákhoz, melyben metódusokként implementálva vannak az alapvető műveletek (például `insert`, `delete`, `select`), amiket a táblákon elvégezhetünk. A feladatunk, hogy minden egyes adatbázisban található táblához létrehozunk egy osztályt, mely a Zend_Db_Table_Abstract osztályból származik. Mikor valamelyik táblára szükségünk van, csak példányosítanunk kell a megfelelő osztályt a konstruktor meghívásával.

A Zend_Db_Table objektum sorait a Zend_Db_Table_Row osztály példányaival írhatjuk le, így minden táblához készítünk egy a sorobjektumokat leíró modellt is. Ilyen objektumot, vagy objektumok összességét kapjuk vissza, ha lefuttatunk például egy lekérdezést, vagy ezeket a „sorokat” használjuk beszúrásnál és rekordok módosításánál is.

A lekérdezéseket a Zend_Db_Select komponenssel valósítottam meg, mely a szabványos SQL SELECT lekérdezés Zend keretrendszerben implementált változata.

```
$users_modell = new Users();
$select = $users_modell->select()->where("acl_id = ?", 3);
$users = $users_modell->fetchAll($select);
```

Zend_Controller

Az MVC tervezési minta központi elemeként ez a komponens a Zend keretrendszerben írt alkalmazások mozgatórugója. A Zend keretrendszer részeként a Zend_Controller_Front továbbá megvalósítja a Front Controller tervezési mintát is, mely egy mechanizmus arra, hogy az alkalmazás belépési pontját centralizálja. Ennek lényege, hogy minden kérés a Front Controllerhez érkezik be, majd ez dönti el, melyik controller melyik műveletére lesz szükség. Egy-egy controller tehát logikailag összetartozó tevékenységeket valósít meg, mely tevékenységeket akcióknak vagy műveleteknek hívunk.

Alkalmazásomban a sztenderdet követve igyekeztem a fontosabb egységeket elkülönítve különböző controller állományokba szedni. Erre a Zend_Controller_Action osztályt használjuk. Így születtek meg az alapvető controllerek, mint a Zend keretrendszerrel felépített rendszerekben általában létező IndexController, AuthController, és ErrorController, továbbá az alkalmazás-specifikus BookController és OrderController. Két további controller szerepel még a listában, melyek az adminisztrátori és a könyvtárosi jogosultságokkal bíró felhasználók feladatait írják le.

Zend_View

Ez a komponens a megjelenítésért felelős, így képviselve a háromrétegű felépítés „legfelső” rétegét, azaz azt a felületet, amit a felhasználó lát az alkalmazásból, és amin keresztül a felhasználói kommunikáció folyik. A controllertől kapott információk alapján jelenít meg adatokat, természetesen HTML és CSS fájlokat használva. A controllerben természetesen definiálnunk kell, hogy mely adatokat szeretnénk, ha a viewscript fájlunk megkapna. Átadhatunk változókban tárolt egyszerű értékeket, tömböket, de akár objektumokat is. Ez a kis kódrészlet az AdminController osztályból való, és jól reprezentálja, hogy a rendeléseket összegyűjtve egyszerűen átadhatjuk a viewnak, $this->view kifejezéssel hivatkozva azt.

```
$orders_modell = new Orders();
$select = $orders_modell->select()->order('order_date DESC');
$orders = $orders_modell->fetchAll($select);
$this->view->orders = $orders;
```

Az így továbbított megrendeléseket tartalmazó rowsetet az orders.phtml viewscript kapja meg, mely egy foreach ciklusban feldolgozva azt kilistázza a sorokat.

```
<? foreach($this->orders as $o) : ?>
    <tr>
        <td><?=$o->users_user_id?></td>
        <td><?=$o->books_book_id?></td>
        <td><?=$o->order_date?><br /></td>
    </tr>
<? endforeach; ?>
```

Zend_View_Helper

A megfelelő megjelenítést segítő segédosztályok találhatóak itt, melyek funkciója, hogy a controllertől kapott nem, vagy nem megfelelően „megformált” adatok hibátlan kiírásáról gondoskodjanak. Ezekből rengeteg van és rendkívül sokféle feladatot látnak el. A projekten belül a library könyvtárban találhatóak, azaz a keretrendszer beépített elemei. Használatuk egyszerű, a viewscript fájlban meghívhatóak $this->helpernév($paraméterek) formában. Ezek közül most csak egyet említenék meg, az általam leggyakrabban használt helpert, melyet a Zend_View_Helper_Url osztály ír le. Ez a viewscript fájlokban elhelyezett linkek létrehozására szolgál. Használatakor paraméterként egy controllert és egy művelet vár, valamint szükséges esetén egy azonosítót is átadhatunk neki.

Zend_Session

A Zend_Session komponens a munkamenetekben tárolt adatok kezeléséért felelős. Ezek szerveroldalon tárolódó adatok, amikhez a kliens csak akkor férhet hozzá, ha azt a szerver elérhetővé teszi például egy kliensoldali kérésre válaszolva. A Zend_Session_Namespace objektumokon keresztül érhetjük el a munkamenetben tárolt adatokat. Ilyen adatok lehetnek például az azonosításkor használtak. A *namespace* (névtér) példányai egy-egy „szeletnek” felelnek meg a PHP-s $_SESSION szuperglobális tömbből.

```
$namespace = new Zend_Session_Namespace();
$namespace->username = $data->username;
```

Zend_Application

Ez a komponens a legelső, mellyel az alkalmazásom fejlesztése során meg kellett ismerkednem, hiszen ez valósítja meg az úgynevezett „bootstrapping” algoritmust, mely a program elindításáért felelős. Itt kerülnek beállításra a környezeti elemek, és az automatikusan betöltődő információk. Az _init() metódusok segítenek abban, hogy inicializáljuk az alapvető komponenseket, mint az adatbázis, a view vagy az autentikációhoz és autorizációhoz szükséges Zend_Auth és Zend_Acl. A bootstrap fájl működéséhez szükség van egy hozzá kapcsolódó konfigurációs fájl létrehozására is, mely tartalmazza annak nevét, elhelyezkedését a könyvtárszerkezetben, továbbá adatbázis- és egyéb beállításokat. Példaképpen a view inicializációját megvalósító metódus a bootstrap.php fájlból a következő kóddal írható le:

```
    function _initView() {
        $view = new Zend_View();
        $view->doctype('HTML4_STRICT');
        $view->headTitle('MYPROJECT');
        $view->headMeta()->setHttpEquiv('Content-Type','text/html;
            charset=UTF-8');
        $view->headLink()->appendStylesheet('css/style.css');
        return $view;
    }
```

Zend_Registry

Globális változók helyettesítésére szolgál ez a komponens, objektumok, változók, tetszőleges értékek tárolására hivatott, a PHP-s $_GLOBAL-lal egyenértékű. Az alkalmazásban bárhonnan elérhetők és oldaltöltődések között is élnek az itt tárolt adatok. A két statikus metódusával tudjuk a kívánt értékeket letárolni, vagy azok értékét lekérdezni bárhol a programon belül.

```
    Zend_Registry::set('db', $db);
    $dbAdapter = Zend_Registry::get('db');
```

Zend_Form

Ez a komponens a bonyolultan megvalósított HTML nyelvű űrlapokat hivatott helyettesíteni. Az űrlap vagy form a felhasználói interakciók egyik alapvető eszköze, így az én alkalmazásomban is több helyen előfordul. Fontos, hogy jól felépítetten alkossuk meg őket, és hogy egyszerűen tudjuk lekezelni a visszakapott vagy postolt adatokat. A Zend keretrendszer ezen eszköze, nemcsak hogy a kódunkat „szebbé” teszi, de mind a formok megalkotása, mind az input adatok felhasználása kézenfekvő. Az MVC architektúrát követve a formot felépítő kód természetesen a controllerben található.

```
$usernameFE = new Zend_Form_Element_Text('username');
$usernameFE->setLabel('Felhasználói név')
     ->setRequired(true);

$passwordFE = new Zend_Form_Element_Password('password');
$passwordFE->setLabel('Jelszó')->setRequired(true);

$submitFE = new Zend_Form_Element_Submit('ok');

$form = new Zend_Form();
$form->addElements(array($usernameFE, $passwordFE, $submitFE));
$form->setAction('identify')->setMethod('post');
$this->view->form = $form;
```

Zend_Paginator

Ezt az eszközt a keretrendszer a lapszámozás vagy lapozás megvalósításához biztosítja. Nagyon praktikus, és mivel alkalmazásomba az elkészítés szinte utolsó fázisában építettem be, így bátran állíthatom, hogy egyszerű volt a programhoz fűzni az apró részeket, még a majdnem kész kódokhoz is. Létrehoztam egy paginator.phtml viewscript fájlt, melyet a többi lapszámozást tartalmazó viewscript hivatkozni tud, és a CSS fájl is tartalmaz a megjelenítésre vonatkozó részleteket.

```
$books = $books_modell->fetchAll($select);
$page = $this->_request->getParam('page', 1);
$paginator = Zend_Paginator::factory($books);
$paginator->setItemCountPerPage(5);
$paginator->setCurrentPageNumber($page);
$this->view->paginator = $paginator;
```

Addon

Az alkalmazásomhoz készítettem egy addont is, a felhasználóknak szóló üzenetek közlésére. Mivel ezek az üzenetek igen gyakoriak, praktikusabbnak találtam az erre szolgáló kódot kiemelni, és külön fájlban letárolni, hogy elkerüljem a redundáns kódrészletek felhalmozását. Ez tulajdonképpen egy olyan programkomponens, ami nem tartozik közvetlenül egyik controllerhez sem, ezért számára különálló fájlt hoztam létre. Ez az addons_UserMessage osztályt leíró php fájl, egyetlen függvényt, a userMessage()-et tartalmazva. A függvény feladata, hogy a paraméterként megkapott változóban tárolt sztringet elhelyezze a névtérben, ahonnan azt kiolvasva a layout betöltéskor megjeleníthet. Így a controllerekben akárhol meghívott függvény automatizálja az üzenetek képernyőre írását.

```
addons_UserMessage::userMessage("A név vagy a jelszó nem
    helyes! Próbáld újra!");
```

