Универзитет у Крагујевцу

Факултет инжењерских наука

Назив студијског програма: Машинско инжењерство

Ниво студија: Основне академске студије

Модул: Информатика у инжењерству

Предмет: Програмски језици

Број индекса: 785/2010

Марко Рашковић

Систем за управљање садржајем портала(cms) применом PHP Yii развојног окружења

завршни рад

Комисија за преглед и одбрану:

1. Др Ненад Грујовић - ментор
2. _____ Датум одбране:_____
3. _____ Оцена:_____

У оквиру овог завршног рада кандидат треба да проучи препоручену, као и ширу литературу која се односи на системе за управљање садржајима портала применом PHP Yii развојног окружења. На бази усвојеног знања кандидат треба да опише Yii развојно окружење, направи мали туторијал за прављење апликације у Yii окружењу, креира систем за управљање садржајем (cms) портала, као и пример административног дела и дела сајта портала у развијеном систему.

Препоручена литература:

[1] Jeffrey Winesett, Agile Web Application Development with Yii1.1 and PHP5, Август 2010, Birmingham.

[2] http://www.yiiframework.com/doc/

ментор:

Др Ненад Грујовић

Крагујевац, xx.xx.2012

_____

Резиме

У првом делу рада изложени су неки основни појмови везани за системе за управљање садржајем и Yii радно окружење. У другом делу су представљене могућности Yii окружења, написан је мањи туторијал који даје увид у једноставност и лакоћу прављења просте веб апликације помоћу Yii окружења, и детаљно је приказан креирани систем за управљање садржајем. На крају је приказан пример портала за размену индивидуалних радова из области роботике.

Кључне речи

Yii радно окружење, систем за управљање садржајем, PHP.

Resume

In the first part are shown some basic futures related with content managment systems, PHP, MySQL, Apache HTTP. Yii framework features, basic tutorial witch show how simply and easy it is creating basic web applications with Yii framework, and detaild review of created cms system are shown in the second part of this study. At the end there is example of portal for individual works exchange in created cms system.

Keywords

Yii framework, content managment system, PHP

Садржај

1. Увод

За потребе креирања и одржавања комплексних веб презентација направљена је посебна група програма за манипулацију садржајем (*CMS – Content Management Systems*). Задатак CMS система је креирање, едитовање, категоризација и публиковање садржаја. Садржај може бити различитих врста, од новинских чланака, преко техничких докумената, па до разних врста каталога – отприлики било какав садржај који ваља адекватно организовати. Поред садржаја, треба организовати и људе који тај садржај креирају, па је тако задатак CMS-а да адекватно расподели улоге свим учесницима који учествују у процесу. Стога CMS треба да обезбеди постојање одређене хијерархије корисника, која омогућава контролу објављивања садржаја, права приступа и праћење процеса креирања садржаја. Све у свему, ради се о једном систему који је у стању да испрати пут креираног садржаја од идеје до објављивања, омогућавајући при том великом тиму људи да ради на њему у строго дефинисаним условима. Све овде речено може се пресликати на CMS-ове, са тим што је садржај, који је њихов крајњи производ, низ веб страница, тачније сајт.

Веб CMS-ови имплементирани су као веб апликације, које кориснику или групи корисника омогућавају да у претраживачу креирају странице сајта. Садржај странице одвојен је од изгледа странице што се лако илуструје на примеру текста који корисник жели да објави – текст је увек исти, али се он може појавити на страницама чији се изглед може производно мењати. За изглед страница су у CMS-овима задужени шаблони (templates), који су у ствари празне странице које доносе одређену визуелну шему. CMS-ови омогућавају корисницима да своје садржаје убацују у различите шаблоне, мењајући тако изглед страница у свега неколико кликова. Сам унос садржаја (најчешће текста и слика) врши се обично путем WYSIWYG едитора, тако да се искуство корисника не разликује пуно од оног код рада у класичним програмима. Подаци које корисник уноси и које CMS користи обично се смештају у базе података, тако да за коришћење одређеног CMS-а сервер мора да подржава одређене базе података и одређене скрипт језике у којима се ови системи програмирају.

CMS-ови направљени су тако да од вас захтевају да током креирања страница испоштујете одређени процес, уносећи поред главног садржаја странице и додатне информације специфичне за веб садржај (*нпр. мета информације*). Садржај који на крају генеришу CMS-ови у складу је са актуелним веб страницама, тако да корисник не мора да се брине о валидности страница. У зависности од специфичних потреба корисника, ови системи се могу проширити неком врстом додатака (plugin, modul, widget), што их чини још моћнијим и кориснијим.

1.1. Коришћене технологије и алати при пројектовању

При пројектовању су максимално коришћене предности отвореног кода и самим тим коришћене су следеће технологије:

- PHP
- MySQL
- JavaScript
- Yii framework
- Jquery

Алати:

- phpMyAdmin
- NetBeans

1.2. PHP

PHP представља наследника алата по називу PHP/FI, написаног 1995. године од стране Расмуса Лердорфа. PHP/FI је представљао скуп алата написаних у Перлу, и аутор га је користио за сопствене потребе. Скуп алата је добио име *алати за личну презентацију* (*Енг. Personal Home Page Tools*), одакле и скраћеница PHP. Како су расле потребе на сајту, аутор је преписао комплетан пројекат у C-у и омогућио да може да комуницира са базама података а корисницима свог сајта да направе сопствене презентације помоћу њега. Расмус је потом објавио своје алате и учинио изворни код доступан свима да би се пројекат брже развијао и да би се грешке брже исправљале.

PHP/FI, чија је пуна дефиниција гласила *алати за личну презентацију/преводилац образаца* (*Енг. Personal Home Page Tools/Forms Interpreter*) је имао само неке ствари заједничке са данашњим PHP-ом - променљиве као у Перлу, аутоматско парсирање променљивих из захтева и уграђени HTML. PHP/FI 2.0 је коначно и званично објављен 1997. године, да би га убрзо заменио PHP 3.0.

Иако се PHP може користити за програмирање конзолних апликација и графичких интерфејса (*библиотека PHP-GTK*) његова основна и главна употреба је у програмирању динамичних страница на Интернету.

До 1997. PHP је стекао неколико хиљада корисника; до 1998. број сајтова на којима је инсталиран PHP 3.0 је порастао на пар стотина хиљада а број корисника истог на пар десетина хиљада. Данас PHP користи неколико стотина хиљада програмера и неколико милиона сајтова.

За разлику од већине програмских језика који поседују почетну функцију (main у C-у, први блок BEGIN u Паскалу, класа која поседује main методу у Јави итд.) него налик на већину скриптних језика, и PHP датотека једноставно садржи скуп инструкција које се извршавају једна за другом, од прве до последње где следи крај програма.

У PHP датотеци, блок који је окружен језичким структурама <?php i ?> се сматра PHP кодом и извршава се, а остатак - ван тих знакова - се сматра текстом који једноставно треба да се испише на стандардни излаз, без интерпретирања. Следи пример једног PHP програма:

```
<!-- наредни део написан у HTML-у се преписује директно на
стандардни излаз -->
<html>
     <body>
          <p>
               <!-- наредни део припада PHP блоку, те се
               извршава -->
               <?php echo "Здраво свете!"; ?>
               <!-- наредни део до краја кода се такође
               преписује директно на стандардни излаз -->
          </p>
     </body>
</html>
```

На овај начин, на стандардном излазу се на крају појављује следећи текст:

```
<html>
     <body>
          <p>
               Здраво свете!
          </p>
     </body>
</html>
```

што представља HTML код спреман за приказ у претраживачу.

PHP код може бити организован у функције и класе, и може се организовати у више датотека. Као почетна датотека, тј. датотека чије инструкције се извршавају прве, узима се она датотека која се да интерпретеру на извршавање.

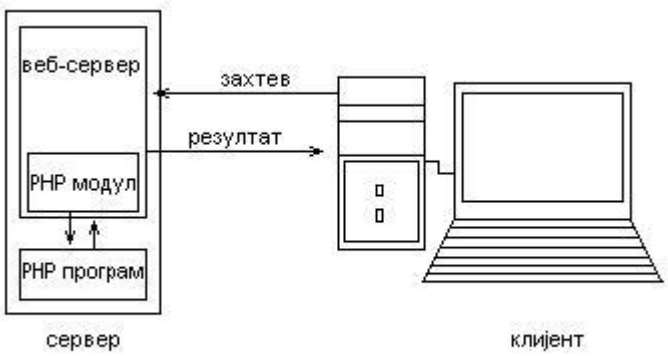
Развојни тим PHP-а се састоји од неколико десетина програмера, и још неколико десетина радника који раде на другим пројектима везаним за PHP, као што је

PEAR и документација PHP-а. Поред овога, PHP-у су добровољно доприносили многи програмери широм света. Брз развој је проузроковао да PHP поседује велики број библиотека и функција, али и проблем неконзистентности у именовању уграђених функција.

Програм који се напише у PHP-у не захтева превођење (*компајлирање*), него се интерпретира при сваком извршавању. PHP интерпретер може радити по CGI принципу, односно тако што ће интерпретер постојати као екстерна апликација која се позива да изврши дату скрипту сваки пут кад буде захтевана од неког корисника, а може бити инсталиран и као модул веб-сервиса. Друга варијанта је данас у највећој употреби јер пружа знатно већу брзину извршавања - интерпретер је на тај начин увек учитан у меморију те се не мора позивати спољашњи програм.

Уобичајен сценарио по ком се извршавају PHP скрипте је следећи:

- клијент (*корисник Интернета који користи неки претраживач*) захтева PHP страницу са сервера
- сервер прослеђује захтев сервису за веб (*програм веб-сервер на серверу*)
- веб-сервер препознаје да се тражи PHP датотека
- не шаље његов садржај клијенту, него га извршава као програм помоћу PHP модула
- излазни текст програма (*стандардни излаз*) се шаље клијенту као резултат захтева
- клијент препознаје врсту резултата (*HTML код, слика, PDF садржај, архива итд.*)
- резултат се приказују клијенту на одговарајући начин



1.3. MySQL

MySQL је вишенитни, вишекориснички SQL систем за управљање базама података. Систем ради као сервер, обезбеђујући вишекориснички интерфејс за приступ бази података.

16. јануара 2008. године, MySQL AB је објавио да је Sun Microsystems откупио MySQL за око милијарду америчких долара.

Следи кратак преглед развоја пројекта:

- MySQL је први пут објављен 23. маја, 1995
- Верзија за Windows је објављена 8. јануара, 1998. за Windows 95 и Windows NT
- Верзија 3.23: бета верзија у јуну 2000, завршна верзија у јануару 2001
- Верзија 4.0: бета верзија у августу 2002, завршна верзија у марту 2003 (додате уније)
- Верзија 4.1: бета верзија у јуну 2004, завршна верзија у октобру 2004 (додата p-дрвета, под-упити и припрема упита унапред)
- Верзија 5.0: бета верзија у марту 2005, завршна верзија у октобру 2005 (додати курсори, процедуре, тригери, прегледи, XA трансакције)
- Верзија 5.1: тренутно у припреми завршне верзије (од новембра 2005) (додају се партиције, интерфејс за plug-ине, репликације на нивоу појединачних записа, табеле са логовима сервера и извршавање унапријед заказаних догађаја)
- Sun Microsystems откупљује MySQL 16. јануара 2008.

Библиотеке за приступ бази података MySQL постоје за већину програмских језика, чији облик зависи од датог програмског језика. Додатно, постоји ODBC интерфејс по називу MyODBC који дозвољава приступ бази података за оне програмске језике који подржавају ODBC интерфејс, као што су ASP и ColdFusion. MySQL сервер и званично подржане библиотеке су углавном писани у програмским језицима C и C++.

За администрацију базе података MySQL, администратори користе или интерфејс у облику командне линије, или графички интерфејс (*MySQL администратор*) и друге. Поред алата које производи фирма MySQL AB, постоји и неколико комерцијалних и некомерцијалних алата приступачних на тржишту. Алат PhpMyAdmin је слободан софтвер чији је интерфејс у облику веб странице а који је написан у програмском језику PHP.

Раним верзијама MySQL-а су недостајале многе особине које су прописане стандардом за релационе базе података, обично због жеље да систем буде брз и ефикасан. Многе, али не и све, такве особине су касније додате, укључујући трансакције и затезнике за релациони интегритет, што је неопходно да база остане у правилном облику и спречи клијенте базе да тај облик покваре.

У документацијама неких ранијих верзија су чак постојале тврдње да такве особине нису ни потребне, него чак штетне. Једна таква тврдња се појављивала у секцији под насловом *разлози зашто НЕ користити удаљене кључеве*, и објашњавала је да је релациони интегритет компликован за имплементацију и да му је једина сврха за скицирање архитектуре базе података.

Још неке критике обухватају начин на који MySQL третира поља вредности NULL и поља подразумеваних вредности, који се одваја од стандарда SQL-а. Начин на који рукује датумима дозвољава уписивање датума са бројем дана који превазилази број дана у датом месецу, а аритметичке операције су рањиве на превелике целобројне вредности и реални бројеви имају проблема са одсецањем децималног дела. Од верзије 5, начин на који сервер третира неисправне вредности, зависи од режима рада сервера, који је подразумевано подешен тако да необично много толерише исте, што критичари жестоко замерају.

Када је објављена верзија 5.0 MySQL-а, Дејвид Аксмарк, кооснивач MySQL AB-а, је рекао да је *MySQL од првог дана критикован што нема процедуре, тригере и прегледе* и да *сада поправљају све те проблеме у једној верзији.* MySQL 5.0 је објављен 24. октобра 2005, након неколико милиона скинутих примерака са Интернета у фази бета верзије 5.0.

1.4. Apache HTTP

Apache HTTP сервер пројекат представља напор заједничког развоја софтвера у циљу стварања робусног и бесплатног HTTP (веб) сервера отвореног кода . Пројекат воде групе волонтера широм света, који користе интернет да комуницирају, планирају и развијају сервер и њему сродну документацију. Пројекат је део Apache Software фондације. Поред тога, стотине корисника су допринели идејама, кодом и документацијом пројекту. У фебруару 1995, најпопуларнији сервер софтвер на веб-у је развијен од стране Rob McCool-а у националном центру за суперкомпјутер апликације, Универзитета у Илиноису. Међутим развој је заустављен после одласка Rob-а из NCSA-а средином 1994-те, и многи вебмастери су развили екстензије и поправили багове. Мала група ових вебмастера, ступила је у контакт преко email-а у циљу координације својих измена (у форми "закрпа"). Brian Behlendorf и Cliff Skolnick су саставили мејлинг листу, простор за размену информација и ауторизацију за програмере на машинама

California Bay Area, са пропусним опсегом дониранim од стране HotWired-a. До краја Фебруара, осам главних сарадника формирало је фондацију Apache Group.

1.5. Инсталација Apache сервера, PHP-а и MySQL-а

Појединачно инсталирање наведених технологија, поред времена, захтева и додатна подешавања. Све то можете превазићи инсталирањем одређених пакета који садрже све поменуте технологије и уз то је све оптимизовано и подешено. Један од тих пакета је XAMPP http://www.apachefriends.org/en/xampp.html који постоји за четири различите платформе: Linux, Windows, Mac OS X и Solaris. Све што је потребно за инсталацију јесте преузимање инсталера са адресе http://www.apachefriends.org/en/xampp-linux.html, а затим покретање инсталације која вам инсталира Apache сервер, PHP, MySQL и Perl.

Одмах после инсталирања покрените сервер и MySQL preko XAMPP control panel-а или командне линије, после чега можете отворити ваш претраживач и покренути localhost, после чега ће се отворити главна страна вашег сервера. Сви фајлови се смештају у директоријум htdocs унутар xampp инсталације одакле се покрећу од стране сервера.

2. Yii радно окружење

Yii представља PHP радно окружење високих перформанси за развој веб 2.0 апликација. Yii помаже веб програмерима да праве комплексне апликације. Yii се изговара Yee или [ji:], и представља акроним за "*Yes It Is!*" Ово је често тачан, и концизан одговор на питања оних који су нови са Yii системом:

Да ли је брз? ... Да ли је безбедан? ... Да ли је професионалан? ... Да ли је добар за мој следећи пројекат? ... Yes, it is! (*Да, јесте!*).

Yii је слободан , радно окружење отвореног кода за развој веб апликација написаном у PHP5 који представља јасан дизајн и подстиче брз развој. Поједностављује развој ваше аплкације и обезбеђује изузетно ефикасан, проширив и одржив крајњи производ. Са екстремно оптимизованим перформансама, Yii је савршен избор за сваку величину пројекта. Међутим, направљен је са софистицираним, и пословним апликацијама у виду. Имате пуну контролу над свим подешавањима у сагласности са вашом пословном апликацијом. Долази у пакету са

алатима који помажу при тестирању и дебаговању ваше апликације, и има разумњиву и свеобухватну документацију.

