Matjaž Kovačič

ANALIZA IN PRIMERJAVA PHP OGRODIJ NA PRIMERU IZDELAVE SPLETNEGA DNEVNIKA

Diplomsko delo

Maribor, september 2010

Diplomsko delo univerzitetnega strokovnega študijskega programa

ANALIZA IN PRIMERJAVA PHP OGRODIJ NA PRIMERU IZDELAVE SPLETNEGA DNEVNIKA

Študent:	Matjaž Kovačič
Študijski program:	UN ŠP Računalništvo in informacijske tehnologije
Mentor(ica):	Izred. prof. dr. Milan Ojsteršek

Maribor, september 2010

FERI
Fakulteta za elektrotehniko, računalništvo in informatiko
Smetanova ulica 17
2000 Maribor

Številka: BRIT-8
Datum in kraj: 12. 03. 2010, Maribor

Na osnovi 330. člena Statuta Univerze v Mariboru (Ur. l. RS, št. 90/2008)

SKLEP O DIPLOMSKEM DELU

1. **Matjažu Kovačiču,** študentu univerzitetnega študijskega programa Računalništvo in informacijske tehnologije se dovoljuje izdelati diplomsko delo pri predmetu Spletno programiranje.

2. MENTOR: **izred. prof. dr. Milan Ojsteršek**

3. **Naslov diplomskega dela:**
ANALIZA IN PRIMERJAVA PHP OGRODIJ NA PRIMERU IZDELAVE SPLETNEGA DNEVNIKA

4. **Naslov diplomskega dela v angleškem jeziku:**
ANALYSIS AND COMPARISON OF PHP FRAMEWORKS ON CASE OF DEVELOPING OF BLOG

5. Diplomsko delo je potrebno izdelati skladno z "Navodili za izdelavo diplomskega dela" in ga oddati v treh izvodih (en vezan izvod in dva nevezana izvoda) ter en izvod elektronske verzije do 12. 03. 2011 v referatu za študentske zadeve.

Pravni pouk: Zoper ta sklep je možna pritožba na senat članice v roku 3 delovnih dni.

Dekan:

UNIVERZA V MARIBORU FAKULTETA ZA ELEKTROTEHNIKO, RAČUNALNIŠTVO IN INFORMATIKO 8

Obvestiti:
- kandidata,
- mentorja,
- somentorja,
- odložiti v arhiv.

ZAHVALA

Zahvaljujem se mentorju za pomoč in vodenje pri opravljanju diplomskega dela.

Posebna zahvala velja staršem, ki so mi omogočili študij.

ANALIZA IN PRIMERJAVA PHP OGRODIJ NA PRIMERU IZDELAVE SPLETNEGA DNEVNIKA

Ključne besede: spletno programiranje, PHP ogrodja, Zend, CakePHP, Codeigniter, Symfony, Seagull

UDK: 004.45:004.738.5(043.2)

Povzetek

V diplomski nalogi smo predstavili funkcionalnosti in naredili primerjavo med petimi PHP ogrodiji: Zend, CakePHP, Codeigniter, Symfony in Seagull. Ogrodja smo uporabili pri izdelavi spletne aplikacije in jih primerjali glede na njihovo učinkovitost pri izdelavi spletnih aplikacij, zmoglivost in varnost spletnih aplikacij ter ustreznost njihove dokumentacije.

INSTRUCTIONS FOR PREPARING A DIPLOMA

Key words: web programming, PHP frameworks, Zend, CakePHP, Codeigniter, Symfony, Seagull

UDK: 004.45:004.738.5(043.2)

Abstract

In this thesis we presented the functionalities and made the comparison between the five PHP frameworks: Zend, CakePHP, Codeigniter, Symfony and Seagull.. Comparison is made regarding to the effectiveness in designing, performace and security of web applications and the adequacy of their documentation

VSEBINA

UPORABLJENE KRATICE

PHP	angl. »PHP: Hypertext Preprocessor«
MVC	angl. »Model View Controller«
CSS	angl. »Cascading Style Sheets«
CTP	angl. »Cake template page«
ACL	angl. »Access Control List«
LDAP	angl. »lightweight directory access protocol«

1 UVOD

PHP (angl. »PHP: Hypertext Preprocessor«) je skriptni jezik, namenjen izdelavi spletnih aplikacij. Uporablja ga velik del spletnih razvijalcev, saj se ga hitro naučijo, ter z lahkoto uporabljajo.

1.1 Področje raziskave

Kljub enostavnosti pa ima jezik PHP hibe, kot so visoka toleranca do napak, neurejenost kode, pomankjkanje podpore objektno orientiranega pristopa itd. Razvijalci včasih torej naredijo napako, ki se ne prikaže takoj, lahko pa povzroča velike težave v nadaljnjem razvoju spletne aplikacije. PHP ogrodja so produkt razvijalcev, ki so želeli poenostaviti in izboljšati proces razvoja spletnih aplikacij v skriptnem jeziku PHP. Pomagajo ustvariti čitljivo in organizirano programsko kodo, omogočajo različne module, predvsem pa poskrbijo da razvijalcem ni potrebno reševati istih problemov večkrat, saj kodo s pomočjo ogrodij zlahka ponovno uporabijo.

1.2 Namen raziskave

Namen raziskave je primerjava več različnih ogrodjih PHP, ter odgovoriti na vprašanje: »Katero izmed njih je najboljše?«. Ko izbiramo ogrodje, moramo upoštevati več kriterijev. Tokrat se bomo osredotočili na naslednje: enostavnost, uspešnost, varnost ter dokumentacija. Smisel raziskave je torej na kratko predstaviti izbrana ogrodja, ter vsakega oceniti po teh štirih kriterijih.

1.3 Struktura dela

V drugem poglavju se najprej seznamimo z osnovami PHP ogrodij, kako je PHP deloval nekoč, kako deluje danes ter na kak način nam ogrodja pomagajo pri razvoju aplikacij.

V tretjem poglavju je podan opis izdelave našega spletnega bloga, kakšno podatkovno bazo potrebujemo ter kako smo izdelali uporabniški vmesnik.

V četrtem poglavju so na kratko predstavljena izbrana PHP ogrodja: CodeIgniter, CakePHP, Zend, Seagull ter Symfony. Za vsakega je opisan potek namestitve, opis datotečne strukture ter kako v njem razvijamo aplikacije.

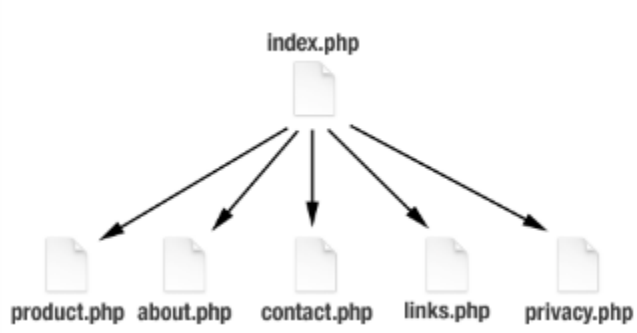
Peto poglavje je namenjeno primerjavi med ogrodji, testiranju vsakega po danih kriterijih, ter rezultati analize.

V šestem poglavju na kratko povzamemo glavne ugotovitve glede primerjave ogrodij.

2 OGRODJA PHP

2.1 Evolucija skriptjega jezika PHP

PHP spada med najpopularnejše skriptne jezike za izdelavo dinamičnih spletnih strani. Za skriptne jezike je značilno, da se interpretirajo in ne prevajajo. Pri prevajanju se celotna aplikacija prevede v izvršljivo datoteko, medtem ko se skriptni jeziki interpretirajo vrstico za vrstico. Obstaja še mnogo drugih skriptnih jezikov, vendar PHP še vedno ostaja največkrat uporabljen med vsemi. Programerje privlači predvsem zato, ker se z malo programskega dela hitro pride do rezultatov. Učenje je mnogo hitrejše v primerjavi z ostalimi jeziki. Delo z jezikom PHP pa se skozi čas spreminja. Nekoč smo za aplikacijo zgradili izhodno datoteko (večinoma imenovana index), iz katere so nas povezave vodile do drugih datotek (Slika 1). Ker je to bil prostorsko in logično neučinkovit pristop smo raje poenostavili stvari ter ustvarili le eno izhodno datoteko, v katero smo s pomočjo jezika PHP dinamično polnili vsebino, glede na zahtevo uporabnika (Slika 2).