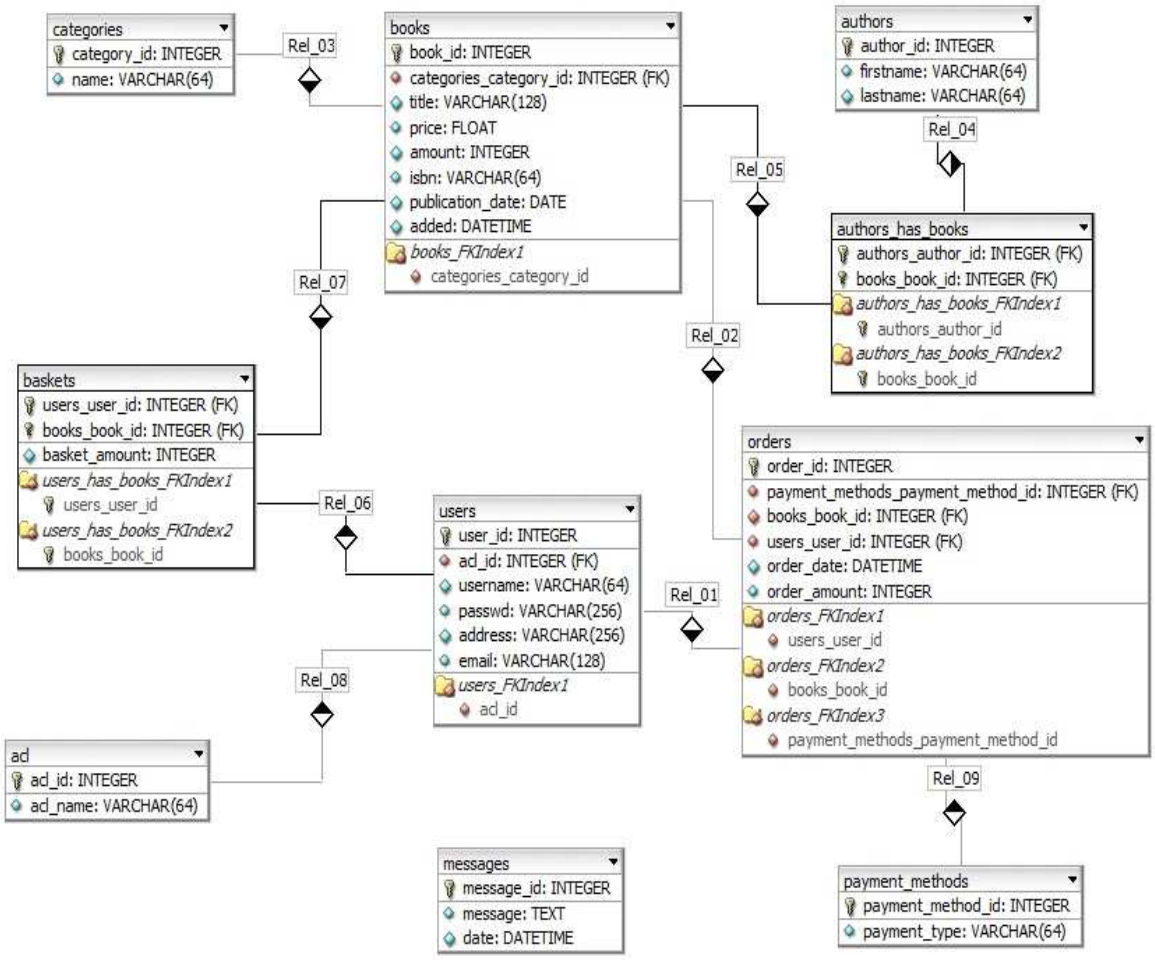
Plugin

Létrehoztam egy plugint is, mely a jogosultság-ellenőrzést segíti. Először a névtér és a Zend_Auth segítségével beállítja a felhasználó szerepkörét, majd a Zend_Registry-ből lekéri a Zend_Acl-t és megvizsgálja, hogy az adott szerepkörbe tartozó felhasználó jogosult-e megtekinteni a választott controller bizonyos műveletéhez tartozó oldalt.

3. Adatbázis

Az alkalmazásom alapja egy SQL adatbázis, melyben minden a szoftver működéséhez szükséges adat le van tárolva. Az adatbázis pontos megtervezése és logikus felépítése egy alkalmazás létrehozásának az egyik legfontosabb része, hiszen ezen áll vagy bukik a későbbi programozói munkánk, és hogy mennyire lesz egyszerű vagy éppen bonyolult az adatbázisunkkal való kommunikáció, és a benne tárolt adatokon végzett műveletek. Létrehozás előtt tehát az első lépés volt átgondolni, hogy melyek lesznek a program által megvalósítandó feladatok, és milyen egyedtípusok vesznek ezekben részt. Az adatbázis megtervezésénél az is szempont volt, hogy egyrészt az egyedek, másrészt a különböző funkciókban használt adatok hogyan kapcsolódnak egymáshoz. Ezen kapcsolatok tanulmányozása után összekötöttem a szükséges táblákat, és kapcsolótáblák is kerültek az adatbázisba. Az adatbázis a program írása során természetesen bővült, és jól strukturált felépítése végett tetszőlegesen tovább bővíthető. A felhasználói funkciók alapján elkészítettem az adatmodellt, valamint az egyedeket és az azok közötti kapcsolatokat ábrázoló diagramot.

Az adatbázis létrehozásánál a konvenciókat követve kisbetűs angol többes számú neveket adtam a tábláknak (pl. `users`), az elsődleges kulcs minden esetben egy autoincrement `táblanév_id` alakú számláló (pl. `user_id`), a külső kulcsok pedig az eredeti táblanevet hivatkozva `eredetitábla_elsődelegeskulcs` alakúak (pl. `users_user_id`), ahol az elsődleges kulcs az előbb definiált alakú. A kapcsolótáblák esetében a név a két összekapcsolt tábla neveit tartalmazza az angol „has” szócskával összekötve (pl. `authors_has_books`). Ezekben csak az egymáshoz kapcsolt táblák azonosítói szerepelnek.

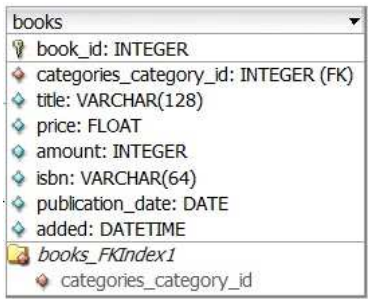


Az adatbázis

A továbbiakban részletesen ismertetem az egyes táblák adatmodellben betöltött szerepét és a közöttük lévő kapcsolatok típusát, azok funkcióját.

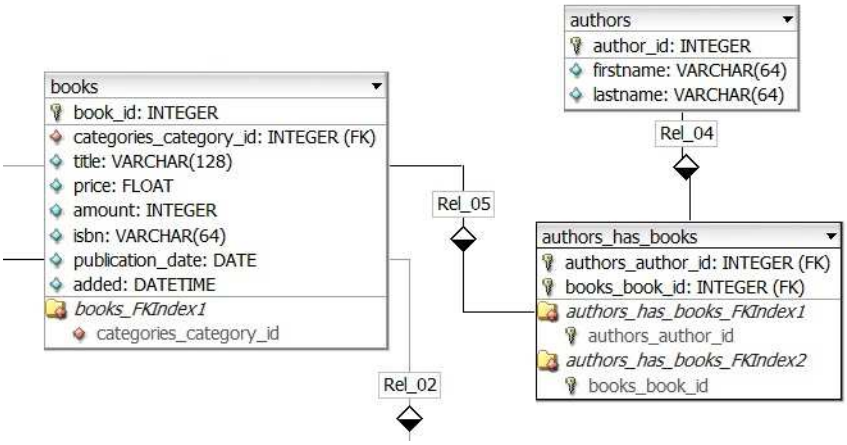
Az adatbázisom központi és egyben a legtöbb kapcsolattal rendelkező táblája az áruházban megtalálható könyveket tároló `books` tábla. Ebben vannak nyilvántartva a könyvek fontosabb attribútumai, kivéve azokat, melyeket külön táblában tároltam. A könyvek esetében a vásárló számára alapvető a cím, a szerző, a publikáció dátuma, továbbá szükség lehet esetenként az ISBN számra is. Itt található a könyvek ára, ez alapján számolódik a vásárlási végösszeg. A szerzőt mint attribútumot körültekintőbben kellett elhelyezni az adatbázisban, hiszen bonyolítja a helyzetet, hogy egy könyvnek több szerzője is lehet. Ezért ezt a jellemzőt külön táblában tároltam, a többi előbb felsorolt tulajdonság a `books` tábla része.

A tábla további mezői a könyvtárosok számára tartalmaznak fontos információkat, ilyen például, hogy az adott könyv mikor lett hozzáadva az adatbázishoz, vagy hogy hány darab van belőle raktáron.



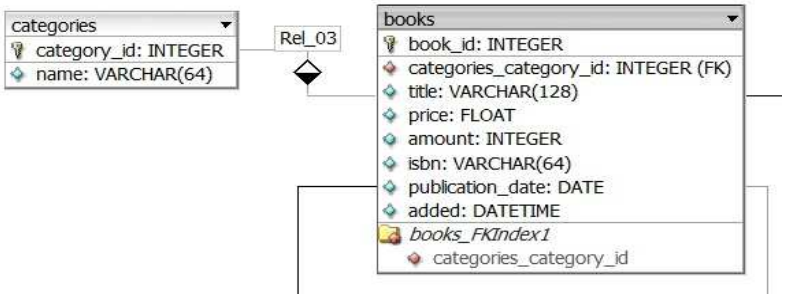
A books tábla

Mint már említettem, a könyvek szerzői egy külön táblában találhatóak, mely az `authors` nevet kapta. Az egyértelmű, hogy egy szerzőnek több könyve is lehet az adatbázisban, mégsem elég az 1:N kapcsolat, hiszen az is előfordulhat, hogy egy könyvnek két vagy akár több szerzője is van. Ebből kifolyólag a `books` és az `authors` táblák között N:M számosságú kapcsolatra lesz szükség, melyet a relációs adatbázisok esetén kapcsolótáblával valósíthatunk meg. A kapcsolótáblákban csak a kapcsolódó táblák elsődleges kulcsai szerepelnek. Így jön létre az `authors_has_books` táblánk, melyhez mind a `books`, mind az `authors` táblák 1:N kapcsolattal kapcsolódnak. Az `authors` táblában találjuk a szerzők nevét, vezetéknév és keresztnév mezőkre bontva.



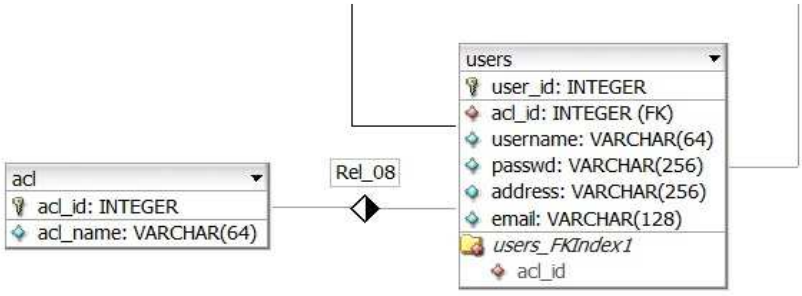
Az `authors` és `authors_has_books` táblák

A könyveknél lényeges tulajdonság továbbá az is, hogy milyen kategóriába sorolhatóak tartalmuk és stílusuk alapján. Amellett, hogy ez lényeges információ a könyvekről, alapvető keresési szempont is a vásárlók számára. Ezt hivatott tárolni a `categories` tábla, melyben az elsődleges kulcson kívül csupán a kategória neve szerepel. Mivel egy kategóriába rengeteg könyv tartozik, viszont egy könyvet csak egy kategóriába soroltam, így ehhez a táblához a `books` tábla 1:N kapcsolattal kapcsolódik, mely kapcsolatot a `books` táblában megjelenő külső kulcs mutat.