2.1. Историја

Yii оснивач је Qiang Xue који је започео Yii пројекат првог Јануара 2008 године. Qiang је претходно развио и одржавао Prado framework. Године стеченог искуства и прикупљене повратне информације програмера из тог пројекта појачало је потребу за брзом, безбедном и професионалном framework-у који би био посебно направљен да задовољи Веб 2.0 развој апликација. Трећег Децембра 2008, после скоро годину дана развоја Yii 1.0 је званично пуштен у јавност.

Са својим изузетно импресивним перформансама у односу на остале PHP framework-ове је одмах добио веома позитивне критике и његова популарност и усвајање су се константно повећавали.

2.2. Могућности Yii радног окружења

Било да сте један програмер који прави прилично једноставан веб сајт, или тим програмера који праве изузетно сложену Веб апликацију, коришћењем Yii-а проширујете ваш тим програмера са додатним професионалним и ефикасним ресурсима.

Можете бити фокусирани на специфичним пословним задацима, и пустите Yii да обезбеди стратегију имплементације свих наведених:

- Model-View-Controller (*MVC - модел-поглед-контролер*) дизајн образац
- Database Access Objects (*DAO*), Query Builder (*Прављење Упита*), Active Record, DB Migration (*DB миграција*)
- Форме и валидација
- AJAX - активне додатке (*widgets*)
- Аутентификација и ауторизација
- Skinning и theming (*измена дизајна апликације*)
- Веб сервиси

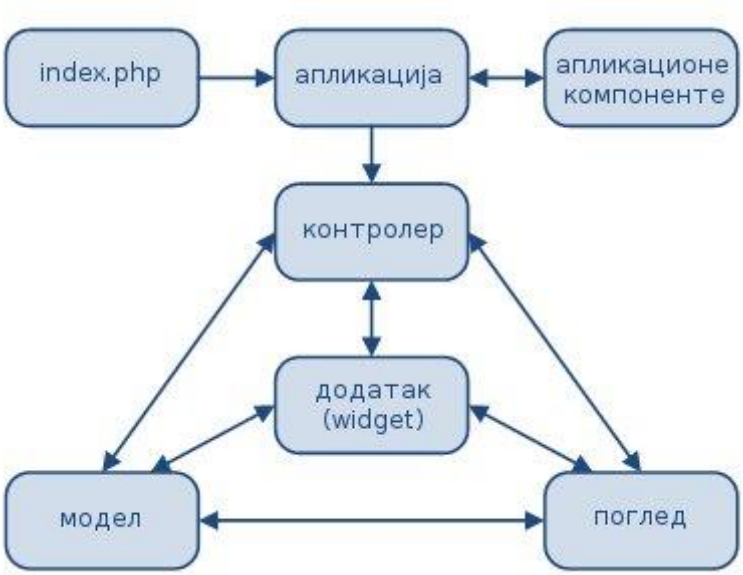
- Интернационализација (*i18n*) и локализација (*l10n*)
- Шема слојевитог кеширања
- Руковање грешкама и логовање
- Безбедност
- Unit и функционално тестирање
- CRUD - аутоматско генерисање кода
- XHTML усклађеност
- Објектно орјентисан
- Пријатељски са кодом треће стране (*PEAR, Zend Framework…*)
- Детаљна документација
- Библиотека екстензија

2.2.1. Model-View-Controller (MVC) дизајн образац

Yii имплементира model-view-controller (*MVC - модел-поглед-контролер*) дизајн, који је широко прихваћен у Веб програмирању. MVC има за циљ да одвоји пословну логику од корисничког интерфејса, тако да програмери могу много лакше да измене један део без утицаја на други. MVC, модел представља информацију (*податак*) и пословну логику. Поглед садржи елементе корисничког интерфејса као што су: текст, улази форме итд. Контролер управља комуникацијом између модела и погледа.

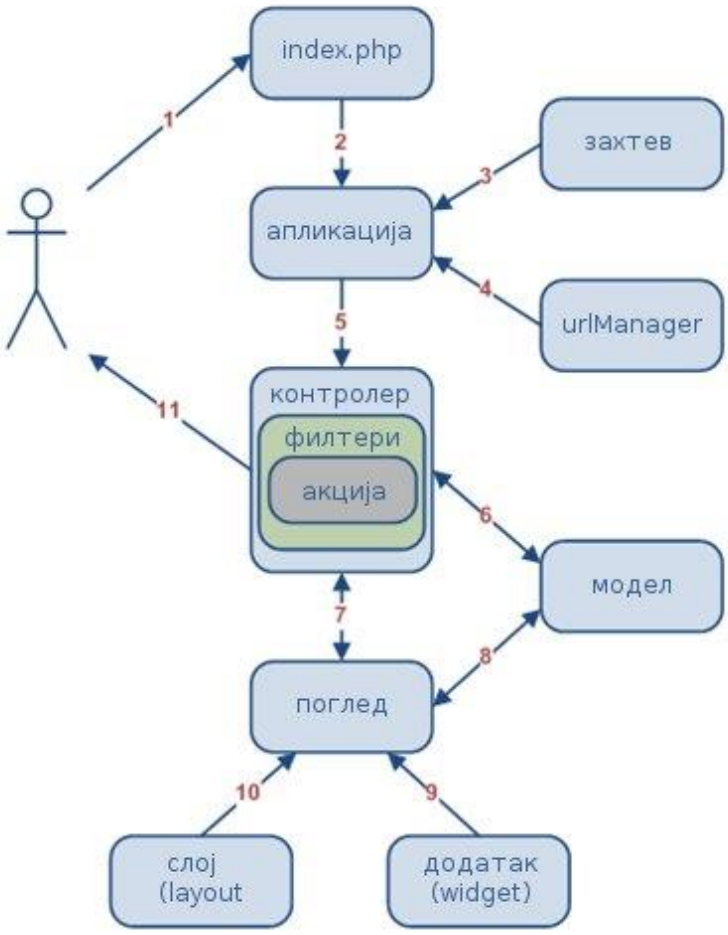
Поред имплементације MVC, Yii такође уводи front-controller, који инкапсулира контекст за обраду захтева. Апликација прикупља неке информације о корисничком захтеву и затим отпрема на одговарајући контролер за даљу манипулацију.

Следећи дијаграм приказује статичку структуру Yii примене:



Типични ток

Следећи дијаграм приказује типичан радни ток за Yii апликацију када се рукује корисничким захтевом:



1. Корисник прави захтев са адресом http://www.example.com/index.php?r=post/show&id=1, веб сервер обрађује захтев извршавањем bootstrap[1] скрипте index.php.
2. Bootstrap скрипта креира инстанцу апликације и покреће је.
3. Апликација добија детаљне информације о захтеву корисника из компоненте апликације која се назива request (*захтев*).
4. Апликација одређује тражени контролер и акцију помоћу апликационе компоненте зване urlManager. На пример, контролер је Post, који се односи на PostController класу, а акција показује који стварни смисао је одређен контролером.
5. Апликација креира инстанцу захтеваног контролера да додатно обради корисничков захтев. Контролер утврђује да акција Show упућује на методу actionShow у класи контролера. Затим креира и извршава филтере (*access control, benchmarking*) у вези са овом акцијом. Акција се извршава само уколико је дозвољено од стране филтера.
6. Акција чита Post модел из базе чији је ID 1.
7. Акција показује поглед show са Post моделом.
8. Поглед чита и приказује атрибуте Post модела.
9. Поглед извршава неке додатке (*widgets*).
10. Приказани резултат погледа је уграђен у layout-у (*слоју*).
11. Акција завршава поглед приказом резултата кориснику.

2.2.2. Рад са базама података

Yii пружа снажну подршку за рад са базом података. Изграђена на врху PHP Data Objects (*PDO*) екстензије, Yii Data Access Objects (*DAO*) омогућава приступ различитим системима за управљање базама података (*DBMS*), у једном јединственом интерфејсу.

[1] Самоодрживи процес који наставља без спољне помоћи.

Апликације развијене коришћењем Yii DAO могу се лако пребацити на другачији DBMS без потребе да се модификује приступни код података.

Yii креатор упита нуди објектно оријентисан метод за изградњу SQL упита, што смањује ризик од SQL injection напада.

И Yii Active Record (*AR*), који се реализује као широко прихваћен објектно-релационо мапиран (*ORM*) приступ, додатно поједностављује програмирање база података. Представља табелу у смислу класе и реда као инстанце, Yii AR елиминише понављање задатака који пишу оне SQL упите који се углавном баве CRUD (*create, read, update и delete – креирање, читање и брисање*) операцијама.

Иако Yii функције везане за базу података могу да обаве скоро све послове, и даље можете да користите сопствене библиотеке за базе података у вашој Yii апликацији. У ствари, Yii framework је пажљиво дизајниран да се користи заједно са другим независним библиотекама.

Data Access Objects (*DAO*)

Data Access Objects (*DAO*) обезбеђује генерички API за приступ подацима који се налазе у различитим системима за управљање базама података (*DBMS*). Као резултат тога, основни *DBMS* може бити промењен у неки други без потребе за променом кода који користи *DAO* за приступ подацима.

Yii *DAO* је изграђен на врху PHP Data Objects (*PDO*), који представља његов продужетак и обезбеђује јединствен приступ подацима за многе популарне *DBMS*-ove, као што су *MySQL*, *PostgreSQL*. Дакле, да бисте користе Yii DAO, PDO екстензија и специфични PDO драјвер (нпр. PDO_MYSQL) треба да буду инсталирани.

Yii DAO углавном се састоји од следећих четири класа:

- *CDbConnection*: представља конекцију ка бази података.
- *CDbCommand*: представља SQL упите.
- *CDbDataReader*: представља ток редова из скупа резултата упита.
- *CDbTransaction*: представља DB трансакцију.

Успостављање конекције са базом података

Да бисте успоставили везу са базом података, креирајте инстанцу од *CDbConnection* и активирајте га. Потребно је име извора података (*DSN*) да би се остварило повезивање са базом података. Корисничко име и лозинка такође могу бити неопходни за успостављање конекције. Изузетак ће бити подигнут у случају да дође до грешке приликом успостављања везе (*нпр. лош DSN или неважећи име/лозинка*).

```
$connection=new CDbConnection($dsn,$username,$password);
// успостављање везе.
$connection->active=true;
......
$connection->active=false;  // затварање конекције
```

Формат *DSN*-а зависи од *PDO* коришћеног драјвера. У принципу, *DSN* се састоји од имена *PDO* драјвера, праћено двотачком, и драјвер-специфичном везом. Погледајте *PDO* документацију за више информација. Испод је листа најчешће коришћених формата *DSN*-а:

- *SQLite*: sqlite:/path/to/dbfile
- *MySQL*: mysql:host=localhost;dbname=testdb
- *PostgreSQL*: pgsql:host=localhost;port=5432;dbname=testdb
- SQL *Server*: mssql:host=localhost;dbname=testdb
- *Oracle*: oci:dbname=//localhost:1521/testdb

Због тога што се *CDbConnection* продужује из *CApplicationComponent,* такође га можемо користити као компоненту апликације. Да бисте то урадили, конфигуришете у *db*-у (или под другим називом) компоненту апликације у конфигурацији као што следи,

```
array(
    ......
    'components'=>array(
        ......
        'db'=>array(
            'class'=>'CDbConnection',

'connectionString'=>'mysql:host=localhost;dbname=testdb',
            'username'=>'root',
            'password'=>'password',
            'emulatePrepare'=>true,  // потребан за неке MySQL
инсталације
        ),
```

```
    ),
)
```

Сада можемо приступити *DB* конекцији преко *Yii::app()->db* који је већ аутоматски активиран, осим ако смо конфигурисали *CDbConnection::autoConnect* да је *false*. Коришћењем овог приступа, једана DB-веза може да се дели на више места у нашем коду.

Извршавање SQL наредби

Када се успостави веза са базом података, SQL наредбе се могу извршити помоћу *CDbCommand*. Један креира инстанцу *CDbCommand* позивом *CDbConnection::createCommand()* са одређеном SQL наредбом:

```
$connection=Yii::app()->db;   // претпостављајући да имате конфигурисану "db" конекцију
// Ако немате, можете експлицитно креирати конекцију:
// $connection=new CDbConnection($dsn,$username,$password);
$command=$connection->createCommand($sql);
// ако је потребно, SQL наредба се може изменити са следећим:
// $command->text=$newSQL;
```

SQL наредба се извршава преко *CDbCommand* преко једног од следећа два начина:

- *execute()*: изводи SQL наредбе које се не тичу упита, као што су *INSERT*, *UPDATE* и *DELETE*. Ако буде успешна, враћа број редова.
- *query()*: извршава SQL наредбу која враћа редове података, као што је *SELECT*. Ако буде успешна, она враћа инстанцу *CDbDataReader* из које се могу добити настали редови података. Због конвенције, скуп *queryXXX()* метода које се такође имплементирају, директно враћају резултате упита.
- *$rowCount=$command->execute();* // извршава не-упитне SQL наредбе
- *$dataReader=$command->query();* // извршава SQL упите
- *$rows=$command->queryAll();* // враћа све редове резултата упита
- *$row=$command->queryRow();* // враћа први ред резултата упита
- *$column=$command->queryColumn();* // враћа прву колону резултата

- *$value=$command->queryScalar();* // враћа прво поље реда резултата упита

Преузимање резултата упита

Након што *CDbCommand::query()* генерише *CDbDataReader* инстанцу, један може да преузме редове резултујућих података позивањем *CDbDataReader::read()*. Такође се може користити CDbDataReader у PHP foreach-у.

```
$dataReader=$command->query();
// позива read() више пута док не врати вредност false
while(($row=$dataReader->read())!==false) { ... }
// коришћење foreach-а за прелажење кроз сваки ред података
foreach($dataReader as $row) { ... }
// враћа све редове од једном у једном низу (array)
$rows=$dataReader->readAll();
```

Обавезујући (binding) параметри

Да бисте избегли SQL *injection* нападе и да бисте унапредили перформансе извршења SQL наредби које се користе више пута, може се "припремити" SQL команда са опционим параметарским чуварима места (*placeholders*) који треба да буду замењени са реалним параметрима у току обавезујућег параметар процеса.

Параметарски чувари места (*placeholders*) могу бити именовани (представљени као јединствени токени) или неименовани (представљени са знаком питања). Позивањем *CDbCommand::bindParam()* или CDbCommand::bindValue() да замени ове чуваре места са стварним параметрима. Обавезујући параметар морају да се ураде пре него што се SQL наредба изврши.

```
// SQL са два чувара места (placeholders) ":username" и
":email"
$sql="INSERT INTO tbl_user (username, email)
VALUES(:username,:email)";
$command=$connection->createCommand($sql);
// замењује чувара места ":username" са стварном username
вредношћу
$command->bindParam(":username",$username,PDO::PARAM_STR);
// замењује чувара места ":email" са стварном email вредношћу
$command->bindParam(":email",$email,PDO::PARAM_STR);
$command->execute();
// убацује још један ред са новим сетом параметара
```

```
$command->bindParam(":username",$username2,PDO::PARAM_STR);
$command->bindParam(":email",$email2,PDO::PARAM_STR);
$command->execute();
```

Методе bindParam() и bindValue() су веома сличне.

2.2.3. Рад са Формама

Прикупљање корисничких података преко HTML форми је један од главних задатака при развоју веб апликација. Поред дизајнирања образаца, програмери треба да попуне формулар са постојећим подацима или подразумеваним вредностима, провере унос корисника, приказују одговарајуће поруке о грешкама за погрешан унос, и сачувају унос за складиштење. Yii у великој мери поједностављује овај ток посла са MVC архитектуром.

Следећи кораци су обично потребни када се ради са формама у Yii-у:

- Направити модел класу која представља податке поља за колекцију.
- Направити контролер акцију са кодом који одговара на слање форме.
- Направити форму у скрипти поглед фајла повезан са акцијом контролера.

2.2.4. Проширење Yii апликације

Проширење Yii апликације је уобичајена активност у току развоја. На пример, када пишете нови контролер, можете проширити Yii наслеђивањем његове CController класе, када пишете нови widget, ви проширујете CWidget или неку постојећу widget класу. Ако је проширени код дизајниран тако да се поново може употребити од стране других програмера, зовемо га продужетак (*extension*).

Нека екстензија обично служи за једну сврху. У Yii смислу, може се класификовати на следећи начин:

- Компонента апликације
- Понашање (*behavior*)
- Додатак (*widget*)

- Контролер
- Акција
- Филтер
- Конзолна команда
- Валидатор: валидатор је класа компоненте проширена са CValidator.
- Помагач (*helper*) помагач је класа само са статичним методама. То је као да глобалне функције користе име класе као свој именски простор.
- Модул: модул је самостална софтверска јединица која се састоји од модела, контролера, погледа и осталих подржаних компоненти. У многим аспектима, модул подсећа на апликацију.Главна разлика је у томе што је модул унутар апликације.