Slika 1: Kako smo gradili spletne strani nekoč

Slika 2: Kako gradimo spletne strani danes

2.2 Kaj so PHP ogrodja

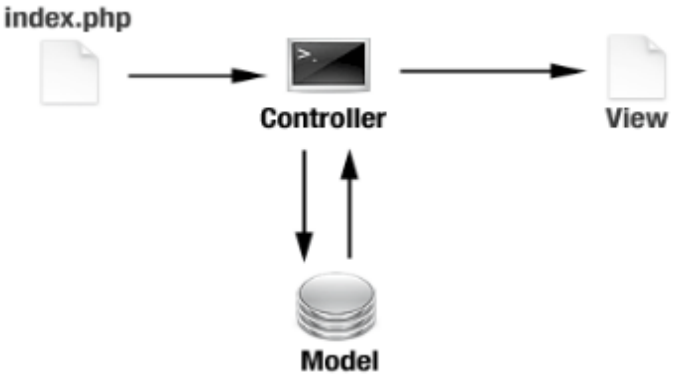
Še kako mikavno je napisati vsako spletno stran znova in znova od temeljev do vrha, vendar to ni vedno najboljša rešitev. Večkrat programiranje pri različnih projektih dolgočasno, saj je potrebno enake probleme reševati znova in znova. Tukaj nam pomagajo PHP ogrodja. Z njihovo pomočjo na enostaven način večkrat uporabimo kodo, ki predstavlja standardni problem pri različnih projektih (kot je npr. Avtentikacija) ter s tem omogočijo hitrejši razvoj. Poskrbijo tudi za osnovno strukturo projekta. S pomočjo te strukture znatno znižamo število napak in lažje zagotovimo stabilnost spletne strani, ter zmanjšamo količino ponavljajoče se kode. Prav tako poskrbijo za pravilno in enostavnejšo interakcijo s podatkovno bazo. S pomočjo vseh teh izboljšav tako programer lažje posveti več časa izdelavi spletne aplikacije in manj za kodiranje.

Ogrodja imajo tudi slabosti:

- pojavijo se lahko težave pri uporabi več ogrodij hkrati,
- potreben je daljši čas učenja ogrodja (daljši čas izdelave prve aplikacije),
- razvijalec je omejen pri razvoju spletne aplikacije zaradi omejenosti ogrodja,
- pojavijo se lahko morebitne težave s postavitvijo v produkcijsko okolje v primeru gostovanja na tujih strežnikih,
- spletna aplikacija je lahko varnostno ranljiva, saj je večina napak ogrodja javno objavljena, zato jih je možno izkoristiti za vdor v spletno aplikacijo.

2.3 Arhitektura MVC

Pri definiranju arhitekture spletne aplikacije želimo, da z uporabo PHP ogrodja modularno in strukturiramo razvijemo spletno aplikacijo. To najbolj učinkovito dosežemo z uporabo arhitekture MVC (angl. »Model View Controller«). MVC loči poslovno logiko od uporabniškega vmesnika. S tem dopušča, da vsak element posebej spreminjamo, brez da bi s tem vplivali na druge. Tako poenostavimo PHP kodiranje, ter pridobimo na času. Na sliki 3 lahko vidimo prikaz arhitekture MVC.



Slika 3: MVC arhitektura

V arhitekturi MVC imamo naslednje elemente:

- Kontroler (angl. »Controller«) – predstavlja poslovno logiko in povezuje ostale elemente med sabo. Sprejema vhodne podatke, jih procesira ali pošlje zahtevek na Model, ter jih vrne v pogled.
- Model – element za interakcijo s podatkovno bazo. Od kontrolerja sprejme zahtevke po podatkih, naredi poizvedbe, ter rezultate vrne nazaj v kontroler.
- Pogled (angl. »View«) – prejema podatke od kontrolerja in jih predstavi končnemu uporabniku. Gre torej za uporabniški vmesnik.

2.4 Zakaj uporabiti PHP ogrodja

Prvi večji razlog za uporabo PHP ogrodij je skrajšanje časa razvoja spletnih aplikacij. Pri izdelavi spletnih aplikacij se razvijalci vedno srečujejo z istimi problematikami. Nekaj teh problematik je:

- Zasnova uporabniškega vmesnika.
- Poslovna logika.
- Manipulacija podatkovne baze.
- Nadzor uporabniškega dostopa.

Drugi razlog pa je stabilnost izdelka. PHP ima enostavno sintakso, kar je razlog, da ga mnogi začetniki radi izbirajo za učenje. Hkrati pa je PHP zelo neobčutljiv na napake in mnogokrat se lahko zgodi, da razvijalec sploh ne ve, da je napravil napako, saj bo spletna stran vseeno delovala. V najslabšem primeru lahko ogrozimo varnost naše spletne strani. Enostavnost kodiranja v PHPju je torej hkrati ena večjih problematik, katera pa se pojavlja manjkrat ob uporabi primernega PHP ogrodja. K stabilnosti prispeva tudi dejstvo, da ogrodje razvijajo za to posvečeni razvijalci, ki spremljajo odkrite napake in skrbijo za popravke ogrodja. To velja še posebej za odprtokoda ogrodja, kjer lahko za popravek poskrbi kar skupnost, ki ga razvijalci ogrodja pregledajo in uveljavijo. Razvijalcu spletne aplikacije (ki uporablja ogrodje) tako ni potrebno skrbeti za napake v ogrodju, ampak le uveljavi popravek ogrodja, ko je ta na voljo. Na ta način lahko na enostaven način odpravi napake v vseh aplikacijah, ki to ogrodje uporabljajo.

Kako izbrati PHP ogrodje

Najprej se moramo seveda vprašati ali ogrodje sploh potrebujemo. Za začetek moramo vedno analizirati projekt, na katerem bomo delali. Potem si postavimo naslednja vprašanja:

- Ali nam bo ogrodje prihranilo čas?
- Ali nam bo ogrodje prihranilo trud?
- Ali bo zaradi ogrodja naša aplikacija bolje varovana?

Če na vsa ta vprašanja odgovorimo pritrdilno, potem je PHP ogrodje prava rešitev za nas. Težko je oceniti katero izmed mnogih obstoječih ogrodij je najboljše, saj na odločitev vpliva mnogo dejavnikov. Zelo je pomembno, kako močno skupnost ima izbrano ogrodje. Več pripadnikov ogrodja namreč pomeni živeče ogrodje, ki se stalno izboljšuje in spreminja. Pomembna je enostavnost grajenja, vsebovani moduli itd. V tej raziskavi bomo testirali pet najpogosteje uporabljenih PHP ogrodij in poskušali poiskati najprimernejšega za izdelavo enostavnega spletnega dnevnika.

3 IZDELAVA SPLETNEGA DNEVNIKA

3.1 Izdelava podatkovne baze

Za enostavni spletni dnevnik potrebujemo vsaj tri tabele v podatkovni bazi. V prvo tabelo hranimo prispevke, v drugo komentarje ter v tretjo uporabnike. Za izdelavo podatkovne baze smo uporabili podatkovno bazo MySQL, saj jo podpirajo prav vsa ogrodja v katerih bomo delali. Ime tabel ter polj v tabelah se lahko razlikuje od tistih v spodnjih tabelah, odvisno od zahtev različnih PHP ogrodij.

Tabele v podatkovni bazi so naslednje:

Tabela 1: Tabela za prispevke

Prispevki	
Polje	**Podatkovni tip**
#ID_prispevka	Integer(7)
Naslov	Varchar(1000)
Datum	Date
Tekst	Text
Avtor	Varchar(100)

Tabela 2: Tabela za komentarje

Komentarji	
Polje	**Podatkovni tip**
#ID_komentarja	Integer(7)
ID_prispevka	Integer(7)
Avtor	Varchar(20)
Komentar	Text
Datum	Date

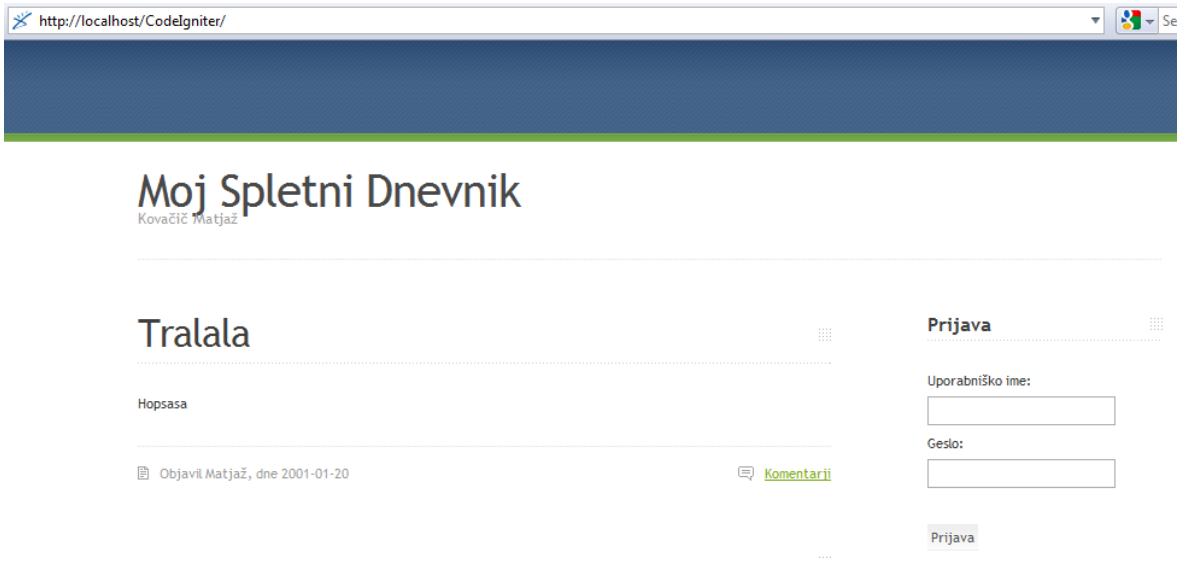
Tabela 3: Tabela za uporabnike

Uporabniki	
Polje	**Podatkovni tip**
#ID	Integer(11)
Username	Varchar(100)
Password	Varchar(255)

