PHP-POHJAISET OHJELMOINTIKEHYKSET

Antti Humalamäki

Opinnäytetyö
Marraskuu 2011

Ohjelmistotekniikka
Tekniikan ja liikenteen ala

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖN
KUVAILULEHTI

Tekijä(t) HUMALAMÄKI, Antti	Julkaisun laji Opinnäytetyö	Päivämäärä 30.11.2011
	Sivumäärä 77	Julkaisun kieli Suomi
	Luottamuksellisuus ( ) saakka	Verkkojulkaisulupa myönnetty (X)

Työn nimi
PHP-POHJAISET OHJELMOINTIKEHYKSET

Koulutusohjelma
Ohjelmistotekniikka

Työn ohjaaja(t)
PELTOMÄKI, Juha

Toimeksiantaja(t)
Sysdrone Oy

Tiivistelmä

Opinnäytetyön tavoitteena oli tehdä Sysdrone Oy:lle vertailevaa tutkimusta PHP-kielellä toteutettujen avoimen lähdekoodin ohjelmointikehyksistä. Työn motivaationa oli löytää työn toimeksiantajan tulevaisuuden projekteja varten sellaisia sovelluskehitysalustoja joista löytyvät valmiina perustoiminnallisuudet kuten käyttäjienhallinta, autentikaatio ja lokalisointi, samalla myös huomioiden sellaiset ominaisuudet kuten kehityksen helppous ja testattavuus. Lisäksi kiinnitettiin huomiota mahdollisiin sisällönhallintaominaisuuksiin, dokumentaation tasoon ja käyttäjäyhteisön aktiivisuuteen.

Tutkittaviksi järjestelmiksi rajautuivat varsinaiset ohjelmointikehykset Symfony, Zend ja Yii sekä sisällönhallintajärjestelmät Joomla ja Drupal. Kaikkia ohjelmointikehyksiä ja sisällönhallintajärjestelmiä tutkittiin empiirisin menetelmin virtuaalikoneympäristössä, mutta myös kirjallisuudesta tietoa hakien.

Kaikista tutkituista kehysjärjestelmistä kerättiin laajasti aineistoa, josta on hyötyä toimeksiantajan tulevaisuuden projektien suunnittelussa. Molempien ohjelmointikehysten ja sisällönhallintajärjestelmien sisällyttäminen samaan tutkimukseen, aiheutti vaikeuksia niiden väliselle vertailulle. Järjestelmät pisteytettiin toimeksiantajan määrittelemien vaatimusten mukaisesti. Yksittäistä parasta järjestelmää ei kuitenkaan voida nimetä, sillä yksittäisen ohjelmistoprojektin vaatimukset vaikuttavat aina siihen, millainen kehysjärjestelmä kannattaa valita.

Avainsanat (asiasanat)
Avoin lähdekoodi, Ohjelmointikehys, PHP, Drupal, Joomla, Symfony, Zend, Yii

Muut tiedot

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

DESCRIPTION

Author(s)	Type of publication	Date
HUMALAMÄKI, Antti	Bachelor´s Thesis	30112011
	Pages 77	Language Finnish
	Confidential ( ) Until	Permission for web publication (X)

Title
PHP SOFTWARE FRAMEWORKS

Degree Programme
Software Engineering

Tutor(s)
PELTOMÄKI, Juha

Assigned by
Sysdrone Oy

Abstract

The aim of the Bachelor's Thesis was to carry out a comparative research of PHP-based open source software frameworks for Sysdrone Oy. The motivation behind this thesis work was to find possible software framework for the assigner's future projects. The framework should preferably have implemented basic functionalities such as user management, authentication, and localization. Additionally, it should consider possible content management features, quality of documentation and activity of community.

The software frameworks Symfony, Zend and Yii, plus content management systems Joomla and Drupal were selected for closer inspection. These systems were studied empirically in a virtual machine environment, while also searching information from written sources.

Plenty of research material was collected for each software framework. This material will be useful as the assigner company plans its future projects. Including both basic software frameworks and more feature-rich content management systems in the same study caused difficulties during the comparison between those groups of systems. Each system was given a score based on the requirements defined by the assigner. One single best system cannot be named - instead it is advisable to choose a framework based on the task at hand.

Keywords
Open source, Software framework, PHP, Drupal, Joomla, Symfony, Zend, Yii

Miscellaneous

SISÄLTÖ

KUVIOT

TAULUKOT

TERMIT JA LYHENTEET

Active record On suunnittelumalli (engl. *design pattern*), jonka mukaan mallina toimivalla oliolla tulee olla tietokannan kenttiä vastaavat samannimiset ominaisuudet (engl. *property*).

Ajax JavaScriptiin perustuva tekniikka, jonka avulla web-sivun sisältöä ja ulkoasua muutetaan, ilman että koko sivua ladataan uudelleen. Tekniikan tarkoituksena on tehdä verkkopalvelusta enemmän työpöytäsovelluksen kaltainen.

Apt-get On Debianiin pohjautuvien GNU/Linux-käyttöjärjestelmien pakettien hallintaan tarkoitettu komentorivityökalu, jonka avulla voidaan ohjelmia ja ohjelmistoja asentaa keskuspalvelimilta automatisoidusti niiden oletuskonfiguraatioilla. Riittää, että tunnetaan asennuspaketin nimi.

Asennusvelho (wizard)

Ohjelmiston asennuksessa avustava sovellus jonka tarkoituksena on automatisoida ja helpottaa monimutkaisia operaatioita.

Automaattitestaus Testauksen automatisoinnilla toteutetaan jatkuvan integraation periaatetta, joka tähtää siihen, että ohjelmisto on mahdollisimman usein julkaisukelpoinen. Käytännössä tämä tarkoittaa sitä, että kaikkia tai keskeisintä osaa testejä suoritetaan automaattisesti integraatiopalvelimella tietyin välein, esimerkiksi laukaistuna aina kehittäjän viedessä uusia muutoksia versionhallintaan.

Avoin lähdekoodi Avoimen lähdekoodin ohjelmistoa saa kuka hyvänsä käyttää, muokata tai jakaa edelleen. Erilaisten vapaan lähdekoodin lisenssien keskeisin tekijä yrityksen kannalta on usein se rajaako lisenssi sitä, että myös vapaan lähdekoodin ohjelmistosta muokatun tuotteen lähdekoodien tulee olla vapaasti jaettavissa samalla lisenssillä.

Bootstrapping Bootstrappingillä viitataan siihen osaan sovellusta, jossa sijaitsee sovelluksen aloituskohta ja siihen liittyvät alustukset.

