SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR

Upotreba MVC frameworka u izradi PHP aplikacija

Nikola Brežnjak
Voditelj: *prof.dr.sc. Nikola Bogunović*

Zagreb, travanj, 2009.

Sadržaj

1. Uvod

U današnje vrijeme je sve veća potreba, bilo osobna bilo od strane neke tvrtke ili organizacije da bude prisutna na Internetu, tj. da ima svoje vlastite stranice. Međutim, kako je tehnologija napredovala nije više zadovoljavajuće imati statičke html stranice sa ponekom slikom, već se danas traže stranice koje omogućuju određenu dinamiku i interakciju korisnika s samom stranicom. Tu uskače PHP programski jezik koji sa svojom jednostavnošću omogućuje izradu kompleksnih dinamičkih stranica (npr. internet trgovina). Međutim svaki zahtjevniji projekt danas ima puno linija koda, koji onda postaje nepregledan i nečitljiv pa su zbog toga razvijeni razni frameworci koji olakšavaju izradu, održavanje dinamičkih web stranica.

U ovom seminarskom radu će se obraditi osnove programskog jezika PHP, prikazati osnovni koncept MVC frameworka kao pomoć pri izradi PHP aplikacija, dati popis dostupnih PHP MVC frameworka i napraviti njihova usporedba, te će se na kraju uzeti CodeIgniter PHP MVC framework i obraditi detaljno, te prikazati izrada jednostavne implementacije u cilju da se prikaže jednostavnost korištenja i učinkovitost ovog frameworka.

2. PHP

2.1. Povijest PHP-a

PHP (*engl. PHP: Hypertext Preprocessor*) je programski jezik interpreterskog tipa (na računalu postoji *interpreter*, program koji čita naredbe te ih izvršava) namjenjen izradi softverskih aplikacija koje se izvršavaju na Windows ili na UNIX-olikim poslužiteljima (*engl. Server*). Podržava proceduralno i objektno orjentirano programiranje, a najčešće se koristi kao razvojna platforma za interaktivne web stranice.

PHP je nastao iz PHP/FI kojeg je 1995. godine napravio Rasmus Lerdorf, kombinirajući Perl skripte na svojim osobnim web stranicama. Taj softver je nazvao 'Personal Home Page Tools / Forms Interpreter'. S vremenom je na to dodavao neke funkcije iz programskog jezika C za komunikaciju s bazama podataka i publiciranje dinamičkih web stranica. Rasmus je javno objavio kôd svog PHP/FI da bi ga svi mogli koristiti, ali i ako žele sudjelovati u budućem razvoju i poboljšanju.

PHP/FI je od početka imao neke od osnovnih funkcionalnosti PHP-a kojeg poznajemo dan danas. Koristio je variable na način Perla, automatsko interperetiranje variabli primljenih iz HTTP formi i omogućavao uključivanje HTML sintakse. Sintaksa mu je bila jako slična Perl-u, uz nešto ograničenja i pojednostavljenja, ali i sa dosta nekonzistetnosti.

Godine 1997. pojavilo se drugo izdanje PHP/FI-a - verzija 2.0, napisano u C-u. Tada ga je koristila grupa od nekoliko tisuća ljudi širom svijeta na oko 50,000 stranica, što je otprilike bilo oko 1% Internet domena u tom trenutku. Iako je već nekoliko ljudi intenzivno sudjelovalo u razvoju, to je i dalje bio uglavnom *one-man* projekt. PHP/FI 2.0 je službeno izdan u studenom 1997. godine, nakon što je dugo bio u beta izdanju.

PHP 3 je prvo izdanje koje sliči na ovo današnje, a stvorio ga je dvojac - Andi Gutmans i Zeev Suraski 1997. godine prepisujući kompletni PHP/FI 2.0 jer ih nije zadovoljavao u razvoju njihovih *e-commerce* web aplikacija. Andi, Rasmus i Zeev odlučili su surađivati u zajedničkom razvoju PHP 3.0 kao službenog nasljednika PHP/FI 2.0. Tako je nastao novi programski jezik nazvan 'PHP' što je danas skraćenica od 'PHP: Hypertext Preprocesor'.

Jedna od najvećih kvaliteta PHP 3.0 je bila mogućnost dodavanja novih funkcionalnosti, uporabe različitih baza podataka, protokola i API-ja. Ključ uspjeha PHP 3.0 je, dakle, bila mogućnost modularnih proširenja, objektno orjetirana sintaksa i primjenjivost za web programiranje. Stabilna verzija PHP 3.0 je službeno izdana u lipnju 1998. godine, nakon otprilike 9 mjeseci testiranja. Tijekom 1998. godine, PHP se širi na desetke tisuća korisnika i stotine tisuća web poslužitelja zauzimajući pri tome udio od oko 10% svih Web poslužitelja.

Od zime 1998., Gutmans i Zeev Suraski su počeli rad na rekonstrukciji jezgre PHP-a. Ciljevi su bili poboljšati karakteristike kompleksnih aplikacija i još više povećati modularnost PHP-a. Novi PHP interpreter, nazvan 'Zend Engine' (od Zeev and Andi), postigao je zadane ciljeve i predstavljen je prvi put sredinom 1999. Nova verzija, PHP 4.0, je bazirana na ovom interpreteru sa novim proširenjima izdana je službeno u svibnju 2000, skoro dvije godine nakon pojave PHP 3. Osim poboljšanja jezgra i novih ekstenzija, značajne novosti su bile podrška za još više web poslužitelja, HTTP sesije (*engl. sessions*), tzv. "*output buffering*" i sigurniji način rukovanja korisničkim podacima.

PHP 4 verziju PHP-a koriste stotine tisuća programera na skoro 10 milijuna web poslužitelja, odnosno nalazi se na preko 20% Internet domena. Izašla je i nova verzija PHP-a (PHP 5) koja je u potpunosti stabilna, te dolazi sa još više modula za podršku raznim novim tehnologijama (npr. XML), te omogućava potpuno objekno orjentirano programiranje.

2.2. Osnove PHP-a

Dinamičke PHP stranice su kao i obične web stranice, osim što se PHP naredbe mješaju s HTML tagovima. Točnije, PHP naredbe su omeđene posebnim tagovima koji to indiciraju. PHP stranice imaju nastavak *.php*, koji kaže web poslužitelju da to nije obična web stranica, nego nešto što traži dodatnu obradu, u ovom slučaju od strane php interpretera. Taj će program proći kroz cijelu web stranicu, te na mjestima označenim s PHP tagovima izvršiti naredbe, eventualno ih supstituirati s ispisom koji oni produciraju, i na kraju takva prerađena stranica bit će poslana klijentu koji će je moći pregledati u svojem web pregledniku.