3.2 Izdelava Spletnega vmesnika

Osnovni izgled

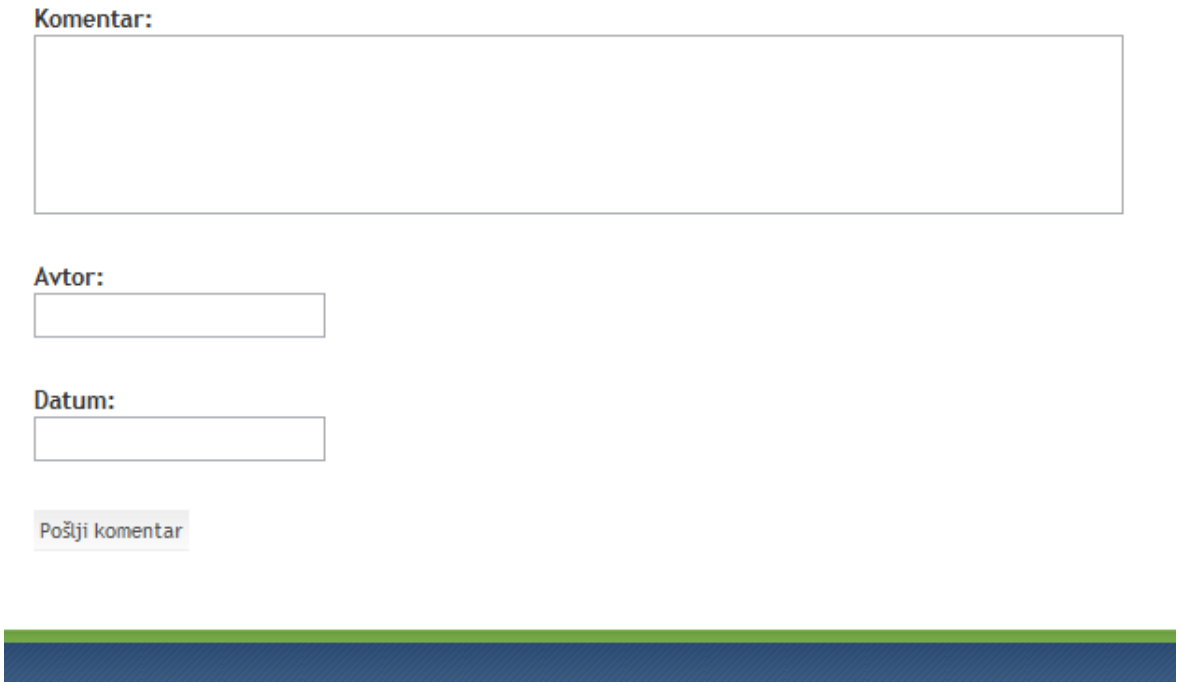
Osnovni izgled je izdelan s pomočjo označevalnega jezika HTML, za obliko poskrbimo s CSS (angl. »Cascading Style Sheets«). Kot kaže slika 4, stran vsebuje le osnovne elemente HTML, ki so potrebni za izpis in vpis podatkov (kot so div, polja za besedilo, tabele).



Slika 4: Izgled prve strani dnevnika

3.3 Funkcionalnosti

Na uvodni strani so v glavnem odseku prikazani naši vnosi v spletnem dnevniku. Ob kliku na povezavo »Komentarji« nas usmeri na novo stran, na kateri se izpiše izbrani vnos v dnevniku, pod njim pa vsi komentarji bralcev. Na dnu lahko vsi bralci dnevnika objavljajo komentarje s pomočjo vnosnega polja (Slika 5). Na desni strani uvodne strani je obrazec za prijavo, po uspešni prijavi pa se izpiše pozdravno sporočilo in gumb za odjavo. Prijavljen uporabnik lahko dodaja nove prispevke v dnevnik.



Slika 5: Izgled vnosnega polja za dodajanje komentarjev

4 PREDSTAVITEV OGRODIJ

Vsa ogrodja imajo eno skupno lastnost in to so zahteve za delovanje. Za vseh pet ogrodij potrebujemo naslednje:

- Spletni strežnik (vsa ogrodja delujejo na strežniku Apache, Zend pa je edini, ki ponuja svoj spletni strežnik.).
- Nameščen PHP jezik (zahtevana minimalna verzija PHP jezika varira od ogrodja do ogrodja).
- MySQL podatkovna baza (vsa ogrodja imajo podporo tudi za druge podatkovne baze.).

Vseh pet ogrodij uporablja MVC arhitekturo, zato je datotečna struktura pri vseh zelo podobna.

4.1 CodeIgniter

CodeIgniter je relativno mlado ogrodje, ustvarjeno za vse PHP razvijalce, ki si želijo enostavnega in hitrega ogrodja za izdelavo spletnih aplikacij. Ponuja preproste rešitve, obsežno knjižnico video vodičev, forumov za podporo, priročnikov za učenje ter močno bazo uporabnikov. To ogrodje je priporočljivo za začetnike, kakor tudi za izkušene razvijalce. Za delovanje CodeIgniter potrebuje PHP verzije 4.3.2.

Postopek namestitve je naslednji:

- Iz uradne spletne strani[1] CodeIgniter prenesemo zadnjo verzijo ogrodja na naš strežnik.
- V datoteki »application/config/database.php« nastavimo podatke o dostopu do naše podatkovne baze. Nastavimo uporabniško ime, geslo ter ime baze, do katere želimo dostopati.
- S tem je namestiteva končana. Ko poženemo naš spletni strežnik, CodeIgniter deluje.

Datotečna struktura ogrodja:

Ogrodje vsebuje več imenikov, za izdelavo spletnega dnevnika se je potrebno osredotočiti le na nekatere pomembnejše. Ti so :

- V imeniku »system/application« se nahaja naša aplikacija.
    - V »system/application/config« so kofiguracijske datoteke, v katerih nastavljamo privzete naslove URL, podatke o podatkovni bazi, idr.
    - V »system/application/controllers« je mesto, kjer ustvarjamo naše kontrolerje, ki predstavljajo poslovno logiko.
    - V »system/application/views« je mesto, kamor shranimo datoteke z uporabniškim vmesnikom.
    - V »system/application/models« je mesto, kjer ustvarjamo modele, s pomočjo katerih beremo podatke iz podatkovne baze.

Kontrolerji:

Datoteke, ki vsebujejo razred, morajo torej biti v mapi »controllers«. Ime datoteke mora vedno imeti malo začetnico, ime razreda veliko začetnico, imena pa morata biti enaka.

Primer za datoteko: **www.moja-stran.com/blog**

```
<?php
class Blog extends Controller
{
    function Blog(){
        parent::Controller();
    }
    function index($lokacija){
        echo »Pozdravljen «.$lokacija;
    }
?>
```

Metoda »*Blog*« predstavlja konstruktor razreda, index pa je metoda, ki se zažene po prevzetem, razen če je v URLju drugačen segment.

Če kličemo: **www.moja-stran.com/blog/index/svet, nam bo metoda index vrnila rezultat »Pozdravljen svet«.**

Pogledi:

Imenik »Views« vsebuje PHP datoteke s pogledi oz. uporabniškim vmesnikom. Teh datotek nikoli ne kličemo direktno, ampak s pomočjo kontrolerjev.

Primer klica: **$this->load->view('blog');**

Dinamične podatke pogledom pošiljamo tako, da dodamo še dodaten segment klicu.

Primer:

```
$lokacija = 'Svet';
$this->load->view('blog',  $lokacija);
```

Modeli:

Imenik »models« vsebuje metode za delo s podatkovno bazo. Ko podatke pridobimo iz podatkovne baze, jih pošljmoe kontrolerju, ta pa pogledu.

Primer:

```
function vrni_prispevke() {
$query = $this->db->get('prispevki');
Return $query->result();
}
```

Knjižnice:

Knjižnice se nahajajo v »system/libraries« in so razredi za posebne namene. Če jih želimo uporabljati jih moramo najprej naložiti v kontroler.