Екстензија може бити компонента која не спада ни у једну од наведених категорија. У ствари, Yii је пажљиво дизајниран тако да скоро сваки део њеног кода може се продужити и прилагодити тако да одговарају индивидуалним потребама.

Компонента апликације

Да бисте користили неку компоненту апликације, прво је потребно да промените конфигурацију апликације додајући нову ставку у конфигурацији компоненти, као што следи:

```
return array(
    // 'preload'=>array('xyz',...),
    'components'=>array(
        'xyz'=>array(
            'class'=>'ext.xyz.XyzClass',
            'property1'=>'value1',
            'property2'=>'value2',
        ),
        // остале конфигурације компоненти
    ),
);
```

Затим, можемо приступити компоненти на било ком месту помоћу *Yii::app()->xyz*. Компонента ће бити лењо (*lazily*) креирана (то јест, креирана када се приступа по први пут), осим ако ми не наведемо при подешавању компоненти.

Понашање (*behavior*)

Behavior (понашање) може да се користи у свим врстама компоненти. Његова употреба укључује два корака. У првом кораку, behavior (понашање) је прикључен на циљну компоненту. У другом кораку, behavior (понашање)метода се позива преко циљне компоненте. На пример:

```
// $name јединствено идентификује понашање у компоненти
$component->attachBehavior($name,$behavior);
// test() је метода из $behavior
$component->test();
```

Чешће се behavior (понашање) везује за компоненту користећи конфигурациони начин уместо позивањем *attachBehavior* методе. На пример, да бисте прикључили behavior (понашање) на компоненту апликације, могли бисмо да користимо следећу конфигурацију апликације:

```
return array(
    'components'=>array(
        'db'=>array(
            'class'=>'CDbConnection',
            'behaviors'=>array(
                'xyz'=>array(
                    'class'=>'ext.xyz.XyzBehavior',
                    'property1'=>'value1',
                    'property2'=>'value2',
                ),
            ),
        ),
        //....
    ),
);
```

Додатак (widget)

Додаци се углавном користе у погледима. У погледу додатак можемо користити као што следи:

```
// додатак којем није потребан body садржај
<?php $this->widget('ext.xyz.XyzClass', array(
    'property1'=>'value1',
    'property2'=>'value2')); ?>
```

```
// додатак који може да садржи body
<?php $this->beginWidget('ext.xyz.XyzClass', array(
    'property1'=>'value1',
    'property2'=>'value2')); ?>
...body садржај додатка...
<?php $this->endWidget(); ?>
```

Акција

Акције се користе од стране контролера да одговори на специфичне захтеве корисника. Можемо је користити у контролер класи као што следи:

```
class TestController extends CController
{
    public function actions()
    {
        return array(
            'xyz'=>array(
                'class'=>'ext.xyz.XyzClass',
                'property1'=>'value1',
                'property2'=>'value2',
            ),
            // остале аксије
        );
    }
}
```

Затим се акцији може приступити путањом `test/xyz`.

Филтер

Филтери се такође користе од стране контролера. Они се углавном извршавају пре или после обраде захтева корисника када се рукује акцијом.

```
class TestController extends CController
{
    public function filters()
    {
        return array(
            array(
```

```
                'ext.xyz.XyzClass',
                'property1'=>'value1',
                'property2'=>'value2',
            ),
            // остали филтери
        );
    }
}
```

У претходном, можемо да користимо плус и минус операторе у првом елементу низа да примените филтере у само ограниченим акцијама. За више детаља, погледајте документацију.

Контролер

Контролер обезбеђује скуп акција које могу бити захтеване од стране корисника. Да бисмо користили екстензију типа контролер, треба да конфигуришемо *CWebApplication::controllerMap* у апликационој конфигурацији:

```
return array(
    'controllerMap'=>array(
        'xyz'=>array(
            'class'=>'ext.xyz.XyzClass',
            'property1'=>'value1',
            'property2'=>'value2',
        ),
        // остали контролери
    ),
);
```

Валидатор

Валидатор се углавном користи у класи модела (*CFormModel* или *CActiveRecord*).

```
class MyModel extends CActiveRecord // or CFormModel
{
    public function rules()
    {
        return array(
            array(
                'attr1, attr2',
                'ext.xyz.XyzClass',
                'property1'=>'value1',
```

```
                'property2'=>'value2',
            ),
            // остала правила валидације
        );
    }
}
```

Конзолна команда

Конзолна команда обично проширује yiic алатку са додатном командом. Можемо га користити конфигурисањем конфигурације за конзола апликације:

```
return array(
    'commandMap'=>array(
        'xyz'=>array(
            'class'=>'ext.xyz.XyzClass',
            'property1'=>'value1',
            'property2'=>'value2',
        ),
        // остале команде
    ),
);
```

Можем
о користити yiic алатку са додатном командом xyz.

Модул

Детаљније о модулима у поглављу 4.7.

2.2.5. Аутентификација и ауторизација

Аутентификација и ауторизација потребни су за Веб страницу која треба да буде ограничено на одређене кориснике То обично укључује корисничко име и лозинку, али може да садржи било које друге методе исказивања идентитета, као што су паметне картице, отисак прста, итд. Аутентификација сазнаје да ли лицу, када се једном идентификује (*mj. ауторизован*), је дозвољено да манипулоше одређеним ресурсима. Ово обично утврђује да ли та особа има одређену ролу за приступ ресурсима.

Yii има уграђен authentication/authorization (*auth*) радно окружење које је једноставно за коришћење и може се прилагодити посебним потребама.

Централни део у Yii auth радном окружењу је унапред декларисана корисничка апликациона компонента која је објектно имплементирана (*IWebUser*) интерфејс. Корисничка компонента садржи сталну информацију о идентитету тренутног корисника. Можемо јој приступити са било ког места помоћу *Yii::app()->user* кода.

Помоћу корисничке компоненте, можемо проверити да ли је корисник пријављен преко *CWebUser::isGuest* (да ли је пријављен); можемо да пријавимо или одјавимо корисника, можемо проверити да ли корисник може обављати одређене послове позивом *CWebUser::checkAccess* и можемо такође добити јединствени идентификатор и друге информације о идентитету корисника.

2.2.6. Теме и скинови

Теме представљају систематичан начин прилагођавања изгледа странице у веб апликацији. Примењујући нову тему, укупан изглед веб апликације може одмах да се промени.

У Yii, свака тема је представљена као директоријум који се састоји од погледа (*view*), облика (*layout*) и осталих фајлова као што су слике, CSS фајлови, JavaScript фајлови, итд. Име теме је име његовог директоријума. Све теме се налазе у истом директоријуму WebRoot/themes. У сваком тренутку, само једна тема може да буде активна.

2.2.7. Веб сервиси

Веб сервис је софтверски систем дизајниран да подржи интероперабилну машина-машини интеракцију преко мреже. У контексту веб апликације, обично се односи на скуп API-ја којима се може приступити преко Интернета и да изврши на удаљеном хостинг систему тражену услугу. На пример, Flex-базиран клијент може позвати функцију имплементирану на страни сервера који ради под PHP веб апликацијом. Веб сервис се ослања на SOAP као његовог основног слоја комуникацијоног протокола.

Yii обезбеђује *CWebService* и *CWebServiceAction* да поједноставе имплементацију веб сервиса у веб апликацији. API-ји су груписани у класе, које се називају сервисни провајдери. Yii за сваку класу генерише WSDL спецификацију која описује који API-ји су доступни и како треба да се позову од стране клијента. Када је API позиван од стране клијента, Yii ће позвати одговарајући провајдер сервис и позвати тражене API-је.

2.2.8. Интернационализација

Интернационализација (*I18N*) односи се на процес дизајнирања софтверске апликације тако да може да се прилагоди разним језицима и регионима без инжењерских промена. За веб апликације, ово је од посебног значаја, јер потенцијални корисници могу бити из целог света.

Yii пружа подршку за I18N у неколико аспеката.

- Обезбеђује локалне податке за сваки од могућих језика и варијанти.
- Пружа сервис поруке и сервис превођење датотека.
- Пружа локално-зависан формат датума и времна.
- Пружа локално-зависан формат бројева.

2.2.9. Кеширање

Кеширање је јефтин и ефикасан начин да се побољшају перформансе веб апликација. Чувајући релативно статичне податке у кешу и извлачи га из кеша када се затражи, можемо уштедети време потребно за генерисање података.

Употреба кеша у Yii-у углавном подразумева конфигурисање и приступање кеш апликационој компоненти. Следећа апликациона конфигурација прецизира кеш компоненту која користи memcache са два кеш сервера.

```
array(
    ......
    'components'=>array(
        ......
```

```
        'cache'=>array(
            'class'=>'system.caching.CMemCache',
            'servers'=>array(
                array('host'=>'server1', 'port'=>11211,
'weight'=>60),
                array('host'=>'server2', 'port'=>11211,
'weight'=>40),
            ),
        ),
    ),
);
```

Када се апликација извршава, кеш компоненти се може приступити преко *Yii::app()->cache* кода.

Yii пружа разне компоненте кеширања које могу да складиште податке у различитим медијима. На пример, *CMemCache* компонента инкапсулира memcache PHP екстензију и користи меморију као медиј за складиштење кеша; *CApcCache* компонента инкапсулира PHP APC екстензију, и *CDbCache* компонента чува кеширане податке у бази података. Следе неке од доступних кеш компоненти:

- *CMemCache*: користи PHP memcache екстензију.
- *CApcCache*: користи PHP APC екстензију.
- *CXCache*: користи PHP XCache екстензију.
- *CEAcceleratorCache*: користи PHP EAccelerator екстензију.
- *CDbCache*: користи табелу базе података да смешта кеш податке.

Кеширање се може користити на различитим нивоима. На најнижем нивоу, користимо кеш да смести један део податка, као што су променљиве, и зовемо га кеширање података. На следећем нивоу, у кешу чувамо фрагмент странице који је генерисан од стране погледа. А на највишем нивоу, чувамо целу страницу у кешу и користимо је по потреби.

2.2.10. Руковање грешкама

Yii обезбеђује радно окружење за комплетно манипулисање грешкама базиран на PHP 5 механизму изузетака. Када се креира апликација за руковање долазним захтевима корисника, он региструје своју *handleError* методу за руковање PHP упозорењима и обавештењима, и региструје *handleException* методу за руковање не ухваћеним PHP изузецима. Према томе, ако се PHP упозорења/обавештења или не ухваћени изузетак јавља у току извршавања апликације, једна од руковаоца грешкама ће преузети контролу и почети неопходну процедуру руковања грешкама.

По подразумеваној вредности, *handleError* (*или handleException*) ће покренути *onError* догађај (*или onException догађај*). Ако се грешком (*или изузетком*) не рукује са било којим руковаоцем, он ће позвати помоћ из *errorHandler* апликационе компоненте.

2.2.11. Пријава

Yii обезбеђује флексибилну и прошириву могућности пријаве. Поруке пријављивања могу се класификовати према нивоима пријаве и категорији поруке. Коришћење ниво и категорија филтере, одабране поруке могу додатно да се усмеравају ка различитим дестинацијама, као што су датотеке, е-пошту, претраживаче, итд.

2.2.12. Порука Логовања

Поруке се могу пријавити позивом било *Yii::log* или *Yii::trace*.

```
Yii::log($message, $level, $category);
Yii::trace($message, $category);
```

Када пријављујете поруку, потребно је да наведете своју категорију и ниво. Категорија је стринг у формату xxx.yyy.zzz. На пример, ако је порука пријављена у CController-у, можемо користити категорију *system.web.CController*. Ниво поруке треба да буде једна од следећих вредности:

- trace (*траг*)
- info (*информација*)

- profile (*профил*)
- warning (*упозорење*)
- error (*грешка*)

2.2.13. Безбедност

Превенција скриптинга на страни сајта (Cross-site scripting)

Скриптинг на страни сајта (*такође познат као XSS*) се јавља када веб апликација прикупља податке од злонамерних корисника. Често ће нападачи убризгати JavaScript, VBScript, ActiveX, HTML, или Flash у рањиву апликацију да превари остале кориснике и прикупља податке из ње. На пример, лоше дизајниран Форум систем може да прикаже унос корисника у форуму без икакве провере. Нападач тада може убризгати део злонамерног JavaScript кода у пост, тако да други корисници кад прочитају овај чланак, неочекивано JavaScript ради на њиховим рачунарима.

Једна од најважнијих мера за спречавање XSS напада је да проверите унос корисника пре него што их прикжете. Можете да урадите HTML-кодирање корисничког уноса за постизање овог циља. Међутим, у неким ситуацијама, HTML кодирање не може бити добар избор, јер онемогућава HTML тагове.

Yii обухвата рад *HTMLPurifier* и омогућава програмерима корисну компоненту звану *CHtmlPurifier* који енкапсулира HTMLPurifier. Ова компонента је способна да уклони сав злонамеран код.

HTMLPurifier компонента се може користити било као додатак или филтер. Када се користи као додатак, *HTMLPurifier* ће прочистити садржај приказан у свом телу. На пример,

```
<?php $this->beginWidget('CHtmlPurifier'); ?>
... овде је приказивање корисничког кода ...
<?php $this->endWidget(); ?>
```

Спречавање фалсификовања захтева на сајту (CSRF)

CSRF напади се јављају када злонамерни веб сајт проузрокује да корисников веб претраживач изврши нежељену акцију на поузданој локацији. На пример, злонамерни веб сајт има страницу која садржи слику чији src (*линк адреса*) показује на банкарски сајт: http://bank.example/withdraw?transfer=10000&to=someone. Ако корисник има пријављен колачић(*cookie*) за банкарски сајт а деси се да посети ову страницу, извршиће се пребацивање 10.000 долара.

Да бисте спречили CSRF нападе, важно је да се придржавате правила за GET захтеве коме треба дозволити да преузме податке али не и да мења податке на серверу. А за POST захтев, треба да садрже неку насумичну вредност која се може препознати од стране сервера да би се потврдило да је форма послата и да су резултати послати натраг.

```
return array(

    'components'=>array(
        'request'=>array(
            'enableCsrfValidation'=>true,
        ),
    ),
);
```

Cookie превенција напада

Заштита колачића од напада је од изузетне важности, зато што се *session ID* обично чувају у колачићима. Ако неко добије на чување *ID session*, он у суштини поседује све релевантне информације сесије.

Постоји неколико противмера за спречавање напада на колачиће.

- Апликација може користити SSL за креирање сигурног комуникационог канала и само да колачић прође аутентификацију преко једне HTTPS конекције.
- Сесија има адекватни важећи рок, укључујући све колачиће и токене сесије.

- Спречити сајт скриптовање које изазива извршавање произвољног кода у претраживачу корисника и тако излаже колачиће нападу.
- Валидација података колачића и провера да ли су измењени.

```
return array(
    'components'=>array(
        'request'=>array(
            'enableCookieValidation'=>true,
        ),
    ),
);
```

2.2.14. Аутоматско генерисање кода

Од верзије 1.1.2, Yii је опремљен интернет алатком за генерисање кода који се зове Gii. Унапређена претходна генерација *yiic shell* алата која ради на командној линији.

Коришћење Gii алата

Gii се имплементира у смислу модула и мора се користити у оквиру постојеће Yii апликације. Да бисмо користили Gii, морамо прво изменити конфигурацију апликације:

```
return array(
    ......
    'modules'=>array(
        'gii'=>array(
            'class'=>'system.gii.GiiModule',
            'password'=>'овде упишите лозинку',
            // 'ipFilters'=>array(...листа IP-а...),
            // 'newFileMode'=>0666,
            // 'newDirMode'=>0777,
        ),
    ),
);
```

У претходно наведеном коду, декларишемо модел gii чија класа је *GiiModule*. Такође одређујемо шифру модула која ће бити затражена када будемо приступали Gii-у.

По подразумеваној вредности из безбедносних разлога, Gii је конфигурисан тако да буду доступан само на *localhost* -у. Ако желимо да буде доступан на другим рачунарима, можемо конфигурисати *GiiModule::ipFilters* као што је приказано у претходном коду.

Због тога што Gii може генерисати и сачувати нове датотеке у постојећој апликацији, морате да се уверите да веб сервер има одговарајућа права. Изнад *GiiModule::newFileMode* и *GiiModule::newDirMode* контролише се како би се нови фајлови и директоријуми генерисали.

Сада можемо приступити Gii-у преко адресе: *http://hostname/path/to/index.php?r=gii*. Овде предпостављамо да је *http://hostname/path/to/index.php* адреса постојеће Yii апликације.