CRUD (Create, Read, Update and Delete) Tiedon luominen, lukeminen, muuttaminen ja tuhoaminen ovat massamuistin käsittelyn perustoiminnot.

CRUD-liittymä Käyttöliittymä, joka toteuttaa kaikki CRUD toiminnot.

Getteri/Setteri Erityisesti olion ilmentyjämuuttujien hakemista ja muuttamista varten luodut metodit, nimetään yleensä muodossa *get/set<Muuttujan nimi>*.

Malli (model) Sovelluksen tiedonkäsittely, suorittaa toiminnot ja tarjoilee ohjaimelle sen tarvitsemaa dataa halutussa muodossa.

Nimiavaruus Nimiavaruus (engl. *Namespace*) on tietojen ja luokkien kapselointiin tarkoitettu nimeämistapa. Luokkien asettaminen kuulumaan nimiavaruuksiin ryhmittelee yhteen kuuluvia käsitteitä, ja helpottaa järjestelmien integraatioita estämällä yhteentörmäykset.

Näkymä (view) Sovelluksen käyttäjälle näkyvät osat, käyttöliittymäkomponentit ja mallista saatujen tietojen esittäminen.

Ohjelmointikehys ”Puolivalmis” ohjelmisto, joka tarjoaa erilaisia valmiita toimintoja kuten esimerkiksi autentikaation jatkokehittäjän käyttöön. Ero ohjelmointikehyksen ja ohjelmointikirjaston välillä

voitaisiin määritellä siten, että siinä missä kehittäjän koodi kutsuu ohjelmointikirjaston koodia, ohjelmointikehys puolestaan suorittaa kehittäjän koodia.

ORM (Object-relational mapping) ORM-järjestelmä virtualisoi tietokantayhteyden, siten että kehittäjä voi hoitaa tietojen tallentamiseen liittyvän ohjelmointityön olioparadigman mukaisesti tuntematta sen tarkemmin käytössä olevan tietokannan toimintaa.

Ohjain (controller) Sovelluksen toimintalogiikka, ohjaa käyttäjältä tulevat käskyt mallille ja mallilta tulevat tiedot näkymään.

PHP (PHP Hypertext Preprocessor) Välikielen kautta tulkattava olio-ohjelmointikieli, jota käytetään enimmäkseen web-sovelluksien kehittämiseen.

Reititin (Router) Palvelinsovelluksissa reitittämisellä tarkoitetaan sovelluksen sitä osaa, joka huolehtii URL-osoitteen purkamisesta parametreiksi. Esimerkiksi osoite http://esimpalvelin.fi/käyttäjä/näytä/123 voitaisiin reitittimessä purkaa siten, että reitittin kutsuisi *käyttäjä*-ohjainta parametreilla *näytä*

ja *123*. Ohjain sitten käskisi näkymää näyttämään käyttäjän numero 123 tiedot.

Riippuvuusinjektio Riippuvuusinjektointi (engl. *dependency injection*) on suunnittelumalli, jossa sovelluksen riippuvuudet tarjotaan sovellukselle ikään kuin palveluina. Nämä palvelut tuodaan (injektoidaan) sovellukseen sen luomisen yhteydessä. Menetelmä väljentää kytköksiä sovelluksen ja sen riippuvuuksien välillä tehden sovelluksesta modulaarisemman. Tämä helpottaa uudelleenkäyttöä ja samalla mahdollistaa palveluiden yksikkötestauksen.

Sisällönhallintajärjestelmä (CMS)

Tässä opinnäytetyössä sisällönhallintajärjestelmällä tarkoitetaan sellaista järjestelmää johon pystytään luomaan sisältöä käyttöliittymän avulla ilman merkittävää ohjelmointityötä. Tällaista järjestelmää voidaan kutsua myös julkaisujärjestelmäksi.

Template Mallipohja, johon järjestelmän näkymä korvaa merkityt muuttujat ohjaimelta saamansa datan perusteella. Mallipohjien käytön tarkoituksena on erotella HTML-koodi ohjelmakoodista ja tehdä web-sovelluksen HTML-rakenteen ja ulkoasun muokkaamisesta nopeampaa.

TODO (To do), lähdekoodissa käytetty kommentti merkintä, jolla osoitetaan, että jokin ominaisuus on vielä keskeneräinen.

URL Verkko-osoite, muodostuu protokollaosasta, erotinmerkeistä (://), tietokoneen osoitteesta ja mahdollisesti myös porttinumerosta, polusta ja parametrijoukosta.

XPath (XML Path Language) XPath-kielen avulla pystytään poimimaan XML- tai HTML-dokumentista osia. XPath-haut ovat keskeinen työkalu selainautomatisoituja hyväksymistestejä suunniteltaessa osoittamaan, mistä testin tarvitsemat hyperlinkit ja tiedot löytyvät.

1 TYÖN LÄHTÖKOHDAT

1.1 Tavoite

Opinnäytetyön tavoitteena oli tehdä vertailevaa tutkimusta PHP-kielellä toteutettujen avoimen lähdekoodin ohjelmointikehyksistä. Työn motivaationa oli löytää toimeksiantajan tulevaisuuden projekteja varten sellainen sovelluskehitysalusta, josta löytyvät valmiina perustoiminnallisuudet kuten käyttäjienhallinta, autentikaatio ja lokalisointi, samalla myös huomioiden sellaiset ominaisuudet kuten kehityksen helppous ja testattavuus. Lisäksi tuli kiinnittää huomiota mahdollisiin sisällönhallintaominaisuuksiin, dokumentaation tasoon ja käyttäjäyhteisön aktiivisuuteen.

1.2 Toimeksiantaja

Työn toimeksiantajana oli Sysdrone Oy. Sysdrone on vuonna 2006 perustettu jyväskyläläinen ohjelmistoalan yritys, joka keskittyy pääasiallisesti terveysteknologian ratkaisuihin. Sysdrone käyttää kehitystyössään enimmäkseen Microsoftin tuotteita kuten C# ja ASP.NET, mutta sillä on myös useita PHP:llä toteutettuja projekteja. Sysdrone Oy toimii osana Protacon konsernia saumattomassa yhteistyössä Protacon Solutions Oy:n kanssa.