A `categories` tábla

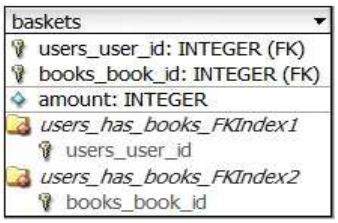
A másik alapvető tábla az adatbázisomban a felhasználókat tárolja. A `users` tábla a `user_id` elsődleges kulcs mellett tartalmazza a felhasználók felhasználói nevét, és a jelszavuk md5-ös kódolással titkosított formáját, azaz az itt található adatok alapján végzi a rendszer a felhasználók azonosítását, illetve be- és kijelentkeztetését. Az `acl_id` letárolásával a felhasználó szerepköre is megállapítható, így bejelentkezéskor ezt is lekérdezi a rendszer, és ennek megfelelően állítja be a felhasználói felületet és a jogosultságokat. Ennek beállítása tagok és könyvtárosok regisztrációjakor automatikusan történik. A további mezőkben az egyéb személyes adatok találhatók, mint a postacím a megrendelések kézbesítéséhez, és az e-mail cím a szükséges kommunikáció lebonyolításához. Mindezen adatok a felhasználói regisztráció során kerülnek az adatbázisba, és később módosíthatóak. E köré a tábla köré is több tábla szerveződik, létrehozva ezzel egy nagyobb egységet az adatbázisban.



A `users` és `acl` táblák

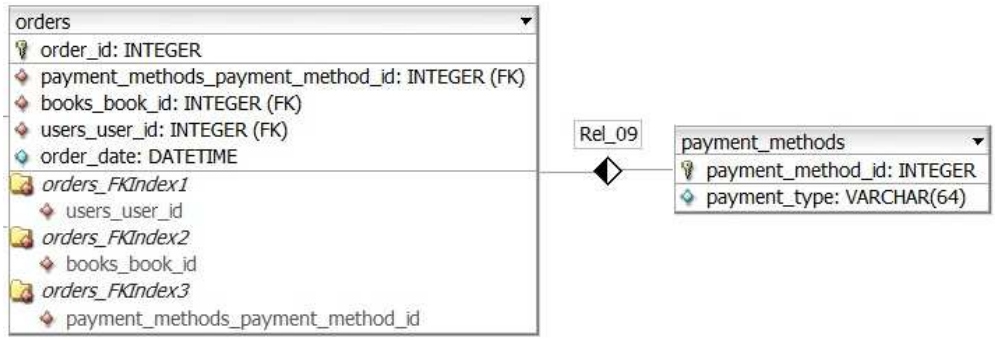
Az úgynevezett `acl` tábla tartalmazza az előbb már említett lehetséges felhasználói kategóriákat és azok megnevezését. Mivel minden felhasználó csak egy ilyen csoportba tartozhat, így a `users` táblában fog szerepelni az `acl` tábla kulcsa, mint külső kulcs, ahogy erről már korábban említést is tettem. Ezen tábla alapján különíthetőek el a különböző jogosultsági csoportok, azaz, hogy a látogatónk vendégként, tagként, könyvtárosként vagy adminisztrátorként érkezik az oldalra. Bejelentkezés után a `users` táblában szereplő `acl_id`-hoz tartozó megnevezést kéri le a rendszer az `acl` táblából, és ez alapján állítja be az oldalakhoz való hozzáférést.

A felhasználókat tartalmazó táblához köthető a `baskets` tábla, hiszen minden regisztrált taghoz tartozik egy bevásárlókosár, amelybe összegyűjti a megvásárolni kívánt termékeket. Ezek a könyvek vannak a `baskets` táblában nyilvántartva. Amikor a kosárba helyeznek egy terméket, az „átkerül” a `books` táblából ebbe a táblába, így biztosítva a konzisztenciát. A `baskets` tábla a `books` táblához is hozzá van kapcsolva, így egy N:M számosságú kapcsolatot megvalósítva a `books` és a `users` táblák között, ugyanis bárkinek lehet több könyv a kosarában, és egy könyv természetesen több kosárban is lehet egyidejűleg. Ennek értelmében a redundáns adattárolást elkerülve csak a két összekapcsolt tábla elsődleges kulcsát tartalmazza, valamint az adott könyv egy taghoz tartozó darabszámát, hiszen előfordulhat, hogy ugyanazon könyvből egynél többet szeretnénk rendelni. Minden további információt szükség esetén a `users` és `books` táblákból lehet lekérdezni.



A `baskets` tábla

Szintén a `users` táblához kapcsoltam a megrendelésekkel kapcsolatos információk tárolására szolgáló `orders` táblát. Ez tartalmazza a tényleges rendeléseket a felhasználói azonosító és a könyvek azonosítója alapján. Az előbb a kosárnál elmondottak nagyrészt igazak, ugyanis ez a tábla is kapcsolódik a `books` táblához. Bekerült azonban egy új mező, a megrendelés dátuma. Ez, míg a kosárba helyezés esetén lényegtelen adat, megrendelésnél annál inkább jelentős, adminisztrátori szemmel nézve nyomon követendő információ.

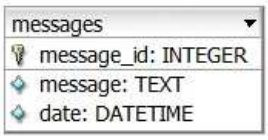


Az `orders` és `payment_methods` táblák

Tartozik az `orders` táblához egy, a fizetési módokat tároló tábla is. Megrendeléskor ugyanis a tagok kiválaszthatják, hogyan szeretnének fizetni a megrendelt termékekért. Ezeket a lehetőségeket tartalmazza a `payment_methods` tábla egy azonosítóval, mint elsődleges kulcs, és a fizetési mód megnevezésével. A tábla kulcsa külső kulcsként szerepel az `orders` táblában.

A mindenképpen szükséges táblák elkészítése után végeztem különböző bővítéseket, hiszen nem mindig láttam előre, hogy milyen adatokra is lesz pontosan szükség. Így bekerült egy-két új mező bizonyos táblákba kiegészítésképpen, és egy további táblát is létrehoztam az adatbázisban. Ez a `messages` nevet viseli, és egy az alkotói folyamat késői szakaszában kialakított funkció miatt hoztam létre. Tulajdonképpen a könyvtárosok és az adminisztrátor

közötti kommunikáció segítését szolgálja, azáltal, hogy lehetővé teszi szöveges üzenetek küldését. Nem kapcsoltam egyik eddigi táblához sem, mert ez a funkció csupán arra való, hogy minél egyszerűbben és gyorsabban értesíteni lehessen az adminisztrátort, ha szükséges. Éppen ezért az üzenet szövegén és a küldés dátumán kívül nem tartalmaz egyéb mezőket a tábla.



A `messages` tábla

Az adatbázis inicializálása nem volt túlságosan komplex feladat, hiszen a benne szereplő táblák legtöbbjének feltöltése az alkalmazás használata közben, űrlapok segítségével történt. A felhasználók adatai regisztrációkor, a könyvekkel, szerzőkkel és dolgozókkal kapcsolatos adatok pedig a különböző menüpontok alatt kitölthető űrlapokkal kerülnek az adatbázisba. Az alkalmazás működése közben töltődnek fel továbbá a `baskets` és `orders` táblák. A `payment_methods` és `acl` táblákat azonban értelemszerűen inicializálni kellett, ezek tartalma a későbbiekben nem módosítható.

4. Fejlesztői környezet

Az alkalmazást Windows 7 operációs rendszer alatt fejlesztettem, Apache webszervert és PHP5-öt használva MySQL adatbázis segítségével. A forráskódok elkészítéséhez PHP Designer 7-et, az egyszerű HTML fájlok és CSS stíluslapok szerkesztéséhez PSPad szövegszerkesztőt használtam. A programot Firefox 3.5.8-as verziójú böngészőhöz fejlesztettem. Az adatbázis egyed-kapcsolat modelljét DBDesigner programmal, az adatbázis kezelését Navicat-tel valósítottam meg.

4.1 PHP Designer 7

Kényelmes PHP fejlesztői környezet és szerkesztőprogram, melyet már évek óta használok. Nemcsak PHP, hanem HTML, CSS, JavaScript fájlok szerkesztésére is alkalmas, valamint bármely PHP és JavaScript keretrendszert is támogat.

4.2 DBDesigner

Egyszerűen használható adatbázis-tervező program, mely teljes grafikus eszközkészlettel rendelkezik, így támogatva a minél egyszerűbb és gyorsabb tervezést. A felhasználónak csupán annyi a dolga, hogy „megrajzolja” a szükséges táblákat, és összekösse azokat valamilyen kapcsolattípussal. Ha kész van az adatbázisunk, kiexportálhatjuk azt SQL scriptbe, melyet bárhol máshol felhasználhatunk. Ha a későbbiekben módosítunk az adatbázison, esetleg új táblát hozunk létre, lehetőség van természetesen csak az adott táblát módosító kód exportálására is.

4.3 Navicat for MySQL

A Navicat nevű szoftver kitűnően alkalmas MySQL adatbázisok fejlesztésére és kezelésére egyaránt. Funkciókban gazdag, mégis egyszerű a használata, ezért is választottam ezt a programot. Az előbb leírt DbDesigner programmal könnyen összehangolható, hiszen csupán egy CREATE utasítást tartalmazó *query* lefuttatásával létrehozható egy tábla, vagy akár az egész adatbázis is. A kapcsolat konfigurálásánál csak arra kell figyelni, hogy az adatok konzisztensek legyenek a fejlesztői környezetben beállítottakkal.

4.4 Böngésző-kompatibilitás

Az alkalmazást igyekeztem úgy elkészíteni, hogy hibátlanul működjön valamennyi böngészőprogramban. Kliens oldalon Firefox 3.5, vagy Internet Explorer 8 böngésző használata szükséges.

5. Felhasználói felület

A megjelenő felület szabványos HTML nyelven készült külső CSS stíluslappal és JQuery kiegészítéssel az interaktív felhasználói üzenetekhez.

A következő részekből áll:

- fejléc: a logóra kattintva a kezdőoldalra jutunk vissza, mely felhasználói csoporttól függően különbözik;
- menüsor: a főbb menüpontokat tartalmazza, szintén különbözik adminisztrátor, könyvtáros és tag esetén;
- bal oldali terület: a fő tartalmi rész, ide töltődnek be az oldalak, azaz az akciók tevékenységei itt jelennek meg:
    - listák, táblázatok
    - űrlapok
    - felhasználói értesítések
    - lapszámozás, ha szükséges, az oldal alján
- jobb oldali terület: a layout része, azaz statikus tartalom; itt találhatóak a kategóriák, linkek, ajánlat, adatmódosítási link; tagok esetén itt jelenik meg a kosár is;
- lábléc: linkek és logó.

A különböző szerepköröknek megfelelő layoutok betöltése a bejelentkezés után történik, és a Zend_Controller_Action osztályba írt preDispatch metódus valósítja meg, miután a Zend_Acl és Zend_Auth elvégezte a szükséges ellenőrzéseket és beállításokat.

```
    public function preDispatch(){
        require_once 'Zend/Session/Namespace.php';
        $namespace = new Zend_Session_Namespace();
        if ($namespace->role == 'admin')
            $this->_helper->layout->setLayout('layoutadmin');
        else if($namespace->role == 'librarian')
            $this->_helper->layout->setLayout('layoutlibrarian');
        else if($namespace->role == 'user')
            $this->_helper->layout->setLayout('layoutuser');
    }
```

5.1 HTML

A HTML nyelv weboldalak készítéséhez kifejlesztett leíró nyelv, mely alapvető internetes sztenderd. Folyamatos fejlődése során eljutottunk a ma használatos 4.01-es verzióhoz, melynek én a HTML4_STRICT változatát használtam alkalmazásom elkészítése során. Ennek sajátossága, hogy nem tartalmazza a már elavult elemeket és attribútumokat, így ha szabályos kódolunk, nem kell a böngészők által támasztott megszorításokhoz alkalmazkodni. A viewscriptek nagyrészt HTML kódból állnak, de a bennük szükséges ciklusok és a controllertől megkapott adatok kiíratásánál kisebb mértékben PHP kódokkal keverednek. Éppen emiatt a kiterjesztésük a klasszikus `.html` kiterjesztéstől eltérően `.phtml`.