Ако постојећа Yii апликација користи *path*-формат адресе, можемо приступити Gii-у преко адресе: *http://hostname/path/to/index.php/gii* .

Проширење Gii-ја

Док уобичајени код генератори који долазе уз Gii могу да генеришу веома моћан код, ми често желимо да их прилагодимо или да креирамо нове да би се уклопили у наше потребе. На пример, можда ћемо желети да генеришемо код у нашем омиљеном стилу кодирања, или можда желите да направите код који подржава више језика. Све ово се може лако урадити са Gii алатом.

Gii се може продужити на два начина: прилагођавање кода шаблону постојећег код генератора, и писање новог генератора кода.

Структура код генератора

Код генератор је смештен под директоријумом чије име је подразумевано и име код генератора. Директоријум се обично састоји од:

```
model/                          root фолдер генератора модела
   ModelCode.php                код модела коришћен за генерисање
>> кода
   ModelGenerator.php           код генератор контролер
   views/                       садржи поглед скрипте за
>> генератор
      index.php                 подразумевана поглед скрипта
   templates/                   садржи шаблон кода
      default/                  подразумевани чшаблон
         model.php              шаблон кода за генерисање кода
>> модел класе
```

Конфигурација изнад упућује Gii да тражи генераторе под директоријуму наведеном као *application.gii* поред подразумеване локације *system.gii.generators*.

Могуће је имати два генератора под истим именом али под различитим претраживачким путањама. У овом случају генератор под путањом дефинисаном раније у *GiiModule::generatorPaths ће* имати предност.

Прилагођавање код шаблона

Ово је најједноставнији и најчешћи начин ширења Gii-ја. Користићемо пример да објаснимо како се прилагођава шаблон кода. Претпоставимо да желимо да прилагодимо код генерисан од стране генератора модела.

Прво смо креирали директоријум под именом *protected/gii/model/templates/compact*. Овде модел значи да ћемо замените подразумевани генератор модела. И *templates/compact* значи да ћемо додати нови шаблон скупа кодова под називом compact (*компактан*).

Затим мењамо конфигурацију наше апликације за додавање *application.gii* у *GiiModule::generatorPaths*.

Сада отворите модел код генератор страницу. Кликните на код шаблон поље. Требало би видети падајућу листу која садржи наш ново креирани шаблон. Међутим, ако изаберемо овај шаблон да генерише код, видећемо грешку. То је зато што тек треба да се стави било која стварна код датотека у нови шаблон.

Копирајте фајл *framework/gii/generators/model/templates/default/model.php* у *protected/gii/model/templates/compact*. Ако покушамо поново генерисање са компактним шаблоном, требало би све проћи успешно. Међутим, генерисани код се не разликује од оног који је генерисао подразумевани (*default*) шаблон.

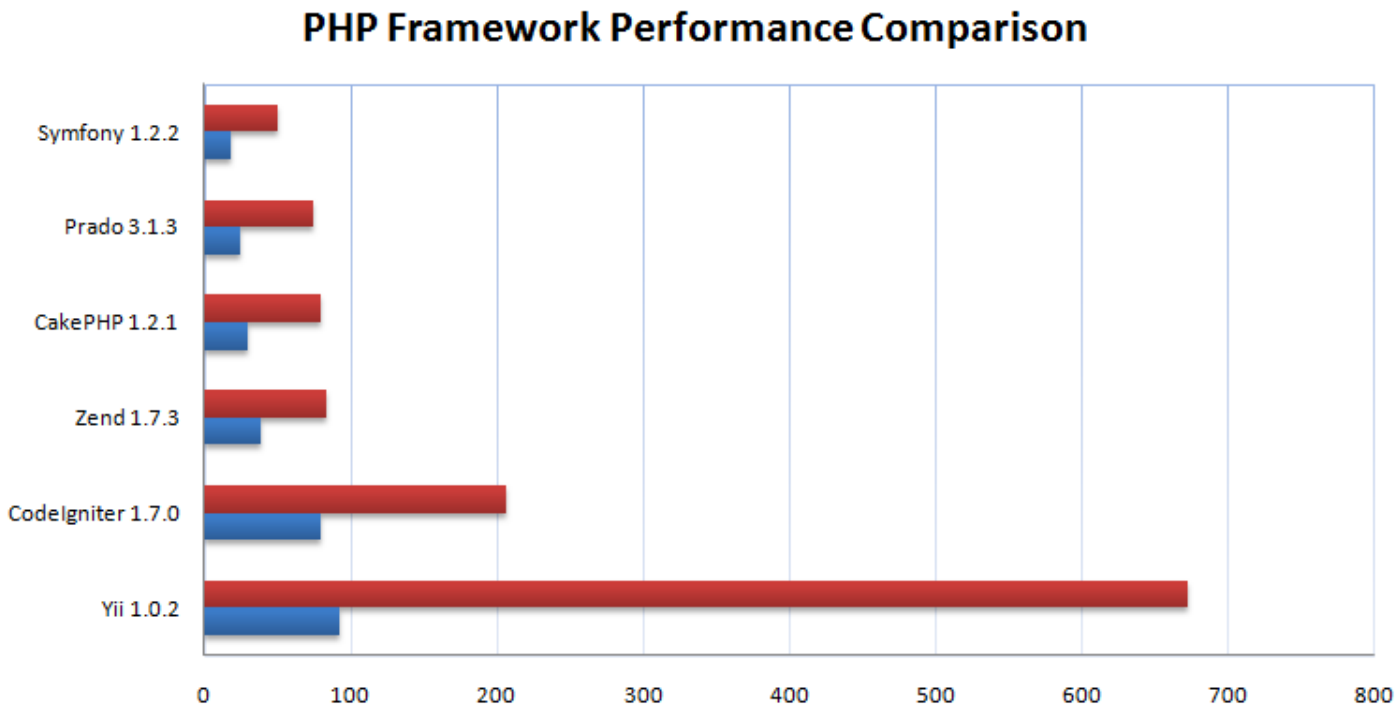
Време је да урадимо право прилагођавање шаблона. Отворите датотеку *protected/gii/model/templates/compact/model.php* за измену. Имајте на уму да ће се овај фајл користити као поглед (*view*) скрипта, што значи да може садржати PHP екстензије. Хајде да изменимо шаблон тако да *attributeLabels()* метода у генерисаном коду користи *Yii::t()* да преведе наслове атрибута:

```
public function attributeLabels()
{
    return array(
<?php foreach($labels as $name=>$label): ?>
            <?php echo "'$name' => Yii::t('application',
'$label'),\n"; ?>
<?php endforeach; ?>
    );
}
```

У сваком код шаблону, можемо приступити неким унапред дефинисаним варијаблама, као што је *$labels* у горе поменутом примеру.

2.3. Перформансе Yii

Yii је framework високих перформанси. Графикон испод показује колико је ефикасан Yii у поређењу са другим популарним PHP радним окружењима. У графикону, RPS је скраћеница за "*захтев у секунди*", која описује колико захтева које апликација записује у радно окружење може да обради по секунди. Већи број, ефикасније радно окружење. Као што можемо да видимо Yii надмашује сва друга окружења. Предност у перформансама код Yii-а је значајна када је укључена APC[1] екстензија.



	Yii 1.0.2	CodeIgniter 1.7.0	Zend 1.7.3	CakePHP 1.2.1	Prado 3.1.3	Symfony 1.2.2
RPS with APC	673	206	83	79	75	50
RPS without APC	93	79	38	29	24	18

[1] Напредно php кеширање

2.4. Yii туторијал

2.4.1. Креирање нове апликације

Да бисмо креирали нову апликацију, користићемо алат који је познат као yiic, и који долази уз yii радно окружење. Он представља алат командне линије који можемо да користимо да брзо направимо нову Yii апликацију. Није обавезно коришћење овог алата, али ће вам уштедети много времена и гарантовати одговарајућу структуру директоријума и фајлова.

Да бисте користили алат за прављење ваше прве Yii апликације, отворите shell прозор, и пронађите место на вашем фајл систему где желите да креирате апликацију. За потребе овог малог туторијала, ми ћемо предпоставити следеће:

- *YiiRoot* је директоријум где имате инсталиран Yii
- *WebRoot* је конфигурисан као root документ вашег веб сервера
- Из командне линије промените на ваш WebRoot директоријум и извршите следеће:

```
% cd WebRoot
% YiiRoot/framework/yiic webapp demo
Create a Web application under '/Webroot/demo'? [Yes|No]
Yes
   mkdir /WebRoot/demo
   mkdir /WebRoot/demo/assets
   mkdir /WebRoot/demo/css
   generate css/bg.gif
   generate css/form.css
   generate css/main.css
```

Ваша апликација је успешно креирана под /Webroot/demo. Webapp команда се користи да креира потпуно нову Yii апликацију. Користи само један аргумент да прецизира апсолутну или релативну путању до директоријума где апликација треба да

буде креирана. Резултат је генерисање свих потребних директоријума и фајлова да би се обезбедила структура Yii веб апликације.

Сада можемо променити у ново креирани demo директоријум и погледати шта је креирано за нас.

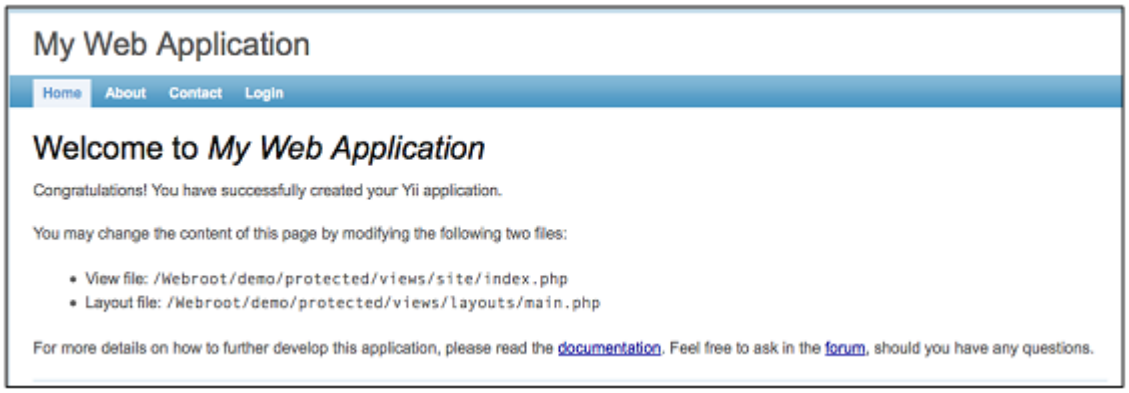
```
% cd demo
% ls -p
assets/    images/    index.php  themes/
css/    index-test.php  protected/
```

Опис ставки високог нивоа које су аутоматски креиране:

```
demo/
index.php    улазни скрипт фајл веб апликације
index-test.php  улазни скрипт фајл за функционално тестирање
assets/    садржи објављене изворне фајлове
css/      садржи CSS фајлове
images/    садржи слике
themes/    садржи изглед апликације
protected/    садржи заштићене фајлове апликације
```

Са извршавањем једне једноставне команде у командној линији, створили смо структуру директоријума и креирали све потребне фајлове. Сви ови директоријуми и фајлови заједно са поддиректоријумима и фајловима које они садрже, може изгледати мало обесхрабрујуће на први поглед. Међутим можемо игнорисати већину њих на самом почетку. Сви ови фолдери и фајлови представљају функционалну веб апликацију. Yiic команда је генерисала довољно кода да покренемо једноставну почетну страну, стандардну контактирајте нас (*Contact Us*) страну да прикаже пример веб форме, и страна за ауторизацију (*login*) са довољно аутоматски генерисаног кода да демонстрира основе ауторизације и аутентификације у Yii. Ако ваш сервер подржава GD2 екстензију графичких библиотека, такође ћете видети CAPTCHA додатак (*widget*) на контакт форми, и апликација ће имати одговарајућу валидацију за ову форму.

Докле год ваш веб сервер ради, требало би да можете да отворите свој претраживач и дођите до адресе http://localhost/demo/index.php. Овде ће вам бити представљена *My Web Application* почетна страна заједно са пријатељским поздравом *Welcome to My Web Application*, а затим следе корисне информације о следећим корацима које треба предузети. Следећа слика показује пример почетне странице:



Цео овај аутогенерисан код ће добити више смисла када прођемо кроз прост пример. Да би смо испробали овај нови систем, хајде да изградимо Hello, World! (*здраво свете*) програм. Hello, World! програм у Yii-у биће проста веб страна апликације која шаље ову поруку нашем претраживачу. Већ смо разговарали о томе да Yii представља Модел-Поглед-Контролер (*Model-View-Controller*) радно окружење. Типична Yii апликација прима захтев од претраживача, налази контролер, а затим позива акцију унутар тог контролера. Контролер затим може да позове одређени поглед да рендерује и врати садржај кориснику. Ако се ради са подацима, контролер ће комуницирати са моделом да одради све Креирај, Читај, Ажурирај и Обриши (*CRUD*) операције над тим подацима. У нашој простој апликацији, све што захтевамо јесте код за контролер и поглед. Не користимо податке, тако да нам модел неће бити потребан. Почнимо наш пример прављењем контролера.

Као што смо радили када смо креирали прву апликацију, поново ћемо позвати yiic команду да нам помогне при креирању нашег контролера. У овом случају, користићемо *yiic shell* команду да стартујемо апликацију са интерактивним shell-ом у коме ћемо позвати остале команде. Да би старторвали shell идите на root ваше демо апликације покретањем следеће команде:

```
%cd /Webroot/demo
```

Затим извршите yiic са следећом shell командом:

```
%YiiRoot/framework/yiic shell

Yii Interactive Tool v1.1

Please type 'help' for help. Type 'exit' to quit.
```

```
>>
```

Сада сте у дијалогу са интерактивним shell-ом. Можете укуцати help да видите листу доступних команди:

```
>> help

At the prompt, you may enter a PHP statement or one of the
following

commands:

 - controller

 - crud

 - form

 - help

 - model

 - module

Type 'help <command-name>' for details about a command.
```

Видимо да су доступне неколико команди. Controller команда изгледа као нешто што нама треба, како желимо да креирамо контролер за нашу апликацију. Можемо сазнати нешто више о овој команди куцајући help controller у командној линији.

```
>> help controller

USAGE

  controller <controller-ID> [action-ID] ...

DESCRIPTION

  This command generates a controller and views associated
with

  the specified actions.

...
```

Из читања помоћи. види се да команда controller ће генерисати контролер, акције и погледе који су у вези са потребним акцијама. Пошто је примарна функција наше апликације да прикаже поруку, назовимо контролер *message*, и назовимо акцију по поруци коју желимо приказати:

```
>> controller message helloWorld
```

```
generate MessageController.php
      mkdir /Webroot/demo/protected/views/message
   generate helloworld.php
   generate index.php
Controller 'message' has been created in the following file:
    /Webroot/demo/protected/controllers/MessageController.php
You may access it in the browser using the following URL:
    http://hostname/path/to/index.php?r=message
>>
```

Требало би да одговори са индикацијом успешног креирања MessageController унутар protected/controllers/ директоријума.

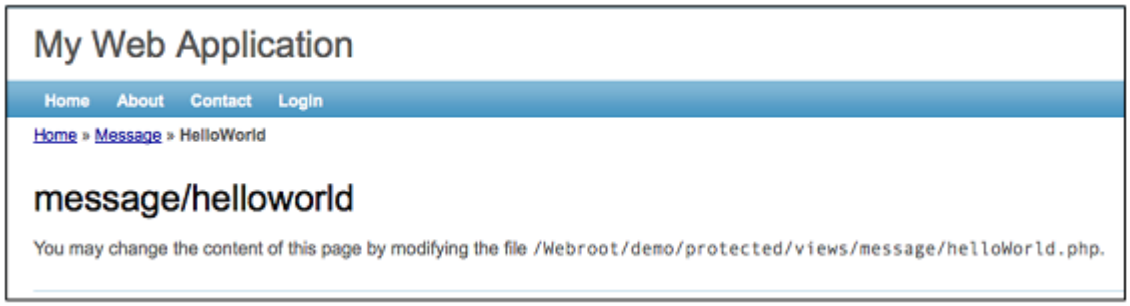
Са једном простом командом, генерисали смо нови PHP контролер фајл, назван MessageController.php, и смештен је на подразумевану локацију protected/controllers/. Генерисана MessageController класа наслеђује главну класу апликације Controller, која се налази на protected/components/Controller.php. Ова класа проширује главну класу радног окружења CController, тако да аутоматски добија сва стандардна понашања контролера. Пошто смо навели actionID параметар, helloWorld, проста акција је такође креирана унутар MessageController-а под називом *actionHelloWorld()*. Yiic алат претпоставља да ова акција, као већина акција дефинисаних унутар контролера, рендерује поглед. Тако да додаје код унутар ове методе за рендеровање поглед фајла под истим именом helloworld.php, и смешта га унутар подразумеваног директоријума за поглед фајлове повезане за овај контролер protected/views/message/. Код који је генерисан за MessageController класу:

```
<?php
class MessageController extends Controller
{
  public function actionHelloWorld()
{
    $this->render('helloWorld');
  }
  public function actionIndex()
  {
```

```
    $this->render('index');
  }
}
```

Видимо да је такође додата и *actionIndex()* метода која једноставно рендерује поглед фајл који је такође аутоматски креиран за нас на дестинацији protected/views/message/index.php. По Yii стандарду захтев који наводи message као controllerID, али не наводи име акције, биће усмерен ка actionIndex() методи. yiic алат је био довољно паметан да је знао да треба креирати и подразумевану акцију.

