Debreceni Egyetem
Informatika Kar

Portálkészítés japán szótár és tananyagok számára PHP nyelven

Témavezető:
Dr. Horváth Géza
egyetemi adjunktus

Készítette:
Pintér Krisztián
programtervező informatikus

Debrecen, 2010

Tartalomjegyzék

1 Bevezetés

Az internet egy olyan globális rendszer, mely számítógép hálózatok millióit kapcsolja össze a TCP/IP protokoll segítségével. Az internet tehát hálózatok hálózata, amelyen rengeteg információs rendszer működik. Ezek közül a legfontosabb a web, amely a kliens-szerver modellen alapszik.

Az internet története az 1960-as évekig nyúlik vissza, amikor is az USA egy hidegháborús hadi fejlesztés keretében létrehozott egy csomagkapcsolt hálózatot, mely az ARPANET (Advanced Research Projects Agency) névre hallgatott. A fejlesztés célja egy olyan információs rendszer kiépítése volt, amelynek nincs központja, minden csomópontja több más csomóponttal áll kapcsolatban Így egy esetleges atomtámadás nem bénítaná meg a teljes hálózatot, a megmaradt részei továbbra is működőképesek maradnának. A hálózat 1969-ben valósult meg és kezdetben négy csomópontja volt. 1971- ben további 15 oktatási és kutatási intézmény csatlakozott, majd később további hálózatok is kapcsolódtak, 1974-re pedig megszületett a ma is használatos TCP/IP protokoll. 1990-ben az ARPANET megszűnt, de az internet tovább fejlődött, és a 90-es évek elejére kialakult a World Wide Web

Mára az internet az egyre inkább terjedő globalizáció egyik legfontosabb eszközévé vált. Az internetnek hála nem léteznek többé földrajzi távolságok a kommunikációban. A világ egyik végében lévő ember kapcsolatba léphet a másikkal pár pillanat alatt.

A mai modern nyelvoktatás már nem csak a szemtől szembeni, közvetlen kommunikáció adta lehetőségeket használja ki. Terjedőben vannak az olyan online felületek is, amik a nyelvet tanulni vágyók rendelkezésére állnak a nyelvtanulás segítésére. Leginkább az angol nyelv elsajátítására kínálnak ilyen online tanfolyamokat, segítséget. Egy böngészőbe beírva az „online angol nyelvtanulás” –t, több mint 240ezer ilyen linket kaphatunk.

Az olyan nyelv esetében, mint a japán, még nagyobb szükség lenne az ilyen online tanfolyamokra, mint az angol esetében. A japán nyelvet mivel jelenleg kevesen ismerik, ezért nagyon kevés az olyan ember, aki az oktatásával foglalkozna, ezért nehezen is elérhetőek,

főleg vidéken. Éppen ezért az online oktatás nagyon kézenfekvő, és kényelmes megoldás lenne a tanulni vágyóknak, és az oktatóknak is. Az online oktatás része kell hogy legyen egy jó online szótár is, ami mára sokkal könnyebben kezelhetővé vált mint egy hagyományos, papír alapú szótár.

Ezen szakdolgozat célja, egy olyan web portál tervezése, és kivitelezése, amelyen közzétehetek japán nyelvi tananyagokat és egy online japán szótárt. Célkitűzéseim között szerepel a portál olyan kialakítása, amely lehetővé teszi hogy a későbbiek folyamán könnyen bővíthető legyen újabb funkciókkal, felhasználói csoportokkal, újfajta tananyagtípusokkal.

A megvalósításában a következő eszközöket használtam fel:

- PHP, JavaScript, CSS programozási nyelvek
- MySQL relációs adatbázis-kezelő rendszer
- Zend Framework PHP 5 alapú keretrendszer
- Netbeans integrált fejlesztői környezet

Néhány alapfogalom tisztázása és a felhasznált technológiák bemutatása után megpróbálom bemutatni a tervezés és a fejlesztés menetét a jelenlegi követelmények feltárásától kezdve a portál elkészültéig.

2 Alapfogalmak

A weben közzétett dokumentumok a kiszolgáló oldaláról tekintve statikus vagy dinamikus, a kliens oldaláról tekintve végleges és változó megjelenésű dokumentumok lehetnek.

Statikus dokumentumok: Már az ügyfélkérések fellépte előtt létre kell hozni és tárolni kell a kiszolgálók gépein. Ezek a dokumentumok a kiszolgálótól gyakorlatilag változatlan formában jutnak el az ügyfelekhez. Valamilyen szabványos dokumentációs nyelv (HTML vagy XML) és kapcsolódó nyelvek (CSS, XSLT, stb.) szabályainak eleget téve kell őket létrehozni.

Dinamikus dokumentumok: A dinamikus dokumentumok esetén az ügyfélkérelmek felléptekor még nem állnak rendelkezésre a kért információt reprezentáló dokumentumok. Az igényelt információ, illetve az azt reprezentáló dokumentumok helyett dokumentumokat generáló programok állnak rendelkezésre. Ezek különböző erőforrásokhoz (fájlok, adatbázisok, más webkiszolgálók, stb.) hozzáférve, azokból a szükséges információt kinyerve, olyan ún. dinamikus dokumentumokat állítanak elő, amelyek már az ügyfél által igényelt információt reprezentálják.

Végleges megjelenésű dokumentumok: Az ügyfelekhez letöltött dokumentumok egy része, akár statikus akár dinamikus eredetű, a letöltés után teljes élettartama során gyakorlatilag változatlan formában jelenik meg az ügyfél böngészőjében. Legfeljebb a dokumentum űrlapjain megjelenített bemeneti mezők tartalma változhat a felhasználóval való interaktivitás során.

Változó megjelenésű dokumentumok: Az ügyfelekhez letöltött dokumentumok jelentős része, akár statikus akár dinamikus eredetű, olyan a dokumentumba beékelt programokat (vagy objektumokkal reprezentált programokat) is tartalmaz, melyeket az ügyfélalkalmazás hajt végre. Ezeknek a programoknak az elsődleges feladata a dokumentumok megjelenési formájának és/vagy tartalmának megváltoztatása a felhasználóval való interaktivitás során.

Statikus weboldalak: A statikus weboldalakat az jellemzi, hogy használatuk előtt bármilyen szövegszerkesztővel vagy valamilyen dokumentumgenerátorral előre elkészítik. Az így előre

elkészített weboldalt a webkiszolgáló gépén a dokumentumok számára kijelölt könyvtárban tárolva, annak statikus, nem változó tartalma letölthető, és az ügyfél böngészőjében megjeleníthető.

Dinamikus weboldalak: A dinamikus weboldalak letöltésekor az ügyfél böngészője nem egy statikus HTML vagy XML fájl lehívását, hanem egy program végrehajtását kezdeményezi a http kapcsolaton keresztül. A kért programot a kiszolgáló elindítja, s ez a program hozzáférve a különböző erőforrásokhoz (adatbázisok, fájlok, stb.) létrehoz egy HTML vagy XML formájú weboldalt, melyet a kiszolgáló elküld a böngészőbe.

Mivel az így előállított weboldal tartalma a hozzáfért erőforrások aktuális tartalmától, s az ügyfél felől érkező adatoktól is függhet, a különböző időpontokban és/vagy különböző ügyfelek általi kérelmekre végrehajtott program különböző weboldalakat generálhat. Ezért ezeket a weboldalakat dinamikus weboldalaknak nevezzük.