Sljedeći isječak koda prikazuje primjer obične web stranice koja ispisuje prikladnu poruku:

```
<html>
<head>
     <title>primjer 1.</title>
</head>
<body>
     <p>Ovo je primjer jednostavne web stranice!</p>
</body>
</html>
```

Sljedeći isječak koda prikazuje primjer obične web stranice s PHP naredbama:

```
<html>
<head>
    <title>primjer 1.</title>
</head>
<body>
     <?php
           echo "Bok, narode!";
     ?>
</body>
</html>
```

Dakle, sve PHP naredbe se pišu unutar bloka *<?php* i *?>*, a odvajaju sa točkom zarez (;). Imena varijabli počinju s dolarom ($) i nije ih potrebno unaprijed deklarirati. Ostatak imena *mora* početi s slovom ili podvučenom crtom (_), ali ime može sadržavati i brojke. Velika i mala slova se razlikuju! Sljedeći isječak kod prikazuje primjere nekoliko varijabli:

```
$var = "Ovo je string";
$Var = "A ovo je različita varijabla od one prije";
$_ok = "I ovo je dobro ime varijable";
$o23456 = 23456;
```

Vrste podataka u PHPu su:

- cijeli broj (integer)
- boolean (istina/laž)
- realni ili decimalni broj (floating point)
- string (niz slova)
- niz (array) : indeks u nizu mora biti ili nenegativni cijeli broj ili string
- razred (object ili class)

Sljedeći isječak koda prikazuje nekoliko varijabli različitih tipova podatka:

```
$a = -1234;      // negativni broj
$b = 0123; // oktalni broj
$c = 0x1A; // heksadecimalni broj

$a = True; // boolean
$a = 1.234;      // decimalni brojevi
$b = 1.2e3;
$c = 7E-10;

$a = 'string s jednostrukim navodnim znakom';
$b = "string koji ima kraj linije na kraju\n";

$a = array( 1 => 'one', 2 => 'two', 3 => 'three' ); # niz string-ova
$b['ab'] = 1.3:              # niz decimalnih brojeva
$b['ac'] = 3.2;

$c[1] = 'jedan';
$c['01'] = 1;                # različita varijabla od $c[1] !

$zz = 0;
$aa[$zz] = True; #boolean tip
$d = array(1,2,3,5,7,11,13,17,19); # niz s cjelobrojnim indeksom
```

Sljedeći isječak koda prikazuje osnovne funkcije za kontrolu toka programa (if, else, while, for):

```
#primjer if naredbe
$a = 1;
$b = 50;
if ( $a > $b)
{
     echo ("a je veće od b");
}
elseif ( $a == $b)
{
     echo ("a je jednako b");
}
else
{
     echo ("b je veće od a");
}
```

```
#primjer while petlje
$i = 0;
$s = $i;
while ($i <= 10)
{
     $s += $i++; // skraćeni zapis za: $s = $s+$i; $i = $i+1;
}
print $s;


#primjer for petlje
$s = 0;
for ($i = 0; $i <= 1000; $i++)
{
     $s += $i;
}
print $s;


#primjer foreach petlje - koristi za iteriranje po članovima nekog niza
$arr = array (1,2,3,5,7,11,13,17,19);
$sum = 0;
foreach ($arr as $val) { $sum += $val; }
print $sum;
$telefon = array (
     'Pero' => '3456-345',
     'Miro' => '5678-432',
     'Toma' => '6545-111',
     'Vera' => '6545-111'
     );


foreach ($telefon as $ime => $broj)
{
     if (($broj == $telefon['Vera'])and ($ime != 'Vera'))
     {
          print "$ime i Vera žive zajedno!\n";
     }
}
```

3. MVC

Model-view-controller (MVC) je arhitekturalni uzorak (engl. Architectural pattern) koji se koristi u programskom inženjerstvu. Uspješna uporaba uzorka izolira poslovnu logiku od korisničkog sučelja, što rezultira programom u kojem je lakše modificirati vizualni izgled aplikacije ili osnovna poslovna pravila bez utjecaja na druge dijelove aplikacije.

U MVC uzorku model predstavlja podatke aplikacije, pogled odgovara elementima korisničkog sučelja (npr. tekst), dok kontroler upravlja komunikacijom podataka i poslovnih pravila za manipuliranjem podacima prema i iz modela.

MVC je prvi opisao Trygve Reenskaug. Postoji nekoliko izvedenica MVC uzorka, a jedan od najpoznatijih (zbog toga što je korišten od strane Microsofta) je Model View Presenter uzorak koji se pojavio 1990 godine i koji je bio dizajniran kao evolucija MVC uzorka, međutim Model-View-Controller i dalje ostaje u vrlo velikoj upotrebi.

Model-view-controller je istovremeno i arhitekturalni uzorak i dizajn uzorak (*engl. design pattern*), ovisno o tome gdje se koristi:

1. Kao arhitekturalni uzorak

   Uobičajeno je razdvojiti aplikaciju u odvojene slojeve koji se izvode na različitim računalima: sloj prezentacija tj. sloj korisničkog sučelja, sloj poslovne logike i podatkovni sloj. U MVC-u je prezentacijski sloj dalje podijeljen na model, view (pogled) i controller.

   MVC je često korišten u web aplikacijama, gdje su pogledi (*engl. view*) stvarne HTML ili XHTML stranice, a kontroler je kôd koji prikuplja podatke i generira dinamički sadržaj unutar HTML ili XHTML stranica. Na kraju, model predstavlja stvarni sadržaj, koji je često pohranjen u bazi podataka ili u XML čvorovima.

   Iako postoji više različitih implementacija MVCa, kontrola protoka je općenito sljedeća:

   a. Korisnik interagira sa sučeljem na neki način (npr. pritisne tipku miša)

   b. Kontroler obrađuje ulazni događaj od korisničkog sučelja, često putem registriranog rukovatelja (*engl. Handler*)

   c. Kontroler obavještava model o korisničkoj akciji, što potencijalno može dovesti do promjene u stanju modela (npr. kontroler ažurira korisnikovu košaricu)

   d. Pogled koristi model da posredno generira odgovarajuće korisničko sučelje (npr. u pogledu se prikazuje sadržaj korisničke košarice). Pogled dobiva svoje vlastite podatke iz modela. Model i kontroler nemaju direktnu vezu sa pogledom

   e. Korisničko sučelje čeka daljnju interakciju korisnika, što ponovno pokreće ciklus

   Razdvajanjem modela i pogleda MVC pomaže smanjiti složenost u arhitekturalnom dizajnu i povećava fleksibilnosti i ponovnu iskorištenost koda.