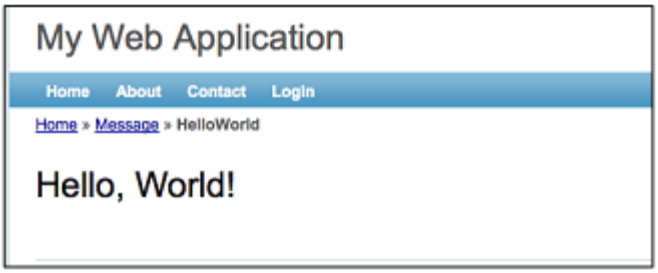
Испробајте апликацију адресом:
http://localhost/demo/index.php?r=message/helloWorld.



Да бисмо ово претворили у Hello, World! апликацију, све што је потребно јесте да прилагодимо *helloWorld.php* поглед да прикаже Hello, World! поруку. Ово се лако ради. Измените фајл protected/views/message/helloWorld.php тако да садржи само следећи код:

```
<?php
$this->breadcrumbs=array(
  'Message'=>array('message/index'),
  'HelloWorld',
);?>
<h1>Hello, World!</h1>
```

Снимите ваш код и погледајте страницу поново у вашем претраживачу:
http://yourhostname/index.php?r=message/helloWorld



Сада имамо нашу комплетну просту апликацију са запањујуће мало кода.

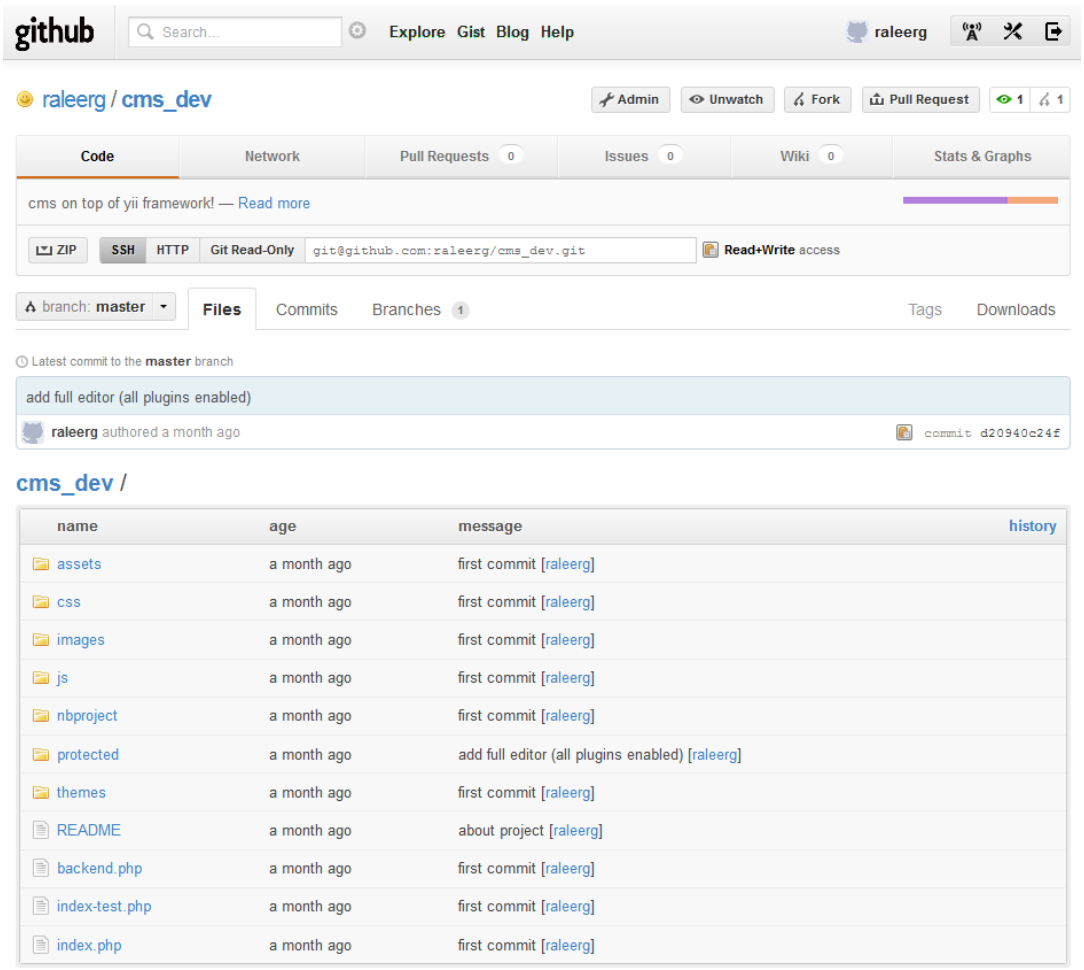
3. Инсталација cms система

3.1. Скидање последње верзије система

Са GitHub -а можете преузети последњу верзију cms система. Такође можете се прикључити и учествовати у даљем развоју система.

GitHub је Веб засновани хостинг сервис за развој софтвера који користи git контролни систем. GitHub нуди комерционалне и бесплатне системе за пројекте отвореног кода. Github пружа следеће услуге:

- Сарадња (*управљање тимовима*)
- Git Wiki
- Праћење багова
- Преглед и дискусије у вези кода



На адреси: https://github.com/raleerg/cms_dev можете прегледати или скинути све фајлове из пројекта. Уколико желите да скинете последњу верзију cms система, на

претходно наведеној адреси кликните на иконицу: ZIP при чему ће се појавити дијалог за снимање архиве на локални диск.

За сарадњу на пројекту неопходно је инсталирати git са адресе: http://git-scm.com/download. У зависности од оперативног система који користите селектујте један од понуђених. После инсталације потребно је извршити подешавање git-а, о чему највише можете прочитати на адреси: http://help.github.com/win-set-up-git/.

3.2 Подешавање система

Након скидања кода са GitHub-а потребно је распаковати zip архив на root вашег система (*ако још нисте инсталирали php, mysql и apach препоручујем http://www.apachefriends.org/en/xampp.html*). Унутар распакованог директоријума налази се и MySQL база са потребним табелама унутар sql dump фајла који је потребно импортовати унутар базе. Прво је потребно помоћу phpmyadmin-а или неког другог алата за манипулацију са базама креирати базу унутар које је потребно импортовати sql dump фајл. Базу именујте cms_dev, а затим селектујте ново креирану базу и кликните на импорт картицу после чега треба кликнути на browse дугме и селектовати sql фајл на дестинацији: cms_dev/protected/data/cms_dev.sql као на следећој слици.

Као што је раније речено цмс ради на yii framework-u који не иде уз cms. Зато је потребно скинути последњу верзију са адресе: http://www.yiiframework.com/download/, а затим распаковати га на серверу где се налази cms систем. Замените путање до yii-а са путањом до ваше дестинације на линији 4 у фајловима:

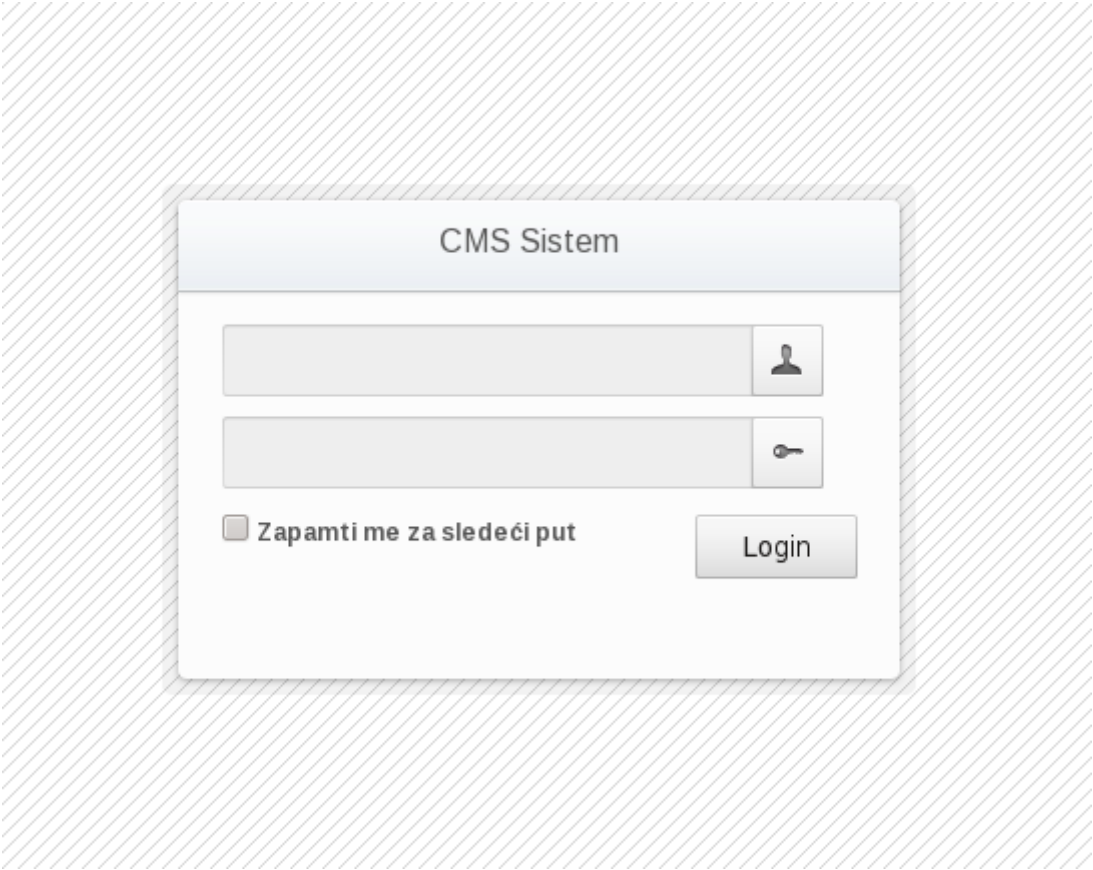
1. backend.php
2. index.php
3. protected/yiic.php

Последње подешавање потребно је извршити у фајлу protected/config/main.php на линији 60, где се налазе подешавања везана за базу података. Упишите име базе и хоста као и username и password.

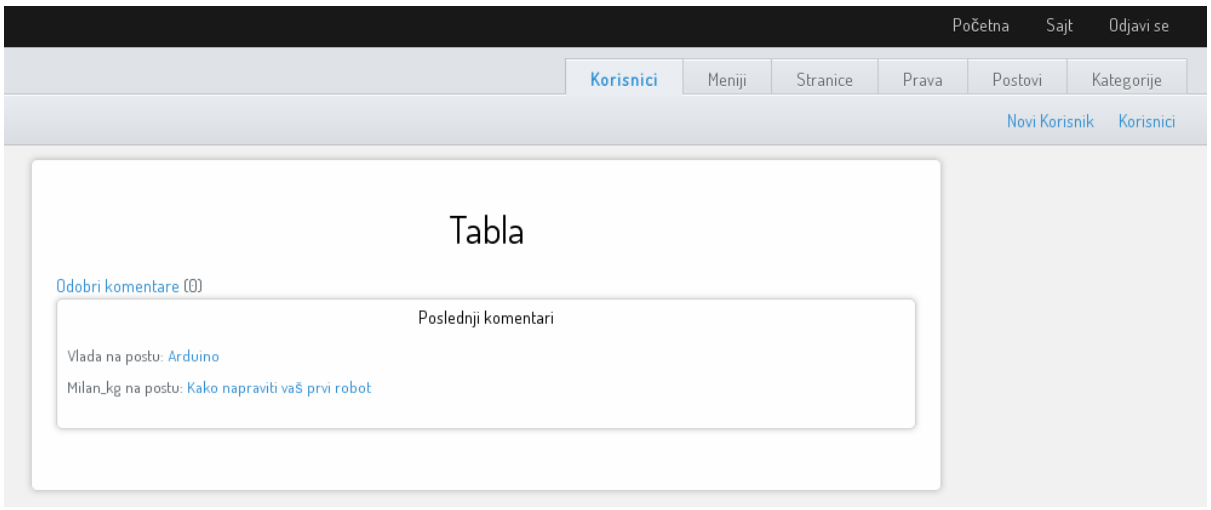
Ваш систем је сада подешен и спреман. На адреси localhost/cms_dev_vx.xx можете покренути тест сајт, а на адреси localhost/cms_dev_vx.xx/backend.php се налази административни део сајта.

4. Администрација система

После инсталације cms система имамо аутоматски пробне податке импортоване заједно са табелама у базу података. То нам омогућава да на root адреси система видимо тест портал, са почетном темом (*изгледом*), као и са тест постовима. Све те податке можемо по жељи мењати или уклонити из административног дела система, који се налази на адерси /cms_dev_vx.xx/backend.php. Да би се приступило административном делу система потребно је укуцати корисничко име и шифру. На свеже постављеном систему постоји корисник admin са шифром admin који користите да уђете по први пут у администрацију а затим направите новог корисника или измените постојећи. Мењање и брисање admin корисника је веома важно због безбедности система и информација на серверу.



Након ауторизације појављује се прва страна система, страна која показује информације о последњим коментарима и коментаре које треба одобрити. Ове информације представљају widget, представља пример додатних програма који могу да олакшају рад на систему својим брзим и приступачним информацијама. О прављењу једног од ових додатних програма биће речи нешто касније.

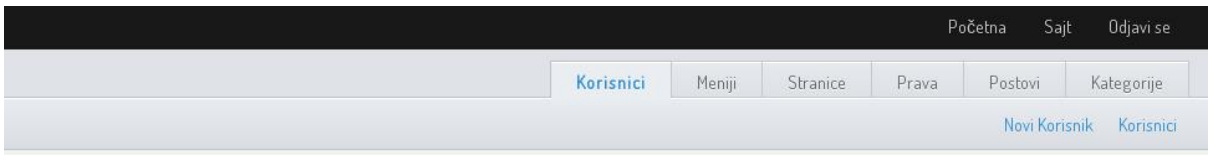


Главни мени се налази на врху стране, и има све главне операције у систему. Преко њега приступа се следећим категоријама:

- Корисници
- Менији
- Странице
- Права
- Постови
- Категорије

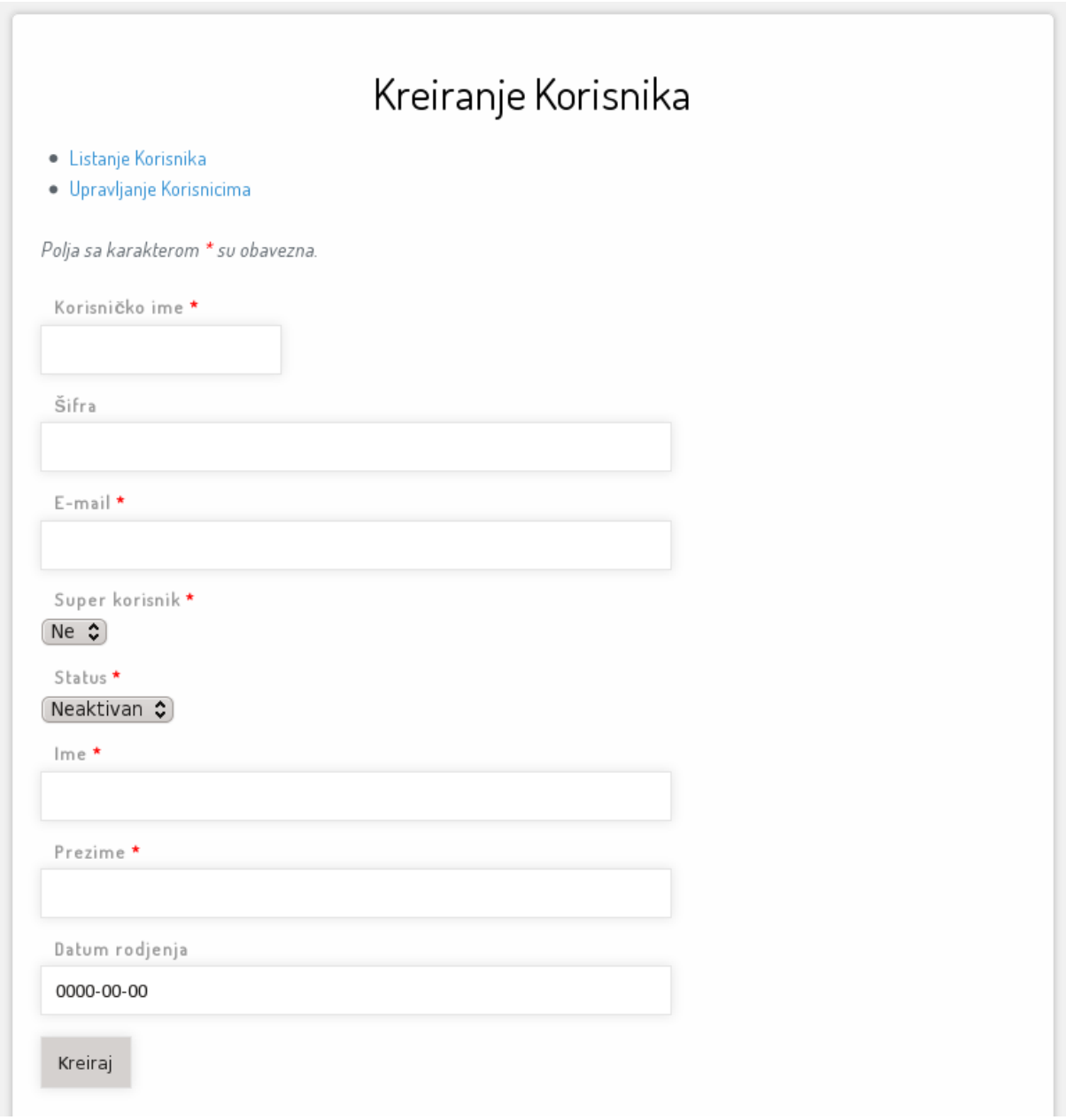
Унутар сваке од наведених категорија налазе се основне операције као што су: креирање, брисање, измена итд.