Primer: **$this->load->library('calendar');**

S pomočjo te knjižnice lahko kasneje ustvarimo koledar. Ustvarjamo lahko tudi svoje knjižnice, jih shranimo v mapo »system/libraries«, ter jih kličemo iz kontrolerjev. V primeru spletnega dnevnika je bilo potrebno ustvariti knjižnico »avtentikacija«, saj je CodeIgniter po privzetem ne vsebuje.

Pomočniki:

Pomočniki (angl »Helpers«) so pomagači, ki nam olajšujejo kodiranje v PHP-ju. V kontrolerjih jih moramo naložiti enako kot knjižnice. Pomočnike uporabimo, če del naše programske kode potrebujemo na več mestih ali pa želimo doseči boljšo organizacijo in s tem preglednost naše programske kode.

Primer: **$this->load->view('form');**

4.2 CakePHP

CakePHP nam ponuja razširljivo arhitekturo za razvoj ter vzdrževanje. Enako kot CodeIgniter tudi CakePHP temelji na arhitekturi MVC, podpira pa tudi ORM (angl. »Object Relational Mapping«). ORM poenostavi delo s podatkovnimi bazami. Odnosi med tabelami so tako shranjeni v CakePHPju in s pomočjo njegovih funkcionalnosti lahko zelo enostavno beremo, pišemo, spreminjamo ter brišemo podatke v bazi.

Postopek namestitve je naslednji

- Iz uradne spletne strani[2] CakePHP prenesemo zadnjo verzijo ogrodja na naš spletni stražnik.
- V datoteki »app/config/database.php« nastavimo podatke o dostopu do naše podatkovne baze. Nastavimo uporabniško ime, geslo ter ime baze, do katere želimo dostopati.
- S tem je namestitev končana. Ko poženemo naš spletni strežnik, CakePHP deluje.

Datotečna struktura ogrodja

Podobno kot pri CodeIgniterju je za nas pomemben le majhen del imenikov:

- V »app« direktoriju se skriva jedro naše aplikacije:
    - V »app/config« se nahajajo konfiguracijske datoteke.
    - V »app/controllers« se nahajajo kontrolerji oziroma poslovna logika našega projekta.
    - V »app/models« se nahajajo modeli, ki jih uporabljamo za interakcijo z bazo.
    - V »app/views« se nahajajo pogledi oziroma naš uporabniški vmesnik.

Podatkovna baza:

Pri delu s CakePHP je imenovanje datotek ter tabel v bazi zelo pomembno. Vse tabele v naši bazi morajo imeti ime v množini. Seveda je zato bolje le-te poimenovati v angleščini. Npr. našo tabelo »prispevki« preimenujemo v »posts«.

Kontrolerji:

Kontrolerji se uporabljajo za upravljanje poslovne logike. Sodijo v mapo »app/controllers« in njihova imena so sestavljena iz imena tabele v podatkovni bazi, ter »_controller«. Uporablja se končnica php. Npr. v primeru tabele »posts« se kontroler imenuje »posts_controller.php«.

Primer kontrolerja:

```
<?php
class PostsController extends AppController {
    function hello_world(){
    }
}
?>
```

Ime razreda v kontrolerju imenujemo z imenom tabele, ki se začne z veliko začetnico in »Controller«.

Modeli:

Modeli se uporabljajo za interakcijo s tabelami v podatkovni bazi. Imena modelov vedno imenujemo v edninski različici imena tabel. Če je torej ime naše tabele »posts«, se mora model imenovati »post.php«. Ime razreda v modelu pa imenujemo isto kot ime modela, le da z veliko začetnico.

Primer:

```
$this->posts->findByTitle('Moj prvi vnos');
```

Pogledi:

Za poglede moramo ustvariti novo podmapo, ki jo imenujemo po kontrolerju. Npr. če je ime kontrolerja »posts_controller« mapo za poglede poimenujemo »posts«.V podmapi ustvarimo datoteko, katere ime je enaka kot ime metode ustvarjene v kontrolerju. Končnica te datoteke je ctp (angl. »Cake template page«).

4.3 Seagull

Seagull je objektno orientirano PHP ogrodje. Mnoge popularne PHP aplikacije so že integrirane v projektu, kot so razne predloge, pripomočki za testiranje in knjižnice. Za začetnike je na voljo veliko vzorčnih aplikacij, s katerimi se lažje učijo.

Namestitev Seagulla je izjemno enostavna:

- Iz uradne spletne strani[3] snamemo zadnjo verzijo Seagulla ter jo shranimo na naš spletni strežnik.
- Sledimo navodilom za namestitev.
- Seagull sam namesti prijazen uporabniški vmesnik ter podatkovno bazo.

Datotečna struktura ogrodja

Seagull deluje na principu MVC kot ostala ogrodja, vendar ima direktorije poimenovane nekoliko drugače, kot ostala ogrodja. Del, ki nas zanima za izdelavo enostavne aplikacije se nahaja v mapi »seagull/modules«. V njej ustvarimo novo mapo, ki jo poimenujemo poljubno oziroma primerno našemu projektu (npr. »HelloWorld«). V tej mapi ustvarimo dodatne tri mape in jih imenujemo: »classes«, »data« ter »templates«. V te shranjujemo kontrolerje, module ter poglede.

Kontrolerji:

Ko ustvarimo datotečno strukturo, lahko začnemo ustvarjati naše kontrolerje. Ime kontrolerja mora biti enako kot ime razreda v njem. Napišimo primer, ki vsebuje metodo, ki pokliče pogled helloWorld.html, ki ga bomo ustvarili v mapi »templates«, ter mu pošlje spremenljivko:

```
<?php
class HellowWorldMgr extends SGL_Manager{
    function display(){
        $output->template = 'helloWorld.html';
        $output->testnaSpremeljivka = 'Hello World!'«;
    }
}
?>
```

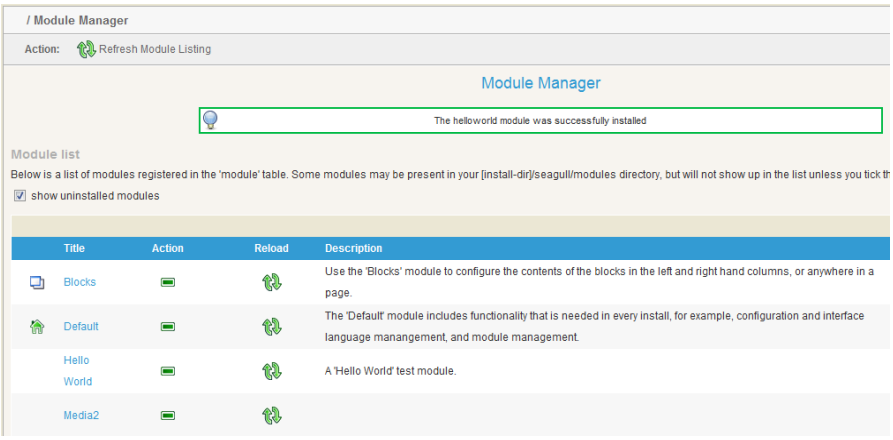
Pogledi:

Pogledi se nahajajo v mapi »templates«. Spremenljivke, ki nam jih pošlje kontroler moramo vedno uporabiti z zavitimi oklepaji. Primer:

<h1> {testnaSpremenljivka} </h1>

Registracija modula:

Vsak novo ustvarjen modul moramo registrirati v uporabniškem vmesniku ogrodja (Slika 6), da ga le-to prepozna.



Slika 6: Upravljanje z moduli

4.4 Zend

Zend se razlikuje od ostalih predvsem zaradi tega, ker ponuja svoj spletni strežnik. S pomočjo le-tega je namestitev ter delo v ogrodju Zend zelo olajšano.

Namestitev ogordja Zend poteka drugače, če se odločimo uporabljati tudi njihov brezplačni spletni strežnik, ki ga najdemo na uradni spletni strani[4], ali pa če uporabljamo svojega. Za oba primera pa imamo na uradni strani dovolj dokumentacije. Namestitev se v obeh primerih razlikuje od ostalih ogrodij, saj je potrebno poganjati namestitvene datoteke iz ukazne vrstice.

Datotečna struktura ogrodja

Ko uspešno končamo namestitev ogrodja, se v izbrani projektni mapi ustvari struktura datotek. Za nas so najpomembnejše datoteke v imeniku »application«. Tam se nahajajo »configs«, »controllers«, »models« ter »views«. Vanje shranjujemo datoteke, s katerimi gradimo naš projekt z uporabo MVC arhitekture.

Kontrolerji:

Kontrolerje shranimo v imenik »application/controllers«. Imena se vedno končajo z »Controller«. Razredi, ustvarjeni v kontrolerjih, morajo imeti enako ime kot kontrolerji, metode pa se končajo z »Action«. Primer kontrolerja IndexController.php:

```
<?php
class IndexController extends Zend_Controller_Action {
    public function indexAction(){
    }
}
?>
```

Pogledi:

Pogledi se nahajajo v mapi »application/view/scripts«. Imenujejo se po akcijah, ki smo jih ustvarili v kontrolerju. Kot končnico datoteke uporabljajo phtml.