A weboldal-generáló programok alapvetően kétfélék lehetnek:

- A kiszolgálótól független, s azzal csak egy jól definiált interfészen keresztül kommunikáló programok. Ezek a programok bármilyen nyelven megírhatók, mint önálló processzek hajtódnak végre, s végrehajtásuk alatt a szabványos I/O csatornán keresztül kommunikálnak a kiszolgálóval. (pl. CGI-programok)

- A kiszolgáló valamilyen modulja által végrehajtott vagy értelmezett programok. Ezek nem önálló processzként, hanem csak mint a kiszolgáló program önálló szálai hajtódnak végre vagy kerülnek értelmezésre (interpreter programok). Önálló szálként végrehajtott tipikus képviselőik a Java szervletek, értelmezett végrehajtású tipikus képviselőik pedig az ASP, PHP, stb. technológiájú weboldalak.

A kiszolgáló oldali programoknak egy másik feladatuk is lehet: ezek dolgozhatják fel és tárolhatják az ügyfelektől érkező információt.

Változó megjelenésű weboldalak: Az ügyfelek böngészőjébe letöltött weboldalak a statikus tartalom közé beékelt szkripteket és speciális objektumokat is tartalmazhatnak.

[1]

3 Az alkalmazott eszközök és technológiák

3.1 CSS (kaszkád stíluslapok)

A CSS egy deklaratív nyelv, amellyel a HTML elemek megjelenítési módját írhatjuk elő. A HTML nyelv nem a dokumentum formázására, hanem a tartalom definiálására lett tervezve. Amikor a <font> és a hozzá hasonló jelölőelemek és a „color” attribútum részei lettek a HTML-nek, a nagy weboldalak fejlesztése -ahol minden egyes lapon meg kellett adni a betűtípusra és a színekre vonatkozó információkat- rémálommá vált. Ennek a problémának a megoldására alkották meg a CSS-t, amely lehetővé teszi, hogy minden formázást elkülönítsünk a HTML dokumentumtól, és egy külön állományban tároljuk.

Egy stílusszabály két fő részből áll: egy szelektorból és egy vagy több deklarációból. Minden deklaráció egy tulajdonságot és egy hozzátartózó értéket tartalmaz:

```
Szelektor {tulajdonság: érték[; tulajdonság: érték]...}
```

A szelektor általában egy HTML jelölőelem, amelynek tulajdonságát vagy tulajdonságait specifikálni akarjuk. Egy szabályban több elemre vonatkozó előírást is megadhatunk, ilyenkor a szelektorokat vesszővel kell elválasztanunk. Lehetőség van továbbá saját, id vagy class szelektorok specifikálására. Az id szelektor egy egyedi HTML elemre vonatkozik, a class szelektorral elemek egy csoportját specifikáljuk.

Példa az id szelektorra:

```
#para1
{
text-align:center;
color:red;
}
```

Példa a class szelektorra:

```
p.center {text-align:center;}
```

Egy HTML dokumentumban egy elemet csak egyetlen osztályba sorolhatunk, és egy elem id

attribútumának egyedinek kell lennie.

A stíluslap beillesztését háromféleképpen tehetjük meg:

- külső állományban (External style sheet)
- dokumentum fejrészében (Internal style sheet)
- dokumentum törzsében (Inline style)

Több stíluslapra is hivatkozhatunk egy dokumentumon belül. Ha különböző stílusinformációk egy elem vonatkozásában ellentmondanak egymásnak, akkor a következő felsorolásban nagyobb sorszámú előírás hajtódik végre:

1. Böngésző alapértelmezés
2. Külső állományban előírt stílus
3. A dokumentum fejrészében előírt stílus
4. Beágyazott stílus a dokumentum törzsében

Azonban ha a külső stíluslapra mutató hivatkozás a HTML dokumentum fejrészében előírt stílus után szerepel, akkor hatástalanítja azt.

[1][8]

3.2 JavaScript

A JavaScript egy kliensoldali, objektumalapú szkriptnyelv, melyet a weblapok interaktívabbá tételére terveztek. A nyelv első változatát a Netscape jelentette meg 1995-ben LiveScript néven. Mivel a nyelv szintaxisa nagyon hasonlított a Java szintaxisához, ezért piaci szempontok miatt a nevét Javascriptre változtatták.

A JavaScript egy interpreteres nyelv. Az interpreter általában egy alkalmazás része, esetünkben a böngészőé.

[1]

3.3 MySQL

A MySQL az egyik legnépszerűbb, többfelhasználós, többszálú relációs adatbáziskezelő-rendszer. A szoftver nyílt forráskódú, a GPL licensz alatt ingyenesen használható.

A MySQL-nek minden fontosabb programnyelvhez van API-ja (Application Programming Interface), így a PHP-hez is. Nagyon sok webalkalmazás mögött ez az adatbáziskezelő-rendszer áll. (lsd. WAMP illetve LAMP szerver összeállítások.) A kiszolgálói eljárás mellett eszközöket biztosít az adatbázisok eléréséhez, SQL parancsok végrehajtásához, felhasználók kezeléséhez, adatbázisok létrehozásához és törléséhez. Ez utóbbi felsorolás a teljesség igénye nélkül készült.[4][9]

3.4 PHP

PHP: **H**ypertext **P**reprocessor

A PHP egy szerveroldali -az -5-ös verziótól kezdve objektumorientált- szkriptnyelv, tehát az ezen készült szkriptek a szerveren hajtódnak végre. A szerveren az interpreter feldolgozza az utasításokat, és ennek eredményét továbbítja a kliensnek. Konfigurálható szerver modulként vagy CGI szkriptként. Többnyire webfejlesztés során alkalmazzák HTML dokumentumokba ágyazva, de önálló programozási nyelvként is megállja a helyét. Szintaktikája sokat kölcsönöz a C és a Perl nyelviből. A PHP egy nyílt forráskódú szoftver.

3.4.1 Szkriptek beillesztése

A PHP állományok egyszerű szövegfájlok, bármilyen szövegszerkesztővel létrehozhatók. A kiterjesztésük általában .php, de a szerver beállításaitól függően .phtml, .php5 kiterjesztés is lehetséges. Az állományon belül jelölnünk kell a szkriptek elejét és végét. Ezt a következő jelölőelemekkel tehetjük meg:

Nyitóelem	Záróelem
`<?php`	`?>`
`<SCRIPT LANGUAGE="php"`	`</SCRIPT>`
`<?`	`?>`
`<%`	`%>`

Utóbbi kettőt engedélyezni kell a PHP beállításaiban. A jelölőelemeken kívül eső szöveget az értelmező nem dolgozza fel, hanem változatlan formában továbbítja azt.

3.4.2 Nyelvi elemek

3.4.2.1 Változók

A változók a program szövegében egy $-el és azt követően a változó nevével jelennek meg. A név egy betűvel vagy aláhúzással kezdődő, és betűvel, számjeggyel vagy aláhúzásjellel folytatódó karaktersorozat. Változó deklarálása:

```
$var_name = value;
```

A PHP gyengén típusos nyelv. A változókat nem szükséges deklarálni, automatikusan konvertálódnak a megfelelő típusúra az értéküktől függően.

3.4.2.2 Típusok

Skalár típusok:

boolean	logikai
integer	egész
float (v. double)	valós
string	sztring

Összetett típusok:

array	tömb
object	objektum

A PHP tömbjei rendezett leképezések, értékeket rendelnek kulcsokhoz. Az érték bármilyen típusú lehet, de a kulcs csak sztring vagy integer lehet. Tömb típusú változó az array() függvénnyel hozható létre, amely kulcs => érték párok vesszővel elválasztott listáját várja

paraméterként. Tömböket szögletes zárójel segítségével is létrehozhatunk:

```
$tömb_név[] = érték;
```

Ilyenkor a $tömb_név nevű változó tömb típusúként jön létre, vagy ha a változó már létezett korábban akkor az utasítás a meglévő tömböt bővíti. A tömb indexelése automatikusan 0-tól kezdődik, ha csak explicit módon nem adunk meg tömbindexet.