Изнад главног менија се налази топ мени са линковима: сајт, који отвара сајт у новом табу, одјави се који је намењен за одјаву из администрационог дела, и почетна који води на почетну административну страну.



4.1. Корисници

Као што је напоменуто на почетку, на свеже инсталираном систему имамо корисника admin са шифром admin који се након прве ауторизације мења. Из главног менија изаберите Нови Корисник после чега се отвара форма за прављење другог корисника.

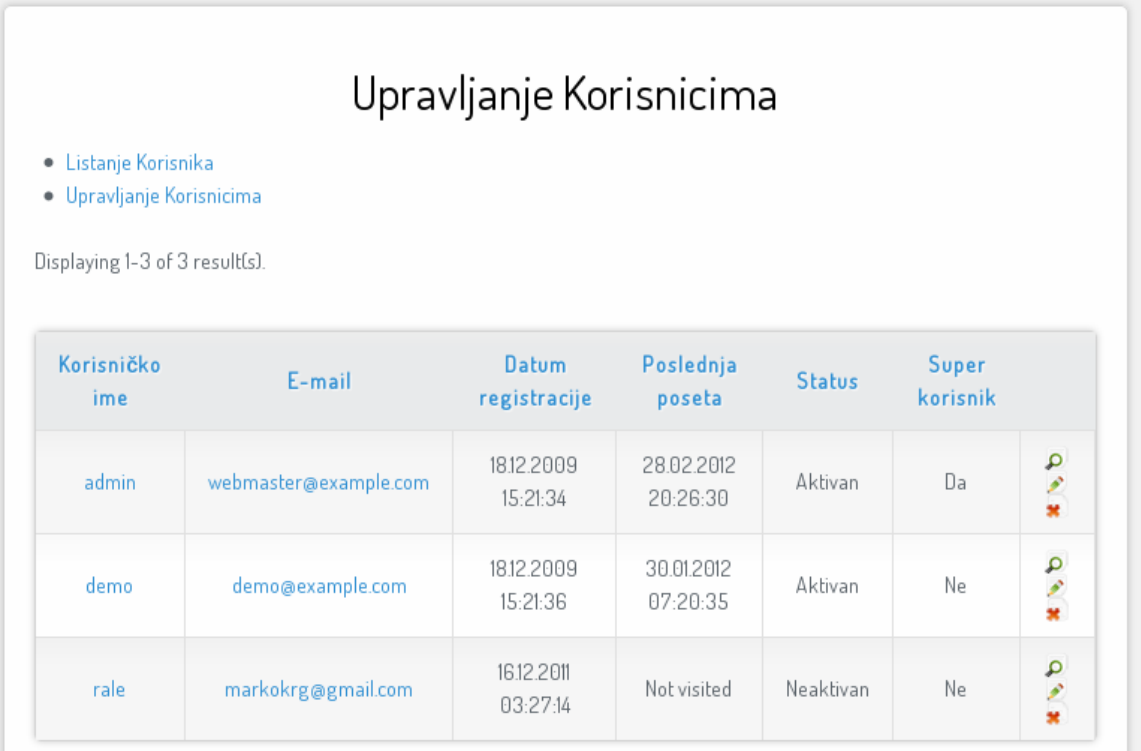


На слици су приказана сва поља потребна да се креира корисник. Битно је издвојити поље за селектовање супер корисника, које представља права корисника. Супер корисник има сва права над свим садржајем система: корисницима, постовима, менијима, страницама и категоријама. Док обичан корисник има права само над

својим садржајем, односно садржајем који је он сам креирао. Овде треба напоменути да је ово основни систем додељивања права корисницима, а да постоји још један који се налази под именом *права* у главном менију. Ово је додатни модул који није имао сврху на презентационом блогу али ће користити на већем пројекту где имамо више степена сигурности и већи број различитих корисника.

Статус као и што само каже представља статус корисника, односно да ли је или није активан. Поља: име, презиме и датум рођења су поља профила корисника, поља која се могу мењати или се додавати нова по потреби. Тренутно ова могућност није потребна па се неће детаљније помињати у даљем тексту.

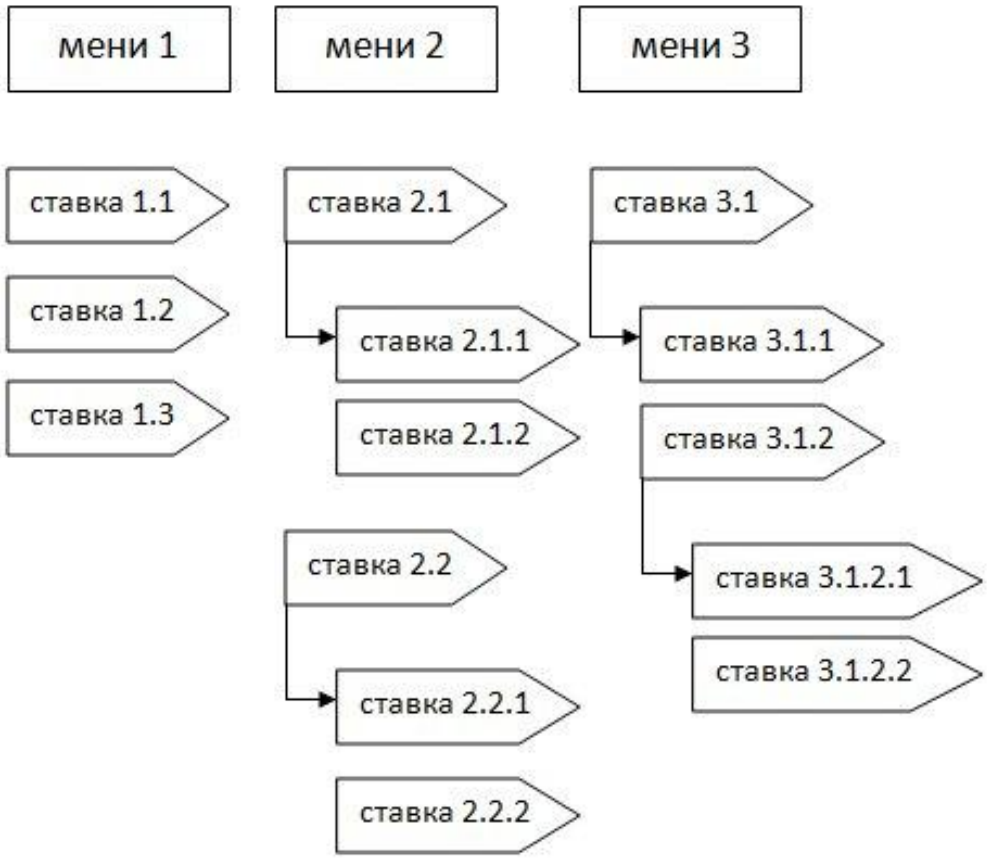
У горњем делу странице приметићемо две операције: *листање корисника* и *управљање корисницима*. Кликом на *управљање*, учитава нам се страница за манипулацију са постојећим корисницима.



У задњој колони табеле излистаних корисника се налазе доступне операције у виду иконица. Прва операција у облику лупе врши приказ корисника из истог реда. Оловка представља операцију измене свих података датог корисника, док је икс знак за брисање корисника. Кликом на брисање појављује се дијалог прозор за потврду брисања корисника.

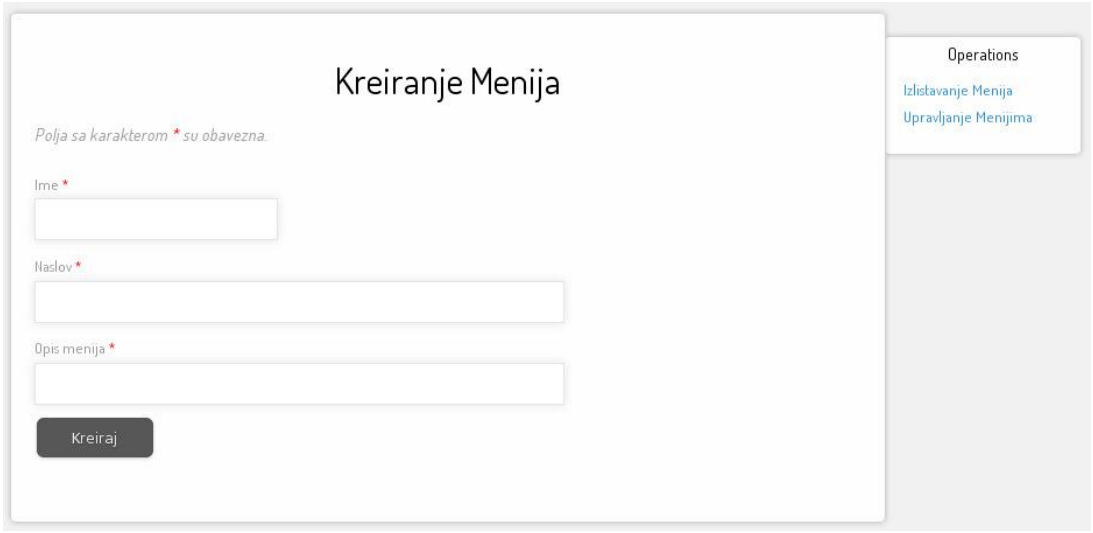
4.2. Менији

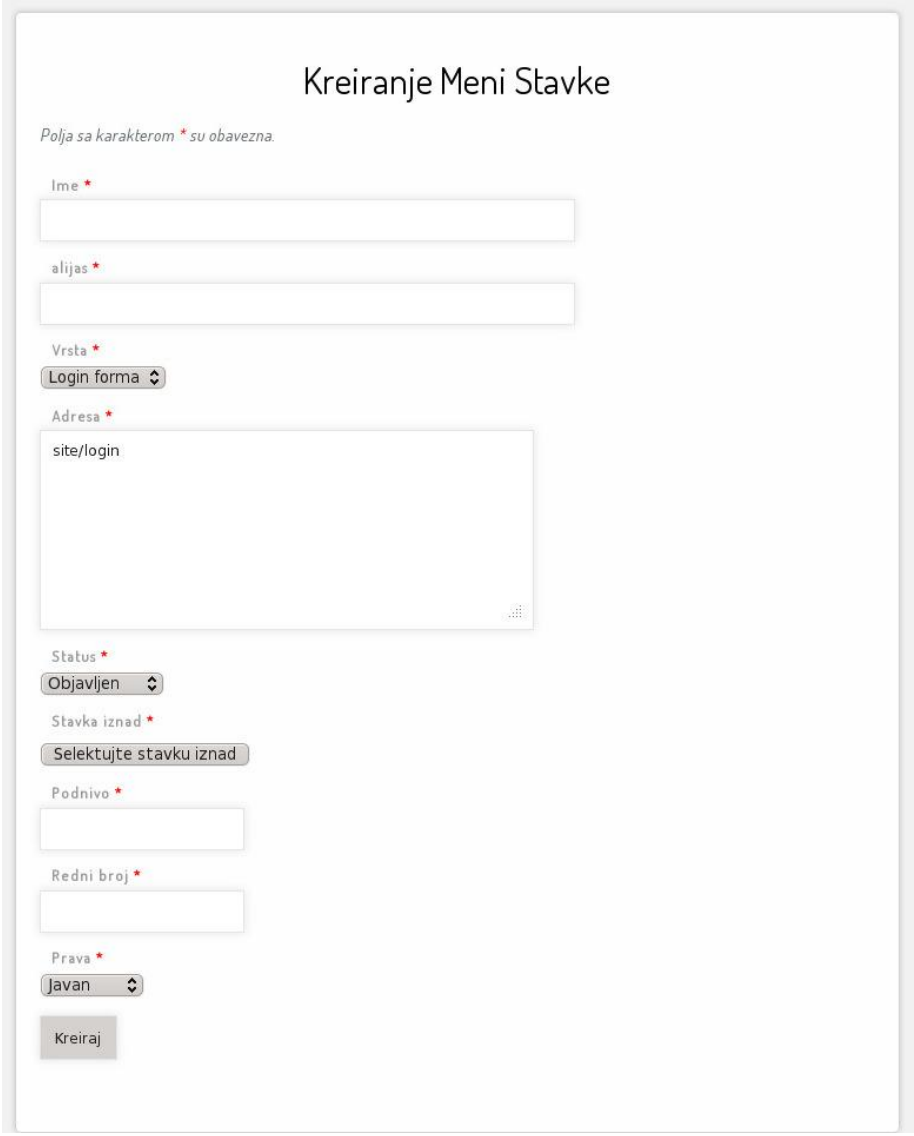
Менији су посебно организовани, и поред мени категорија поседују и мени ставке. Ради бољег разумевања структуре и организације погледајте следећу шему. Мени 1, мени 2 и мени 3 представљају имена менија. Менији су састављени из структурално сложених мени ставки. Мени ставке могу бити различитог типа и различитих позиција. Основни типови ставки јесу следећи: страница, контакт форма, почетна страна, линк (линк ка мануелно написаној адреси), логовање/одлоговање.



На графику су приказана три различита типа менија, први је мени са једни нивоом, други са два нивоа и трећи са три нивоа. У суштини се може направити n број нивоа, али за наш пројекат је довољно користити два нивоа. Приказ менија на сајту се врши преко widget-а о коме ће бити више речи касније.

Кликом на *креирај нови мени* отвара нам се форма за креирање са пољима: име, наслов и описом менија. После креирања новог менија потребно је направити одређени број ставки, због чега се отвара нова страна где на десној страни имамо операцију *креирање ставке*.