> *Protacon Group on suomalainen teknologia-alan suunnittelu- ja palveluyritys, joka tuottaa asiakkailleen luotettavia, tehokkaita ja joustavia ratkaisuja tuotannon, ylläpidon ja projektoinnin tarpeisiin. Tavoitteena on parantaa asiakkaiden kilpailukykyä.*
>
> (Protacon 2011)

1.3 Tarve

Protacon Solutionsilla on käytössään talon sisällä rakennettu PHP-pohjainen kehysjärjestelmä web-projekteja varten, jota myös Sysdronen puolella useissa projekteissa käytetään. Kuitenkin on olemassa mahdollisuus, että tietyn tyyppisiin projekteihin kyseistä kehysjärjestelmää ei voida käyttää. On esimerkiksi mahdollista, että jotkin julkishallinnon ratkaisut vaativat järjestelmän lähdekoodien olevan täysin avointa. Tällöin talon oman kehysjärjestelmän käyttö ei ole mahdollista, vaan pitää löytää jokin avoimen lähdekoodin järjestelmä. Myöskään kokonaisen kehysjärjestelmän rakentaminen tyhjästä ei ole järkevää, kun kerran valmiitakin ratkaisuja on olemassa. Tämän opinnäytetyön tavoite ei ollut löytää täydellistä ohjelmointikehystä, vaan tehdä valmistelevaa tutkimusta tukemaan toimeksiantajaa valinnan tekemisessä.

1.4 Vaatimukset

Työn toimeksiantaja oli määritellyt kahdeksan vaatimusta, joihin opinnäytetyössä tulisi löytää vastaukset jokaisen kehysjärjestelmäkandidaatin osalta:

- *Autentikaatio/käyttöoikeuksien hallinta/käyttäjien hallinta*
- *Käytännön käyttö/ohjelmoinnin helppous*
- *CRUD-liittymien teko*
- *Lokalisointi*
- *Kehittäjäyhteisön aktiivisuus/tulevaisuuden näkymät/onko taustalla isoja toimijoita jotka käyttävät kehysjärjestelmää.*
- *Dokumentaatio*
- *Testattavuus (PHPUnit/Selenium)*
- *CMS-Ominaisuudet*

(Särkkä 2011).

Yleisesti ottaen tutkimuksella pyrittiin löytämään avoimen lähdekoodin kehysjärjestelmä, jossa on mahdollisimman kattavasti näitä vaatimuksista valmiina ja hyvin toteutettuna.

Työn toimeksiantaja määritti tutkittujen vaatimusten pisteytystä varten suhteelliset painoarvot:

- asennus 10 %
- käyttäjienhallinta 10 %
- käyttökokemus / helppous 30 %
- CRUD-toteutus 10 %
- dokumentaatio 20 %
- lokalisointi 10 % ja
- testattavuus 10 %.

2 TYÖN SUUNNITELMA JA ALOITUS

2.1 Esitutkimus

Aivan aluksi koottiin lista mahdollisimman useasta tarjolla olevasta vapaan lähdekoodin ohjelmointikehys- ja sisällönhallintajärjestelmä-vaihtoehdoista. Tämä alustava taulukko 1 koottiin Wikipedian ja phpframeworks.net-sivuston tuntemien järjestelmien perusteella. Sellaisia kehysjärjestelmiä, joihin ei ollut tullut päivityksiä kuluneen vuoden aikana, ei otettu tarkasteltavien listalle mukaan lainkaan.

TAULUKKO 1. Järjestelmäkandidaatit

Nimi	**Versio**	**Julkaisun päivä-määrä**
Symfony	2.0.1	26.8.2011
FuelPHP	1.0.1	23.8.2011
Zend	1.11.10	3.8.2011
CMS made simple	1.9.4.3	27.8.2011
Agile toolkit	4.1	22.8.2011
Drupal	7.7	28.7.2011
Prado	3.1.10	27.7.2011
CakePHP	1.3.11	26.7.2011
Kohana	3.2	24.7.2011
Agavi	1.0.6	23.7.2011
PHPDevShell	3.0.4	28.6.2011
Yii	1.1.8	26.6.2011
Kajona	3.4.0	19.6.2011
Lithium	0.10	18.6.2011
Outglow	2.1.6	5.6.2011
Modx	2.1.0	24.5.2011
Rain Framework	2.3.1	3.5.2011
Joomla!	1.6.3	18.4.2011
Alloy	0.7.2	12.4.2011
CodeIgniter	2.0.3	7.4.2011
e107	0.7.25	6.4.2011

(Taulukko 1 jatkuu seuraavalla sivulla)

(Taulukko 1. jatkoa edelliseltä sivulta)

Horde	4.0	5.4.2011
AppFlower	1.0	30.3.2011
Quick PHP	1.4.3	21.3.2011
PHP on TRAX	0.17.0	15.3.2011
DooPHP	1.4.1	23.2.2011
Solar	1.1.2	4.2.2011
SilverStripe (Sapphire)	2.4.5	2.2.2011
Qcodo	0.4.20	24.12.2010
Seagull	0.6.8	3.12.2010
OpenDelight	1.1.4	30.11.2010

Opinnäytetyön suorittamisen ajaksi lukittauduttiin ohjelmointikehysten taulukossa 1 näkyviin versioihin, vaikka opinnäytetyöprosessin aikana tutkituista järjestelmistä julkaistiinkin uusia versioita.

2.2 Karsinta

Aivan ensimmäiseksi kaikki taulukossa 1 mainitut kehysjärjestelmät asennettiin testiympäristöön ja silmäiltiin niiden oletus- ja esimerkkikäyttöliittymät lävitse. Joukosta pudotettiin pois sellaiset järjestelmät, joilla selvästikään ei ollut mahdollisuuksia täyttää keskeisimpiä vaatimuksia, ja sellaiset järjestelmät, joista selvästi paistoi läpi harrastelijamaisuus.

Viimeiseksi listalta pudotettiin pois myös sellaiset järjestelmät, joilta ei teknisestä lupaavuudestaan huolimatta löytynyt referenssejä tunnettujen www-sivustojen toteutuksiin.

Jäljelle jäivät sisällönhallintajärjestelmät Joomla! ja Drupal sekä ohjelmointikehykset Symfony, Zend ja Yii.

Kuten kuviossa 1 esitetyistä Googlen hakumääristä voidaan vertailla, jäljelle jääneiden ryhmästä selvästi tunnetuimpia ovat Joomla! ja Drupal, ne kun helppojen CMS-ominaisuuksien ansiosta soveltuvat hyvin myös tavanomaiseen kotisivukäyttöön. Kenties juuri näiden

sisällönhallintaominaisuuksien puuttumisen vuoksi varsinaiset ohjelmointikehykset olivat työläämpikäyttöisiä ja siten jäivät selvästi marginaaliin.