Postavitve:

Zend nam omogoča konsistentne postavitve za vse strani oziroma poglede. Za to moramo postavitve najprej omogočiti. Ko so postavitve omogočene, se v imeniku »application« ustvari nova mapa, rezervirana za postavitve. Gradimo jih podobno kot poglede ter z enako končnico phtml. S tem dosežemo enotno postavitev za vse strani.

Modeli:

Modeli se uporabljajo za interakcijo s podatkovno bazo. S pomočjo ukaza v ukazni vrstici hkrati ustvarimo podatkovno bazo, v njej tabele ter podmape v imeniku »models«. V podmapah se odvijajo vse operacije, povezane s podatkovno bazo, ki so: pisanje, branje, spreminjanje in brisanje.

Ostale knjižnice:

Na voljo je še mnogo ostalih knjižnic, ki nam pomagajo pri delu s elementi HTML. Naštejmo jih nekaj:

- Zend_Date (za manipulacijo časa in datuma),
- Zend_Form (za ustvarjanje form),
- Zend_Navigation (za ustvarjanje menijev, linkov itd.).

4.5 Symfony

Symfony je ogrodje, ki omogoča gradnjo kompleksnih spletnih aplikacij hitreje in z manj truda. Olajša nam gostovanje, nadgradnjo in nadzor uporabnikov naše aplikacije. Temelji na tehnologiji vtičnikov(angl. »Plugin«). Uporablja jo mnogo naprednih razvijalcev, zato ima veliko skupnost uporabnikov in obsežno zbirko vodičev, vtičnikov in brezplačnih aplikacij.

Namestitev za Symfony je malce bolj komplicirana od ostalih ogrodij in zahteva boljše poznavanje spletnih strežnikov. Na uradni spletni strani [5] so navodila za namestitev korak za korakom. Iz strani snamemo zadnjo verzijo ogrodja ter sledimo navodilom. Ko uspešno končamo z namestitvijo in ustvarimo projekt, se nam postavi osnovna datotečna struktura.

Datotečna struktura ogrodja

Po namestitvi in uspešno ustvarjenemu projektu se nam zgradi datotečna struktura, ki temelji na arhitekturi MVC. Symfony temelji na logiki javni del – skrbniški del (angl. »frontend-backend«), kar pomeni, da je za vsakega izmed njiju ustvarjen posebni imenik. »Frontend« oz. javni del je namenjen anonimnim uporabnikom, ki želijo le obiskati spletno stran, »backend« oz. skrbniški del pa je namenjen razvijalcem.

Razvoj aplikacij poteka nekoliko drugače, kot v ostalih ogrodjih. Čeprav tudi Symfony temelji na arhitekturi MVC, je ustvarjanje direktorijev bolj zapleteno in sprotno. Ko uspešno ustvarimo aplikacijo, je naslednji korak ustvariti javni del, ki je namenjen izpisu podatkov za končnega uporabnika. Ko javni del ustvarimo, nam Symfony ustvari še datotečno strukturo, kjer nato ustvarjamo module po sistemu arhitekture MVC.

5 ANALIZA IN PRIMERJAVA OGRODIJ

Pri tolikšnem številu ogrodij se seveda poraja vprašanje, katerega izbrati. Ker je na to vprašanje vse prej kot lahko odgovoriti, moramo upoštevati mnoge različne faktorje, funkcionalnosti in tako ugotoviti kateri je najprimernejši za nas.

Tabela 4: Primerjalna tabela lastnosti ogrodij[6].

	CakePHP	CodeIgniter	Zend	Seagull	Symfony
PHP 4	✔	✔	✘	✔	✘
PHP 5	✔	✔	✔	✔	✔
MVC	✔	✔	✔	✔	✔
Moduli	✔	✔	✔	✔	✔
Avtentikacija	✔	✘	✔	✔	✔
AJAX	✔	✘	✔	✔	✔
Več pod. baz	✔	✔	✔	✔	✔
Predloge	✘	✔	✘	✔	✘

Ogrodja bomo ocenjevali po naslednjih kriterijih:

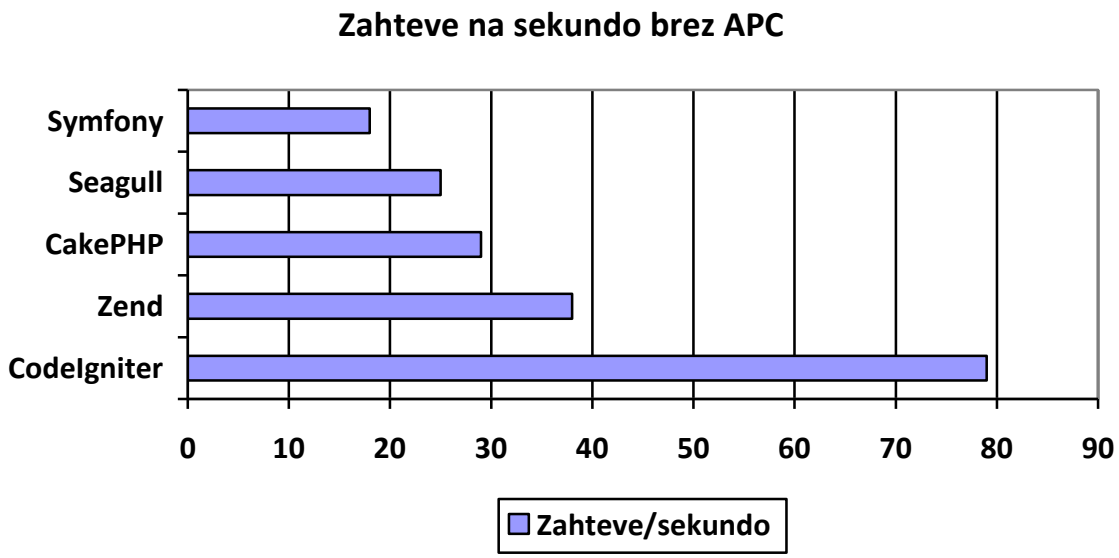
- Zmogljivost
- Varnost
- Enostavnost
- Dokumentacija

Vsako ogrodje bomo točkovali po vseh štirih kriterijih. Na koncu bomo sestavili metriko ter sešteli katero ogrodje zbralo največ točk.

5.1 Zmogljivost

Pri ocenjevanju uspešnosti različnih ogrodij nas zanima predvsem, koliko zahtev na sekundo lahko obdela vsako od ogrodij. Čeprav ima naprimer Zend svojo platformo, se rezultati ne razlikujejo dosti. Za primerjavo smo izdelali enostavno »Hello World« aplikacijo in jo pognali v vseh ogrodjih. Za pravično merjenje nismo uporabili podatkovne baze, izklopili vse pomočnike za razhroščevanje ipd. Meritve so bile opravljene z ApacheBench orodjem, ki je namenjeno ugotavljanja zmogljivosti aplikacije na spletnem strežniku. Vsako aplikacijo smo poganjali 30 sekund ter na koncu izpisali rezultate meritve.

Vsako aplikacijo smo pognali z in brez PHP pospeševalca APC (Alternative PHP Cache).

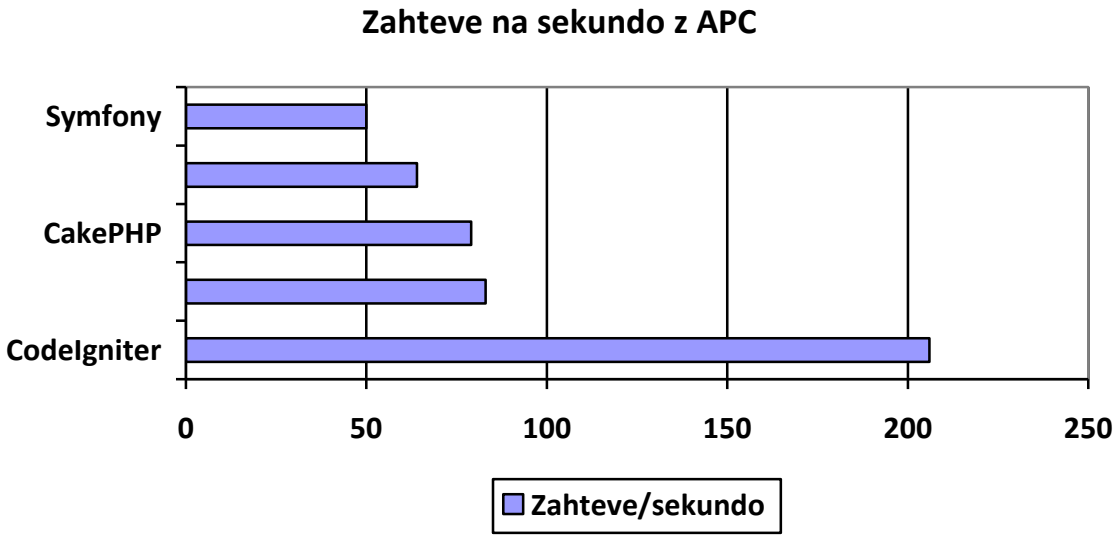


Slika 7: Rezultati meritve števila zahtev na sekundo brez APC (več je boljše).