Kreiranje Menija
Polja sa karakterom * su obavezna.
Ime *
Naslov *
Opis menija *
Kreiraj
Operations
Izlistavanje Menija
Upravljanje Menijima

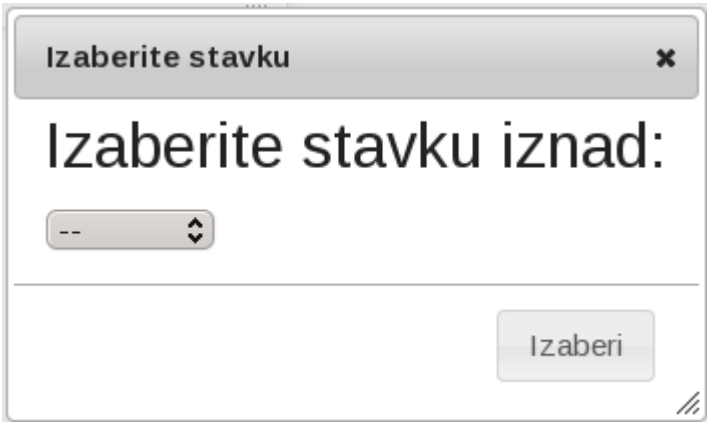

Kreiranje Meni Stavke
Polja sa karakterom * su obavezna.
Ime *
alijas *
Vrsta *
Login forma
Adresa *
site/login
Status *
Objavljen
Stavka iznad *
Selektujte stavku iznad
Podnivo *
Redni broj *
Prava *
Javan
Kreiraj

Код форме за креирање ставки приметићемо претходно поменутог селект поља *врста*. Селектовањем вредности, поље *адреса* се у зависности од селектованог мења:

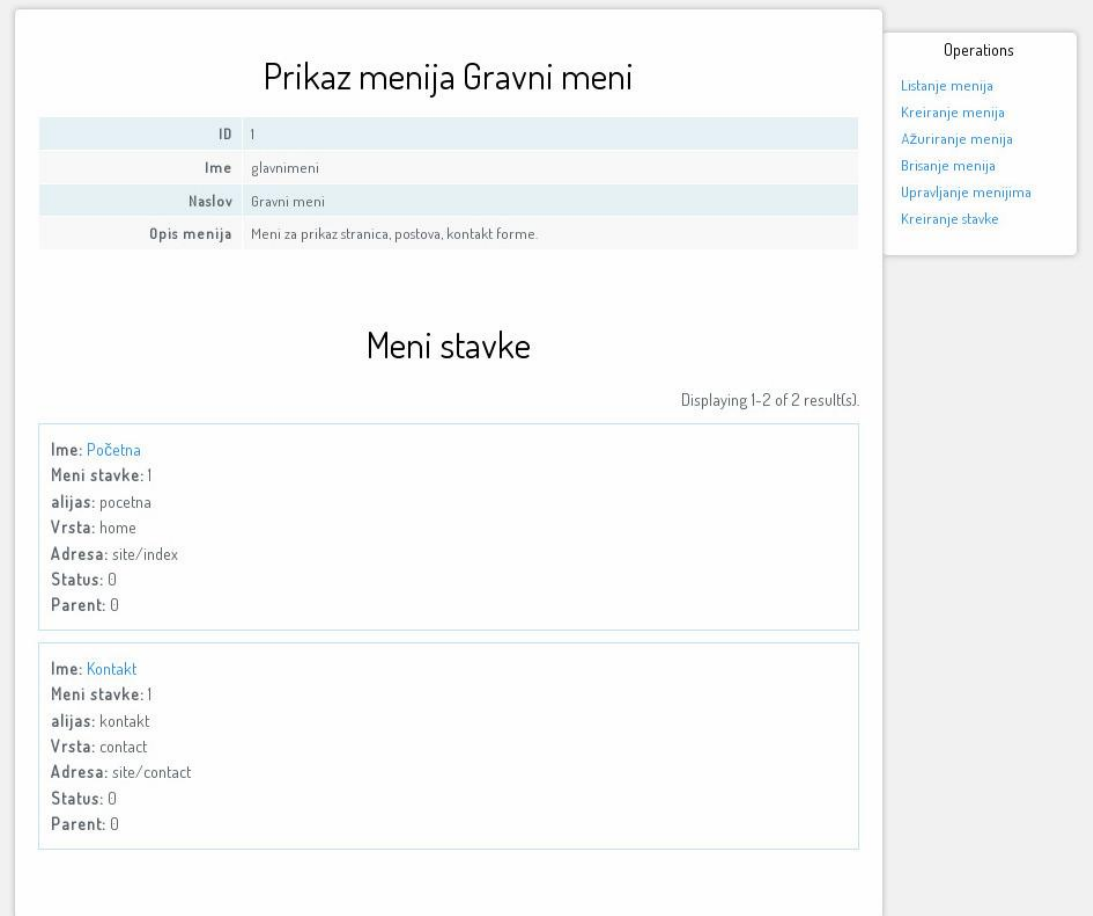
- Страница – појављује се ајах дугме које отвара дијалог прозор за селекцију доступних страница.
- Контакт – уписује *site/contact* унитар поља адреса.
- Почетна – уписује *site/index* унитар поља адреса.
- Линк – корисник ручно уписује адресу у поље адреса.
- Форма за логовање – уписује *site/login* унитар поља адреса.

Сваки унос сем уноса линка има следећу структуру: *контролер/акција*. Због тога је остављено да се поље адреса може ручно изменити у случају да је корисник направио нови контролер или додао нову акцију. Што се тиче форме за логовање, тај линк је динамичан и зависи од тога да ли је корисник већ улогован или није. Уколико јесте, појављује се линк одјавите се.

Помоћу ајах дугмета селектујте ставку изнад, могуће је направити структуру која је приказана на претходном дијаграму. Кликом на дугме отвара нам се дијалог прозор у коме селектујемо ставку у којој ће бити приказана ново креирана ставка.

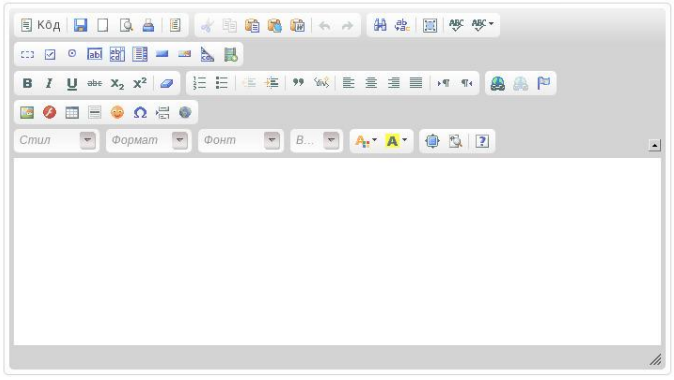


Као и све остало, и мени ставке могу бити јавне или приватне, што омогућава приступ само ауторизованим корисницима. Начин приказа селектованог менија и свих његових ставки као и операције можете видети на следећој слици.



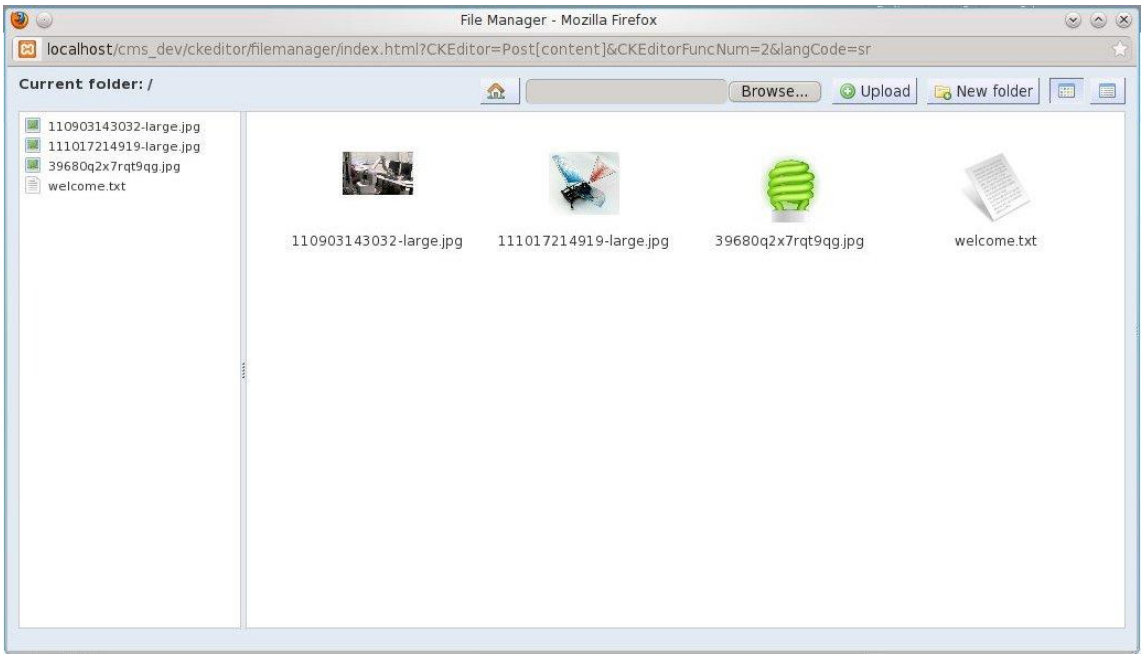
4.3. Странице

Странице и постови користе WYSIWYG едитор који поред стандардних операција код уноса html-а има и могућност слања слика на сервер и њихово позивање на страници, као и могућност убацивања видеа са сајтова као што су youtube, vimeo и други. На почетку је било више кандидата за едитор система, али после тестирања два едитора која спадају у категорију бољих на веб-у избор је пао на CKEdit http://ckeditor.com/.



CKEditor је преведен на српски језик и има велики број додатака (*plugins*) који се појављују у едитору у виду иконица. WYSIWYG едитор значи да уношењем текста и слика у едитор, омогућено нам је да видимо како ће садржај изгледати на сајту јер сваки унос преводи у html код.

Систем је веома прилагодљив, тако да је интегрисана могућност прављења галерија слика као и приказ слика у FancyBox-у http://fancybox.net/. Са FancyBox-ом имам дугогодишња искуства и нашао је место скоро у свим мојим пројектима. Лак је за оптимизацију и интеграцију, а уз то и привлачног је изгледа. Слање слике на сервер и њена селекција се врши преко операције слика у облику мале иконице слике. Када кликнемо на поменуту операцију отвара нам се прозор где поред операција за подешавање селектоване слике имамо и дугме Претражи сервер, која нам отвара нови прозор за слање или селекцију слика.

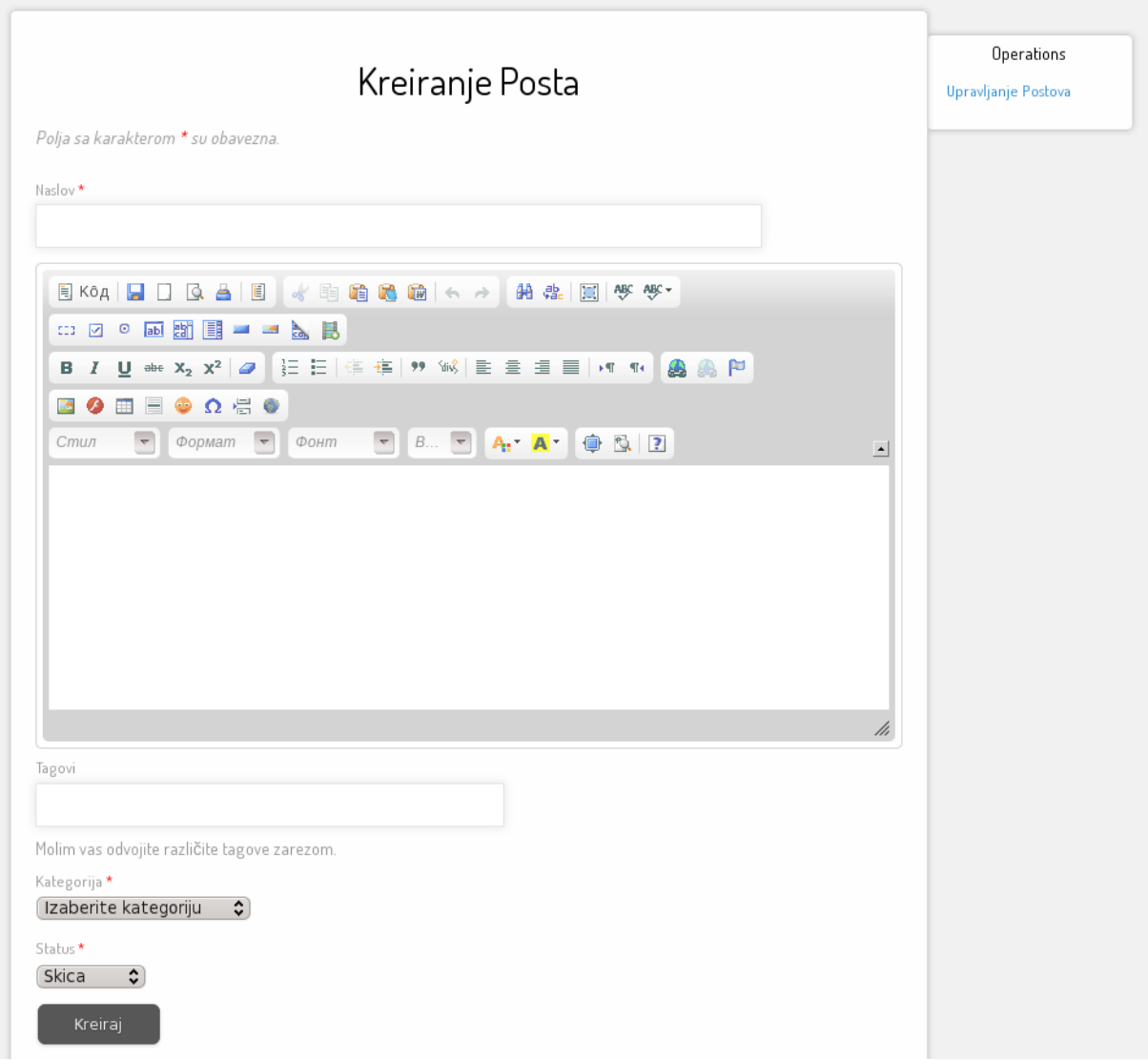


Са леве стране се налази листа доступних слика у приказу листе, док са десне стране имамо прегледнији приказ такође доступних слика. Уколико желимо да пошаљемо нову слику на сервер кликнемо на дугме *Browse* после чега се отвара прозор за селекцију фајла на локалном диску. После селекције кликне се на *Upload* и ваша слика ће се појавити на листи доступних слика, после чега је селектујете. После селекције жељене слике враћате се на прозор са опцијама слике.

Ако желимо да закачимо FancyBox на слику, из доступних опција слике изаберемо Напредни тагови и у поље Stylesheet класе упишите *slika*. Ако желите да вам селектована слика буде део галерије слика са странице упишите *grupa*. То је све што је потребно да се интегрише галерија.

4.4. Постови

За писање постова користимо исти едитор како код прављења страница, тако да иста процедура при писању садржаја важи и код постова. Кликом на *Креирај Пост* отвара се форма са пољима које можете видети на следећој слици:



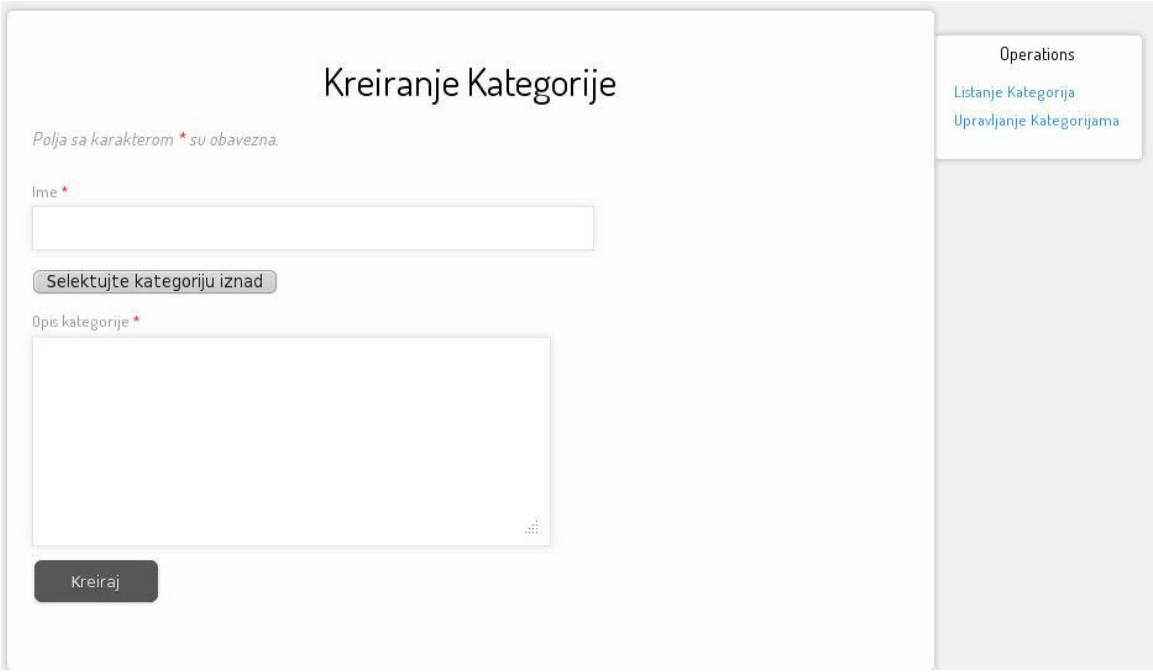
У систему су постови организовани у одређене категорије чије креирање ћемо покрити касније. Тренутно се у систему користи приказ категорија са једним нивоом, али се за касније потребе може без проблема додати нови поглед. На сајту односно на предњем делу система се налази widget (*додатак*) за приказ категорија и постова унутар њих. О овом и другим widget-има биће речи нешто касније. Због такве структуре и организације постова на форми имамо селект поље на коме селектујемо доступне категорије. Изнад тог поља се налази поље *Тагови* где се тагови везани за пост одвајају

зарезима и касније омогућавају претрагу постова као и њихово сортирање. На страници за листање постојећих постова појављују се постови у односу на права која дотични корисник поседује. Ако је обичан корисник видеће само своје постове, а у супротном све.