5.2 CSS

A HTML 4.0-s verzió óta minden formázást elősegítő elem eltávolítható a HTML fájlból, és külön stíluslapban gyűjthető össze. A sztenderdet követve alkalmazásom megjelenéséért jórészt egy CSS fájl, azaz lépcsőzetes stíluslap felel, melyet az internetről letölthető ingyenes felületek közül választottam ki, majd kisebb átalakításokat végezve rajta a saját igényeimhez alakítottam. Természetesen a program bizonyos részei tartalmaznak ugyan HTML nyelvű alapvető formázó elemeket, de könnyebb a módosítás és átláthatóbb a kód, hogy ha elszeparáljuk a formázásért felelős kódrészleteket. Így a viewscript fájlok legtöbb eleme csak a `class` és `id` attribútumokkal van ellátva. A könyvtárszerkezetben a public mappa css almappája tartalmazza a felhasznált stíluslapot, melyet a layoutban kell hivatkozni ahhoz, hogy automatikusan betöltődjön.

5.3 JQUERY

A JQuery egy praktikus és könnyen tanulható JavaScript eszköztár vagy könyvtár, mely segítségével viszonylag egyszerűen és kevés munkával látványos produktumot hozhatunk létre. Nagy előnye, hogy kis mérete ellenére nagy tudású keretrendszer, és szinte bármely feladatra találunk már megírt plugint, a hozzátartozó dokumentációval. És ami talán még fontosabb, bármely böngészőben futni fognak a helyesen megírt JQuery kódok.

A könyváruházamban egyelőre csak a felhasználóknak szánt értesítő üzenetek elkészítésénél kapott helyet a JQuery, de a tervezett fejlesztések további elemek beépítését is magukba foglalják.

Én is kihasználtam, hogy készen letölthetjük a nekünk tetsző plugint, így a hivatalos honlapról kiválasztottam egy a már elkészült felület színsémájához illő *dialogot* a userMessage-ek kiírására, és beépítettem a programomba a következő kódrészlettel, mely a layoutból való:

```
<script type="text/javascript">
     $(function(){
          $('#dialog').dialog({
               autoOpen: true,
               width: 300,
               position: 'center',
               buttons: {
                    "Ok": function() {
                         $(this).dialog("close");}}
          });
     });
</script>
```

És a kód által a megfelelő CSS-sel létrehozott párbeszédablak:

Az üzeneteket megjelenítő dialog

6. Az alkalmazás működése

6.1 A vendégek szemszögéből

A programomat úgy építettem fel, hogy bizonyos funkciók a nem regisztrált felhasználók számára is elérhetőek legyenek. Hiszen amikor valaki rátalál a honlapunkra, legalább minimális szinten biztosítanunk kell, hogy megismerkedjen a lehetőségekkel. Éppen ezért a vendégek számára látható felhasználói felület a tagok számára elkészítettel azonos. Láthatóak tehát az alapvető menüpontok, melyekből a felhasználónk fel tudja térképezni, hogy mire alkalmas a program, és hogy eleget tesz-e az általa támasztott elvárásoknak. A Regisztráció menüpont természetesen elérhető, hiszen ezen keresztül válhat taggá az oldalt először látogató, továbbá a Könyvespolc menüpont, azaz a könyveink listája, és a Keresés menüpont is bizonyos korlátozásokkal. Azért tartottam fontosnak, hogy kipróbálhassák a keresést is, mert én is gyakori látogatója vagyok bizonyos regisztrációt igénylő oldalaknak, de általában – annak ellenére, hogy 1-2 percet vesz igénybe – csak akkor regisztrálok, ha rátaláltam az általam vágyott termékre. Elérhető még a Tudnivalók, és a Kapcsolat menüpontok is, hogy megbizonyosodjanak vendégeink arról, hogy megbízható a honlapunk, és hogy megfelelően zajlik a vásárlás menete, ahogy azt az egyéb webes áruházak esetében megszokhatták. A vendég szerepkörbe tartozó felhasználók tehát bizonyos korlátozásokkal lekérdezhetnek adatokat, de módosítási jogunk nincsen semmilyen tekintetben sem. A regisztrált felhasználókra vonatkozó adatokhoz természetesen nem férnek hozzá.

6.2 A tagok vagy felhasználók szemszögéből

Felhasználó vagy tag lesz egy az oldalunkra látogatóból, hogyha a Regisztráció menüpontban kitölti az adataival a regisztrációs űrlapot. Ezután bármikor az oldalra látogatva a Belépés menüpontban kell megadnia a felhasználói nevét és a jelszavát a belépéshez. A tagok számára a legfontosabb honlapunk gyors és egyszerű kezelhetősége mellett, hogy minden számukra szükséges információt és funkciót könnyen elérhessenek. Azért látogatják az áruházat, hogy nézelődjenek, böngésszenek a könyvek között, és adott esetben vásároljanak is. Ehhez minden menüpont rendelkezésükre áll, s ha megtalálták, amit keresnek, azonnal meg is rendelhetik. A Keresés menüpontban több szempont szerint kereshetnek, illetve kategóriák vagy ABC sorrend szerint tekinthetik meg a könyvek listáját. Fontos számukra az is, hogy keresgélés közben ne kelljen gondot fordítani arra, hogy egyesével rendeljék meg a termékeket, vagy

észben tartsák, hogy melyeket fogják megrendelni. Erre szolgál a Kosár, mely bejelentkezés után bármikor megtekinthető, törölhetünk belőle, kiüríthetjük, és a vásárlás végösszegét is láthatjuk. Ha mindent rendben talál a vásárló, akkor véglegesítheti a rendelést. A regisztrált tagok megváltoztathatják vagy kiegészíthetik a regisztrációkor megadott személyes adataikat is az Adatmódosítás link alatt.

6.3 A könyvtárosok szemszögéből

A könyvtárosok tulajdonképpen a virtuális áruház dolgozói, azaz feladatuk a hagyományos eladói szerepkörrel egyezik meg. Számukra a felhasználókétól különböző felhasználói felületre lesz szükség, hiszen ők, mint vásárlók nem érintettek, viszont olyan feladatokat látnak el a rendszerben, amelyekhez a tagoknak, vagy vendégeknek nincs hozzáférésük. Erre tekintettel más menüpontokat tartalmaz a hozzájuk rendelt felhasználói felület. A könyvtáros számára az egyik legfontosabb funkció a terméklista folyamatos karbantartása. Ehhez szükség van az adatbázissal kommunikálva ahhoz új terméket hozzáadni, módosítani a már meglévőket, vagy törölni azokat. A másik alapvető feladatuk az alkalmazás eseményeinek és a felhasználók tevékenységeinek figyelése, hiszen a rendszer helyes működése az ő felelősségük; az ő dolguk, hogy szükség esetén értesítsék az adminisztrátort, és kikérjék a segítségét, ha a felmerülő problémákkal nem tudnak megküzdeni. Ennek érdekében láthatják a felhasználók adatait, a megrendelésekről is hozzáférnek alapvető információkhoz, továbbá üzeneteken keresztül kommunikálhatnak az adminisztrátorral. Hangsúlyoznám azonban, hogy módosítási joggal csak a könyvekkel kapcsolatos adatokra, vagy a saját adatikra vonatkozóan bírnak, a tagokkal és rendelésekkel kapcsolatos információkat megváltoztatni nem tudják.

6.4 Az adminisztrátor szemszögéből

A rendszer adminisztrátorának biztosítani kell a korlátlan hozzáférést, hiszen ő tartja karban a programot, és gondoskodik a megfelelő működésről. A másik lényeges igény a tagok adataihoz, tevékenységeihez, és a megrendelésekhez való hozzáférés, hogy mindig naprakész lehessen. Az adminisztrátor nem csak hozzáfér, de bármely adatot módosítani is tud. Erre a különböző menüpontokban van lehetőség, de mivel az adatbázist is ő tartja karban, így bármely adatot láthat, és meg is változtathat. Figyelemmel kíséri a rendszert, de leginkább akkor avatkozik be, mikor a dolgozók nem tudnak megoldani egy felmerülő problémát, legyen az például egy felhasználói igény, vagy a rendszer működésében adódó zavar.

7. Felhasználói dokumentáció

Ez a fejezet a webes könyváruházam felhasználói dokumentációja. A program fő célja, hogy korszerű és „divatos” technológiák segítségével, jól átlátható felhasználói felülettel valósítson meg egy általános kereskedelmi alkalmazást, melyet interneten keresztül webes felület segítségével tudunk elérni.

A felhasználói dokumentáció alapvető tartozéka bármely számítógépes szoftvernek és elsődleges feladata, hogy a felhasználókat vagy a lehetséges felhasználói csoportokat részletesen megismertesse a program használatával, lépésről lépésre elemezve az egyes menüpontokat és funkciókat. Ezért a programom elkészítése és tesztelése után az én szakdolgozatomban is helyet kapott egy ilyen leírás, mellyel igyekszem bemutatni mindazt, amit egy felhasználónak tudni szükséges.

Mint már korábban említettem, több felhasználói csoport eltérő jogosultságokkal használhatja a szoftvert, s ehhez a felhasználói dokumentáció is igazodik, külön fejezetekben taglalva az egyes csoportokra vonatkozó funkciókat és lehetőségeket.

7.1 Felhasználói csoportok

Az alkalmazást négy különböző felhasználói csoport használhatja, ezek a vendég, a tag vagy általánosabban felhasználó, a könyvtáros és az adminisztrátor. A különböző csoportokhoz természetesen különböző felület, jogosultságok és funkciók tartoznak, melyeket a későbbiekben részletesen kifejtek.

7.2 Felület és menüsor

A menü általában egy honlap vagy alkalmazás legfontosabb, leggyakrabban használt elemeit gyűjti össze, hogy a felhasználók a lehető leggyorsabban megtalálják az információkat, elérjék a szükséges oldalakat. Az én programom is követi a hagyományokat, így a menüsorból közvetlenül elérhetőek az általa biztosított alapvető szolgáltatások. Bizonyos menüpontok elérhetősége függ attól, hogy az adott felhasználó rendelkezik-e a szükséges jogosultsággal. A felhasználói felület valójában négy különböző felületet foglal magába, melyek bejelentkezés után szerepkörtől függően töltődnek be. Ezek az alapvető felépítésükben nem különböznek,

viszont minden szerepkörhöz más-más menüsor tartozik, és bizonyos elemek a jobboldali tartalmi részben is változnak.

7.3 Vendég

Már korábban említettem, hogy a vendégek számára nem készítettem külön felület, a web-áruházaknál szokásos módon pontosan azt látják, amit a regisztráció után is fognak. Amely menüpontok eléréséhez nincsen jogosultságuk, a rendszer párbeszédablakban megjelenítve figyelmezteti őket. Az számukra elérhető oldalak a Regisztráció, Könyvespolc, Újdonságok, Keresés, Tudnivalók és a Kapcsolat menüpontokban találhatóak. Ezek működését a Regisztráció kivételével a tagokra vonatkozó részben ismertetem részletesen.