KUVIO 1. Sisällönhallintajärjestelmät Googlen trendeissä

2.3 Virtuaalikone testiympäristönä

Kehityskokeiluja helpottamaan päätettiin perustaa virtuaaliympäristö. Virtuaalikoneeksi valittiin VirtualBox ja siihen asennettavaksi käyttöjärjestelmäksi tuorein palvelinversio Ubuntu GNU/Linuxista. Virtuaalikoneelle määriteltiin kaksi virtuaalista verkkokorttia, ensimmäinen Internetiin pääsyä varten ja toinen siksi, että voidaan suorittaa selainautomatisoituja Selenium-testejä isäntäkoneen Windows 7 -ympäristöstä käsin.

Virtuaalikoneelle haettiin asennuspaketit kaikista ohjelmointikehyksistä.

Automaattitestejä varten luotiin Visual Studio 2010:llä testiprojekti. Harkittiin, että automaattitestejä voisi käyttää myös testivetoiseen

kehittämiseen. Esimerkiksi voitaisiin ensin tehdä selainautomatisoitu testi sisään kirjautumista varten ja sitten vasta luoda käyttäjätunnukset järjestelmään. Ideasta kuitenkin luovuttiin, kun havaittiin järjestelmien toimivan varsin tavalla toisiinsa nähden.

Lisäksi yksikkötestien suorittamista varten asennettiin PHPUnit. Pelkkä *apt-get install* ei tässä tapauksessa kuitenkaan riittänyt, vaan ensin täytyi Pearin avulla asentaa PHPUnitin riippuvuudet.

```
sudo pear channel-discover pear.phpunit.de
sudo pear channel-discover pear.symfony-project.com
sudo pear channel-discover components.ez.no
sudo pear update-channels
sudo pear upgrade-all
sudo pear install --alldeps phpunit/PHPUnit
sudo apt-get install phpunit
```

3 TOTEUTUS

3.1 Symfony

3.1.1 Yleisesti

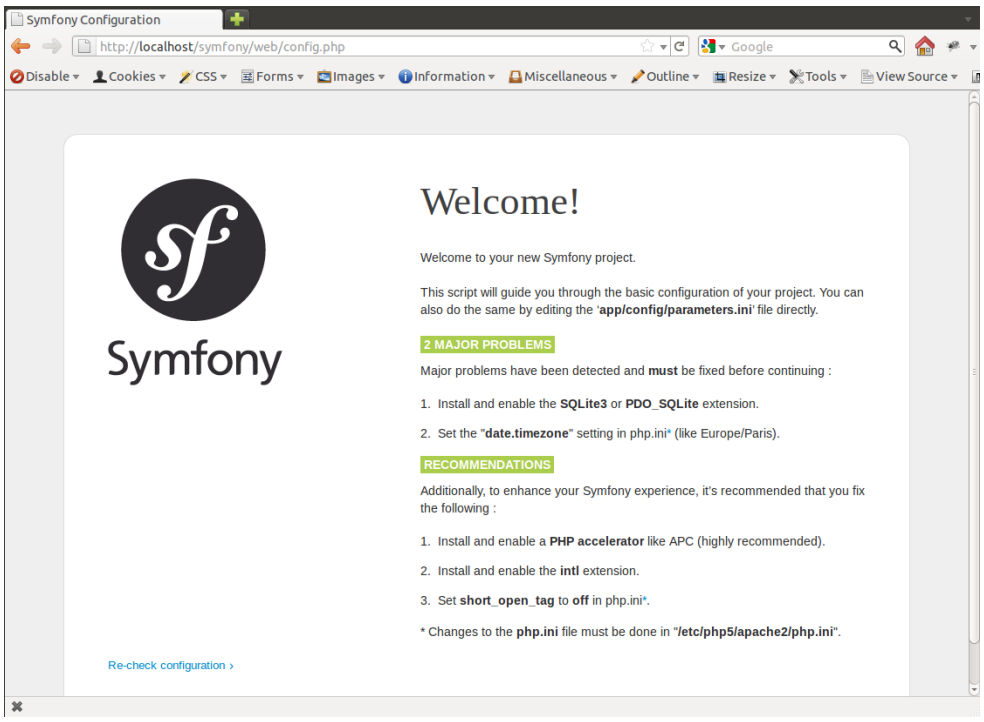
Symfony on ranskalaisen Sensio Labsin kehittämä MIT-lisenssin alainen ohjelmointikehys. Symfonyn äskettäin julkaistu ja suurilta osin uusiksi kirjoitettu 2. versio pyrkii tarjoamaan käyttäjilleen uudelleenkäyttömahdollisuuksia ja skaalattavuutta toteuttamalla riippuvuusinjektiomallia (Symfony Blog 2011).

Symfony antoi itsestään varsin akateemisen oloisen ensivaikutelman. Syvällisesti mietitty arkkitehtuuri on omiaan antamaan toivoa tämän ohjelmointikehyksen kestävästä tulevaisuudesta.

3.1.2 Asennus

Symfonyn voi asentaa hakemalla asennuspaketin symfony.com:ista tai vaihtoehtoisesti kloonaamalla suoraan Git-varastosta. Symfonyn asennus oli suoraviivaista: purettiin asennuspaketti ja mentiin selaimella kohteeseen *config.php*, jolloin Symfony itse ilmoitti havaitsemistaan puutteista järjestelmän konfiguraatiossa. Tosimaailmassa tietoturvasyistä vain pelkkä bootstrap-tiedoston sisältävä kansio asetettaisiin julkiseksi (Symfonyn tapauksessa *web* kansio), mutta virtuaaliympäristössä tällä ei ole merkitystä, joten koko Symfonyn rakenne purettiin julkiseen kansioon */var/www*.

Tässä tapauksessa puutteena oli SQLite-tietokantamoottorin ja php.ini:n aikavyöhykemäärityksen puuttuminen. Muutokset tehtiin järjestelmään, ja tietokantamääritysten jälkeen Symfony oli valmis käytettäväksi. Kuviosta 2 nähdään Symfonyn tervetuloruutu.



KUVIO 2. Symfony välittömästi asennuksen jälkeen

Ei-suljetussa ympäristössä projektia tehdessä Symfonyn asentaminen vaatii paljon tarkempaa perehtymistä konfiguraatioihin, mutta nyt tyydyttiin siihen, että oletusasetuksilla saatiin toimiva järjestelmä, jonka päälle kehittämistä voitiin kokeilla.

Asennuksen jälkeen tarpeeton *config.php* on syytä poistaa tietoturvasyistä.