Objektumot létrehozni példányosítással lehet, melynek operátora a `new`.:

```
$objektum_név = new osztálynév;
```

Speciális típusok:

resource	Külső erőforrásra hivatkozó referencia
NULL	Érték nélküli változó típusa

3.4.2.3 Kifejezések

A kifejezések operátorokból és operandusokból állnak. A PHP mindent kifejezésnek tekint ami rendelkezik érték komponenssel. A legegyszerűbb kifejezés egyetlen operandusból áll, amely konstans vagy változó lehet.

3.4.2.4 Operátorok

Értékadás:

Az értékadás az '=' operátorral, operátorral történik, illetve az értékadó operátor kombinálható az aritmetikai operátorokkal, a sztringeknél használt '.' konkatenáció, és a tömbök '+' unió operátorával.

Aritmetikai operátorok

−	negálás
+	összeadás
−	kivonás
*	szorzás

/	osztás
%	maradékképzés

Bitműveletek:

&	bitenkénti és
|	bitenkénti vagy
^	bitenkénti kizáró vagy
~	bitenkénti tagadás
<< és >>	bitléptető operátorok

Összehasonlító operátorok:

==	értékegyenlőség
===	típus- és értékegyenlőség
!==	igaz, ha a === hamisat ad vissza
!=, <>	nem egyenlő
<, >	kisebb, nagyobb
<=, >=	Kisebb-egyenlő, nagyobb-egyenlő

Logikai operátorok:

and, &&	logikai és
or, ||	logikai vagy
xor	logikai kizáró vagy
!	logikai tagadás

Sztring operátorok:

Sztringekkel a konkatenáció művelet végezhető, melynek operátora a '.'.

Tömb operátorok:

+	unió
==	egyenlőségvizsgálat kulcs-érték párokra
===	egyenlőségvizsgálat kulcs-érték párokra, rendezettségre és típusra
!==	igaz, ha a === hamisat ad vissza
!=, <>	igaz, ha a == hamisat ad vissza

Típus operátorok:

```
objektum_név instanceof osztály_név
```

Az `instanceof` operátor igaz logikai értéket ad vissza, ha az `objektum_név` objektum az `osztály_név`, vagy egy leszármazott osztályának a példánya.

3.4.2.5 Konstansok

A konstansok értéke skalár típusú lehet. Létrehozásuk a define utasítással történik.:

```
define( név, érték );
```

3.4.2.6 Megjegyzések

A megjegyzések hasonlóak a C, C++ illetve a Unix shell megjegyzéseihez:

```
// egysoros c++ stílusú megjegyzés
# egysoros Unix stílusú megjegyzés
```

Az egysoros megjegyzések a sor végéig, vagy a ?> illetve %> kódblokk zárójelig tartanak, ha azok a sor vége előtt szerepelnek.

```
/*
     több sorból
     álló megjegyzés
*/
```

A többsoros megjegyzés az első */-ig tart.

3.4.2.7 Vezérlési szerkezetek

Elágazások:

Kétirányú elágaztató utasítás:

```
if( kifejezés ) utasítás
[elseif( feltétel ) utasítás]...
[else utasítás]
```

Amennyiben a kifejezés nem logikai típusú, automatikusan logikai típusúra konvertálódik. Ha az értéke igaz, végrehajtódik a mögötte álló utasítás. Ha az érték hamis, és szerepel else-ág, az ott szereplő utasítás hajtódik végre.

Többirányú elágaztató utasítás:

```
switch( kifejezés )
{
case kifejezés:
     utasítások
     break;
[case kifejezés:
     utasítások
     break;]...
[default:
     utasítások]
}
```

Kiértékelődik a kifejezés, majd az értéke sorra összehasonlításra kerül a case ágakban szereplő kifejezések értékeivel. Ha az érték egyezik, az adott ágban szereplő utasítások végrehajtódnak. A `break` utasítást kell használnunk a kilépéshez, különben a következő case ágon folytatódik a vizsgálat. Amennyiben egyik ágban sincs egyezés (vagy nem léptünk ki a `break` utasítással), de van default-ág, végrehajtódnak az ott szereplő utasítások.

Ciklusok:

Kezdőfeltételes ciklus:

while

```
while(kifejezés)
     utasítás
```

vagy

```
while(utasítás):
     utasítások
     ...
endwhile;
```

Végfeltételes ciklus:

```
do{
     utasítások
}while(kifejezés)
```

Előírt lépésszámú ciklus:

```
for (kifejezés1; kifejezés2; kifejezés3)
     utasítás
```

Ez utóbbi három ciklus szemantikailag ugyanúgy viselkedik mint a C nyelv megfelelő ciklusai.

foreach ciklus:

```
foreach(tömb_kifejezés as $érték)
     utasítás
```

vagy

```
foreach( tömb_kifejezés as $kulcs => $érték)
     utasítás
```

Ezt a ciklust arra használjuk hogy asszociatív tömb elemeit járjuk be vele egyenként. Az iteráció során az aktuális tömbelem értéke az $érték változóba kerül, emellett a második alak használata esetén az aktuális kulcs értéke a $kulcs változóba kerül. A ciklus nem magán a tömbön, hanem a tömb egy másolatán dolgozik.

3.4.2.8 Függvények

A PHP-ban nagyon sok beépített függvény van. Saját függvényt létrehozni a következőképpen lehet:

```
function függvénynév( formális_paraméterlista ){
     törzs
}
```

A függvény visszatérési értékét a `return` utasítással lehet meghatározni, mely egyben be is fejezteti a függvényt. Amennyiben a `return` elmarad, a visszatérési érték NULL lesz.

A paraméterátadás érték szerinti, de az aktuális paraméter előtt megadott & jellel cím szerinti paraméterátadás is megvalósítható. A paramétereknek megadhatunk alapértelmezett kezdőértéket. A PHP a 4-es verziótól kezdve képes változó hosszúságú paraméterlistát kezelni.

Arra is van lehetőségünk hogy névtelenül hozzunk létre függvényt, vagy hogy változónak függvényt adjunk értéknek.

[2][7][8]

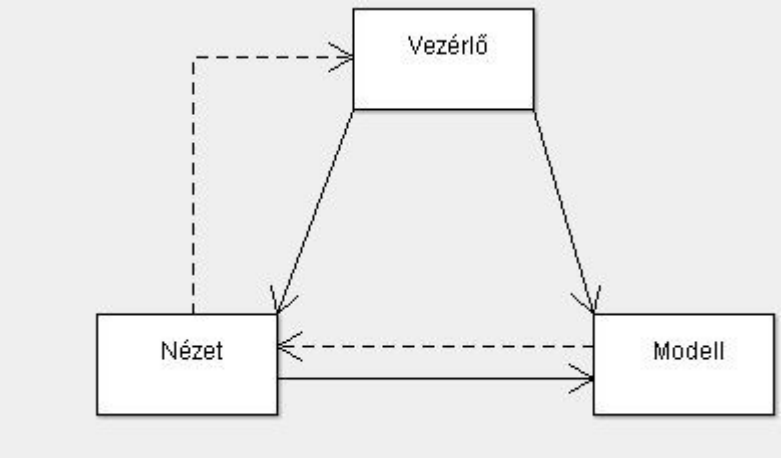
3.5 Zend Framework

A Zend Framework egy PHP 5-re épülő, nyílt forráskódú, objektumorientált keretrendszer. A keretrendszer sok különálló komponensből áll, melyek a 3.5.1 fejezetben bemutatott kategóriákba sorolhatók. Minden komponens osztályokból áll.