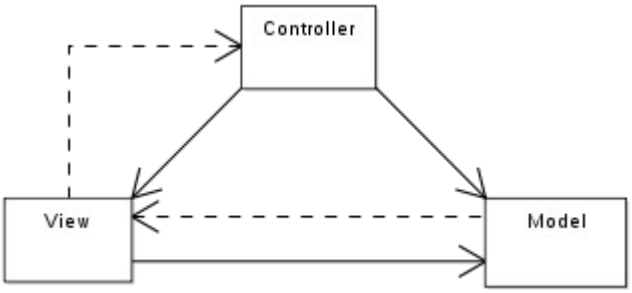
2. Kao dizajn uzorak
   MVC obuhvaća više od arhitekture programa nego što je tipično za dizajn uzorak. Kada se o njemu govori kao o dizajn uzorku, MVC je fundamentalno jednak Observer uzorku.

   a. Model
   Mnogi programi koriste mehanizam perzistencije (kao što je baza podataka) za pohranu podataka. MVC nema izričito definiran sloj pristupa podacima, jer je podrazumijeva da se nalazi na sloju ispod ili je enkapsuliran u modelu

   b. Pogled (*engl. View*)
   Čini model u obliku pogodnom za interakcije, obično jedan korisnički element za inerakciju. Višestruki pogledi mogu postojati za jedan model za različite svrhe.

   c. Kontroler (*engl. Controller*)
   Procesi i odgovari na događaje (tipično korisničke akcije) koji neizravno mogu izazvati promjene u modelu.

Na slici 3.1 je prikazan dijagram koji prikazuje veze između pojedinih dijelova MVC uzorka. Potrebno je napomenuti da pune linije označavaju direktnu vezu, dok iscrtkane linije predstavljaju indirektnu vezu.



Slika 3.1. Veza između modela, pogleda i kontrolera u MVC uzorku

4. Pregled dostupnih PHP MVC frameworka

Nekoliko popularnijih PHP MVC frameworka:

1. Zend Framework (6.5)
2. CakePHP (7.1)
3. Symfony (6.6)
4. Code Igniter (7.7)
5. PRADO (6.5)

U zagradama su prikazane prosječne ocjene (maksimalno 10) preuzete sa stranice http://www.coldscripts.com/php/frameworks. Budući da je na toj stranici i na još nekoliko njih Code Igniter dobio najveći broj glasova, moja upotreba Code Ignitera je opravdana.

5. CodeIgniter

5.1. Što je CodeIgniter?

CodeIgniter je moćan PHP *framework* koji je razvijen za PHP programere koji trebaju jednostavan i elegantan alat za stvaranje potpuno funkcionalnih web aplikacija. Omogućuje brzo stvaranje aplikacija pružajući bogate kolekcije knjižnica (*engl. Library*) za često korištene zadatke, ali i jednostavno sučelje i logičnu strukturu za pristup tim knjižnicama.

5.2. Temelj CodeIgniter-a

CodeIgniter se temelji na MVC (*engl. Model-View-Controller*) razvojnom obrascu, čiji je cilj razdvajanje aplikacijske logike od prezentacije. MVC rješava taj problem tako da razdvaja pristup podacima i poslovnu logiku od prezentacije podataka i korisničke interakcije, uvodeći posredničku komponentu nazvanu kontroler (*engl. Controller*).

- **Model**

  – obuhvaća specifičnu domensku reprezentaciju informacija s kojima aplikacija upravlja. Objekti modela tipično sadrže mehanizme komuniciranja sa sustavima stalne pohrane podataka (npr. baze podataka)

- **View**

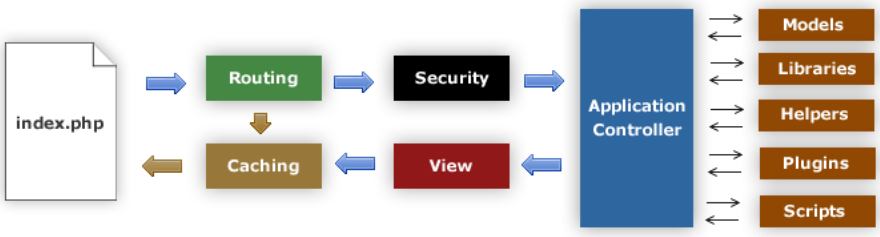
  – predstavlja informaciju koja se prezentira korisniku, najčešće u obliku web stranice. Za jedan model može postojati nekoliko pogleda, svaki sa određenom svrhom

- **Controller**

  – služi kao posrednik između Modela i View-a, tako da obrađuje i odgovara na događaje (*engl. Events*) koji su tipično korisničke akcije koje mogu uzrokovati promjene na modelu

Kod web aplikacija, *View* je zapravo HTML stranica, *Controller* je kôd koji prikuplja dinamičke podatke i generira sadržaj. Na kraju, *Model* je predstavljen aktualnim sadržajem koji je obično spremljen u bazu podataka ili XML datoteke.

Na slici 5.1 je prikazan protok podataka u *CodeIgniter*-u.



Slika 5.1. Protok podataka u *CodeIgniter*-u

Opis pojedinih dijelova sa slike 5.1:

- *index.php* – prednji kontroler koji inicijalizira osnovne resurse potrebne za pokretanje *CodeIgniter*-a
- *Routing* – pregledava HTTP zahtjev da bi odredio što treba s njime napraviti
- *Caching* – ako postoji cache datoteka onda se ona direktno šalje Internet pregledniku, zaobilazeći pri tome normalno izvođenje sustava
- *Security* – prije nego se učita aplikacijski kontroler, filtriraju se HTTP zahtjevi i svi korisnički podaci koji su predani
- *Application Controller* – učitava model, osnovne knjižnice (*engl. Core libraries*), te sve ostale resurse koji su potrebni za procesiranje specifičnog zahtjeva
- *View* – finalizirani pogled se generira i šalje Internet pregledniku. Ako je uključen međuspremnik (*engl. Cache*), *View* se prvo sprema u međuspremnik kako bi se naredni isti zahtjevi mogli poslužiti iz međuspremnika (brže posluženi zahtjevi)

5.3. Instalacija *Codeigniter*-a:

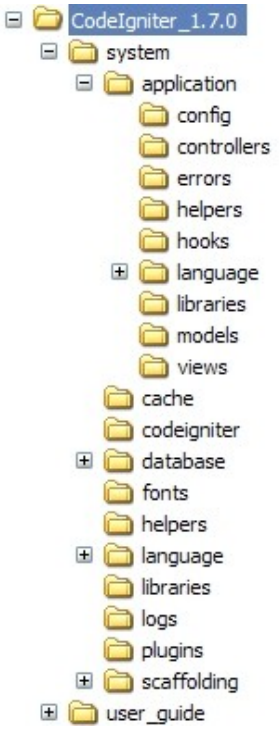
Instalacija CodeIgniter-a je veoma jednostavna i svodi se doslovno na otpakiravanje arhive i podešavanje par konfiguracijskih datoteka. Postupak instalacije je sljedeći:

- Preuzeti najnoviju verziju sa stranice http://codeigniter.com/download.php
- Otpakirati skinutu datoteku u `htdocs` direktorij vašeg poslužitelja (na nekim poslužiteljima se taj direktorij naziva `public_html`). `Htdocs` direktorij je korijenski (*engl. root*) direktorij kojeg vide korisnici koji pristupaju određenoj stranici

5.4. Osnovni građevni blokovi CodeIgniter-a

5.4.1. Struktura direktorija CodeIgniter frameworka

Struktura direktorija *CodeIgniter*-a je prikazana na slici 5.2. Najvažniji je *application* direktorij, jer se u njemu nalazi sva logika aplikacije. U direktoriju *config* se nalaze konfiguracijske datoteke, kao što je datoteka za konfiguraciju baze podataka, te datoteka za općenite postavke *CodeIgniter* frameworka. Direktoriji *controllers, models* i *views* su namijenjeni za MVC. Direktorij *libraries* služi za spremanje korisničkih knjižnica. Direktorij *hooks* sadrži datoteke koje proširuju funkcionalnosti same jezgre *CodeIgniter*-a bez potrebe za modificiranjem izvornog koda jezgre. Direktorij *errors* sadrži datoteke za prikaz i obradu grešaka.



Slika 5.2. Struktura direktorija CodeIgniter-a

5.4.2. Upravljanje URL-ovima

URL-ovi (*engl. Uniform Resource Locator*) u *CodeIgniter*-u su segmentno bazirani (*engl. Segment based*), npr.:

```
primjer.com/vijesti/clanak/moj_clanak
```

Segmenti u URL-u, sljedeći Model-View-Controller pristup, su najčešće predstavljeni:

```
primjer.com/klasa/funkcija/ID
```

Prvi dio segmenta predstavlja *klasu kontrolera* (*engl. Controller Class*) koja se treba pozvati. Drugi dio segmenta reprezentira *funkciju* klase, tj. *metodu*, koja se treba pozvati. Treći dio predstavlja *ID* i varijable koje će se prenjeti kontroleru.

Po pretpostavljenim postavkama `index.php` datoteka će biti uključena u URL, te će on izgledati ovako:

```
primjer.com/index.php/news/article/my_article
```

Da bi se to izbjeglo treba u `.htaccess` datoteku dodati ovaj dio koda:

```
<IfModule mod_rewrite.c>
    RewriteEngine On
    RewriteBase /Seminar/
```

```
    RewriteCond %{REQUEST_URI} ^system.*
    RewriteRule ^(.*)$ /index.php/$1 [L]

    RewriteCond %{REQUEST_FILENAME} !-f
    RewriteCond %{REQUEST_FILENAME} !-d
    RewriteCond $1 !^(index\.php|images|robots\.txt|css)
    RewriteRule ^(.*)$ index.php/$1 [L]
</IfModule>

<IfModule !mod_rewrite.c>
    ErrorDocument 404 /index.php
</IfModule>
```

Ovdje je bitno uočiti `RewriteBase` opciju, koju treba postaviti na naziv točnog direktorija u kojem se aplikacija nalazi. Također, treba u konfiguracijskoj datoteci Apache poslužitelja (`httpd.conf`) odkomentirati liniju

```
LoadModule rewrite_module modules/mod_rewrite.so
```

i svako pojavljivanje teksta `AllowOverwrite` treba zamijeniti sa `AllowOverwrite All`, te također u konfiguracijskoj datoteci *CodeIgniter*-a `config.php` (nalazi se u `/application/config/` direktoriju) treba podesiti sljedeću liniju:

```
$config['index_page'] = "";
```

Datoteka `.htaccess` se na Windows operacijskim sustavima može napraviti tako da se pokrene program `notepad`, upiše traženi kod i nakon toga odabere `Save As...` opcija kod koje se kao tip datoteke odabere `All types (*.*)` i upiše naziv datoteke ".htaccess", baravno bez navodnika.

Također, u `config.php` datoteci je moguće podesiti da datoteke imaju sufiks poput `.html` ili `.php`.

5.4.3. Kontroleri

Kontroler je datoteka razreda (*engl. Class file*) koja je imenovana na način da se može povezati sa URI-em. Npr., ako imamo URI:

```
primjer.com/index.php/blog/
```

tada bi *CodeIgniter* pokušao pronaći i učitati kontroler naziva blog.php. Ove datoteke se stavaljaju u `application/controller/` direktorij. Svi nazivi klasa moraju početi velikim slovom i nasljediti baznu klasu Controller. Primjer:

```
<?php
class Blog extends Controller {

     function index()
     {
           echo 'Hello World!';
     }
```

```
}
?>
```

Ukoliko se želi koristiti i konstruktor za pojedinu klasu u bilo kojem kontroleru, potrebno je obavezno dodati ovaj dio koda:

```
parent::Controller();
```

Razlog zbog kojeg je ova linija potrebna je stoga što će lokalni konstruktor nadjačati onaj u roditeljskoj kontroler klasi, pa ga stoga trebamo ručno pozvati. Konstruktor je funkcija istog imena kao i klasa, a služi za postavljanje inicijalnih vrijednosti.

CodeIgniter može učitati podrazumijevani (*engl. Default*) kontroler ako se URI ne zada, što će biti slučaj kada netko zatraži korijenski URL stranice (npr. www.google.com). Da bi se podesio podrazumijevani kontroler treba u datoteku `routes.php`, koja se nalazi u `/application/config/` direktoriju, dodati:

```
$route['default_controller'] = 'Blog';
```

Gdje je *Blog* naziv klase kontrolera. Ako se sada učita glavna `index.php` datoteka bez sprecificiranja URI segmenata, podrazumijevano će se prikazati "Hello World" poruka (ako uzmemo prijašnji primjer kontrolera Blog).

5.4.4. Funkcije

U prethodnom primjeru naziv funkcije je `index()`. Funkcija `index()` se učitava uvijek kada se drugi segment URI-a ne zada. Drugi način da se u prethodnom primjeru ispiše "Hello World" je da se ukuca sljedeći link:

```
example.com/index.php/blog/index
```

Drugi segment URI-a određuje koja funkcija iz kontrolera se poziva. Npr., ako u prijašnji primjer dodamo funkciju `komentari()`

```
<?php
class Blog extends Controller {

     function index()
     {
          echo 'Hello World!';
     }

     function komentari()
     {
          echo 'Ispis komentara!';
     }
}
?>
```

i pozovemo tu funkciju na način:

```
example.com/index.php/blog/komentari
```

dobit ćemo ispis "Ispis komentara!".