3.1.3 Autentikaatio ja käyttäjien hallinta

Symfonyn käyttövaltuutukset hoidetaan *app/config/security.yml* -tiedostoa konfiguroimalla. Ohjelmointikehys huolehtii autentikoimattoman käyttäjän ohjaamisesta kirjautumissivulle tämän yrittäessä päästä kirjautumisen vaativalle alueelle. Kehittäjä joutuu kuitenkin itse tekemään lomakeautentikaation tarvitsemat ohjaimen ja lomakkeen.

3.1.4 Käyttökokemus

Symfonyn sivuasetteluun käytetään saman tekijän julkaisemaa mallipohjamoottoria (engl. *template engine*) nimeltä Twig, jota voi siis käyttää myös ilman Symfonyä. Muut ohjelmointikehykset käyttävät HTML-koodin sekaan upotettua PHP:tä varsinaisen mallipohjamoottorin sijasta. Twigin helposti opittavasta syntaksista huolimatta mallipohjamoottorin käyttö aiheuttaa jonkin verran ylimääräistä opiskelua kehittäjälle. Twigiin käytetty opiskeluaika on kuitenkin tarpeen, sillä mallipohjamoottorin käyttöönoton jälkeen voidaan olla varmoja siitä, että näkymä on asiallisesti eriytetty ohjelmakoodista, mikä edesauttaa järjestelmän testattavuutta ja ylläpidettävyyttä.

Kehitystä helpottamaan Symfony tarjoaa työkalut Web Debug Toolbar ja Web Profiler. Kun Debug Toolbar on kytketty päälle, se näyttää jokaisella sivulla yleisiä tietoja kuten suoritusajan, muistinkäytön sekä käytettyjen ohjaimen ja toiminnon nimet. Web Profiler puolestaan kertoo tarkempia tietoja järjestelmän konfiguraatiosta, http-pyynnöstä, mahdollisista poikkeuksista, sessiosta, kuuntelijakutsuista ja tietokantahauista.

3.1.5 CRUD-liittymän toteutus

Aluksi luotiin uusi moduuli nimellä oppari. Symfony kutsuu itse moduuleitaan bundle-nimellä, joten käytetään sitä nimitystä tässäkin. Bundlen luominen tapahtuu symfonyn konsolityökalun avulla.

```
php app/console generate:bundle --namespace=oppari/oppariBundle
--format=yml
```

Komento loi bundlea varten kansion ja rekisteröi bundlen *app/AppKernel.php*:hen. Jotta bundlen tiedostot näkyisivät ulospäinkin, tulee sille lisätä oma reititys app/config/routing.yml-tiedostoon.

Symfonyn oletus ORM oli aiemmin Propel, mutta uudemmissa versioissa on siirrytty Doctrineen.

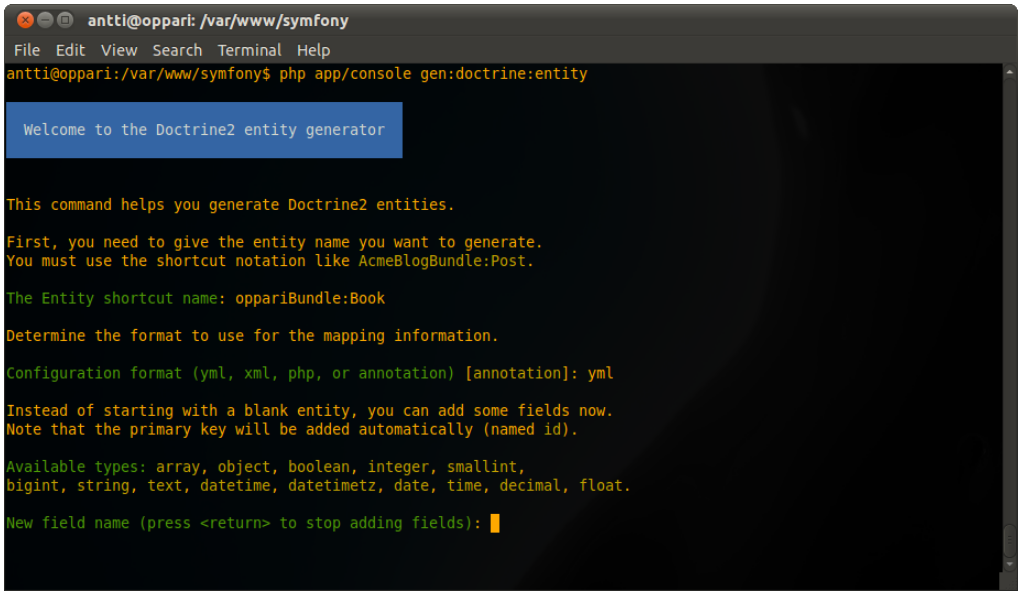
Jotta CRUD voitaisiin generoida, tarvitaan ensin tietotyyppi, jota siinä käytetään, Symfony kutsuu näitä tietotyyppejä nimellä Entity. Entity generoidaan samalla *console*-komentorivityökalulla kuin bundlekin, jonka osaksi Entity tullaan liittämään. Komento luo halutun tyyppisen kuvaustiedoston tietotyypille bundlen *Resources/config/doctrine* kansioon ja haluttaessa myös valmiin luokan kuvaamaan tietotyyppiä gettereineen ja settereineen. Tehdessä opittiin, että kun Entitylle halutaan luoda myös CRUD-liittymä konsolityökalun avulla, tämä luokka on syytä jättää generoimatta, koska sen olemassaolo estää CRUD-liittymän luomisen.

```
php app/console generate:doctrine:entity
```

Kuvion 3 mukainen konsolikomento käynnistää tietotyypin luontivelhon, jolle syötetään kenttiä ja niiden tyyppejä yksi kerrallaan. Menetelmä soveltuu hyvin prototyypittelyyn, mutta kestävämpi ratkaisu Entityjen luomiseen olisi asentaa käyttää skeematiedostoa tietokannan kuvaamiseen ja luoda tietokannan taulut konsolikomennoilla (Symfony Cookbook 2011):

```
php app/console doctrine:mapping:import
php app/console doctrine:generate:entities
```

Myös skeeman luominen olemassa olevasta tietokannasta on mahdollista *doctrine:build-schema* -komennolla.