3.5.1 A Zend Framework komponensei

3.5.1.1 Modell-nézet-vezérlő (Model-View-Controller - MVC)

A webes alkalmazások forrásszöveg részei funkciójuk szerint általában a következő kategóriák valamelyikébe sorolhatók: megjelenítés, üzleti logika, adatelérés. Az MVC tervezési minta ezeket élesen elkülöníti egymástól.



- **Modell:** Idetartoznak az alkalmazás azon részei, melyek az alapvető funkcionalitását határozzák meg, tehát az üzleti logikát és az adatok elérését és kezelését megvalósító eljárások.

- **Nézet:** Ez a rész határozza meg hogy a modell hogyan jelenjen a felhasználónak, tipikusan a felhasználói felületnek egy eleme. Gyakran adatokat is fogadnak a felhasználótól. Egyazon modellhez több különböző nézet is tartozhat.

- **Vezérlő:** Összeköti az előbbi kettőt. Kezeli a modelleket és meghatározza hogy mikor melyik nézet jelenjen meg. Feldolgozza a felhasználói inputokat és minden nézetnek továbbítja a szükséges adatokat.

A Zend Framework az MVC-t a Front Controller mintával együtt valósítja meg, amely az alkalmazás belépési pontjainak centralizációjára nyújt megoldást. A Front Controller megvalósításának fő komponensei a router és a dispatcher. Az előbbi meghatározza hogy melyik műveletnek (action) kell végrehajtódnia, az utóbbi pedig futtatja azt, és továbbiakat amennyiben szükséges. Amikor egy kérés érkezik a böngészőtől, a router és a dispatcher az URL alapján meghatározza hogy mely vezérlőnek kell lefutnia. Ezután a vezérlő a modell és a nézet komponensekkel együttműködve előállítja a böngészőnek visszaküldött választ. [11]

3.5.1.2 Parancssoros eszköz és gyors alkalmazásfejlesztés (Rapid Application Development – RAD)

A keretrendszerben írt projektek egy bizonyos szintig azonos koncepció szerint készülnek. Tartalmaznak műveleteket, vezérlőket, modelleket, adatbázist, a megjelenítéshez kapcsolódó szkripteket stb., és ezeknek van egy hierarchiájuk az alkalmazás könyvtárszerkezetében. A komponens ezek automatizált létrehozására kínál eszközöket. Pl a `create project jportal` parancs hatására létrejön az alkalmazás alapvető könyvtárstruktúrája néhány automatikusan előállított szkripttel.

3.5.1.3 Adatbázis

Ez a komponens az adatbázisokhoz biztosít egy interfészt. Az egyik alapvető osztálya a `Zend_Db_Adapter`, amely segítségével az alkalmazást összekapcsolhatjuk egy relációs adatbáziskezelő-rendszerrel. Másik alapvető osztálya a `Zend_Db_Table`, amely megvalósítja a Table Data Gateway tervezési mintát. Ez egy objektum orientált interfészt nyújt az adatbázistáblák kezeléséhez. A mintát követve csak ezen osztály leszármazottai fognak SQL kódot tartalmazni az alkalmazásban. A Zend Framework alkalmazások többségében ezek az osztályok képviselik az MVC-ben a modellt.

3.5.1.4 Nemzetköziesítés (i18n) és lokalizáció (l10n)

A nemzetköziesítés a tervezésnek az a folyamata, melynek során az alkalmazást több nyelvre és kulturális sajátosságra (pénznem, dátumformátum, különböző mértékegységek stb.) átültethetővé tesszük. Ez utóbbit hívjuk lokalizációnak.

3.5.1.5 Autentikáció, autorizáció és munkamenet kezelés

Az autentikáció a felhasználó azonosításának folyamata. Ez általában felhasználónévvel és jelszóval történik. Az autorizáció során eldöntjük, hogy egy adott felhasználó hozzáférhet-e egy adott erőforráshoz. Előbbire a Zend_Auth, utóbbira a Zend_Acl osztály szolgál, amely egy ACL (access control list) alapú jogosultság rendszert valósít meg. A munkamenetek kezelésében a Zend_Session osztály nyújt segítséget.

3.5.1.6 Webszolgáltatások

Különböző webszolgáltatások nyújtásához és eléréséhez, illetve más online alkalmazások API-jához kínál eszközöket, mint például a Google alkalmazásai.

3.5.1.7 Mail, Formats, Search

A mai webalkalmazások többnyire rendelkeznek keresővel, támogatják az e-mailben történő kommunikációt, az Ajax technikával megvalósított kommunikációt, stb. A komponens ezek fejlesztéséhez nyújt beépített eszközöket.

3.5.1.8 Core Infrastructure

Ide tartoznak azok az osztályok melyek nem fértek bele egyik más kategóriába sem. Itt kaptak helyet az alkalmazás konfigurálását, hibajavítását, naplózását, az osztályok betöltésének automatizálását segítő, és még sok más osztályok.

[5][10]

4 Tervezés

A tervezés során feltárom a követelményeket, majd -mivel az alkalmazás architektúrája jórészt adott az MVC által- rögtön rátérek az adatmodell előállítására.

4.1 Általános követelmények

1. Moduláris felépítés: új funkciók beépítése megoldható legyen a rendszer minimális változtatásával.

2. A rendszer használata legyen könnyen megtanulható. A felhasználói felület legyen könnyen átlátható.

3. A felhasználók és a jogosultságok kezelése adminisztrátor segítségével történjen.

4. A megfelelő jogosultságokkal legyen lehetőség tananyagok hozzáadására, módosítására, törlésére.

5. A felhasználók bejelentkezése felhasználónévvel és jelszóval történjen. A felhasználóknak lehetőséget kell biztosítani jelszavuk módosítására, illetve elfelejtett jelszó esetén új jelszó igénylésére.