Ako želimo da funkcija nema javna (*engl. Public*) prava pristupa, onda se samo doda "_" kao prefiks naziva funkcije, kao npr.:

```
function _utility()
{
  // todo:...
}
```

Ako URI sadrži više od dva segmenta, onda će oni biti preneseni u funkciju kao parametri. Npr., ako URI izgleda ovako:

```
example.com/index.php/proizvodi/cipele/sandale/456
```

Funkciji `cipele()` će biti preneseni URI segmenti 3 i 4 ("sandale" i "456"):

```
<?php
class Proizvodi extends Controller {

     function cipele($sandale, $id)
     {
           echo $sandale;
           echo $id;
     }
}
?>
```

5.4.5. Pogledi

Pogled (*engl. View*) je web stranica ili neki njezin dio (*eng. Fragment*). Pogledi se nikad ne pozivaju direktno, već ih uvijek mora učitati kontroler. Datoteke koje predstavljaju poglede se spremaju `u /application/views/` direktorij. Npr., izgled datoteke `blogview.php`:

```
<html>
<head>
<title>Moj Blog</title>
</head>
<body>
     <h1>Dobrodošli na moj Blog!</h1>
</body>
</html>
```

Da bi se učitao pojedini pogled treba ukucati:

```
$this->load->view('nazivPogleda');
```

Primjer pozivanja pogleda iz prijašnjeg primjera Blog kontrolera:

```
<?php
class Blog extends Controller {

     function index()
     {
           $this->load->view('blogview.php');
     }
}
?>
```

Prosljeđivanje dinamičkih podataka pogledu od strane kontrolera se radi navođenjem polja (*engl. Array*) ili objekta (*engl. Object*) kao drugog parametra u pozivu pogleda. Primjer kontrolera Blog:

```
<?php
class Blog extends Controller {

     function index()
     {
           $data['title'] = "My Real Title";
           $data['heading'] = "My Real Heading";

           $this->load->view('blogview.php', $data);
     }
}
?>
```

Primjer datoteke blogview.php:

```
<html>
<head>
<title><?php echo $title;?></title>
</head>
<body>
     <h1><?php echo $heading;?></h1>
</body>
</html>
```

5.4.6. Modeli

Modeli (*engl. Models*) su PHP klase koje su dizajnirane za rad sa informacijama iz baze podataka. Primjer (Blog) klase modela koja sadrži funkcije za unos, izmjenu i dohvat podataka:

```
class Blogmodel extends Model {
    var $title   = '';
    var $content = '';
    var $date    = '';

    function Blogmodel()
    {
        parent::Model();
    }

    function get_last_ten_entries()
```

```
    {
        $query = $this->db->get('entries', 10);
        return $query->result();
    }

    function insert_entry()
    {
        $this->title   = $_POST['title']; // please read the below note
        $this->content = $_POST['content'];
        $this->date    = time();

        $this->db->insert('entries', $this);
    }

    function update_entry()
    {
        $this->title   = $_POST['title'];
        $this->content = $_POST['content'];
        $this->date    = time();

        $this->db->update('entries', $this, array('id' =>$_POST['id']));
    }
}
```

Naziv datoteke bi bio `blogmodel.php` i bio bi spremljen u `/application/models/` direktorij *CodeIgniter* framework-a. Modeli se učitavaju iz kontrolera i to na sljedeći način:

```
$this->load->model('Model_name');
```

Nakon što se model učita njegove se funkcije pozivaju na sljedeći način:

```
$this->Model_name->funkcija();
```

Primjer kontrolera koji prvo učita model, a zatim posluži (*engl. Services*) model:

```
class Blog_controller extends Controller {
    function blog()
    {
        $this->load->model('Blog');

        $data['query'] = $this->Blog->get_last_ten_entries();

        $this->load->view('blog', $data);
    }
}
```

Model se kod učitavanja ne povezuje direktno sa bazom podataka, već to treba eksplicitno navesti:

```
$this->load->model('Model_name', '', TRUE);
```

5.4.7. Pomoćne funckcije

Pomoćne funkcije (*engl. Helper functions*) olakšavaju rad sa nekim zadacima. Svaka pomoćna funkcija je jednostavno kolekcija funkcija u određenoj kategoriji. Postoje URL Helpers,

koji pomažu kod stvaranje linkova (*engl. Links*), Form Helpers koji pomažu kod kreiranja form elementa, Text Helpers koji provode različita formatiranja teksta, itd. To su u biti jednostavne proceduralne funkcije, od kojih svaka obavlja određeni dio posla neovisno o nekoj drugoj funkciji.

CodeIgniter ne učitava pomoćne funkcije automatski, već to treba napraviti ručno, a jednom kad se pomoćna funkcija učita ona postaje dostupna globalno u svim kontrolerima i pogledima. Način učitavanja pomoćne funkcije je sljedeći:

```
$this->load->helper('name');
```

6. Implementacija

Za implementaciju sam odabrao izradu jednostavnog Bloga. Implementirane su opcije za dodavanje i pregledavanje postova, te dodavanje i pregledavanje komentara na postove. Također, radi jednostavnosti primjera nije implementirano logiranje ili registriranje administratora, već je namjera bila pokazati jednostavnost realizacije osnovne ideje upotrebom CodeIgniter frameworka.

6.1. Model baze podatka

Za spremanje podataka koristim `MySql` bazu podataka koja dolazi u `XAMPP` instalacijskom paketu. Imam jednu bazu podataka naziva `blog`, koja u sebi sadrži dvije tablice:

- `entries` – tablica koja služi za spremanje podataka o novim postovima. Njezina struktura je prikazana u tablici 6.1.

Tablica 6.1. Tablica entries

Entries
id INT(11) NOT NULL AUTO_INCREMENT PRIMARY KEY
title VARCHAR(128) NULL
body TEXT NULL

- `comments` – tablica koja služi za spremanje podataka o novim komentarima na određene postove. Njezina struktura je prikazana u tablici 6.2.

Tablica 6.2. Tablica comments

Comments
id INT(11) NOT NULL AUTO_INCREMENT PRIMARY KEY
entry_id INT(11) NULL
body TEXT NULL
author VARCHAR (100)

U tablici `entries` id označava jednistvenu identifikaciju određenog posta, `title` je naslov posta a `body` sam tekst posta. U tablici `comments` podatak `entry_id` služi za praćenje posta kojem je komentar bio dodan, dok `body` služi za spremanje samog teksta komentara, a `author` za spremanje autora dotičnog komentara.