KUVIO 3. Symfonyn komentorivityökalu

Kun Entity on luotu kuviossa 3 esitetyllä tavalla, voidaan sille luoda CRUD-liittymä.

```
php app/console generate:doctrine:crud
```

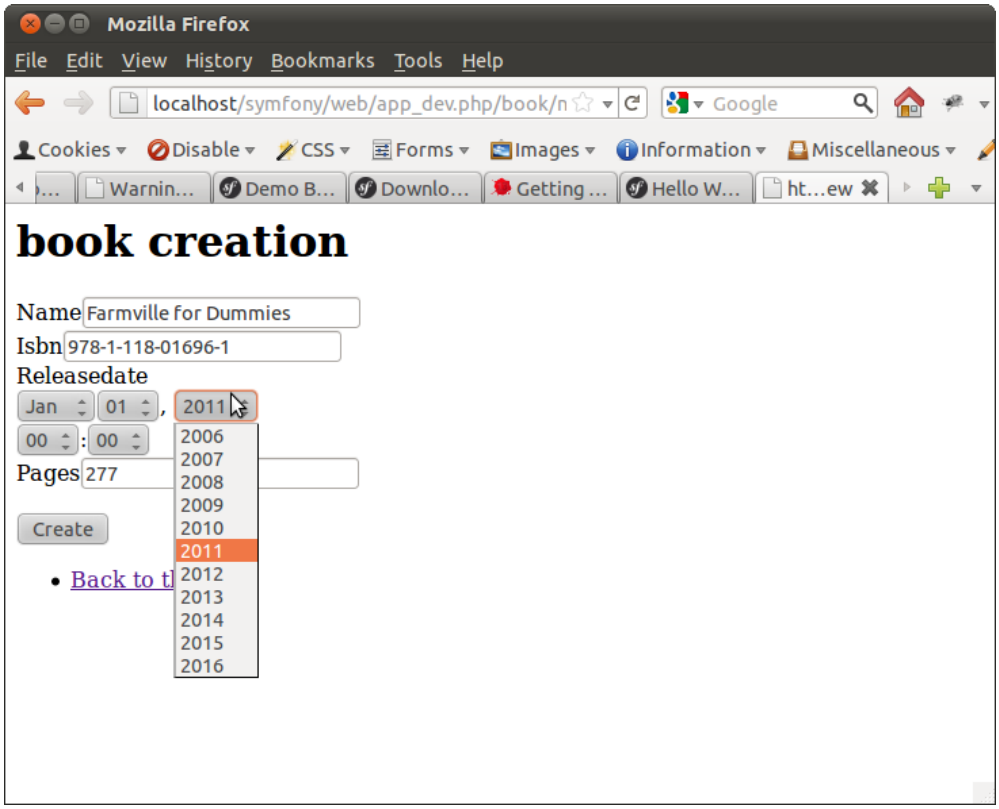
Minkäänlaista toimivaa reititystä generointityökalu ei jostain syystä suostunut tekemään. Kunnollinen reititys lisättiin bundlen *Resources/config/routing.yml* -tiedostoon.

```
oppariBundle_book:
    pattern: /book
    defaults: { _controller: oppariBundle:book:index }
```

Kun tämän jälkeen selaimella mentiin osoitteseen http://localhost/symfony/web/app_dev.php/book saatiin vastaukseksi tietokantayhteyden epäonnistumisesta kertova virheilmoitus. Ja toden totta, tietokantaahan ei ollut tarvinnut asennuksen yhteydessä luoda. Käytiin luomassa Symfonyä varten uusi MySQL-tietokanta *app/config/parameters.ini* -tiedoston asetuksia vastaavaksi. Tämän jälkeen *console*-työkalulla luotiin tarvittava taulu.

```
php app/console doctrine:schema:create
```

Tietokannan korjaamisen jälkeen liittymä toimi. Symfony ei tarjoa valmiita teemoja lomakkeille eikä listauksille, joten konsolityökalun avulla toteutettu liittymä (Kuvio 4) näyttää melko karulta ilman tyylitiedostoilla muotoilua.



KUVIO 4. Valmis muokkausliittymä

3.1.6 Dokumentaatio ja käyttäjäyhteisö

Amazonin kirjahaulla löydettiin kymmenkunta Symfonyä käsittelevää kirjaa. Mutta valitettavasti ne kaikki oli tehty Symfonyn ensimmäistä versiota varten. Symfonyn käyttöoppaan mukaan versio kaksi on kirjoitettu kokonaan uusiksi ja siten paljon kehittyneempi kuin ensimmäinen versio, tuoden mukanaan muutoksia moniin toimintoihin.

Suuret eroavaisuudet Symfonyn ensimmäisen ja toisen version välillä havaittiin käytännön tasolla siinä vaiheessa, kun Symfonyä käsittelevien yli vuoden vanhojen kirjojen tiedot eivät pitäneetkään enää

lainkaan paikkaansa. Esimerkiksi kielikäännökset toteutettiin ensimmäisessä versiossa staattisen *"__"*-metodin avulla, siinä missä toisessa versiossa on siirrytty käyttämään *Translator*-luokkaa. Onneksi virallisen käyttöoppaan ohjeet tarjosivat riittävät tiedot ainakin tämän opinnäytetyön vaatimuksiin.

Symfony ei ole – kenties kompleksisuutensa vuoksi – onnistunut keräämään kovin paljoa käyttäjiä verrattuna muihin tutkittuihin järjestelmiin. Kuitenkin joitakin varsin tunnettuja tukijoita löytyy; kuten Delicious, Yahoo bookmarks ja Dailymotion.com (Symfony Blog 2006, 2007 & 2009). Suomalaisia Symfonyllä toteutettuja sivustoja ovat mm. www.inttikaverit.com ja www.kysymys.fi (Symfonians.net 2011).

3.1.7 Lokalisointi

Symfonyn lokalisointi on toteutettu XML-, YAML- tai PHP-muotoisia tiedostoja jäsentävällä *Translator* luokalla. Aivan aluksi se kytkettiin päälle poistamalla kommenttimerkki *translator*-rivin alusta *app/config/config.yml* -tiedostossa.

Kielikäännöksiä voidaan tehdä ohjaimessa *translator*-luokan *trans*-metodia kutsumalla. (Symfony käyttöopas 2011)

```
$this->get('translator')->trans('Translate this!');
```

Jos käännettävään merkkijonoon sisältyy muuttuja, täytyy se parametrisoida ja välittää muuttujien arvot *trans*-metodille taulukkomuodossa toisena parametrina.