4.2 Rendszerkövetelmények

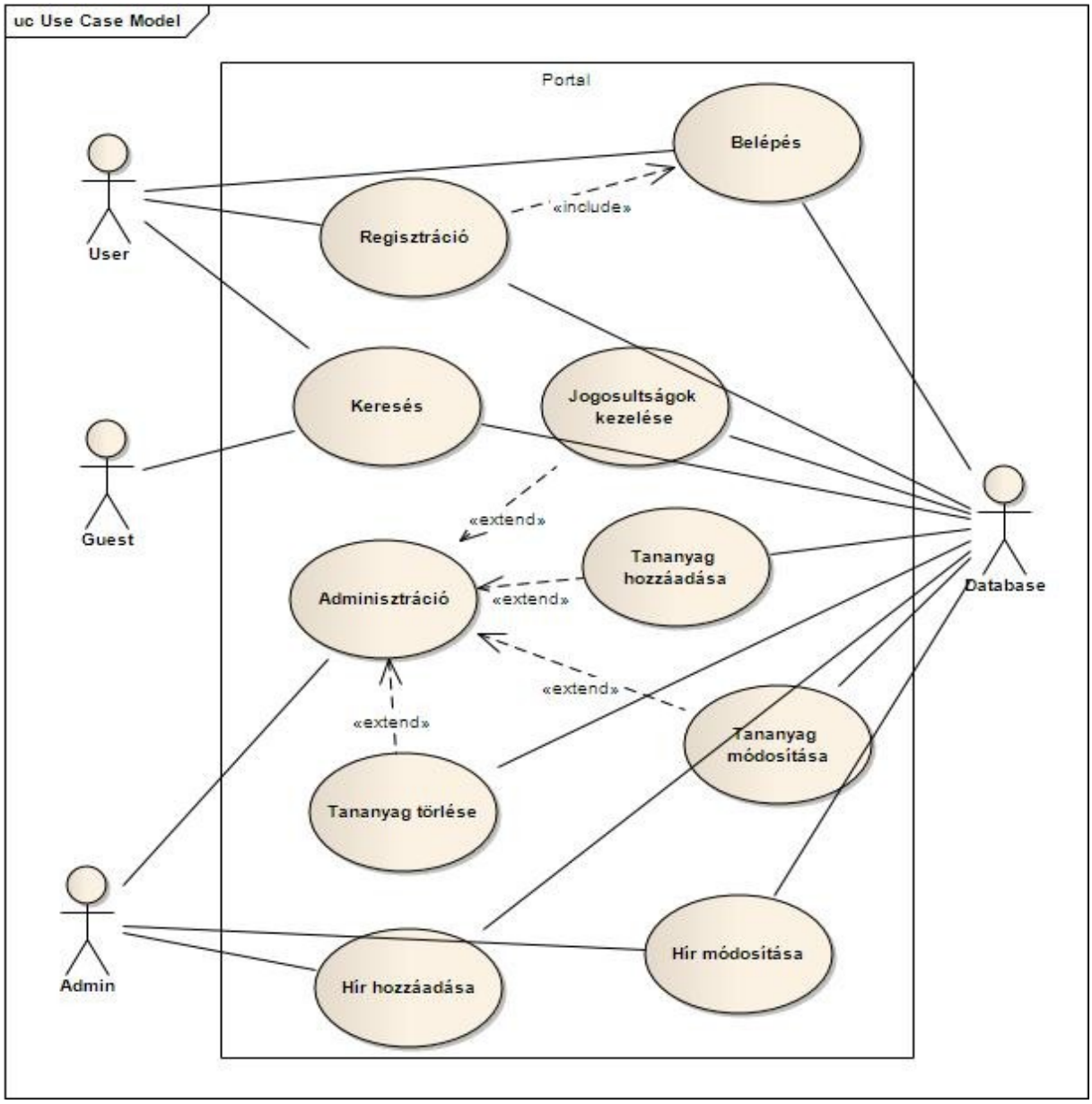
Operációs rendszer: A rendszernek platformfüggetlennek kell lennie.

Programozási nyelv: PHP, Javascript, CSS

4.3 Fogalomszótár

Guest:	Olyan látogató, aki nincs regisztrálva a rendszerben.
User:	Az alkalmazás rendszerében regisztrált személy. Van azonosítója, felhasználóneve, jelszava és e-mail címe.
Admin:	Az a személy aki a jogosultságok kezelését végzi.
Adatbázis:	Az alkalmazás mögött rejlő, az alkalmazás és a felhasználók adatait tároló adatközpont.
Tananyag:	A portálon közzétett dokumentumok. Van azonosítója és címe.
Hír:	Az adminisztrátor állítja elő. A portállal kapcsolatos eseményekről közöl információkat, ilyen például egy új tananyag felkerülése.

4.4 Használati esetek



4.5 Forgatókönyvek

A forgatókönyvek a használati esetek normális lefolyásának menetét írják le.

A felhasználó bejelentkezése a rendszerbe:

- A felhasználó megtekinti a bejelentkezési felületet
- A felhasználó megadja a felhasználónevet és a jelszót

- A rendszer ellenőrzi a felhasználónév és a jelszó érvényességét

  → ha a rendszer elfogadja a felhasználónevet és jelszót, engedélyezi a bejelentkezést

  → ha a rendszer nem fogadja el a felhasználónevet vagy a jelszót, nem engedélyezi a bejelentkezést

A felhasználó regisztrációja a rendszerbe:

- A felhasználó megtekinti a regisztrációs felületet
- A felhasználó kitölti a regisztrációs űrlapot
- A felhasználó megadja a felhasználónevet és a jelszót

  → Ha a felhasználónév még nem foglalt, és a jelszó megfelel az adott kritériumoknak, a rendszer engedélyezi a regisztrációt, és a felhasználó adatait, felhasználónevét és jelszavát elmenti az adatbázisba.

  → Ha a felhasználónév foglalt, vagy a jelszó nem felel meg a kritériumoknak, a rendszer új felhasználónevet és/vagy jelszót kér a felhasználótól

Jogosultságok kezelése

- Az adminisztrátor megnyitja az adminisztrációs felületet
- Az adminisztrátor lekéri a felhasználók és/vagy szerepkörök listáját
- Az adminisztrátor kiválasztja a listából azokat az elemeket amelyeket módosítani kíván és megadja az új jogosultságokat
- A módosítások rögzítésre kerülnek az adatbázisban.

Tananyag hozzáadása

- Az arra jogosult felhasználó megtekinti a tananyagok kezelőfelületét
- A felhasználó kiválasztja a tananyag feltöltése opciót
- A felhasználó kitölti a tananyaghoz tartozó űrlapot
- A felhasználó feltölti a tananyagot
- A módosítások rögzítésre kerülnek az adatbázisban

Tananyag módosítása

- Az arra jogosult felhasználó megtekinti a tananyagok kezelőfelületét
- A felhasználó kiválasztja a módosítani kívánt tananyagot
- A felhasználó elvégzi a szükséges módosításokat a tananyaghoz tartozó űrlapon.
- A módosítások rögzítésre kerülnek az adatbázisban, és a rendszer megmutatja a módosítások eredményét a felhasználónak.

Tananyag törlése

- Az arra jogosult felhasználó megtekinti a tananyagok kezelőfelületét
- A felhasználó kiválasztja a törlendő tananyagot
- A rendszer megerősítést kér a felhasználótól a törlésre vonatkozóan
  - → Ha a felhasználó megerősítési a törlést, végrehajtásra kerül
  - → Ellenkező esetben a törlés nem kerül végrehajtásra