U `controller` direktoriju CodeIgniter frameworka imam jedan kontroler naziva `Blog` čiji se izvorni kod nalazi u datoteci `blog.php` čiji je sadržaj sljedeći:

```
<?php
class Blog extends Controller {

     function Blog()
```

```
	{
		parent::Controller();
		$this->load->helper('url');
		$this->load->helper('form');
	}

	function index()
	{
		$data['title'] = "ITmaster Blog";
		$data['slogan'] = "najbolji slogan koji ste ikad vidjeli";
		$data['css'] = "http://localhost/Seminar/css/default.css";
		$data['main'] = "blog";

		$data['query'] = $this->db->get('entries');

		$this->load->view('blogview', $data);
	}

	function comments()
	{
		$data['title'] = "ITmaster Blog";
		$data['slogan'] = "najbolji slogan koji ste ikad vidjeli";
		$data['css'] = "http://localhost/Seminar/css/default.css";
		$data['main'] = "blog";

		$this->db->where('entry_id', $this->uri->segment(3));
		$data['query'] = $this->db->get('comments');

		$this->load->view('commentview', $data);
	}

	function comment_insert()
	{
		$this->db->insert('comments', $_POST);
		redirect('blog/comments/'. $_POST['entry_id']);
	}

	function unos_novog_obrada()
	{
		$this->db->insert('entries', $_POST);
		redirect('blog/');
	}

	function unos_novog()
	{
		$data['title'] = "ITmaster Blog";
		$data['slogan'] = "najbolji slogan koji ste ikad vidjeli";
		$data['css'] = "http://localhost/Seminar/css/default.css";
		$data['main'] = "blog";

		$this->load->view('unosview', $data);
	}
}
?>
```

Popis i značenje pojedinih funkcija iz izvornog koda Blog kontrolera:

- `Blog()` – konstruktor koji učitava pomoćne funkcije za jednostavniji rad sa URL-om i formama

- `index()` – glavna funkcija koja se poziva kada se drugi segment URI-a ne zada. Ona čita cijeli sadržaj tablice `entries` i sprema u varijablu `query`, te učitava pogled `blogview`, kojem šalje sve potrebne varijable, koji služi za prikaz svih postova
- `comments()` – preuzima podatke iz tablice `comments` i učitava pogled `commentview`, kojem pritom šalje sve potrebne varijable, koji služi za prikaz svih komentara na dotični post
- `comment_insert()` – unosi podatke preuzete iz `_POST` varijable (varijabla okruženja (*engl. Environment variable*)) u tablicu `comments`, te preusmjerava korisnika na stranicu za pregled svih komentara
- `unos_novog_obrada()` – unosi novi post u tablicu `entries` i preusmjerava korisnika na stranicu sa popisom svih postova
- `unos_novog()` – učitava pogled `unosview` koji služi za prikaz stranice za unos novog posta

U `view` direktoriju CodeIgniter frameworka imam tri datoteke (`blogview.php`, `unosview.php` i `commentview.php`) koje predstavljaju poglede. Primjer sadržaja datoteke `blogview.php`:

```
<html>
<head>
      <meta http-equiv="content-type" content="text/html; charset=UTF-8">
      <link rel="stylesheet" type="text/css" href=<?=$css?> media="screen">
      <title><?=$title?></title>
</head>

<body>
<div class="outer-container">
      <div class="inner-container">
            <div class="header">
               <div class="title">
                   <span class="sitename"><a href=<?=$main?>><?$title></a></span>
                   <div class="slogan"><?=$slogan?></div>
               </div>
            </div>

            <div class="main">
            <div class="content">
                  <?php if ($query->num_rows() > 0): ?>
                   <?php foreach($query->result() as $row): ?>
                     <h2><?=$row->title?></h3>
                     <p><?=$row->body?></p>
                     <p><?=anchor('blog/comments/'.$row-id, 'Komentari'); ?></p>
                     <hr>
                   <?php endforeach; ?>

                  <?php else:echo"Nema postova!"; ?>
                  <?php endif; ?>
            </div>
      <div class="navigation">
      <ul>
<li><p class="center nobottom"><?anchor('blog/', "Pregled postova");?></p></li>
<li> <p class="center nobottom"><?anchor('blog/unos_novog/', "Unos novog
posta"); ?></p></li>
</ul>
</div>
```

```
                    <div class="clearer"> </div>
            </div>
            <div class="footer">
                    <p class="center nobottom">Nikola Breznjak &copy;
 2008.<br></p>
                    <div class="clearer"></div>
            </div>
      </div>
</div>
</body>
</html>
```

Podebljani dio označava php kod koji služi za ispisivanje varijabli i prikaz podataka iz tablice `entries`.

Jedini bitni dio datoteke `unosview.php` je sljedeći:

```
<?=form_open('blog/unos_novog_obrada');?>
      <p class="center">Naslov novog posta:<br/>
            <input type="text" name="title" />
      </p>

      <p class="center">Tekst novog posta:<br/>
            <textarea name="body" cols="60" rows="10"></textarea>
      </p>

      <p class="center"><input  type="submit" value="Objavi post!" /></p>
</form>
```

Dakle, u ovoj se datoteci definira forma za unos novog posta koja ima dva polja (polje za unos naslova novog posta i polje za unos samog teksta posta). Obradu forme obavlja funkcija `unos_novog_obrada` koja se nalazi u `Blog` kontroleru.

Jedini bitni dio datoteke `commentview.php` je sljedeći:

```
<?php if ($query->num_rows() > 0): ?>
      <?php foreach($query->result() as $row): ?>
            <p><?=$row->body?></p>
            <p><?=$row->author?></p>
            <hr>
      <?php endforeach; ?>

      <?php else:echo"Nema komentara na ovaj post!"; ?>
      <?php endif; ?>

      <?=form_open('blog/comment_insert');?>
            <?=form_hidden('entry_id', $this->uri->segment(3));?>
            <p class="center">Dodaj novi komentar: <br/>
                  <textarea name="body" cols="60" rows="10"></textarea>
            </p>

            <p class="center">Ime: <br/><input  type="text" name="author" /></p>
            <p class="center"><input  type="submit" value="Komentiraj!" /></p>
      </form>
```

Dakle, prvo se provjeri da li postoje komentari, te ako postoje onda se sa `foreach` naredbom dohvaćaju svi komentari na konkretni post i ispisuje sam tekst komentara i njegov autor. Također, tu se nalazi forma koja služi za prihvat novih komentara. Obradu forme obavlja funkcija `comment_insert` koja se nalazi u `Blog` kontroleru.

7. Pokretanje programa

Da bi se mogla isprobati moja implementacija potrebno je slijediti par jednostavnih koraka.

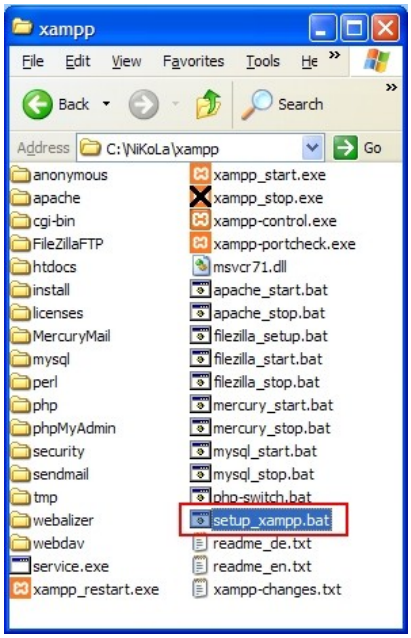
7.1. Instalacija XAMPP-a

Program XAMPP omogućuje jednostavnu instalaciju Apache web poslužitelja, PHP poslužitelja, te MySQL baze podataka. Program se može preuzeti sa sljedeće adrese: http://www.apachefriends.org/en/xampp-windows.html#641.