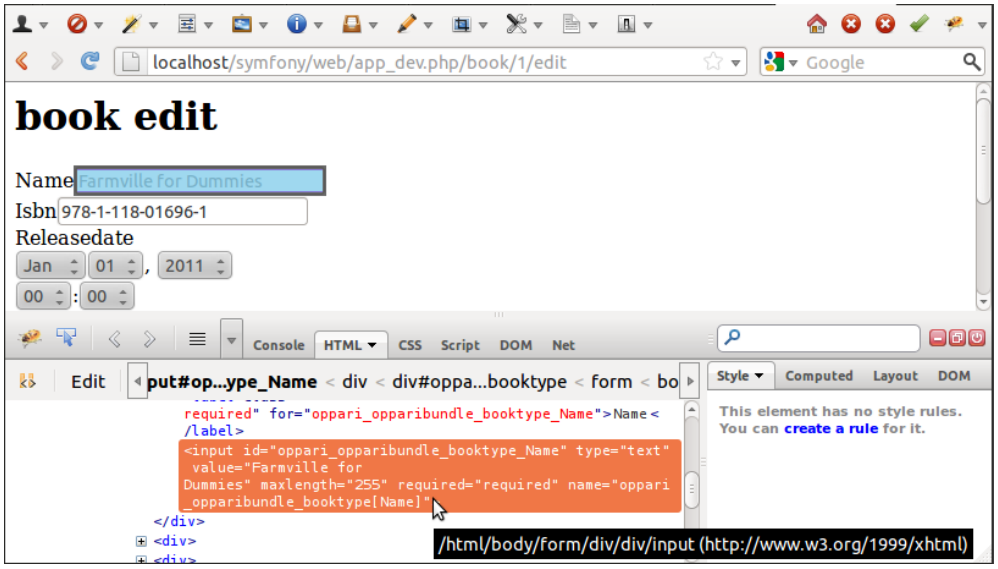
```
$name = 'Jorma';
$this->get('translator')->trans('Nice to meet you %name%.',
array('%name%' => $name));
```

Käännöksiä voidaan upottaa myös suoraan Twig-lomakkeeseen, tässä edelliset erimerkit Twigin ymmärtämässä muodossa. Muuttuja *%name%* viedään Twig-lomakkeelle ohjaimesta.

```
{% trans %}Translate this!{% endtrans %}
{% trans %}Nice to meet you %name%.{% endtrans %}
```

3.1.8 Testattavuus

Generoidussa CRUD-listauksessa ei ollut juuri millään HTML-elementeillä *id*-attribuutteja, mikä vaikeuttaa selainautomatisoitujen hyväksymistestien kirjoittamista. Muokkausnäkymässä oli kuitenkin kaikilla kentillä kuvion 5 mukaiset, automaattisesti nimetyt *id*-attribuutit. Symfony kuitenkin tarjoaa testien kirjoittamista varten erityislaatuisen työkalun, sfBrowserin. SfBrowser on eräänlainen selainsimulaattori, jonka avulla hyväksymistestejä voidaan suorittaa ilman oikeaa selainta. Koska testien ja sivuston välillä ei tällöin ole http-kerrosta, nopeutuu hyväksymistestien suorittaminen huomattavasti verrattuna Seleniumilla tehtäviin testeihin (Bergmann & Priebsch 2011, 165-168).



KUVIO 5. Automaattisesti nimetty id-attribuutti (”oppari_opparibundle_booktype_Name”)

Symfony ohjelmointikehys itsessään on varsin kattavasti yksikkötestattu. Ensimmäinen versio käytti yksikkötestien suorittamiseen kehittäjien itse kirjoittamaa Lime-alustaa, mutta nykyisessä 2. versiossa yksikkötestit ajetaan suositulla PHPUnit-testikehyksellä. (Mts. 162).

3.2 Zend

3.2.1 Yleisesti

Zend on Israelilaisyhdysvaltalaisen Zend Technologies yhtiön kehittämä uuden BSD-lisenssin alainen ohjelmointikehys. Kehyksen puolesta puhuu se, että osa Zendin tekijöistä on mukana myös PHP-ytimen kehittämisessä.

Zend poikkeaa muista tutkituista ohjelmointikehyksistä siten, että se ei pyri luomaan kehittäjälle kokonaisratkaisua, vaan kaikkia Zendin luokkia voi käyttää myös yksitellen. Siksi Zend on kehittäjälle helppo omaksua, kun tämän ei tarvitse tietää kaikkea kerralla. Zendissä toteutuva ohjelmointikehyksen selkeä erottaminen kehitetystä sovelluskoodista varmistaa, että ohjelmointikehystä päivitettäessä ei törmätä konflikteihin.

Uuden toimivan Zend-projektin pystyttäminen vaatii paljon työtä, mikä aiheuttaa riskitekijän ketterän kehityksen menetelmillä tapahtuvassa sovelluskehityksessä. Asiakas voi pettyä kun aikaa kuluu näkyvien tuotosten sijasta konepellin alla tapahtuvien töiden tekemiseen. Toisaalta järjestelmän ylläpito helpottuu kun kehittäjät tuntevat sen syvällisemmin.

3.2.2 Asennus

Zendin asennuksen esivaatimuksina oli PHP 5.2.4 ja web-palvelin, joka tukee web-osoitteiden uudelleenkirjoittamista (engl. *rewrite engine*). Apache2:ssa web-osoitteen uudelleenkirjoittamisesta huolehtii *mod_rewrite*-moduuli. Zendin asennus oli työssä käytetyssä Ubuntu ympäristössä erittäin yksinkertaista. Asennettiin *apt-get*:llä *zend-framework*-paketti ja lisättiin sen asennuspolku *php.ini*:in *include_path*-muuttujaan.

```
include_path = ".:/usr/share/php: /usr/share/php/libzend-framework-php"
```

Projektien luomista ja koodin generointia varten Zend tarjoaa komentorivityökalun *zf.sh*, joka asennettiin Ubuntun pakettienhallinnan avulla.

```
apt-get install zend-framework-bin
```

Komentorivityökalun asentamisen jälkeen sen avulla luotiin uusi projekti.

```
zf create project zend-oppari
```

Projektin luominen tarkoittaa käytännössä sitä, että *zf.sh* loi kansion nimeltä *zend-oppari* alle projektin peruskansiorakenteen ja sijoitti sinne konfiguraatio-tiedoston, bootstrap-tiedoston ja tervehdysviestin näyttävän index-ohjaimen.

```
├── application
│   ├── Bootstrap.php
│   ├── configs
│   │   └── application.ini
│   ├── controllers
│   │   ├── ErrorController.php
│   │   └── IndexController.php
│   ├── models
│   └── views
│       ├── helpers
│       └── scripts
│           ├── error
│           │   └── error.phtml
│           └── index
│               └── index.phtml
├── docs
│   └── README.txt
├── library
├── public
│   └── index.php
└── tests
    ├── application
    │   └── controllers
    │       └── IndexControllerTest.php
```

```
    ├── bootstrap.php
    ├── library
    └── phpunit.xml
```

Tässä oletusrakenteessa on joitakin huomion arvoisia tiedostoja ja kansioita. Mallit, ohjaimet ja näkymät sijoitetaan omiin kansioihin. Ulkoiset kirjastot sijoitetaan *library*-kansioon. Varsinainen sovelluksen aloituskohta on *public/index.php*, *public*-kansio täytyy siksi konfiguroida Apachesta julkiseksi.