Hír hozzáadása

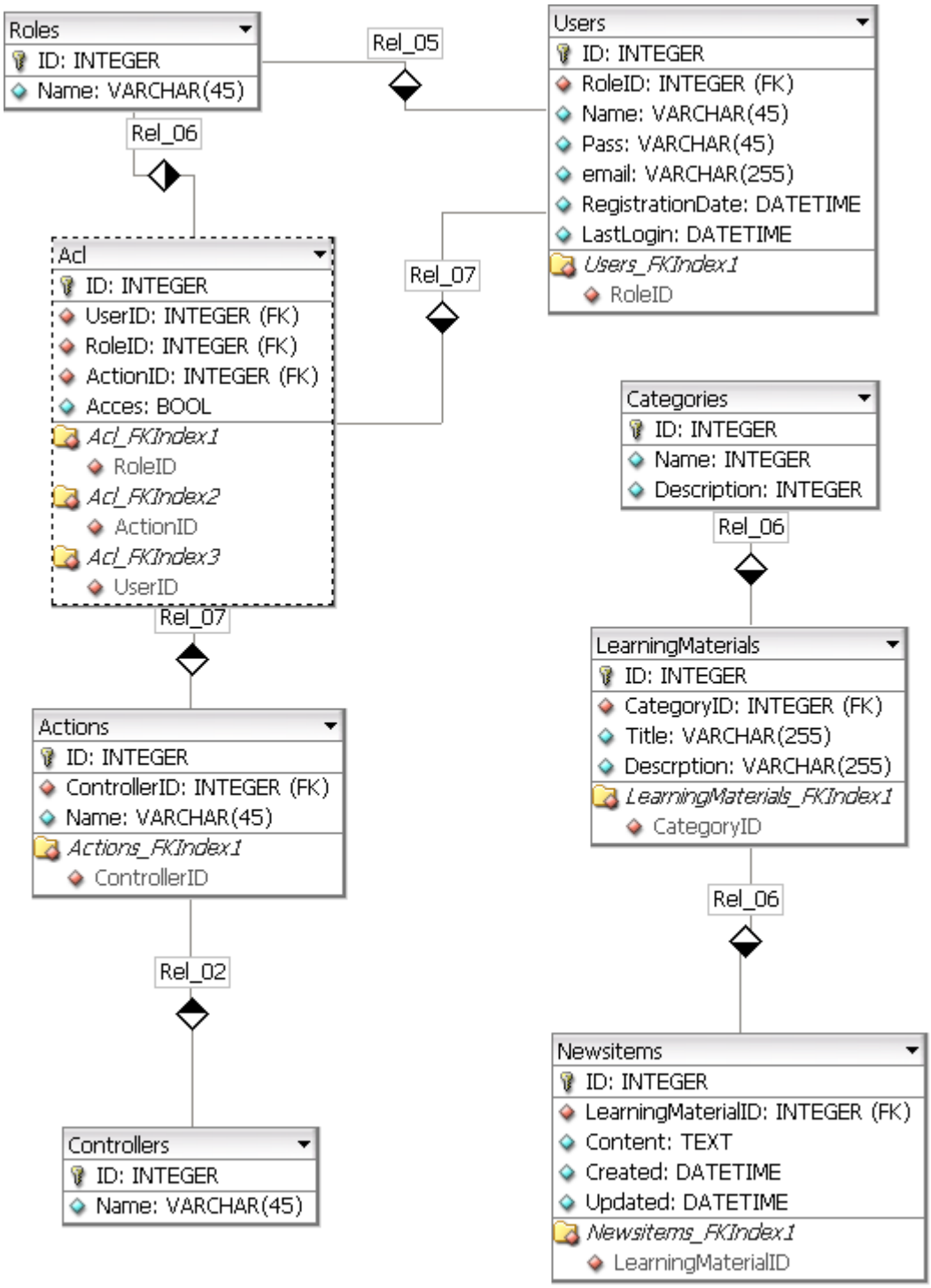
- Az adminisztrátor kiválasztja a hírek kezelőfelületét
- Az adminisztrátor kiválasztja az új hír hozzáadását.
- Az adminisztrátor megadja a hír szövegét és az esetlegesen hozzátartozó tananyagot
- A módosítások rögzítésre kerülnek az adatbázisban

Hír módosítása

- Az adminisztrátor kiválasztja a hírek kezelőfelületét
- Az adminisztrátor kiválasztja a módosítandó hírt és lekéri a hír űrlapját.
- Az adminisztrátor módosítja a hír adatait az űrlapon.
- A módosítások rögzítésre kerülnek az adatbázisban

4.6 Adatbázis tervezés

Az adatbázis tervezése során igyekeztem minél egyszerűbb adatbázist létrehozni, és ezzel egy időben minimálisra csökkenteni a redundanciát és az ebből fakadó anomáliákat. A táblák és a közöttük lévő kapcsolatok:



A táblák szerkezete:

Users tábla:

Mezőnév	Típus	Leírás
ID	INTEGER	Elsődleges kulcs
RoleID	INTEGER	A Roles táblára hivatkozó idegen kulcs
Name	VARCHAR(45)	Felhasználónév
Pass	VARCHAR(45)	Jelszó
email	VARCHAR(255)	A felhasználó e-mail címe
RegistrationDate	DATETIME	A felhasználó regisztrációjának dátuma
LastLogin	DATETIME	A felhasználó legutóbbi bejelentkezésének ideje

A felhasználók adatait és az azonosításukhoz szüksége felhasználónevet és jelszót tartalmazza. Minden sorához tartozik egy rekord, amely a felhasználó szerepkörét határozza meg. Ennek a jogosultságkezeléskor lesz majd szerepe.

Roles tábla:

Mezőnév	Típus	Leírás
ID	INTEGER	Elsődleges kulcs
Name	VARCHAR(45)	A szerepkör/felhasználói csoport neve

A fentebb említett szerepköröket tartalmazza. Egyelőre csak három féle szerepkör megvalósítását tervezem (guest, user, admin), de szeretném ha a további fejlesztések során lehetőségem lenne újabb szerepköröket meghatározni.

Acl tábla:

Mezőnév	Típus	Leírás
ID	INTEGER	Elsődleges kulcs
UserID	INTEGER	A Users táblára hivatkozó idegen kulcs
RoleID	INTEGER	A Roles táblára hivatkozó idegen kulcs
ActionID	INTEGER	Az Actions táblára hivatkozó idegen kulcs
Access	BOOL	Engedélyezett a hozzáférés?

Ezen tábla alapján épül majd fel a jogosultságkezelést szabályzó Acl objektum. A táblázat

sorai vonatkozhatnak egy szerepkörre vagy egy bizonyos felhasználóra. A hozzáférést műveletre lehet lehet engedélyezni vagy megtiltani.

Actions tábla:

Mezőnév	**Típus**	**Leírás**
ID	INTEGER	Elsődleges kulcs
ControllerID	INTEGER	A Controllers táblára hivatkozó idegen kulcs
Name	VARCHAR(45)	A művelet neve

Controllers tábla:

Mezőnév	**Típus**	**Leírás**
ID	INTEGER	Elsődleges kulcs
Name	VARCHAR(45)	A controller neve

Ez utóbbi két tábla szintén a jogosultságkezelés miatt került az adatbázisba. A két tábla között 1:n kapcsolat van, az Actions oldalán kötelezően.

Categories tábla:

Mezőnév	**Típus**	**Leírás**
ID	INTEGER	Elsődleges kulcs
Name	VARCHAR(45)	A kategória neve
Description	VARCHAR(255)	A kategória rövid leírása

A tananyagok kategóriáit tartalmazza. A szerepkörökhöz hasonlóan itt is szerettem volna biztosítani a kategóriák bővíthetőségét.

LearningMaterials tábla:

Mezőnév	Típus	Leírás
ID	INTEGER	Elsődleges kulcs
CategoryID	INTEGER	A Categories táblára hivatkozó idegen kulcs
Title	VARCHAR(255)	A tananyag címe
Description	VARCHAR(255)	Rövid leírás a tananyagról

A közzétett tananyagok adatait tartalmazza.