Tabela 5: Rezultat meritev brez APC:

	CodeIgniter	**Zend**	**CakePHP**	**Seagull**	**Symfony**
Zahtevki/sekundo	79	38	29	25	18

V hitrosti procesiranja oziroma uspešnosti se od izbranih orodij najbolje odreže CodeIgniter, kot vidimo v sliki 7. Zanimajo nas še rezultati, če vključimo PHP pospeševalec APC.

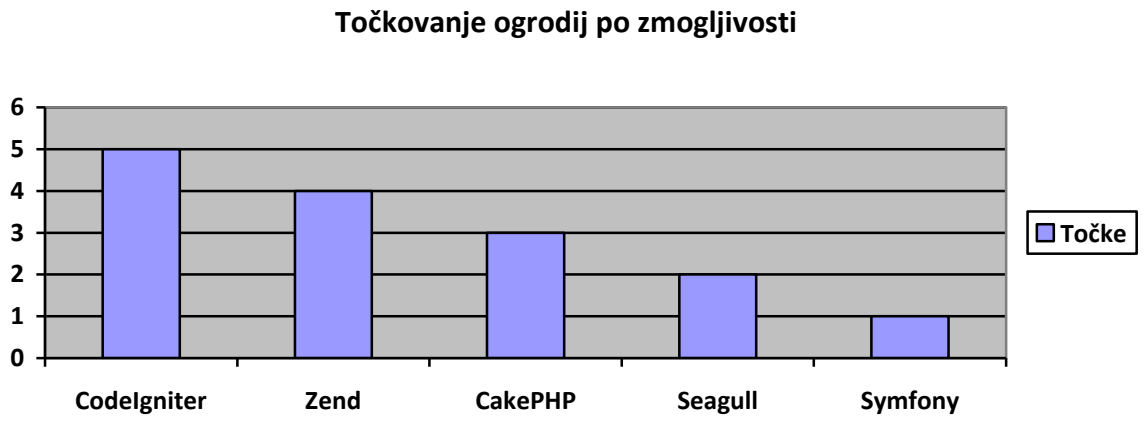


Slika 8: Rezultati meritve števila zahtev na sekundo z APC (več je boljše).

Tabela 6: Rezultati meritev z APC:

	CodeIgniter	Zend	CakePHP	Seagull	Symfony
Zahtevki/sekundo	206	83	79	64	50

Tudi z vključenim pospeševalcem se je najbolje odrezal CodeIgniter, kot vidimo na sliki 8. Ogrodja smo pri zmogljivosti točkovali z oceno od 1 do 5 (Slika 9).



Slika 9: Točkovanje ogrodij po zmogljivosti.

5.2 Varnost

Varnost je zelo težko ocenjevati, saj obstaja mnogo različnih načinov za ogrozitev varnosti spletne aplikacije. Pomembno je predvsem, da imajo ogrodja vgrajene sisteme za kontrolo dostopa ter avtentikacijo.

CakePHP:

CakePHP vsebuje oboje; kontrolo dostopa in pomočnika pri avtentikaciji. S pomočjo generatorja »cake bake« nam omogoča generiranje seznama za preverjanje dostopa ACL (angl. »Access Control List«). Postopek izvedbe je sicer zapleten, a imamo za to primerno dokumentacijo na uradni spletni strani. CakePHP sam zakriptira gesla.

CodeIgniter:

Varnost je ena izmed največjih pomankljivosti v ogrodju CodeIgniter. Ogrodje namreč ne vsebuje niti pomočnika za avtentikacijo, niti kontrole dostopa. Knjižnico sicer lahko implementiramo sami, vendar s tem porabimo dosti časa in truda.

Zend:

Zend prinaša množico modulov za avtentikacijo. Nekateri izmed njih so:

- Database Auth – avtentikacija s pomočjo poadtkovne baze.
- LDAP Auth – avtentikacija preko imeniških storitev po standardu LDAP (ngl »lightweight directory access protocol«).
- Open ID Auth – avtentikacija preko oddaljenih strežnikov OpenID, tako da geslo sploh ne pride do naše aplikacije.

S pomočjo teh modulov dodatno zavarujemo vatnost uporabnikov ter omogočamo avtentikacijo prek oddaljenih OpenID strežnikov.

Seagull:

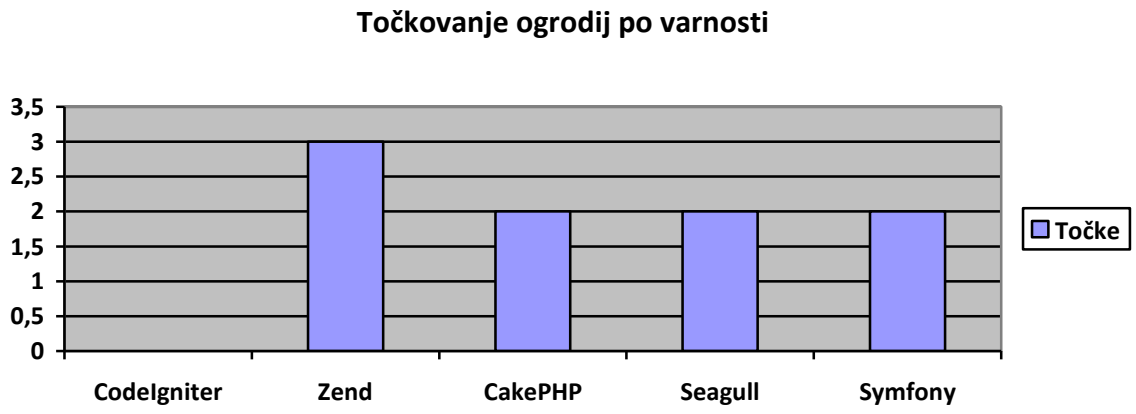
Tudi ogrodje Seagull vsebuje module za pomoč pri avtentikaciji. Uporabniki se ločujejo v osnovnem načinu na anonimne ter registrirane. Ko se uporabnik registrira, lahko ogrodje naloži omejitve do podatkov na strani s pomočjo njegove avtorizacijske sheme, ki temelji na vlogah uporabnikov. To shemo lahko konfiguriramo tudi po svoje. Za prihodnost napovedujejo še podporo drugih sistemom kot LDAP, POP3 itd.

Symfony:

Pri ogrodju Symfony si je za pomočnike pri avtentikaciji in avtorizaciji potrebno prenesti vtič iz uradne strani. Imenuje se sfGuardPlugin. Ta vtič poskrbi za avtentikacijo uporabnikov in omogoča nadzor dostopa. Deluje zelo enostavno in je namenjeno za osnovno avtentikacijo. Napredni razvijalci ga lahko razširijo in prilagodijo po svojih potrebah.

Tabela 7: Primerjalna tabela varnosti ogrodij

	CakePHP	CodeIgniter	Zend	Seagull	Symfony
Avtentikacija	✓	✗	✓	✓	✓
Avtorizacija	✓	✗	✓	✓	✓
Dodatni moduli	✗	✗	✓	✗	✗



Slika 10: Točkovanje ogrodij po varnosti.

Najvarnejšo ogrodje torej Zend s tremi točkami (Slika 10).

5.3 Enostavnost

Enostavnost je kriterij, ki ga je najtežje objektivno ocenjevati. Želimo oceniti kompleksnost namestitve ogrodij, čas za učenje ter za razvoj preprostih aplikacij v njih, zato je ta kriterij prav tako pomemben kot vsi ostali. Hitrost učenja novega ogrodja zavisi od predhodnjega poznavanja arhitekture ogrodja (v tem primeru MVC), jezika PHP ter nasploh principov delovanja ogrodij. Konvencija imen je prisotna v vseh ogrodjih in lahko na začetku povzroča nekaj težav, saj pri nekaterih ogrodjih vključuje vse od podatkovne baze, kontrolerjev, modelov idr.

CakePHP:

Namestitev: Precej enostavna. Iz spletne strani snamemo datoteke na spletni strežnik, konfiguriramo nekaj nastavitev in ogrodje je pripravljeno.

Uporaba: Ogrodje je precej enostavno za uporabo, večina direktorijev je razporejena logično. Po privzetem uporablja predlogo za izdelavo strani, ki jo je potrebno spremeniti. Učenje je enostavno saj na spletni strani najdemo veliko vodičev (angl. »tutorial«), video vodičev in primerov.

CodeIgniter:

Namestitev: Trivialna. Iz spletne strani snamemo datoteke na spletni strežnik, konfiguriramo le podatkovno bazo in ogrodje je pripravljeno.