U navedenom primjeru je skinuta EXE verzija u 7-ZIP formatu. Nakon što se program skine, potrebno ga je pokrenuti, te se nakon toga dobije dijalog kao na slici 7.1. Tu treba odabrati mapu u koju će se program XAMPP otpakirati. U ovom primjeru odabrana je mapa C:\NiKoLa.

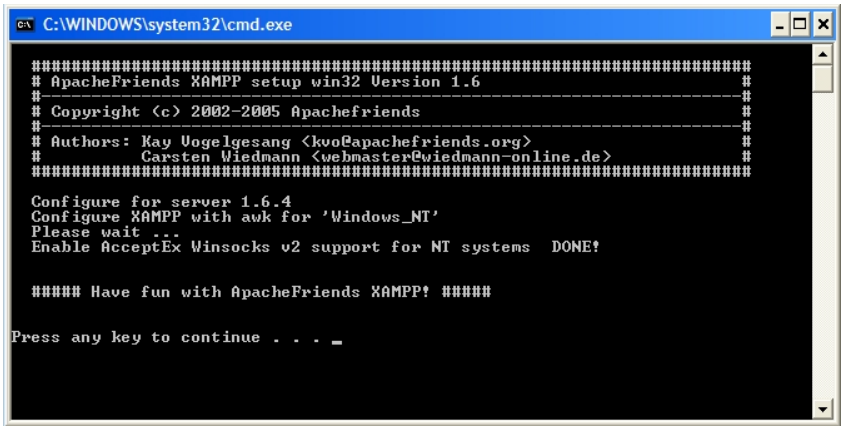
Slika 7.1. Definiranje mape za otpakiravanje programa XAMPP

Nakon što se program otpakira dobijemo novu mapu u mapi C:\NiKoLa, te nam potpun put do programa XAMPP sada glasi C:\NiKoLa\xampp. Sada je potrebno otvoriti taj direktorij i pokrenuti datoteku setup_xampp.bat, kao što je prikazano na slici 7.2.



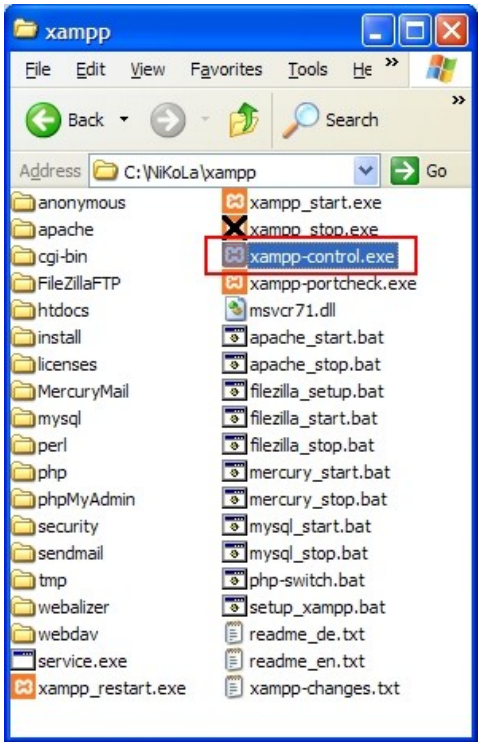
Slika 7.2. Pokretanje datoteke setup_xampp.bat

Nakon što se datoteka `setup_xampp.bat` pokrene i njezino izvršavanje završi dobije se prozor poput ovog na slici 7.3, te je za kraj instalacije potrebno pritisnuti bilo koju tipku na tipkovnici.



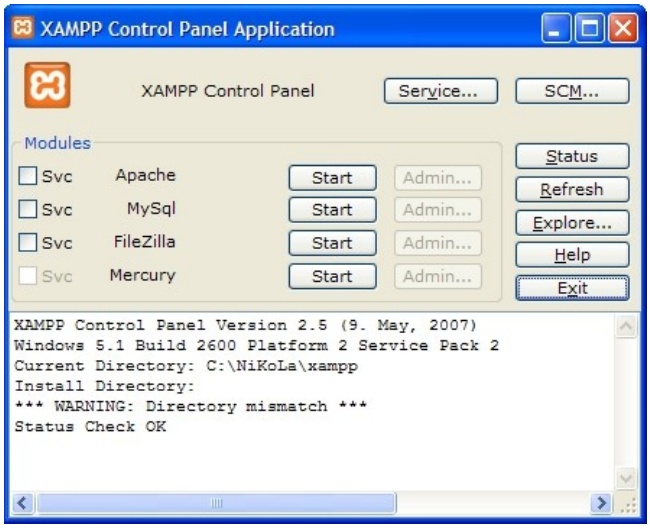
Slika 7.3. Rezultat izvođenja programa setup_xampp.bat

Naime, ovime su svi putovi (*engl. paths*) automatski podešeni što je potrebno zbog normalnog rada programa `XAMPP`. Za jednostavnije pokretanje pojedinih poslužitelja koje nudi program `XAMPP` postoji program `xampp-control.exe`, kao što je prikazano na slici 7.4.



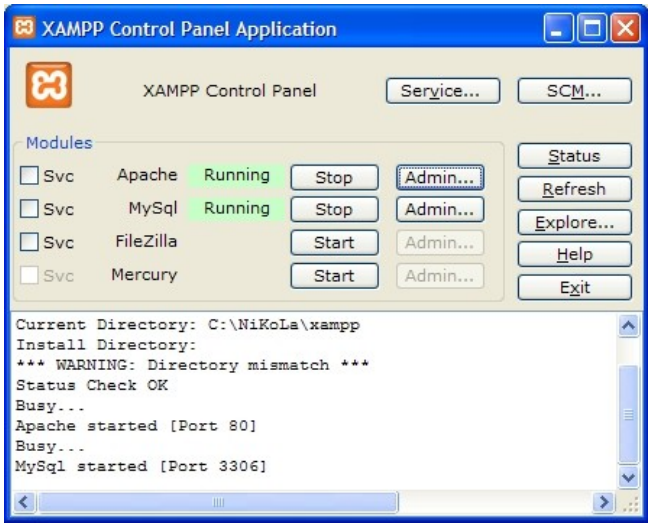
Slika 7.4. Pokretanje programa za jednostavnu kontrolu `XAMPP`-a

Na slici je prikazano grafičko sučelje programa `xampp-control.exe`. Kao što je i intuitivno jasno, pokretanje pojedinog poslužitelja (*engl. Server*) se ostvaruje klikom na tipku `Start`.