Newsitems tábla:

Mezőnév	Típus	Leírás
ID	INTEGER	Elsődleges kulcs
LearningMaterialID	INTEGER	A LearningMaterials táblára hivatkozó idegen kulcs
Content	TEXT	A hír szövege
Created	DATETIME	A létrehozás dátuma
Updated	DATETIME	A legutóbbi módosítás dátuma

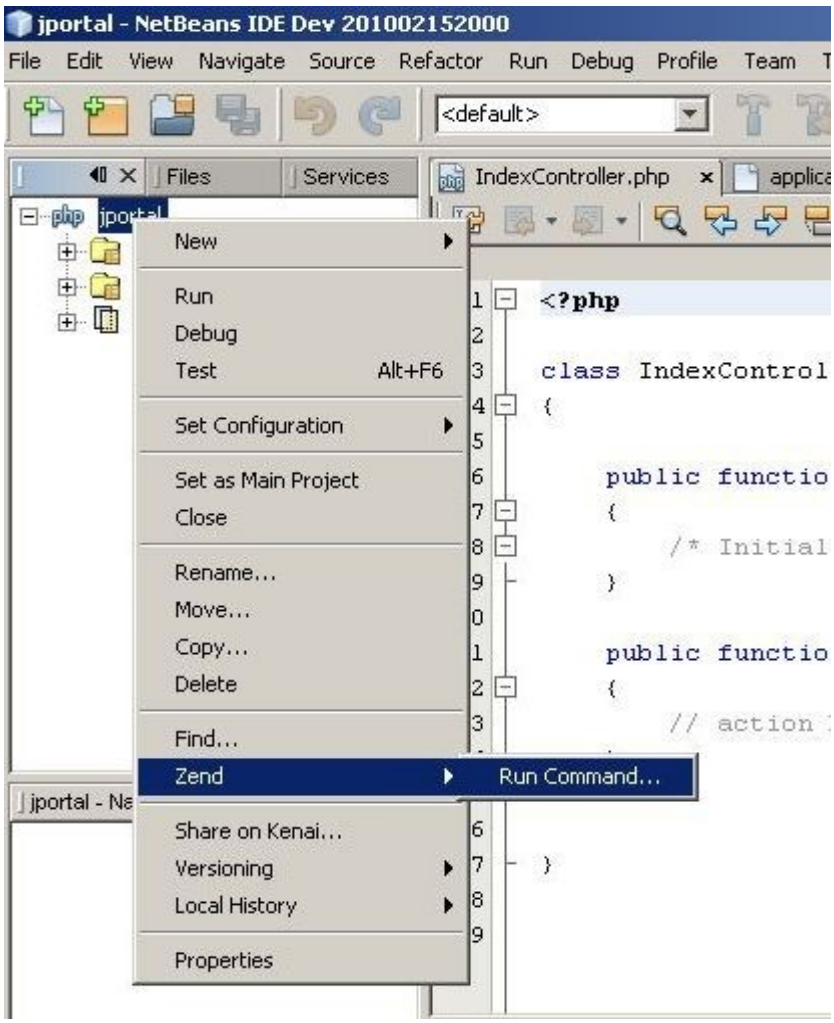
Az admin által létrehozott híreket tárolja

5 Kivitelezés

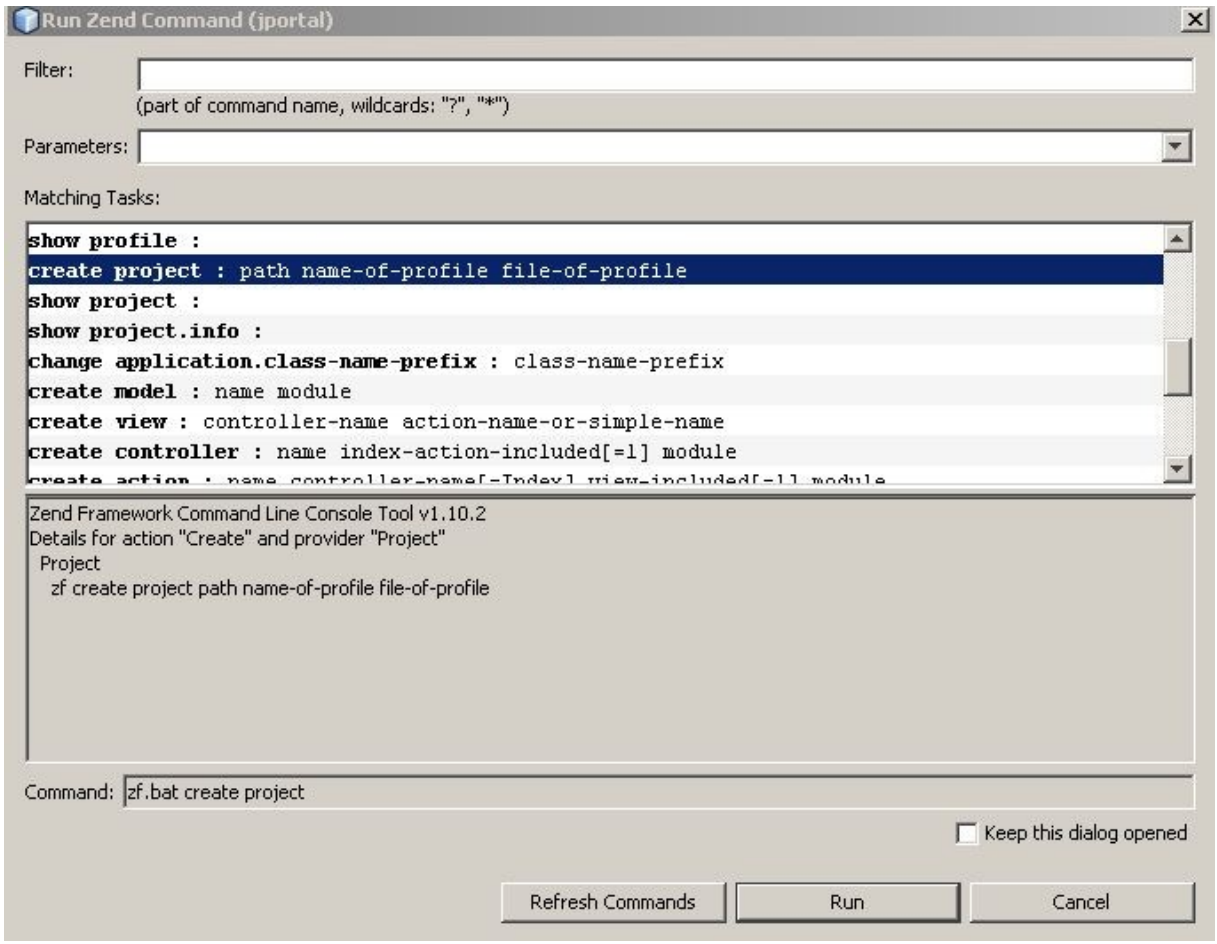
A kivitelezés során az általam alkalmazott fejlesztői környezet a Netbeans volt, melynek 6.9-es beta verziójába Zend Framwork támogatást is beépítettek a PHP mellé. Az IDE nagyon sok segítséget nyújtott a kódolás során, különös tekintettel a kódszínezésre, az automatikus kódkiegészítésre és a parancssoros eszköz használatának megkönnyítésére.

5.1 A projekt létrehozása

Miután a Netbeansben létrehoztunk egy új PHP projektet, kiválasztva Zend Framework használata opciót, jöhet az alkalmazás könyvtárszerkezetének kialakítása. A Zend Framework parancsait a következőképpen hívhatjuk elő.



Ezután a következő ablak fogad minket.



Itt kiválasztva a create project parancsot paraméterként megadva a projekt elérési útját, létrejön az alapvető könyvtárstruktúra, ami valahogy így fog kinézni.:

portal/

- application/
  - configs
  - controllers
  - models
  - views/
    - filters
    - helpers
    - scripts

- library
- public/
    - css
    - img
    - js
- test

Az application könyvtár tartalmazza az alkalmazás kódját, benne találhatóak az MVC-t megvalósító osztályok kódjai, valamint a konfigurációs állományok. A public könyvtárban találhatóak azok a fájlok amelyek az alkalmazáson kívülről elérhetőek. Ezek közé tartozik az index.php, melyen keresztül el lehet indítani az alkalmazást.

5.2 A bootstrap folyamata

A bootstrap az a folyamat, amely során megtörténik az alkalmazás inicializálása és konfigurálása. A folyamatot leíró kód megadható az index.php állományban, vagy az appilication/Bootstrap.php állományban, hogy aztán az index.php-ban csak hivatkozzunk rá. Én egy harmadik utat választottam, nevezetesen a Zend_Application osztályt, mely a konstruktorában megadott konfigurációs állomány adatai alapján elvégezte az összes szükséges inicializációt.