4.5. Категорије

Као што је поменуто, категоријама се организују постови. Категорије могу бити:

- Категорија првог нивоа
- Категорија другог нивоа



Код креирања категорија једино шта треба напоменути јесте дугме *Селектујте категорију изнад* које отвара дијалог прозор у коме се налазе све креиране категорије првог нивоа. Овај дијалог нам омогућава прављење категорије другог нивоа.

4.6. Додаци (Widgets) у администрацији

Widget-и или додаци у систему представљају мање додатке у виду програма. Захваљујући великој модуларности и проширивости система имамо могућност да врло

брзо направимо додатак. Додаци наслеђују главну yii класу *CWidget* као на следећем примеру.

```
class ImeDodatka  extends CWidget {

}
```

Оно што је за нас најбитније јесте како позвати већ постојећи додатак као што је рецимо главни мени у администрацији. За позивање додатка користимо следеће:

```
<!--?php $this--->widget('ImeDodatka'); ?>
```

У већини случајева наведено нам је сасвим довољно за покретање додатка са својим стандардним подешавањима. Међутим често је потребно навести и одређене параметре као што је то случај код нашег додатка за приказ менија, где се може послати параметар за падајући мени или обичан, статични. У додатку се онда бира између два различита погледа за приказ менија. Позивање се врши са следећом линијом кода:

```
<?php $this->widget('AdminMenu',array()); ?>
```

Администрација поседује следеће додатке:

- AdminMenu (главни административни мени)
- RecentComments (последњи коментари)
- TagCloud (за препознавање тагова приликом куцања)
- UserIdentity (прикупљање више података у вези корисника)

Додаци су мали програми којима дајем увек предност, јер за кратко време може да се напише користан програм који поседује све предности објектне архитектуре система. Нешто детаљније о *CWidget* класи можете наћи на адреси: http://www.yiiframework.com/doc/api/1.1/CWidget.

4.7. Модули коришћени у администрацији

Модул је независна софтверска јединица која се састоји од модела, погледа, контролера и осталих подржаних компоненти. У многим аспектима модул потсећа на апликацију. Једина разлика је што модул не може функционисати сам већ само у некој апликацији. Корисници могу приступити контролерима модула као што приступају нормалним контролерима апликације.

Модули су корисни у више сценарија. Веће апликације можемо изделити на више модула, а сваки модул да буде развијен одвојено.

Оно што је добро код yii-а је што има базу модула коју можете скинути на адреси: http://www.yiiframework.com/extension/ и врло лако интегрисати у систем. Модули се инсталирају тако што се скинута архива отпакује у директоријум /cms_dev_vx.xx/protected/backend/modules уколико желите модул у администрацији ili u /cms_dev_vx.xx/protected/modules за ваш сајт.

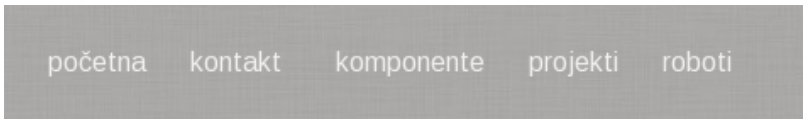
Модули коришћени у администрацији:

- User – модул за управљање корисницима система
  http://www.yiiframework.com/search/?q=user
- Rights – модул за комплекснију контролу права корисника
  http://www.yiiframework.com/extension/yii-user-management

5. Блог за размену индивидуалних пројеката из области роботике

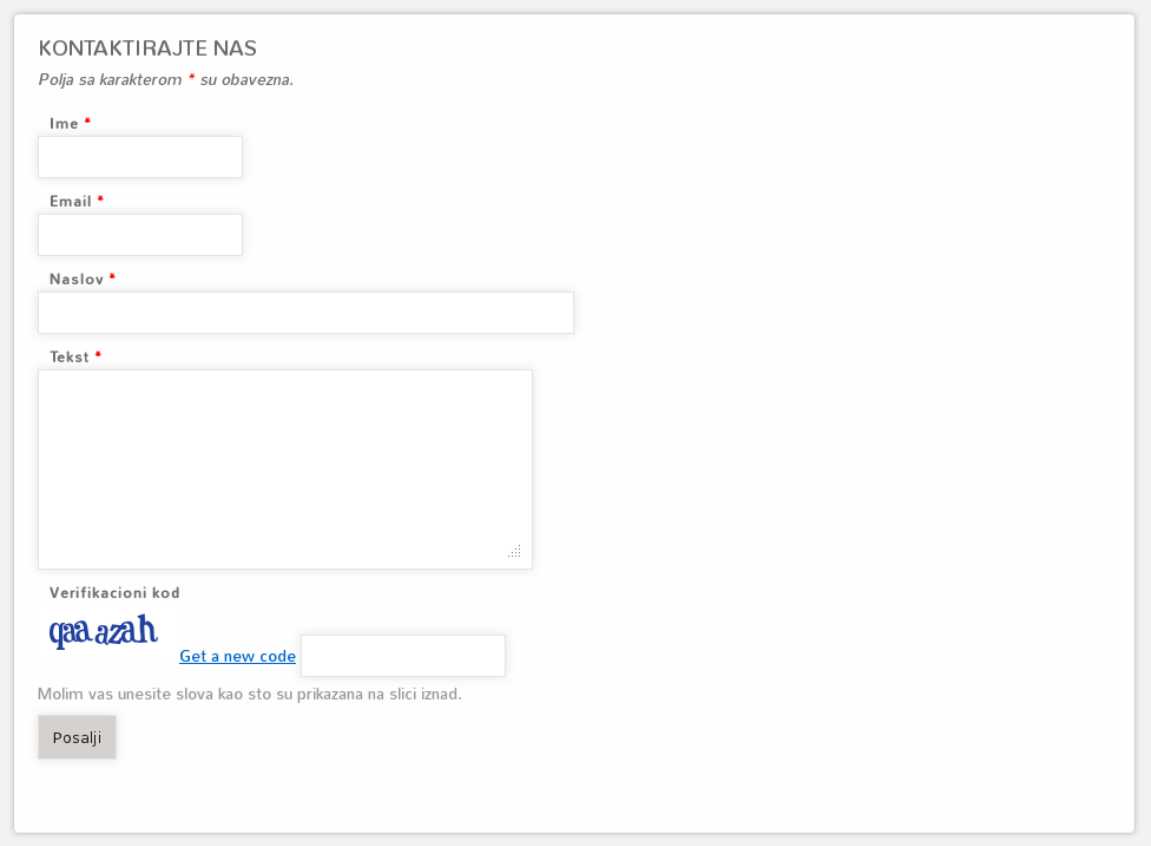
Ради презентовања креираног система за управљање садржајем, креиран је пример блога за размену индивидуалних пројеката, објављивања вести и опис компоненти из области роботике. Овај блог представља прости сајт, чија је намена искључиво презентативна и самим тим садржи само основне и најкорисније компоненте. Сајт садржи два различита менија:

- Главни мени (*један ниво*)
- Мени за листање категорија (*један ниво*)



Главни мени у себи садржи само две ставке: почетна и контакт. Ставка почетна као што само име говори, представља линк који води до почетне (*прве*) стране. Контакт представља линк до стране са контакт формом, који у овом случају има адресу: *r=site/contact*.

Ова контакт форма је основна, и садржи следећа поља: име, email, наслов, текст и верификациони код. Као и сваку компоненту, и контакт форму можете накнадно изменити и додати или избацити одређена поља. Верификациони код је код за врификацију пошаљиоца, односно утврђивање да ли је пошаљиоц особа или скрипта. Поља са звездицом су обавезна и морају се попунити јер форма поседује ајах верификацију сваког поља. Ако неко поље није попуњено како треба или је празно, поље ће добити црвену боју, док у супротном зелену. Текст порука стиже на email администратора сајта.



Мени за листање категорија је мени односно додатак (*widget*) који листа само креиране и објављене категорије. Постоји више погледа које можемо користити:

- menucat.php – погледа за приказ категорија првог нивоа
- menucat_home.php – поглед за посебан приказ категорија првог нивоа на почетној страни сајта.
- multicat_multilevel.php – поглед за приказ категорија првог и другог нивоа, у падајућем менију.

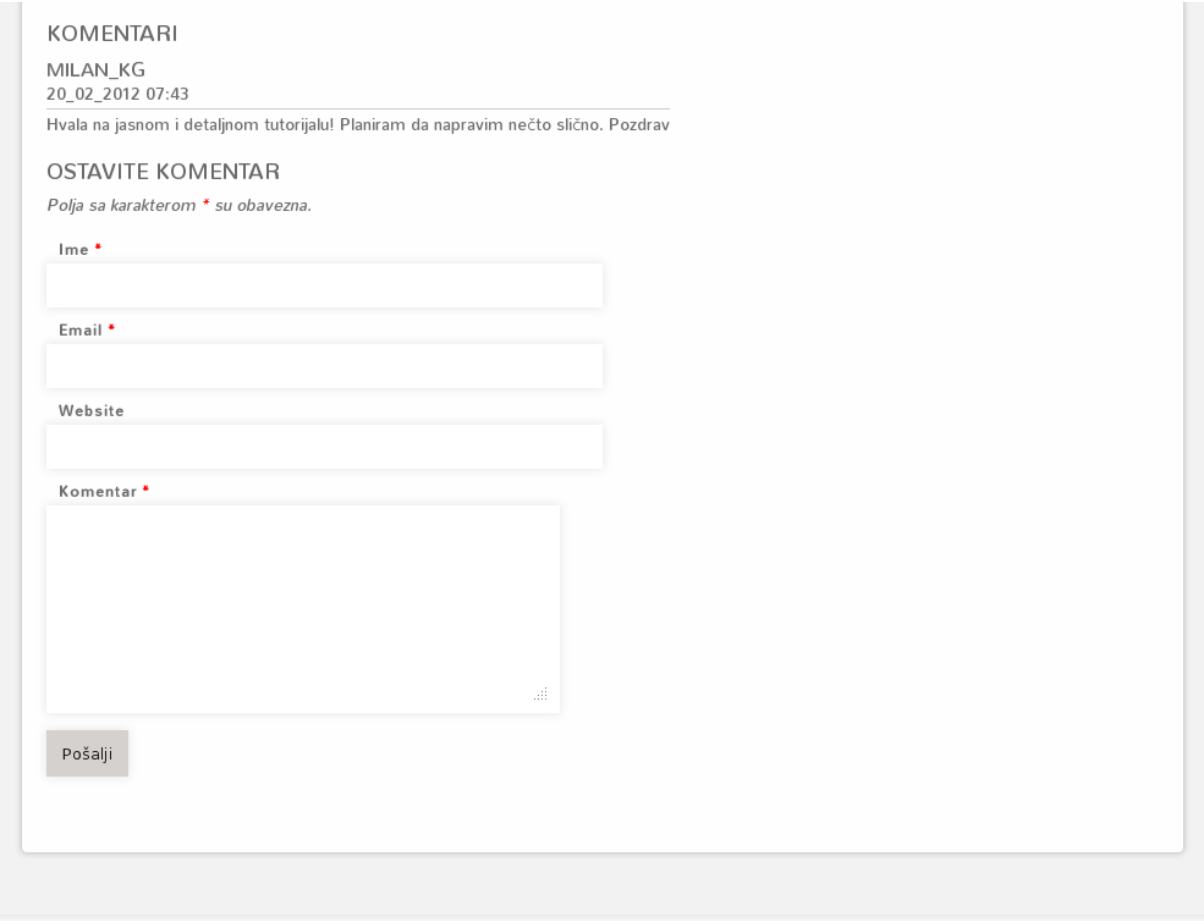
У менију за приказ категорија имамо три категорије првог степена после чијег селектовања се на страни у случају постојања листају подкатегорије.

- компоненте
- пројекти
- роботи

Компоненте садрже опис компоненти у вези са роботиком као што су: контролери, мотори, сензори, чипови, итд. Тренутно у овој категорији се види опис Arduino контролера.

Пројекти садрже индивидуалне пројекте из области роботике, и објашњења или упутства како да се направи робот или нека од компоненти. Поред обичног текста и слика могуће је видети и ведео. Ако уђемо у категорију пројекти, видећемо један пример пројекта, који садржи упутство за прављење вашег првог робота.

Као што се може приметити испод описа поста имамо индикатор коментара који показује да имамо један коментар на нашем посту. Форма за коментар се налази на дну сваког поста, а одобрени постови од стране администратора или постера се налазе излистани изнад форме. Као пример коментара имамо коментар од корисника чије име је MILAN_KG и који одговара на пост о упуству за прављење првог робота.



Категорија роботи садржи вести из области роботике које су категоризоване у три категорије:

- свемирска истраживања
- медицина
- индустрија

Ако селектујемо категорију индустрија, видећемо листу вести поређаних по датуму објављивања.

20_02_12

Leteći roboti sagradili kulu od 6 metara

Bili smo veoma impresionirani mogućnostima ovih letećih robota, kada smo prvi put objavili ovaj dogadjaj krajem Novembra. Sada FRAC centar u Orleanu, Francuska je objavio video swarm robota u akciji tokom izložbe. Pod nazivom "Flight Assembled Architecture" prezentacija flote quadrocopter-a koji grade kulu od 6 metara sa 1,500 cigala od poliester pene. Svaki quadrocopter je opremljen sa prilagodljivom elektronikom i senzorima zbog precizne kontrole letenja, dok takodje pruža pre-programirane putanje letenja, gde uključuje lukove i spirale.

KOMENTARI: 0

20_02_12

Leteći GRASP Lab roboti

GRASP Lab na univerzitetu Pensilvanija je ove nedelje objavio video koji pokazuje njihov novi pogled na GRASP leteće robote. Sada pokazuju robote sa mnogo složenijim ponašanjem nego ranije, u fliti letećih uredjaja koji se kreću u čoporima, kreće se kroz prostor sa preprekama, prevrće se u letu i zadržava poyiciju, i menja formaciju letenja. Video pokazuje da je razvojni tim , Alex Kushleyev, Daniel Mellinger, and Vijay Kumar, u stanju da prikaže složeno autonomno ponašanje. Ključna reč je okretan. Blogovi entuzijasta za robotiku su na objavljeni video rekli da su impresionirani na preciznost pri izvodjenju promene formacija, i generalno na znanje robota gde je čije mesto u prostoru.

KOMENTARI: 0

6. Закључак

У овом раду смо се упознали са техникама и веб технологијама које се користе за израду како основних тако и комплексних веб апликација. Све се више наших података сели на веб сервере који поред већ широко прихваћеног складишћења личних и пословних података све више са њима и управља. Ту се јавља стална потреба за развијањем и усавршавањем система за управљање подацима као и радних окружења (*frameworks*) који нам омогућавају лакше и брже пројектовање веб апликација.

У већем делу рада су се описале могућности и карактеристике радног окружења Yii као једно од бољих решења од гомиле других сличних. При одабиру радног окружења, обратио сам пажњу да поред високих перформанси обезбеди и лаку интеграцију нових технологија и самим тим омогући прављење савременијих апликација. Већ сада видимо велике могућности *HTML5* апликација као и повећане потребе за језицима као што су *JavaScript*.

CMS систем направљен коришћењем Yii окружења за потребе овог рада, представља систем који осликава све позитивне стране Yii окружења а нарочито брзину рада. Данас постоји велики број сличних система од којих се CMS систем овог рада издваја високим перформансама и модуларношћу. Треба издвојити и позитивну страну отвореног кода који се везује за бржи развој апликације.

Будуће генерације система за управљање садржајем се већ сада назиру, пратећи развој свих веб апликација иду ка већој динамици и независности. Захваљујући компанијама као што је Google и њиховим иновацијама у области веб технологија можемо предвидети скору велику миграцију података и апликација са персоналних рачунара на веб сервере. Очекује се већа експанзија cloud система као и оперативних системима заснованим искључиво на веб технологијама као што је B2G. Време ће показати да ли ће се на прави начин искористити потенцијал веб-а и испунити наша велика очекивања!

7. Литература

1. Jeffrey Winesett, Agile Web Application Development with Yii1.1 and PHP5, Август 2010, Birmingham.
2. Alexander Makarov, Yii 1.1 Application Development Cookbook, Август 25, 2011.
3. http://www.yiiframework.com/doc/
4. http://www.yiiframework.com/tutorials/
5. http://help.github.com/create-a-repo/