Slika 7.5. Grafičko sučelje za upravljanje poslužiteljima

Za naš primjer potrebno je pokrenuti poslužitelje `Apache` i `MySql`, te nakon toga je obavezno potrebno postaviti lozinke za pristup `XAMPP` direktoriju i `MySql` bazi podataka, što se radi klikom na tipku `Admin...` kao što je prikazano na slici 7.6.

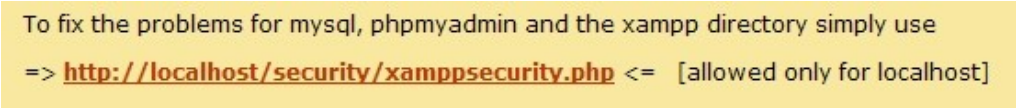


Slika 7.6. Pokrenuti poslužitelji `Apache` i `MySql`

Nakon što kliknemo na tipku `Admin`, pokreće se podrazumijevani (*engl. Default*) web preglednik i automatski otvori stranicu http://localhost/xampp. Ovdje je potrebno kliknuti na link `Security`, kao što prikazuje slika 7.7.

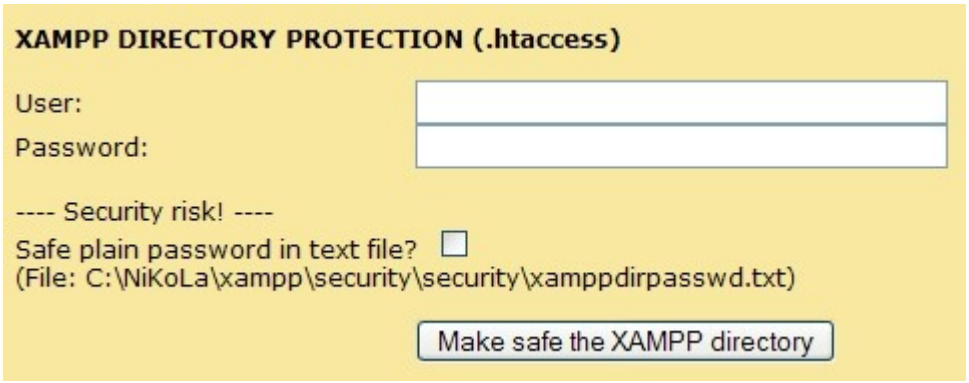
Slika 7.7. Odabir linka Security za podešavanje lozinke

Nakon klika na link `Security` se dolazi do stranice sa popisom sigurnosnih postavki, ali budući da mi želimo izmijeniti trenutne postavke tražimo tekst sa linkom http://localhost/security/xamppsecurity.php, kao što je prikazano na slici 7.8.



Slika 7.8. Odabir linka za promjenu lozinke

Na kraju konačno dolazimo do samog mjesta gdje se mjenja lozinka, te je tu potrebno postaviti lozinku za `Apache` poslužitelj i `MySql` bazu podataka. Oba korisnička imena (*engl. User name*) neka budu **`root`**, a također obje lozinke neka budu **`test123`** u svrhu testiranja moje implementacije. Na slici 7.9 je prikazan izgled mijenjanja korisničkog imena i lozinke za `Apache` poslužitelj.

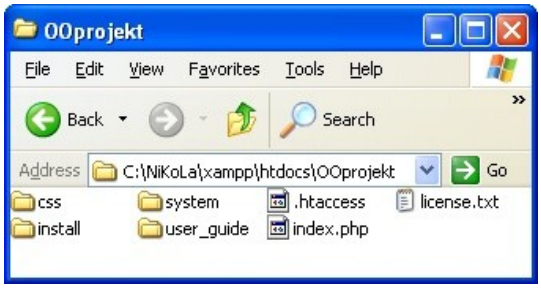


Slika 7.9. Mijenjanje korisničkog imena i lozinke za poslužitelj Apache

Na kraju je još potrebno napraviti izmjene u `httpd.conf` datoteci prema uputama u poglavlju 1.4.2.

7.2. Otpakiravanje

Arhivu `Seminar.rar` koja je poslana kao privitak u email-u treba raspakirati u `htdocs` direktorij instalacije `XAMPP`-a. Na slici 7.10 je prikazan sadržaj direktorija otpakirane arhive.



Slika 7.10. Sadržaj direktorija raspakirane arhive Seminar.rar

Nakon što su u prethodnom odjeljku podešena korisnička imena i lozinke za poslužitelje `Apache` i `MySql`, potrebno je još stvoriti bazu podataka i napraviti potrebne tablice koje koristi moj program. Srećom, ovaj dio se radi automatizirano, i to tako da se u web preglednik upiše sljedeća adresa: http://localhost/Seminar/install/install.php. Ukoliko sve prođe bez greške trebao bi se pojaviti ispis poput ovog na slici .

Uspio sam se spojiti na MySql bazu podataka...

Stvorio sam novu bazu podataka **blog**...
Stvorio sam tablicu **entries**...
Stvorio sam tablicu **comments**...

Proces instalacije baze podataka i podešavanje tablica je gotov...
Radi sigurnosnih razloga izbrišite ovu datoteku!

Kliknite ovdje za odlazak na glavnu stranicu...

Slika 7.11. Uspješno provedena instalacija baze podataka i podešavanje tablica

Slijedeći upute potrebno je obrisati datoteku `install.php` zbog sigurnosnih razloga, te kliknuti na poveznicu `ovdje` koja vodi na glavnu stranicu za pregled svih postova. Nakon ovog koraka je moguće isprobavati rad ove implementacije dodavanjem novih postova i komentara.

8. Zaključak

CodeIgniter je veoma moćan PHP *framework* koji uistinu olakšava izradu dinamičkih web aplikacija. Ima odličnu grupu ljudi koji rade na njegovom održavanju i poboljšavanju, te je i na kraju krajeva potpuno besplatan za korištenje. Omogućuje brzo stvaranje aplikacija pružajući bogate kolekcije knjižnica za često korištene zadatke, ali i jednostavno sučelje i logičnu strukturu za pristup tim knjižnicama. Dojam osobnog korištenja je odličan, jer doista pojednostavljuje i skraćuje neke rutine koje su se trebale ručno pisati svaki puta pri izradi web aplikacija.

9. Literatura

1. Luke Welling: PHP and MySQL Web Development
2. David Upton: CodeIgniter for Rapid PHP Application Development
3. Ellislab, Inc.; http://codeigniter.com/user_guide/
4. Thomas Myer: Professional CodeIgniter

10. Sažetak

U ovom seminarskom radu su obrađene osnove programskog jezika PHP, prikazan osnovni koncept MVC frameworka kao pomoć pri izradi PHP aplikacija, dan je popis dostupnih PHP MVC frameworka i napravljena njihova usporedba. Na kraju je detaljno opisan CodeIgniter PHP MVC framework, te prikazana izrada jednostavne implementacije u cilju prikaza jednostavnosti korištenja i učinkovitosti ovog frameworka.