Regisztráció

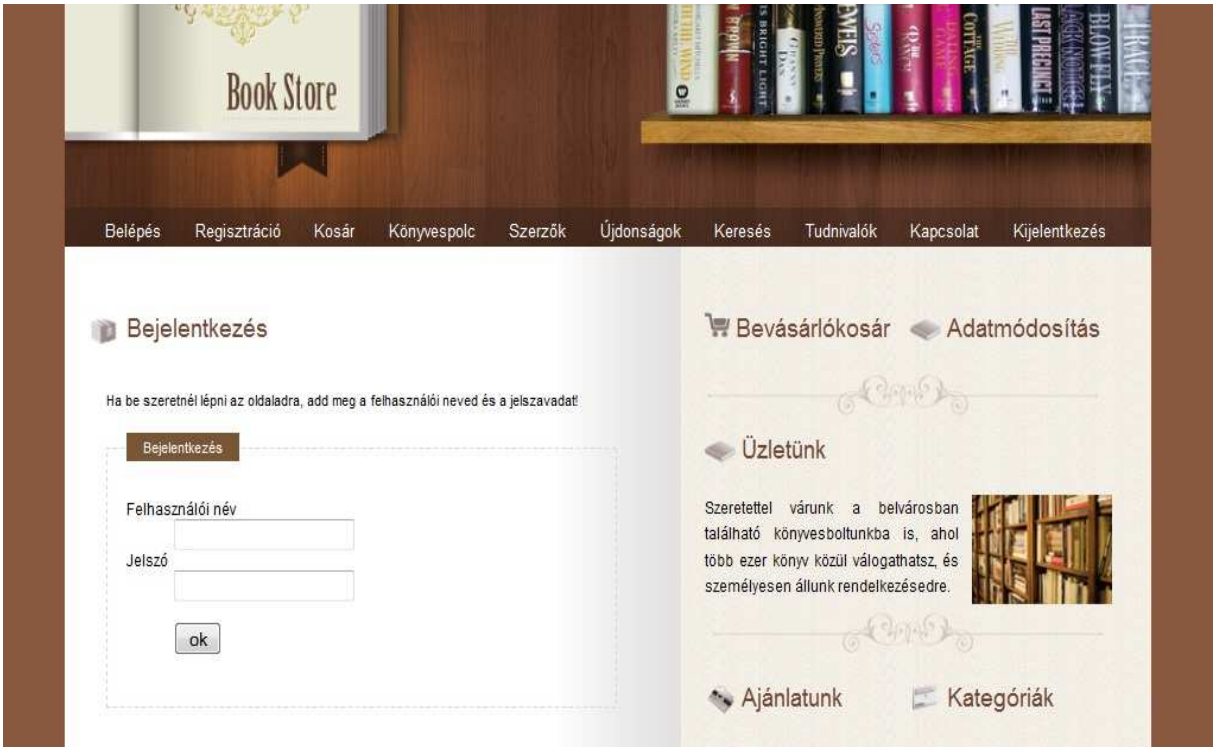
Ahhoz, hogy egy az oldalamra látogató vendég taggá válhasson, bizonyos adatok megadásával regisztrálnia kell. Ez egy regisztrációs űrlap segítségével történik, mely követi a megszokott sztenderdeket. A kitöltendő mezők közül a felhasználói név és a jelszó megadása természetesen kötelező, hiszen ezek alapján végzi a rendszer a későbbiekben az ellenőrzést és a beléptetést. A többi mező megadása opcionális, akár később is elvégezhető, például ilyen a postacím mező, amely a megrendelések kézbesítéséhez szükséges. Minden itt megadott adatunk a későbbiek folyamán bármikor módosítható az adatmódosítás linkre kattintva. Amint regisztráltunk, a rendszer értesít minket a sikeres regisztrációról, majd felkínál egy linket, mely a bejelentkező oldalra irányítja át a frissen taggá vált felhasználót.

7.4 Tag

A regisztrált tagok számára készített felület – mint már említettem – a vendégekével azonos, de tagjaink számára minden menüpont hozzáférhető. Minden könyvekkel kapcsolatos információt megtekinthetnek, kereshetnek, és vásárolhatnak is. Számukra már a jobboldali sáv is tartalmaz funkciókat, itt találják ugyanis a kosárra és az adatmódosításra vonatkozó linkeket.

Bejelentkezés

Amikor egy felhasználó az oldalra érkezik, rögtön a bejelentkező oldal fogadja. Ugyan a felhasználónévvel nem rendelkező vendégek is nézelődhetnek az áruházban, mégis leggyakrabban tagok látogatják az ilyen jellegű honlapokat. A bejelentkezés menüpontra kattintva ugyanez az oldal töltődik be. Bejelentkezéskor a korábban általunk választott felhasználói nevet és a jelszót kell megadni. Amennyiben a név és a jelszó is helyes, azaz a rendszer megtalálta őket az adatbázisban, elvégzi a beléptetést. Innentől kezdve bármely menüpontra kattintva a program természetesen „emlékszik” az adott felhasználóra, egészen addig, míg a Kijelentkezésre nem kattint.



A bejelentkezési panel

Üdvözlő oldal

Ha helyesen adtuk meg a felhasználói nevünket és a jelszavunkat, akkor a rendszer a rendszer egy oldalt tölt be, ahol először is üdvözli a felhasználót a nevén szólítva, másrészt kilistázza a legújabb termékeket. Ha elnavigálunk az oldalról, az a későbbiekben is bármikor elérhető, hiszen külön menüpont hivatkozik ide. Ez az oldal az Újdonságok menüpont alatt található, melyről később részletesebben írok.

Kijelentkezés

Ez a menüpont végzi a felhasználó programból való kijelentkeztetését, mely akkor szükséges, mikor egy felhasználó befejezte a vásárlást, el szeretne navigálni az oldalról, netán másik felhasználói névvel szeretne belépni. Ilyenkor a rendszer törli a felhasználó ideiglenesen tárolt adatait, és a bejelentkező oldalt tölti be.

Könyvespolc

Ez az alapvető menüpont az áruházban kapható termékek listáját tartalmazza. Ha ide kattintunk, apró dobozokban jelennek meg a könyvek ABC sorrendben. A dobozok felett látható a könyv címe, a dobozokban pedig a könyv borítója. Ha ezek valamelyikére kattintunk, a választott könyvről részletes információkat tartalmazó oldal töltődik be, ahol bővebben tájékozódhatunk. Akár a doboz alján lévő „kosárba” gomb segítségével, akár a részleteket tartalmazó oldalról azonnal kosárba helyezhetjük az adott terméket.

Kosár

A kosár megtekintéséhez először be kell jelentkezni a felhasználónak, azaz kosárral csak az rendelkezik, aki az oldalon regisztrált tagként van nyilvántartva. Belépés után a felület jobboldali sávjában található Bevásárlókosár feliratra kattintva érhető el. A kosár arra szolgál, hogy a vásárló által már kiválasztott, de még meg nem rendelt termékeket tárolja. Ide gyűjthetjük nézelődés közben a könyveket, majd eldönthetjük, hogy valóban szeretnénk-e vásárolni. Ha erre a menüpontra kattint a felhasználó, a korábban kosárba helyezett könyveket láthatja, feltüntetve a címet, a darabszámot és az árat. Minden sor végén található egy „törlés” link, mely arra szolgál, hogy ha a kedves vásárló meggondolja magát, minden gond nélkül visszatehesse az adott könyvet a könyvespolcra. Ilyenkor az a kosárból törlődik, és

visszakerül az eredeti helyére az adatbázisban. A lista alján látható a vásárlási végösszeg, továbbá két link, a „kosár kiürítése” és a „vásárlás”. A kiürítéssel megkíméljük a vásárlót attól, hogy a kosara tartalmát egyesével kelljen törölni, azaz ide kattintva minden kosárban lévő könyv törlődik és visszakerül a könyvespolcra. A „vásárlás” linkkel tudunk továbblépni, ha befejeztük a válogatást, és rendelni szeretnénk. Ekkor a Megrendelés megerősítése oldalra jutunk, ahol még egyszer ellenőrizhetjük a megvásárolni kívánt termékek listáját. A lista alján található linkek segítségével visszaléphetünk a kosárhoz, vagy véglegesíthetjük a megrendelésünket. Ilyenkor a rendszer egy oldalra irányítja át a vásárlót, ahol egy legördülő listából választhatja ki a kívánt fizetési módot, mely lehet készpénz vagy átutalás. Ha ezzel megvagyunk, a vásárlás véglegesítése után a rendszer értesít a sikeres megrendelésről, és a kosár kiürül.

Keresés

Ha a keresés menüpontra kattintunk, két panel jelenik meg egymás alatt. A felül található Gyorskeresés funkció arra használható, hogy bármilyen szót vagy karaktersorozatot beírva a keresés az adatbázisban mind a szerzők nevére, mind a könyvek címére lefut. Azaz az eredmény tartalmazhat szerzőket és könyveket is, melyeket egymástól különválasztva jeleníthetünk meg. Mind a könyvek listájában, mind a szerzők felsorolásakor minden sor után található egy link, ahol könyv esetében a könyv részleteit, szerző esetén pedig az általa publikált és az üzletben megtalálható könyvek listáját találjuk.

A második panelben különböző keresési feltételek közül választhatunk, mint a szerző, kategória vagy cím. Ezek közül bármelyikre kattintva egy új oldal töltődik be, ahol cím és szerző esetén beírhatjuk a keresett kifejezést vagy nevet, kategóriák szerinti keresés esetében pedig legördülő menüből választhatjuk ki a megfelelő kategóriát. Bármely keresési feltételt adjuk meg, az eredményt mindig egy táblázatban kapjuk, ahol fel van tüntetve a könyv címe, a szerzője és az ára is, valamint a sorok végén egy „kosárba” linket látunk. Ha megtaláltuk, amit keresünk, és erre a linkre kattintunk, a terméket azonnal a kosarunkba helyezhetjük.

Újdonságok

Az újdonságok menüpontot azért hoztam létre, hogy a lehető leggyorsabban, már a menüből el lehessen érni a legfrissebb könyveket. Ugyan a bejelentkezés után is ez a lap töltődik be, de nem biztos, hogy minden vásárló az új könyvek átböngészésével kezdi a látogatást. Ez a lista

a könyvek listájához hasonlít, de eddig nem árusított könyvek révén nagyobb dobozokba helyeztem a termékeket, és a hozzájuk tartozó rövid leírást is megjelenítettem. Az egyes dobozok jobb alsó sarkában lévő „tovább” linkre kattintva a szokásos könyvekre vonatkozó részletes leírás töltődik be, ahol már nemcsak a cím van feltüntetve, hanem az ár is, és a kosárba helyezés is lehetséges.

Az Újdonságok menüpont

Tudnivalók

Mint minden internetes honlapon, ahol valamilyen szolgáltatás vagy termék értékesítése zajlik, fontos, hogy a vásárlók megrendelés előtt tájékozódjanak a vásárlási feltételekről és tudnivalókról. Ezeket az információkat az én áruházam vásárlójaként is megtalálják, a Tudnivalók menüpontra kattintva. A betöltődő oldalon olvashatjuk a megrendeléssel kapcsolatos feltételeket, hogy mely adatokat szükséges megadni, a fizetési és kézbesítési lehetőségeket és az alapvető szabályokat.