5.3 A vezérlők és műveletek létrehozása.

A projekt létrehozásánál bemutatott módszerhez hasonlóan történhet. Amikor egy vezérlőt létrehozunk Amikor egy vezérlőt létrehozunk, létrejön egy php állomány a conrollers mappában egy hozzátartozó "index" művelettel, és egy könyvtár a view/scripts/ könyvtáron belül a vezérlő nevével, benne egy, az indexhez tartozó view-szkripttel. Műveletet csak vezérlőhöz kapcsolódóan lehet létrehozni, mindegyik művelethez tartozik egy vezérlő. Természetesen a művelet létrehozásakor is létrejön a hozzátartozó view-szkript.

5.4 Az adatbázis létrehozása.

Az adatbázis létrehozásához a phpMyAdmin nevű alkalmazást használtam, ami egy PHP-ban írt eszköz MySQL adatbázisok elérésére. Miután létrehoztam vele egy adatbázist, az alábbi szkripttel hoztam benne létre a táblákat.

```
CREATE TABLE Acl (
ID INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
RoleID INTEGER UNSIGNED NULL,
ActionID INTEGER UNSIGNED NULL,
Access BOOL NULL,
PRIMARY KEY(ID),
INDEX Acl_FKIndex1(RoleID),
INDEX Acl_FKIndex2(ActionID)
);

CREATE TABLE Actions (
ID INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
ControllerID INTEGER UNSIGNED NULL,
Name VARCHAR(45) NULL,
PRIMARY KEY(ID),
INDEX Actions_FKIndex1(ControllerID)
);

CREATE TABLE Categories (
ID INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
Name INTEGER UNSIGNED NULL,
Description INTEGER UNSIGNED NULL,
PRIMARY KEY(ID)
);

CREATE TABLE Controllers (
ID INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
Name VARCHAR(45) NULL,
PRIMARY KEY(ID)
);

CREATE TABLE LearningMaterials (
ID INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
CategoryID INTEGER UNSIGNED NULL,
```

```
Title VARCHAR(255) NULL,
Description VARCHAR(255) NULL,
PRIMARY KEY(ID),
INDEX LearningMaterials_FKIndex1(CategoryID)

);

CREATE TABLE Newsitems (
ID INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
LearningMaterialID INTEGER UNSIGNED NOT NULL,
Content TEXT NULL,
Created DATETIME NULL,
Updated DATETIME NULL,
PRIMARY KEY(ID),
INDEX Newsitems_FKIndex1(LearningMaterialID)
);

CREATE TABLE Roles (
ID INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
Name VARCHAR(45) NULL,
PRIMARY KEY(ID)
);

CREATE TABLE Users (
ID INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
RoleID INTEGER UNSIGNED NULL,
Name VARCHAR(45) NULL,
Pass VARCHAR(45) NULL,
email VARCHAR(255) NULL,
RegistrationDate DATETIME NULL,
LastLogin DATETIME NULL,
PRIMARY KEY(ID)
INDEX Users_FKIndex1(RoleID)
);
```

5.5 A modellek létrehozása.

Mint azt már korábban említettem, az MVC mentén tervezett webalkalmazások nagy részében az adatbázis képezi a modell alapját. Igaz ez az én esetemben is. Az adatbázissal való kapcsolat a bootstrap során jött létre. Ezek után nem maradt más hátra mint a táblák, a táblák közötti kapcsolatok és a tábla sorainak modellezése. Ahogy a 3.5.1.3 fejezetben már szót ejtettem róla, a Zend_Db_Table megvalósítja Table Data Gateway mintát. Három fő komponense van, ezek: Zend_Db_Table_Abstract, Zend_Db_Table_Rowset, és Zend_Db_Table_Row, melyek megvalósításai illetve kiterjesztései az adatbázis egy tábláját, egy lekérdezés eredményhalmazát, illetve egy rekordját modellezi. Példa a táblák modelljére:

```
class Application_Model_Categories extends
                                        Zend_Db_TableAbstract
{
     protected &_name = 'Categories';
     protected $_dependentTables = array('LearningMaterials');
}

class Application_Model_LearningMaterials extends
                                        Zend_Db_Table_Abstract
{
     protected &_name = 'LearningMaterials';
     protected $_referenceMap = array(
          'Category' => array(
          'columns' => array('CategoryID'),
          'refTableClass' => 'Application_Model_Categories',
          'refColumns => array('ID')
          )
     );
}
```

A két osztály, mint ahogy az nevükből is látszik,a Categories és a LearningMateriels táblákat modellezik, ezt az osztályon belül a $_name változó értéke jelöli. A két tábla között 1:n kapcsolat áll fenn, melyet az 1 számosság oldalán a$_dependentTables, az n számosság oldalán a $_referenceMap változó ír le.

5.6 A felület kialakítása

A felhasználói felület kialakításakor a Zend_Layoutot használtam, amely a Zend_View-t felhasználva nagy segítségemre volt abban hogy egy egységes megjelenést adjak a különböző nézeteknek. A layout szkript az application/layouts/scripts/layout.phtml fájlban található. Azok a kinézetet befolyásoló kódok szerepelnek benne, melyek minden nézet esetén azonosak. Az éppen aktuális nézethez tartozó kódot a `layout()->content` változó tartalmazza.

Hiragana

Katakana

6 Összefoglalás

A portál készítése során megvalósítottam azokat a funkciókat melyeket a legszükségesebbnek tartottam. És bár mind funkcionalitását, mind kinézetét tekintve bőven van még mit fejleszteni rajta, az elkészült alkalmazás jó alapja lehet ennek a fejlesztésnek. Új funkciók a dolgozatban bemutatottakkal analóg módon építhetőek be a rendszerbe, főként az MVC mintából adódó architektúrának köszönhetően.

Dolgozatomban megpróbáltam olyan technológiákat bemutatni, melyek egyrészt kiválóan alkalmazhatóak a webfejlesztés területén, másrészt ingyenesek, bárki számára szabadon hozzáférhetőek.

7 Irodalomjegyzék

Magyar nyelvű irodalom:

- [1] Gál Tibor: Webprogramozás, Műegyetemi Kiadó Budapest, 2004
- [2] Matt Zandstra: Tanuljuk meg a PHP 5 használatát 24 óra alatt, Kiskapu, 2005
- [3] Jeffrey D. Ullman, Jennifer Widom: Adatbázisrendszerek, Panem, 1998
- [4] George Reese, Randy Jay Yarger, Tim King: A MySQL kezelése és használata, Kossuth, 2003

Angol nyelvű irodalom:

- [5] Rob Allen, Nick Lo, Steven Brown: Zend Framework in Action, Manning Publications, 2009

Magyar nyelvű internetes források:

- [6] http://hu.wikipedia.org/wiki/Japán_nyelv

Angol nyelvű internetes források:

- [7] http://www.php.net/manual/en/index.php
- [8] http://w3schools.com/
- [9] http://dev.mysql.com/doc/refman/5.1/en/
- [10] http://framework.zend.com/manual/en/manual.html
- [11] http://en.wikipedia.org/wiki/Model-view-controller
- [12]http://maff.ailoo.net/2009/04/set-up-a-zend-framework-application-using-zend_application-including-phpunit-setup/
- [13]http://en.wikipedia.org/wiki/Internet

8 Köszönetnyilvánítás

Szeretnék köszönetet mondani témavezetőmnek, Dr. Horváth Gézának a dolgozatírás során adott tanácsaiért.