Uporaba: CodeIgniter je enostaven, ima logično razporejene direktorije, zato zahteva minimalno porabljenega časa za izdelavo enostavne spletne aplikacije. Enako kot CakePHP ponuja na spletni strani dosti vodičev za začetnike, ter primere enostavnih aplikacij.

Zend:

Namestitev: Je lahko zelo mučna, čeprav je dobro dokumentirana na spletni strani. Če ne želimo uporabljati Zendovega spletnega strežnika, moramo imeti dobro poznavanje našega sistema. Namestitev je najlažja, če si pomagamo z uporabniškimi nasveti in komentarji na uradni spletni strani.

Uporaba: Precej enostavna. Direktoriji so logično ustvarjeni in razporejeni. Razvoj enostavne aplikacije ne traja pretirano dolgo, razpoložljivi moduli pa so dobro dokumentirani na spletni strani. Dokumentacija za razliko od drugih ogrodij omogoča uporabniške komentarje.

Seagull:

Namestitev: Izjemno enostavna. Iz spletne strani snamemo datoteke na spletni strežnik, ter poženemo namestitveni program. Sledimo navodilom, sistem za nas ustvari direktorije, podatkovno bazo, uporabniški račun in vse ostalo kar je potrebno za delovanje ogrodja.

Uporaba: Postavljena je logično razporejena arhitektura MVC z malce drugačnimi imeni, kar na začetku povzroča malo zmede, če s to arhitekturo ne delaš prvič. Tam ustvarjamo module, ki jih moramo kasneje omogočiti preko prijaznega uporabniškega vmesnika. Učenje uporabe je malce počasnejše zaradi zmedene in obširne dokumentacije.

Symfony:

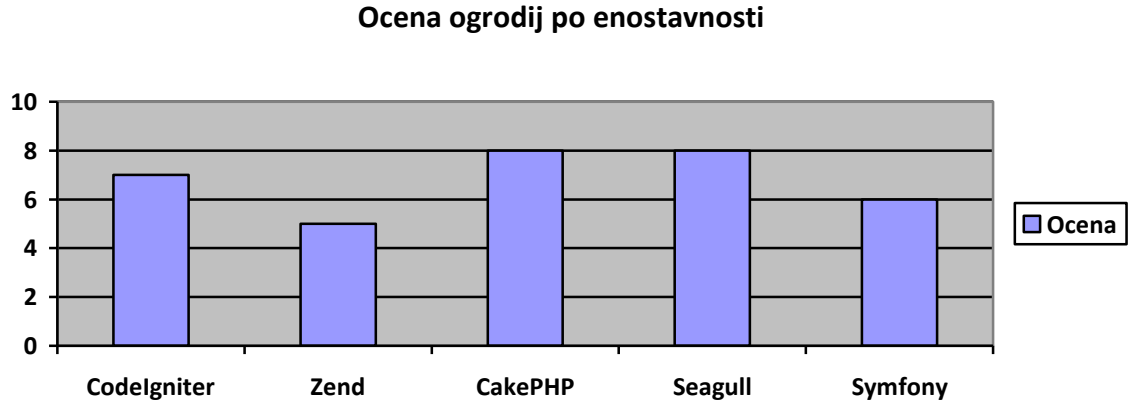
Namestitev: Je zapletena, saj je dokumentacija zelo nejasna. Zelo je podobna namestitvi ogrodja Zend, saj s pomočjo ukazne vrstice ustvarjamo nove mape, podmape in izvršujemo posebne ukaze za izdelavo baze.

Uporaba: Zelo dobro zamišljena arhitektura temelji na logiki javni del – skrbniški del, ki pa je vse prej kot enostavna za začetnike na področju ogrodij. Ogrodje je razširljivo z mnogimi vtiči, ki jih snamemo iz uradne spletne strani.

Ogrodja bomo točkovali od 1 do 3 v vsaki od kategorij.

Tabela 8: Tabela točkovanja po enostavnosti ogrodij (več je boljše)

	CakePHP	CodeIgniter	Zend	Seagull	Symfony
Namestitev	**2**	**2**	**1**	**3**	**1**
Učenje	**3**	**3**	**2**	**3**	**2**
Razvoj enostavne aplikacije	**3**	**2**	**2**	**2**	**3**



Slika 11: Točkovanje ogrodij po enostavnosti.

Kot kaže slika 11 sta se po kriteriju enostavnosti najvišje uvrstila CakePHP ter Seagull.

5.4 Dokumentacija

CakePHP:

Dobra, zaključena in obširna. Na uradni strani najdemo primere, vodiče za začetnike, ter video vodiče o ogrodja in izgradnji enostavnih aplikacij. Na spletni strani je abecedno razporejena dokumentacija, v kateri najdemo pomoč za vsak modul.

CodeIgniter:

Odlična. Na uradni strani sta poudarjena dva video vodiča, ki dodobra razložita delovanje tega ogrodja. Za nadaljne učenje je na voljo logična in zaporedno napisana dokumentacija s primeri ter obrazložitvami.

Zend:

Obširna, na spletni strani najdemo kratek vodič za začetnike, ter obširno abecedno urejeno dokumentacijo za module. Na voljo so tudi uporabniški komentarji, kar v nekaterih primerih pride zelo prav. Manjkajo le video vodiči za začetnike.

Seagull:

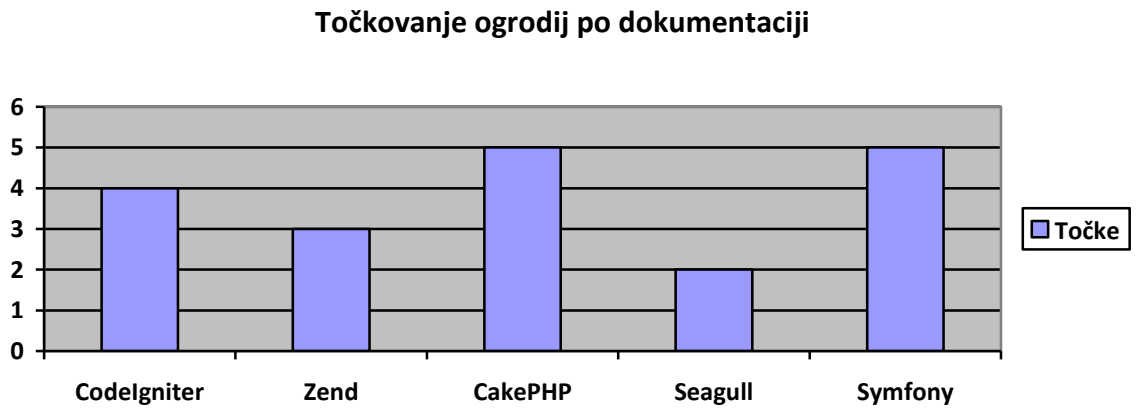
Dobra navodila za namestitev, ter dokumentacija za vse razrede, metode itd., ki pa so na žalost ponekod nejasna. Video vodiči niso na voljo.

Symfony:

Zelo obširna, abecedno urejena, dobro je opisan vsak vtič, ki ga lahko snamemo in z njim razširimo ogrodje. Na uradni strani so še povezave do knjig z dodatno dokumentacijo.

Tabela 9: Primerjalna tabela dokumentacij ogrodij.

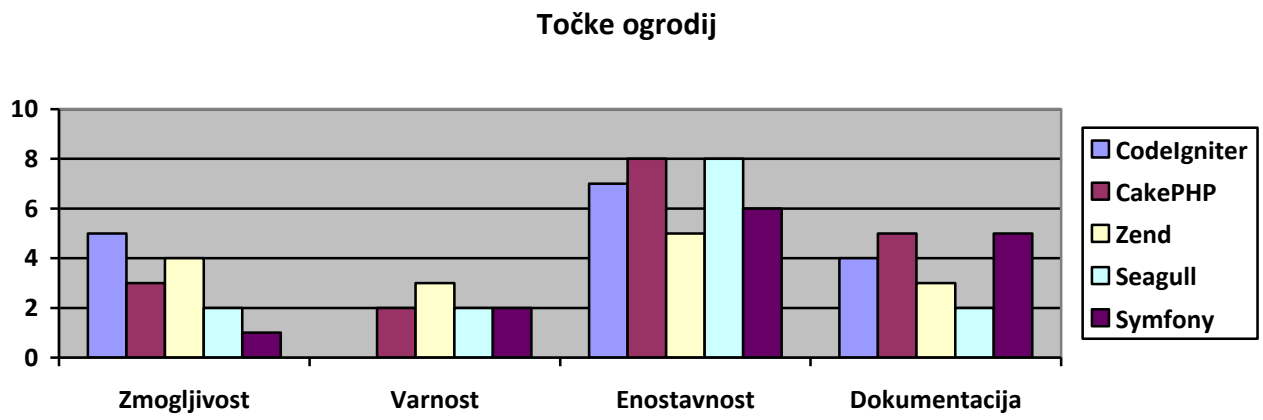
	CakePHP	CodeIgniter	Zend	Seagull	Symfony
Abecedni razpored	✓	✗	✓	✓	✓
Iskalnik po dokumentaciji	✓	✓	✗	✗	✓
Video vodiči	✓	✓	✗	✗	✓
Forum	✓	✓	✗	✗	✓
Možnost komentiranja	✗	✗	✓	✗	✗
Navodila za namestitev	✓	✓	✓	✓	✓