Kapcsolat

Egy másik alapvető menüpont, mely szinte minden internetes oldal tartozéka. Arra szolgál, hogy az áruházról és az elérhetőségekről információt szolgáltassunk. Másrészt lehetőséget biztosít a felhasználók számára – legyenek tagok vagy csupán vendégek – hogy kapcsolatba lépjenek az adminisztrátorral vagy a tulajdonossal. Ezt nagyon fontosnak tartom, hiszen a kereskedelemben rendkívül sok múlik a vásárlók véleményén. Ezúton jelezhetik igényeiket, feltehetik esetlegesen felmerülő kérdéseiket, értesíthetnek minket a honlappal kapcsolatos problémákról, vagy egyszerűen csak véleményt nyilváníthatnak.

Linkek

Ez a lista nem képezi szerves részét a webáruház tartalmi részének. A jobb oldalsó sáv alsó részén található, és funkciója csupán annyi, hogy egy kis ízelítőt adjon olyan honlapokról, melyek érdekesek lehetnek a könyvek és a kultúra iránt érdeklődők számára. A linkekre kattintva automatikusan új ablakban nyílik meg az adott honlap.

Szerzők

Ebben a menüpontban egy listát találunk táblázatos formában, mely ABC sorrendben tartalmazza az áruház adatbázisában szereplő írókat. Az írók listájában minden sor végén egy „könyvek” linket látunk, mely az adott szerző könyveinek listáját tölti be. Fontos felhívni a figyelmet arra, hogy előfordulhat olyan eset, mikor egy a listában szereplő íróhoz nem talál a rendszer könyveket, melyeket kilistázhatna. Ez úgy lehetséges, hogy vagy töröltük a korábban itt szereplő könyvet vagy könyveket, vagy a vásárlói megrendelések során kimerült a raktár. Ilyen esetben nem töröljük a szerzőt, hiszen bármikor újabb könyveket tölthetünk fel a neve alá. Ha kilistázzuk egy szerző könyveit, a Könyvespolc menüponthoz hasonlóan kis dobozokban látjuk a könyveket, melyekben fel van tüntetve a könyv ára, és a „kosárba” link, azaz innen közvetlenül a kosarunkba tehetjük a vágyott könyvet. Ha a képre vagy a kép felett látható címre kattintunk, a részletes leírást adó oldal töltődik be, ahol a további információkról tájékozódhatunk.

Részletek

Ez nem egy külön menüpont, mégis fontos megemlíteni a könyvek részletes leírását megmutató oldalt. Ilyen oldalra jutunk, ha a webáruházon belül bárhol egy könyv képére kattintunk, sok esetben pedig a könyvcímek, vagy a „tovább” illetve „részletek” linkek navigálnak minket ide. Ezen a lapon megtalálhatók az alapvető információk, mint szerző, cím és fogyasztói ár, de további adatok is megjelennek a választott könyvről. A bal oldalon a már előbb is látott képet találjuk, jobb oldalon pedig a szerző és cím kiírás alatt először egy rövid leírást a könyvről, majd rendre a kategória, nyelv, ISBN szám, megjelenés dátuma és az adatbázisba regisztrálás dátuma szolgáltatnak plusz információkat. Szem előtt kell azonban tartsuk, hogy a könyvek feltöltésénél a könyvtáros csak a rendelkezésére álló adatokat tudja bevinni a rendszerbe, így a részletes leírások hiányosak lehetnek.

Adatmódosítás

Miután egy felhasználó beregisztrál az oldalra, az általa megadott személyes adatai bekerülnek a rendszer adatbázisába. Ha azonban a későbbieknek bármilyen változtatás történik adataiban, vagy csak szeretné módosítani például a felhasználói nevét, szükség lehet az adatbázisbeli módosításokra is. Ilyen eset lehet például a vásárló kézbesítési címének megváltozása, vagy új e-mail cím létrehozása. Hogy a vásárlóim megváltoztathassák adataikat, létrehoztam az adatmódosító oldalt, mely a jobb oldali sávban található közvetlenül a kosár mellett. A feliratra kattintva betöltődik az előbb említett oldal, ahol a regisztrációnál korábban látott űrlap található azzal a különbséggel, hogy a mezők ki vannak töltve az előzőleg megadott adatokkal. Ha üres mezőt látunk, az azt jelenti, hogy a regisztrációnál nem töltöttük ki ezeket a mezőket. Az adatmódosításnál természetesen megtehetjük, továbbá bármely más adatot is átírhatunk. Ha készen vagyunk a változtatásokkal, az „ok” gombra kattintva az adatok módosulnak az adatbázisban.

Az egyetlen adat, amelyet nem tudunk itt módosítani, az a jelszó mező. Arra egy külön linket hoztam létre, itt végezhetjük el a szükséges módosítást.

Jelszómódosítás

Ha egy felhasználó módosítani szeretné a jelszavát, csak az adatmódosító oldalon található linken keresztül teheti meg, ehhez a funkcióhoz ugyanis nem hoztam létre saját menüpontot.

Ha betöltjük az oldalt, a már megszokott külsejű űrlap fogad minket. Először meg kell adnunk a régi jelszavunkat, majd begépelhetjük az újat. A biztonság kedvéért kétszer kell az új jelszót megadni. A rendszer először ellenőrzi a régi jelszó helyességét, majd az újonnan megadott két szó egyezését. Ha mindent helyesen adtunk meg, legközelebbi belépéskor már az új jelszavunkkal tudunk belépni.

7.5 Könyvtáros

A könyvtáros az a felhasználónk, aki az adatbázis naprakészen tartásáért felelős, és alapvető adminisztratív feladatokat lát el. Ezért természetesen szükséges volt, hogy a vásárlóktól elkülönítsem őket, mind az adatbázisban, mind a számukra elérhető funkciók tekintetében. Ehhez pedig a felhasználói felületen is változtatni kellett. Ugyan látványra a felhasználói felület nem sokban különbözik a vásárlók számára létrehozottól, de a menüsorban szereplő menüpontok megváltoztak. Alapvető funkciók egy könyvtáros számára, hogy megtekintheti a könyvek és szerzők listáját, a megrendeléseket nyomon követheti, és legfontosabb, hogy ő felel a könyvek adatbázisba való rögzítéséért, és a szükséges adatmódosítások elvégzéséért. Ha a jobb oldalsó sávra tekintünk, észrevehetjük, hogy eltűnt a kosár link is, ugyanis könyvtárosként erre sem lesz szükség.

Terméklista

Ez a menüpont tulajdonképpen a tagok számára elérhető könyvespolcnak felel meg, a könyvtárosok számára átalakítva. A korábban bemutatott esztétikusabb „dobozos” megjelenítést egy egyszerű táblázatra cseréltem, hiszen itt inkább a könyvek adatai fontosak a honlap kinézete és a könyvborítók színes látványa helyett. A táblázatban a könyvek lényegesebb tulajdonságait jelenítettem meg, ezzel biztosítva, hogy könyvtárosaink minden szükséges információhoz hozzáférjenek. Látható, hogy a könyvcímek mind hivatkozások, ezeken keresztül a már korábban bemutatott „részletek” oldal tölt be. Itt minden a könyvről tárolt adat fel van tüntetve. A sorok végén található „módosít”, illetve „töröl” linkekkel elvégezhetjük az adatmódosítást, vagy törölhetjük a könyvet az adatbázisból. A törléssel természetesen nem a könyv darabszámát nullázzuk, hanem a teljes rekordot eltávolítjuk, és a termék kikerül az adatbázisból. Alapértelmezés szerint ilyenkor a könyv szerzője természetesen nem törlődik, de ha szükséges, ezt elvégezhetjük a szerzők menüpontban.

A módosítás link alatt egy a feltöltésnél megjelenő formot látunk, kitöltve az eddig ismeretes adatokkal. Bármely adatot módosítani lehet, vagy az eddig hiányzó információkkal kitölthetjük az üres mezőket. Az ok gombbal menthetjük a változtatásokat az adatbázisba.

Bejelentkezés után is a terméklista egy speciális változata fogadja a könyvtárost, mely oldalra a fejléc logója is hivatkozik. Itt a feltöltés időpontja szerinti fordított sorrendben jelennek meg a könyvek az előbb bemutatott táblázatban.

Feltöltés

Az új könyvek terméklistára való rögzítéséhez a könyvtárosnak egy már a korábbiakban látottakhoz hasonló űrlapot kell kitölteni. A cím, szerző, kategória, ár és darabszám mezők kitöltése kötelező. Ezek közül a kategória egy legördülő listából választható ki, a többi mező szövegesen illetve számmal adható meg. A további mezők kitöltése technikailag opcionális ugyan, de a célunk, hogy minél pontosabb adatokkal rögzítsük a könyveket az adatbázisba. Ha a későbbiekben új információ jut a könyvtáros birtokába valamely könyvről, természetesen lehetősége van módosításokat végezni. A leírás mezőbe hosszabb szöveget gépelhetünk, mely a könyvek listázásánál nem látszik, csak akkor jelenik meg, mikor egy felhasználó a könyvre kattint. Ide bármi kerülhet, akár egy idézet a könyvből, akár egy kritikarészlet, vagy a szerző saját gondolatai a művéről. A képfeltöltésnél a szövegmezőbe vagy a tallózás gombra kattintva a már megszokott intézőablak ugrik fel, ahol a könyvtárosunk megkeresheti a megfelelő képet a feltöltendő könyvhöz. A rendszert igyekeztem úgy felépíteni, hogy a könyvtárosnak minél egyszerűbb dolga legyen, és ne kelljen bonyolult és hosszadalmas feladatok elvégzésével bajlódnia. Ezt a célt szolgálja a képfeltöltés implementációja is, hiszen még a fájlnévre sem kell figyelni, a rendszer átnevezi a számára megfelelő módon. Ha végeztünk a mezők kitöltésével, és az ok gombra kattintunk, az adatok rögzítésre kerülnek az adatbázisban. Feltöltés után a program értesít a sikeres feltöltésről, majd felajánlja újabb könyv feltöltését. Innentől a termék megtalálható lesz a könyvek listájában, keresés közben is rátalálhatunk, de a tagok számára létrehozott Újdonságok menüpontban is automatikusan az adatbázishoz legkésőbb hozzáadott könyvek fognak megjelenni, így erre sem kell a könyvtárosoknak külön gondot fordítani.

A feltöltési űrlap

Felhasználók

A felhasználók listája csak tájékoztató jellegű a könyvtárosok számára, ők nem végezhetnek adatmódosításokat. Ezt csak maga a tag vagy az adminisztrátor teheti meg. Szükség lehet viszont adatellenőrzésre, és a regisztrációk nyomon követésére. A táblázatban látható a felhasználók neve, azonosítója, e-mail címe és postacíme.

Megrendelések

A tagok tevékenységeinek figyelemmel kísérését segíti a Megrendelések menüpontban látható táblázat. A megrendelések csak véglegesítés után kerülnek ide, és ekkor már sem a megrendelő, sem a könyvtáros nem tudja azokat módosítani, vagy törölni.

Kategóriák

Itt a könyvtáros megtekintheti az összes könyvkategóriát, és módosításokat végezhet. Nem túl gyakori, de szükség lehet a kategória törlésére, melyet az adott kategória sorában szereplő „törlés” linkkel tehetünk meg. Ugyanezen az oldalon, lentebb a táblázat alatt új kategóriát lehet az eddigiekhez adni. A szövegmező kitöltés után az ok gombra kattintunk, és az új kategória bekerül a listába; a továbbiakban akár feltöltésnél, akár keresésnél megjelenik a legördülő listában.

Adminisztrátor

Ez a menüpont rendkívül fontos a rendszer megfelelő működéséhez, hiszen kommunikációs csatornát biztosít a dolgozó és az adminisztrátor között. Ez azért szükséges, mert a könyvtárosok hatásköre korlátozott, viszont előfordulhatnak olyan esetek vagy problémák, mikor a „tejhatalmú” adminisztrátor közbeavatkozása elengedhetetlen. Példaképpen vegyük azt az esetet, mikor egy felhasználónk véglegesített egy rendelést, de még kikézbesítés előtt visszavonná azt. Ilyenkor a könyvtáros nem fér hozzá az adatbázishoz, így nincs jogosultsága törölni a megrendelést, viszont értesíteni tudja az adminisztrátort a felmerült problémáról. A funkció lényege, hogy a dolgozó rövid, szöveges üzenetet küldhet az adminisztrátornak, aki azt bejelentkezés után azonnal elolvashatja a kezdőlapján.

7.6 Adminisztrátor

Az adminisztrátori jogosultságokkal bíró felhasználó egyrészt a rendszer teljes körű karbantartásáért felel, másrészt a többi felhasználói csoporttal ellentétben korlátlan hozzáféréssel rendelkezik a rendszerrel és a felhasználókkal kapcsolatos minden információhoz. Az adminisztrátor felhasználói felülete is csak a menüsorban tér el az előbbiekben leírtaktól, de fontos megjegyezni, hogy bár menüpontszerűen nincs minden funkció feltüntetve, az adminisztrátor szerepköréből adódóan a honlap egészére nézve bármely művelethez hozzáférhet szükség esetén. A számára megtekinthető listákban már mindenhol ott szerepel a módosítás oszlop is, az adminisztrátor ugyanis a könyvtárosokkal és a megrendelésekkel kapcsolatosan is végezhet műveleteket.

Felhasználók

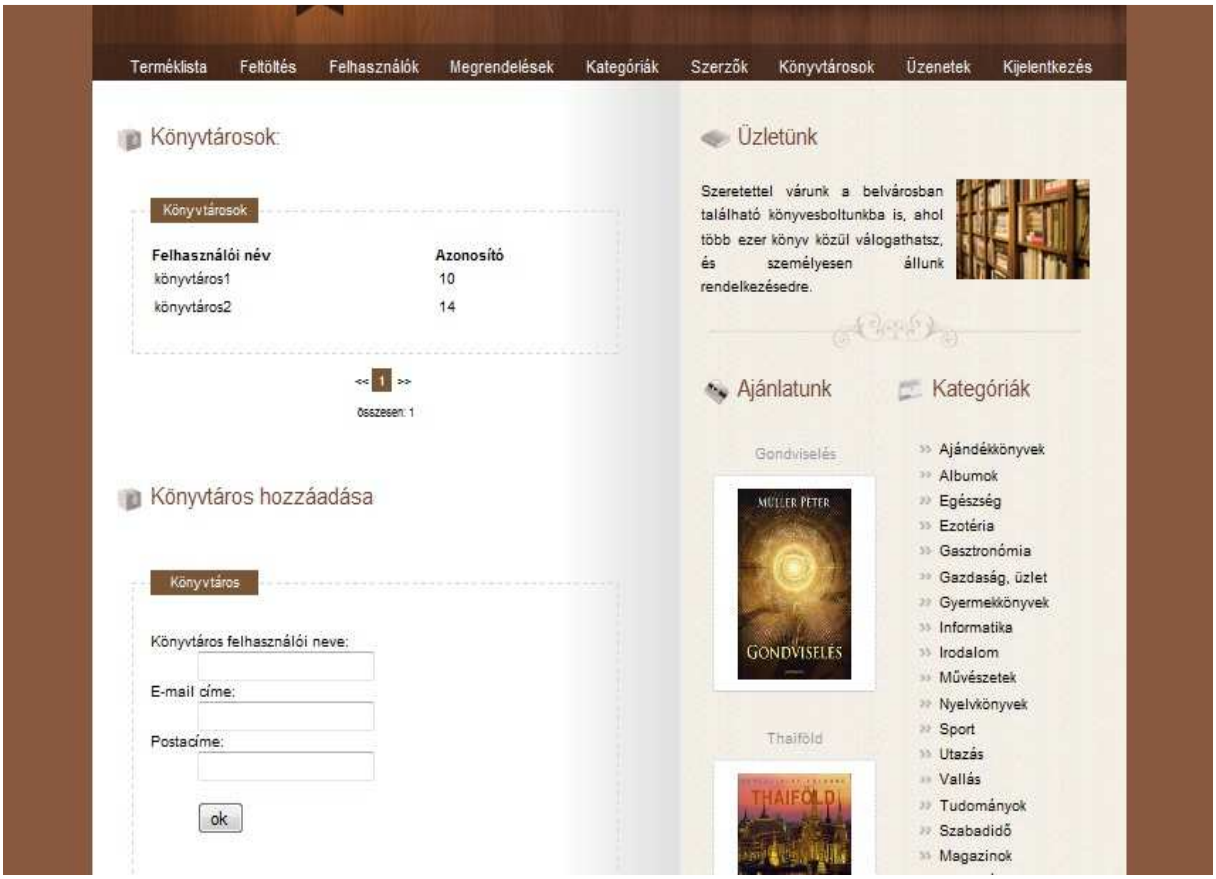
A könyvtárosok számára elérhető felhasználókat tartalmazó táblázat kibővített változatát láthatjuk, ha erre a menüpontra kattintunk. Az adminisztrátornak jogosultsága van a felhasználók adatainak módosítására is, így például törölheti a szabályokat megszegő tagot. Az adatmódosítás ilyen formája persze ritka, de lényeges, hogy csak a felhasználó saját maga, és az adminisztrátor fér hozzá ehhez a funkcióhoz.

Megrendelések

Ez a menüpont a felhasználók megrendeléseit tartalmazza. A táblázatra igazak a korábban a könyvtáros szerepkörnél leírtak, azzal a módosítással, hogy egy új oszloppal bővült. Most már minden sor végén egy „törlés” linket láthatunk, melynek segítségével az adminisztrátor törölheti a megrendelést, persze csak ha a megrendelő tag még kézbesítés előtt jelzi az erre vonatkozó igényét.

Könyvtárosok

Ebben a menüpontban a könyvtáros szerepkörbe tartozó felhasználók listáját tekintheti meg az adminisztrátor. Itt végezhetjük a dolgozókkal kapcsolatos adminisztrációs feladatokat. A táblázat minden sorában egy könyvtáros adatait látjuk. Itt tudjuk a könyvtárost eltávolítani a rendszerből, vagy kicsit lentebb haladva az oldalon egy a már megszokott panelben vehetünk fel új dolgozót a Könyvtáros hozzáadása cím alatt. Új könyvtáros regisztrálásakor meg kell adnunk a felhasználói nevét, és opcionálisan az e-mail- és postacímét is beállíthatjuk. Fontos, hogy ezt a műveletet csak az adminisztrátor végezheti, hiszen a rendszer alapbeállítását úgy adtam meg, hogy a Regisztráció menüpontban regisztrálók szerepköre automatikusan tag legyen. Könyvtárosunk rögzítése után, adatait természetesen már az első belépéskor megváltoztathatja az adatmódosítási linkre kattintva.



A könyvtárosok listája

Üzenetek

Mint már korábban a könyvtárosoknál leírtam, lehetőség van köztük és az adminisztrátor közötti kommunikációra a honlapon belül, rövid üzenetek segítségével. A dolgozók által küldött üzeneteket az Üzenetek menüpontban tekintheti meg az adminisztrátor. Egy egyszerű táblázatban jelennek meg az üzenet szövegével, és a beérkezés dátumával. Ezeknek az üzeneteknek csak az adminisztrátor értesítése a funkciója az esetlegesen felmerülő problémákról vagy felhasználói kérésekről, ezért nincsen szükség további adatok megjelenítésére.

7.7 Jogosultság-ellenőrzés

A rendszer bejelentkezéskor ellenőrzi, hogy az illető mely felhasználói csoport tagja, azaz mely szerepkör vonatkozik rá. Ez alapján korlátozza az oldalakhoz való hozzáférést. Amennyiben egy olyan oldal megtekintését kíséreljük meg, melyhez nincs jogosultságunk

hozzáférni, az alkalmazás egy új oldal betöltésével egyidejűleg egy felugró párbeszédablakban közli ezt. Az ok gomb megnyomása után az oldalon található linkre kattintva bejelentkezhetünk a megfelelő felhasználó névvel, ha eddig nem tettük meg, és megtekinthetjük a kívánt oldalt, ha az ahhoz szükséges szerepkör vonatkozik ránk.

Felhasználói üzenet jogosulatlan hozzáférés esetén

8. Programozói dokumentáció

Ebben a fejezetben a programom működését programozói szemszögből közelítem meg. A felhasználói dokumentációban részletesen elemzek minden funkciót, de arról nem esik szó, hogy azok hogyan valósulnak meg, és mi is történik valójában a háttérben. Túl hosszadalmas lenne azonban végigkövetni az alkalmazás minden tevékenységét kódszinten és megmagyarázni minden sort, ezért csak egy részfolyamatot fogok részletesebben elemezni. Próbáltam olyan részt választani, mely lényegi funkciókat tartalmaz, több controller munkája szükséges hozzá, és természetesen az adatbázissal történő kommunikáció is bemutatható a segítségével.

Talán a leginkább alkalmas minderre, ha végigkövetünk egy megrendelést a könyv kiválasztásától a megrendelés véglegesítéséig. Tegyük fel most, hogy a felhasználó bejelentkezés után az új könyvek listáját átböngészve talál valami kedvére valót. Induljunk el tehát az új könyveket kilistázó newbooksActiontől, mely a BookController része. Az akciók megírásakor minden esetben az első feladat, hogy megvizsgáljuk, mely táblákat fogjuk használni, és hogy az azokhoz tartozó modellt példányosítsuk. A newbooksActionben elsőként példányosítjuk a `books` táblához tartozó modellt, majd egy `select` utasítás segítségével beállítjuk, hogy a könyvek hozzáadás dátuma szerinti fordított sorrend szerint legyenek rendezve. Az összes könyvet egy tömbbe gyűjtve átadjuk a paginator objektumnak, mely a megadott beállítások szerint fogja listázni a könyveket az oldalon. A termékeket listázó oldalon a felhasználó ezután bármely könyvborítóra vagy címre kattinthat, hiszen ezek a megfelelő könyvazonosítóval hivatkozzák a detailsActiont. A részleteket mutató oldalt végigolvasva a vásárló megbizonyosodik arról, hogy megtalálta, amit keresett. Ekkor a kosárba gombra kattintva lefut a tobasketAction. Ez az művelet már a megrendeléshez kapcsolódik, ezért az OrderControllerben található. Futás közben a `users`, `baskets` és `books` táblákat használja, ezért az ezekhez tartozó modelleket példányosítani kell, mielőtt munkához kezdenénk. Az első tevékenység, amit az akció elvégez, hogy beazonosítja a felhasználót, a Zend_Auth getInstance statikus metódusa segítségével lekérdezve a felhasználó azonosítóját. A felhasználón kívül szükséges információ, hogy melyik könyvről is van szó. A választott könyv azonosítóját az előzőleg lefutott, azaz a detailsActiontől kapja meg, tehát itt csak lekérdezzük a paramétert. Létrehoztam ezután a könyvekhez tartozó modellen egy egyszerű `select` utasítást, mely a már ismert könyvazonosító alapján