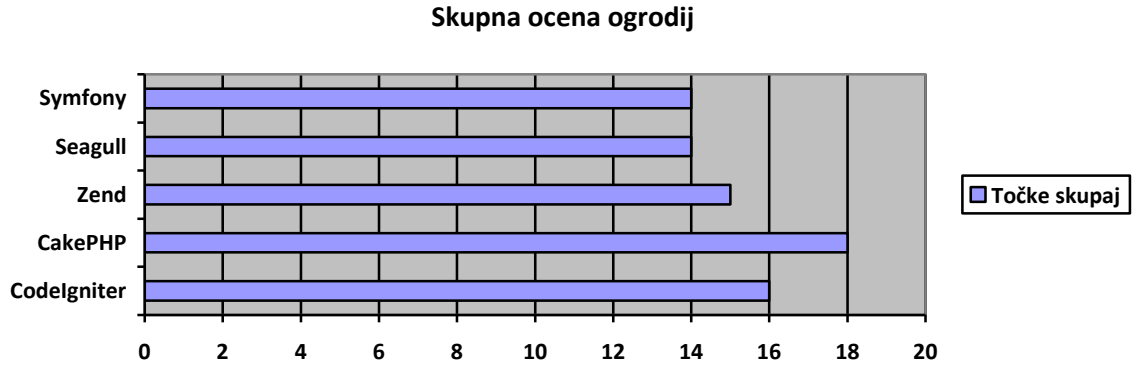
Slika 12: Ocena ogrodij po dokumentaciji.

V seštevku točk sta po kriteriju dokumentacije najvišje ogrodja Symfony ter CakePHP, kot kaže slika 12.

5.5 Rezultati



Slika 13: Točke ogrodij



Slika 14: Končne ocene ogrodij

Tabela 10: Metrika končnih rezultatov

	CakePHP	CodeIgniter	Zend	Seagull	Symfony
Zmogljivost	**3**	**5**	**4**	**2**	**1**
Varnost	**2**	**0**	**3**	**2**	**2**
Enostavnost	**8**	**7**	**5**	**8**	**6**
Dokumentacija	**5**	**4**	**3**	**2**	**5**
Skupaj	**18**	**16**	**15**	**14**	**14**

Odločitev katero izmed izbranih PHP ogrodij torej ni enostavna, saj imajo vsa svoje slabosti in prednosti. Kot kažejo rezultati te raziskave se je na področju hitrosti najbolje odrezalo ogrodje CodeIgniter, najslabše pa Symfony. Kljub temu, da so merjenja hitrosti karseda objektivna, je hitrost ogrodij med sabo težko primerjati, saj ima vsako ogrodje svoje module, knjižnice, razhroščevalnike itd, zaradi česar lahko pride do variacij pri rezultatih. Na področju varnosti, se je prav nasprotno CodeIgniter odrezal najslabše. Tukaj je na prvem mestu ogrodje Zend, saj ima mnogo modulov, ki nam olajšujejo vzpostavitev varnostnega sistema za spletno aplikacijo. Seagull in CakePHP sta se izkazala za podobno enostavna ogrodja, čeprav ima CakePHP malce več zahtev, ko pride do imenovanja podatkovnih baz, kontrolerjev, modulov itd. Dokumentacija pri nobenem ogrodju ne izstopa, najbolje pa sta se odrezala Symfony in CakePHP. Ogrodja se med seboj najbolj razlikujejo po hitrosti (Slika 13), najmanj pa po dokumentaciji.

V skupnem seštevku točk je na prvem mestu CakePHP, kot vidimo v sliki 14.

Poleg opisanih obstajajo še druga merila in kriteriji, s katerimi bi lahko ocenjevali in primerjali obravnavana ogrodja, npr.: fleksibilnost, razširljivost, omogočanje različnih modulov predlog, velikost skupnosti uporabnikov itd.

6 SKLEP

Namen raziskave je bila predstavitev petih pogosto uporabljenih PHP ogrodij, predstaviti njihovo delovanje, njihove prednosti ter slabosti. V drugem poglavju smo spoznali jezik PHP ter njegove lastnosti, v tretjem poglavju smo preleteli elemente za izdelavo enostavnega speltnega dnevnika. Četrto poglavje je bilo namenjeno predstavitvi izbranih ogrodij, peto pa analizi in primerjavi le-teh.

Primerjali smo jih po več različnih kriterijih in po analizi se je na področju zmogljivosti najbolje odrezal CodeIgniter, na področju varnosti Zend, na področju enostavnosti CakePHP ter Seagull in na področju dokumentacije CakePHP ter Symfony. V skupnem točkovanju je največ točk zbral CakePHP.

Kljub vsemu pa je izbira ogrodja stvar odločitve posameznega razvijalca aplikacij, saj ima vsak svoje kriterije in subjektivni pogled na ogrodja.

7 VIRI, LITERATURA

[1] ***CodeIgniter - Open source PHP web application framework,*** http://codeigniter.com/, obiskano 23. septembra 2010.

[2] ***CakePHP: the rapid development php framework,*** http://www.cakephp.org/, obiskano 23. septembra 2010.

[3] ***Seagull PHP Framework :: Overview,*** http://seagullproject.org/, obiskano 23. septembra 2010.

[4] ***PHP Web Application Server - PHP Developer tools - PHP Training & Certification,*** http://www.zend.com/en/, obiskano 23. septembra 2010

[5] ***symfony | Web PHP Framework,*** http://www.symfony-project.org/, obiskano 23. septembra 2010.

[6] **Wikipedia:** ***Comparison of web application frameworks,*** http://en.wikipedia.org/wiki/Comparison_of_web_application_frameworks#PHP, obiskano 23. septembra 2010.

UNIVERZA V MARIBORU

FERI
Fakulteta za elektrotehniko, računalništvo in informatiko

Smetanova ulica 17
2000 Maribor

IZJAVA O USTREZNOSTI DIPLOMSKEGA DELA

Podpisani mentor MILAN OSTERKORN izjavljam, da je
(ime in priimek mentorja)

student MATJAŽ KOVAČIČ izdelal diplomsko
(ime in priimek študenta-tke)

delo z naslovom: ANALIZA IN PRIMERJAVA PHP OGRODIJ NA PRIMERU IZDELAVE SPLETNEGA DNEVNIKA
(naslov diplomskega dela)

v skladu z odobreno temo diplomskega dela, Navodili o pripravi diplomskega dela in mojimi navodili.

Datum in kraj: MARIBOR, 23.9.2010

Podpis mentorja:

UNIVERZA V MARIBORU

FAKULTETA ZA ELEKTROTEHNIKO, RAČUNALNIŠTVO IN INFORMATIKO

(ime fakultete)

IZJAVA O ISTOVETNOSTI TISKANE IN ELEKTRONSKE VERZIJE ZAKLJUCNEGA DELA IN OBJAVI OSEBNIH PODATKOV AVTORJA

Ime in priimek avtorja (avtorice): MATJAŽ KOVAČIČ

Vpisna številka: E1001480

Študijski program: RAČUNALNIŠTVO IN INF. TEHNOLOGIJE

Naslov zaključnega dela: ANALIZA IN PRIMERJAVA PHP OGRODIJ NA PRIMERU IZDELAVE SPLETNEGA DNEVNIKA

Mentor: izred. prof. dr. Milan Ojsteršek

Somentor: /

Podpisani-a MATJAŽ KOVAČIČ izjavljam, da sem za potrebe arhiviranja oddal-a elektronsko verzijo zaključnega dela v Digitalno knjižnico Univerze v Mariboru. Zaključno delo sem izdelal-a sam-a ob pomoči mentorja. V skladu s 1. odstavkom 21. člena Zakona o avtorskih in sorodnih pravicah (Ur. l. RS, št. 16/2007) dovoljujem, da se zgoraj navedeno zaključno delo objavi na portalu Digitalne knjižnice Univerze v Mariboru.

Tiskana verzija zaključnega dela je istovetna elektronski verziji, ki sem jo oddal-a za objavo v Digitalno knjižnico Univerze v Mariboru. Podpisani-a izjavljam, da dovoljujem objavo osebnih podatkov, vezanih na zaključek študija (ime, priimek, leto in kraj rojstva, datum zagovora, naslov zaključnega dela) na spletnih straneh in v publikacijah UM.

Kraj in datum: 28. 9. 2010, Maribor Podpis avtorja (avtorice):