megkeresi a hozzá tartozó sort a táblából. Erre azért van szükség, mert a kosarat reprezentáló tábla nem tartalmaz az azonosítón kívül információt a könyvről, de szükséges a `books` tábla `amount` attribútumának értékét tudni. Most, hogy tudjuk a felhasználói és könyvazonosítót is, beolvasunk minden rekordot a kosárból, és egy foreach ciklusban ellenőrzünk minden könyvet. Ez azért fontos, mert ha találunk olyan rekordot, melynek mind a felhasználói azonosítója, mind a könyvazonosítója egyezik az aktuális azonosítókkal, akkor az azt jelenti, hogy a vásárlónk kosarában már megtalálható az adott könyv, így ismételt kosárba helyezés esetén annak csak a darabszámát kell megemelni. Ha a foreach ciklus egyezés nélkül futott le, és az adott könyv `books` táblában lévő `amount` attribútuma nagyobb, mint nulla, akkor új sort hozunk létre a `baskets` táblában. Beállítjuk a szükséges azonosítókat, a darabszámot egyre állítjuk, majd elmentjük a rekordot. Ha a könyv elfogyott, a program értesíti erről a felhasználót egy userMessage segítségével, majd egy return utasítással befejezi az akció a működését, és továbbadja a vezérlést, így nem futnak le a további utasítások. Ha ezen a ponton túljut az akció, akkor minden esetben kosárba helyezés történt, ezért a könyv darabszámát csökkentjük eggyel, majd mentjük a rekordot.

```
    public function tobasketAction(){
        $user_modell = new Users();
        $user = Zend_Auth::getInstance()->getIdentity();
        $userID = $user->user_id;
        $bookID = $this->_request->getParam('id');

        $books_modell = new Books();
        $select = $books_modell->select();
        $select->where("book_id = ?", $bookID);
        $book = $books_modell->fetchAll($select)->current();

        $baskets_modell = new Baskets();
        $basketbooks = $baskets_modell->fetchAll();

        foreach($basketbooks as $basketbook) :
           if($basketbook->books_book_id == $bookID &&
               $basketbook->users_user_id == $userID){
                $exists = true;
```

```
                $basketbook->basket_amount = ((int)
                  ($basketbook->basket_amount)+1);
                $basketbook->save();
                break;
            }
        endforeach;
        if(!($exists == true)){
            if($book->amount > 0){
                $order = $baskets_modell->fetchNew();
                $order->users_user_id = $userID;
                $order->books_book_id = $bookID;
                        $order->basket_amount = 1;
                        $order->save();
            }
            else {
                addons_UserMessage::userMessage('Sajnos a
                   választott könyv jelenleg nincs raktáron!');
                return;
            }
        }
       $book->amount = ((int)($book->amount)-1);
       $book->save();
       addons_UserMessage::userMessage('A könyv a kosárba került');
    }
```

Ahhoz, hogy a kosárban található cikkeket megrendelhessük, először a kosarat listázó oldalra kell navigálni, melyet a listbasketAction valósít meg. Működése egyszerű, hiszen csak a felhasználóhoz tartozó kosárban található könyvek összegyűjtése a cél. A felhasználó azonosítóját lekérjük, majd megkeressük az általa jelzett sort a `users` táblában. A kosártartalom megállapításához egy úgynevezett findManyToManyRowset metódust használtam, ami – mint a neve is jelzi – N:M kapcsolat esetén használatos. Ebben az esetben ez szükséges, hiszen a `baskets` tábla kapcsolótáblaként funkcionál a felhasználók és a könyvek táblái között. Miután összegyűjtöttem a kosárban lévő könyveket, egy foreach

ciklussal végighaladva a tömbön kiszámoltam a vásárlási végösszeget, majd átadtam a viewnak a szükséges változók értékeit.

A kosártartalom megtekintésekor még lehetőség van akár csak egy könyvet törölni, akár az egész kosarat kiüríteni, de most tegyük fel, hogy a felhasználó vásárolni szeretne, így a „vásárlás” gombra kattint. Az ezután megjelenő oldal – mely a confirmorderAction eredménye – csak a megvásárolandó könyveket tartalmazza, már a törlésre és a kiürítésre vonatkozó linkek nélkül, hogy a vásárló még egyszer ellenőrizhesse a cikkeket. Amint a vásárlás véglegesítésére kattint, a betöltődő oldalon egy legördülő listából kiválaszthatja a kívánt fizetési módot. Ezért a lapért a paymentmethodAction felel. Ebben egy form található, melynek a fizetési módokat listázó eleme egy Zend_Form_Element_Select. Ehhez egy ciklusban egyesével kell hozzáadni a `payment_methods` tábla egyes rekordjait, megadva az azonosítót és a megnevezést, hiszen ezek alapján valósul meg a legördülő lista. A másik elem egy egyszerű Zend_Form_Element_Submit, mely az adatok postolásáért felelős. Létrehoztam egy Zend_Form-ot, hozzáadtam az elemeket, és beállítottam, hogy a feldolgozást végző művelet az orderAction legyen, azaz a postolt adatokat ez kapja meg. Ez az akció végzi a tényleges megrendelést, azaz innentől a felhasználó már nem tudja azt visszavonni.

A tag azonosítása után lekérjük a postolt adatokat, ebben az esetben a fizetési mód azonosítóját. Példányosítjuk a Books, Baskets és Orders osztályokat, majd egy `select` utasítással kiválogatjuk a felhasználóhoz tartozó könyveket a kosárból. Ezeken egy ciklusban végighaladva könyvenként létrehozunk egy új sort az `orders` táblában és beállítjuk a megfelelő attribútumokat. Minden megrendelés mentése után töröljük a `baskets` táblából a megrendeléshez tartozó sort, így ürül ki a kosár.

```
    public function orderAction(){
        $user = Zend_Auth::getInstance()->getIdentity();
        $userID = $user->user_id;

        $data = $this->_request->getPost();
        $payment_method = $data[payment_methods_payment_method_id];

        $books_modell = new Books();
        $baskets_modell = new Baskets();
```

```
        $select = $baskets_modell->select();
        $select->where("users_user_id = ?", $userID);
        $books = $baskets_modell->fetchAll($select);
        $orders_modell = new Orders();

        foreach($books as $row) :
            $order = $orders_modell->fetchNew();
            $order->books_book_id = $row->books_book_id;
            $order->users_user_id = $userID;
            $order->order_date = Zend_Date::now();
            $order->payment_methods_payment_method_id =
                $payment_method;
            $order->order_amount = $row->basket_amount;
            $order->save();
            $row->delete();
        endforeach;
        addons_UserMessage::userMessage('Sikeres megrendelés!');
    }
```

9. Továbbfejlesztési lehetőségek

Az alkalmazásom fejlesztése közben folyamatosan tesztelnem kellett a már késznek gondolt részleteket, így a végső fejlesztési fázishoz egyre közeledve rengeteg átalakítást végeztem. Minden egyes funkció beépítése újabb és újabb ötleteket szült, melyekből idő hiányában nem mind valósult meg a programban, viszont tervezem további bővítések és javítások elvégzését. A teljesség igénye nélkül néhány pont, melyeken szeretnék még alakítani:

- adatbázis:
    - a táblákba további mezők felvétele, például könyvek esetén a könyv nyelve
    - kategóriák átszervezése, részletesebb kidolgozása, alkategóriák létrehozása
- eddigi menüpontok:
    - keresésnél összetett vagy részletes keresés implementálása
    - listáknál és táblázatoknál rendezések megvalósítása
    - üzenet megvalósítása minden felhasználó részére
- felület:
    - menüsor átszervezése többszintűre
    - JQuery elemek beépítése, például a menüsornál, illetve a listák és táblázatok esetében
    - layoutba dinamikus elemek beszúrása, például a gyorskeresés kiemelése
- egyéb funkciók létrehozása:
    - előrendelési lehetőség raktáron nem lévő könyvekre
    - fórum és vendégkönyv kialakítása

10. Összefoglalás

Alkalmazásom elkészítésével és szakdolgozatom megírásával úgy érzem, sikerült megvalósítani a dolgozatom elején megfogalmazott célokat, és a terveknek megfelelően megismerkedni a programozás egy új területével. Célom egy életszerű alkalmazás felépítése volt a mai rendszerek technológiai és eszmerendszerét követve, valamint a fejlesztés során használt eszközök és technológiák megismerése, az azokhoz tartozó szakirodalom tanulmányozása.

Szakdolgozatom megírását megfontolt tervezés előzte meg, majd következett a hosszabb ideig tartó fejlesztés időszaka, végül a program létrehozási folyamatának és működésének dokumentálása.

A webes alkalmazásfejlesztés terén szerzett alapvető tudásomat és tapasztalataimat reményeim szerint sikerrel fogom a későbbiekben is alkalmazni. A választásom azért esett egy webáruház létrehozására, mert a webes fejlesztés talán manapság egyik legfontosabb és leggyakrabban használt területébe nyerhettem így bepillantást. Hogy minél életszerűbb legyen mindez, egy egyre népszerűbbé váló keretrendszert, a Zend keretrendszert választottam.

Mint általában a különböző nyelvekhez létrehozott keretrendszerek, a Zend is azt a célt szolgálja, hogy megkönnyítse a programozási folyamatot. A leglényegesebb előnye, hogy az általánosságban használt eszközöket készen kapjuk, összegyűjtve, így szabadon gazdálkodhatunk mindazzal, amit a keretrendszer nyújt. Rengeteg időt és munkát spórolhatunk így meg, ráadásul az előre leimplementált komponensek egyszerűen beépíthetőek az alkalmazásba.

Az alkalmazásom felépítése követi a ma forgalomban lévő webes áruházak általános struktúráját, ezáltal tetszőlegesen bővíthető. Megírásakor törekedtem arra, hogy amennyire ez lehetséges, minél általánosabbak legyenek az elkészített kódrészletek. Ebben a keretrendszer és az MVC architektúra is segítségemre volt, így a létrehozott alkalmazás megfelelő kiindulópont lehet a későbbiek folyamán tetszőleges webes rendszer fejlesztéséhez.

Irodalomjegyzék

- Rob Allen – Nick Lo – Steven Brown: Zend Framework in action, 2009, Manning Publications Co.
- W. Jason Gilmore: Easy PHP websites with the Zend Framework, 2009
- Keith Pope: Zend Framework 1.8 Web Application Development, 2009, Packt Publishing Ltd.

Internetes hivatkozások:

- Zend keretrendszerrel kapcsolatos információk
  http://framework.zend.com/apidoc/core/
  http://framework.zend.com/manual/en/manual.html

- PHP
  http://php.net/

- JQuery
  http://jquery.com/
  http://plugins.jquery.com/

- HTML és CSS nyelv
  http://w3schools.com/css/default.asp
  http://w3schools.com/html/default.asp

Köszönetnyilvánítás

Ezúton szeretnék mindenkinek köszönetet mondani, aki hozzájárult szakdolgozatom elkészítéséhez, különös tekintettel témavezetőmre, Kósa Márkra, aki ötleteivel és szaktudásával támogatta a munkámat.