3.2.3 Autentikaatio ja käyttäjien hallinta

Zendmäinen käyttäjien valtuutus toteutetaan *Zend_Acl*-luokkaa perimällä. Esimerkki on mukailtu Simon Mundyn kirjoittamasta artikkelista Zend Developer Zonessa.

```
class MyAcl extends Zend_Acl {
  public function __construct(Zend_Auth $auth) {
    parent::__construct();

    // Määritellään resurssit
    $this->add(new Zend_Acl_Resource('home'));
    $this->add(new Zend_Acl_Resource('forum'));
    $this->add(new Zend_Acl_Resource('admin'));

    // Määritellään käyttäjätasot
    $this->addRole(new Zend_Acl_Role('guest'));
    $this->addRole(new Zend_Acl_Role('member'), 'guest');
    $this->addRole(new Zend_Acl_Role('admin'), 'member');

    // Määritellään käyttäjätasokohtaiset valtuutukset
    // resursseille.
    $this->allow('guest', 'home');   // Vapaa pääsy etusivulle
    $this->allow('member', 'forum'); // Käyttäjille pääsy
                                     // keskustelufoorumiin
    $this->deny('member', 'forum', update');
                                     // Foorumipäivitysten esto
    $this->allow('admin');           // Pääkäyttäjille pääsy
                                     // kaikkialle

    // Viedään pääsy-objekti toiminnolle
    $this->allow('member', 'forum', 'update', new
MyAcl_Forum_Assertion($auth));
  }
}
```

Valtuutussääntöjä voi hallita myös yksittäisten toimintojen tasolla, kuten esimerkissä tehdään *update*-toiminnolle. Esimerkissä *update*-toiminnolle injektoidaan *Zend_Auth*-objekti, jotta toiminto saisi helposti tietoonsa kuka käyttäjä toimintoa käyttää (Mundy 2007).

ACL-pääsylistoihin perustuvien valtuutusten sijasta olisi hienostuneempaa käyttää rooleihin sidottuja valtuutuksia (Porebski, Przystalski, Nowak, 2011, 430). Onneksi Zendin modulaarisuus mahdollistaa juuri sellaisen valtuutusten käsittelyn toteuttamisen kuin kehittäjä haluaa.

3.2.4 Käyttökokemus

Koska web-käyttöliittymille on ominaista, että kaikilla alasivuilla tietyt elementit toistuvat on järkevää luoda järjestelmän kaikille alasivuille yhtenäinen asettelu (Lyman 2009, 18). Zendissä tämä tapahtui komentamalla:

```
zf enable layout
```

Komento kytki Zendin layoutit päälle ja loi tiedoston */application/layouts/scripts/layout.phtml*, jota muokkaamalla päästiin kokeilemaan omien asettelujen tekemistä. Oletuksena tiedosto tarjosi pelkästään minimaalisen sisällön tulostamisen:

```
<?php echo $this->layout()->content; ?>
```

Tämän mallipohjan ympärille onkin helppo lähteä rakentamaan omaa sisältöä. On syytä huomata, että oletus layoutista puuttuu oikeastaan aivan kaikki, *head*- ja *body*-tageja myöten.

CSS-muotoiluja pystyttiin tekemään luomalla *public*-kansion alle kansio nimellä *css*. Tähän kansioon luotiin tiedosto *oppari.css*. Tiedosto otettiin käyttöön lisäämällä alustuksia *bootstrap.php*-tiedostoon. Tiedosto on kuvattu tarkemmin liitteessä 12.

```
$view->headLink()->setStylesheet($view-
>baseUrlr().'css/oppari.css');
```

Nyt äsken luotuun *layout.phtml*-mallipohjaan lisättiin sivun otsikkotiedot. Otsikkotiedoissa tulostettiin sivulle bootstrapissä määritellyt otsikkoteksti, dokumenttityyppi sekä linkitys juuri luotuun *oppari.css*-tyylitiedostoon.

```
<html><head>
<?php
echo $this->headMeta();
echo $this->headTitle();
echo $this->headLink();
?></head>
<body><?php echo $this->layout()->content; ?></body>
</html>
```

Näin omat tyylit saatiin sivulle käyttöön.

Zend tarjoaa tehokkaan tuntuisen pohjan verkkosovellusten rakentamiseen. Kantavana ideana on, että kaikki kehittäjän toteuttamat luokat perivät jotakin Zendin omaa luokkaa. Kantaluokka siis lähinnä tarjoaa apuja kyseisen toiminnon varsinaiseen toteuttamiseen. Zendin oliot antavat kehittäjälle runsaasti vapautta tehdä omanlaisensa toteutus. Mutta toisaalta herättävät ajatuksen siitä, olisiko kuitenkin tuottavuuden kannalta parempi, jos tarjotut toiminnot olisivat hieman lähempänä valmista.

3.2.5 CRUD-liittymän toteutus

Zendin CRUDin pohjat syntyivät komentoriviltä *zf.sh*-työkalun avulla Allenin (2010) opasta mukaillen. CRUDin aiheeksi valittiin kirjatietokanta. Ensiksi luotiin ohjain, ja sitten sille toiminnot lisäystä, muokkausta ja poistoa varten. Tietojen listaus päätettiin näyttää oletustoiminnossa (*indexAction*).

```
zf create controller books
zf create action add books
zf create action edit books
zf create action delete books
```

Ohjaimeen oletustoiminto (*indexAction*) kirjojen listaamista varten toteutettiin seuraavaksi. Tämä ohjain kokonaisuudessaan on nähtävissä liitteessä 1.

```
  public function indexAction() {
    $books = new Application_Model_DbTable_Books();
    $this->view->books = $books->fetchAll();
  }
```

Tiedon säilytystä varten luotiin MySQL-tietokanta, ja kannalle käyttäjätunnus *zend-user* salasanalla *zend-pass*.

```
create database zend;
grant all on zend.* to 'zend-user'@localhost identified by
'zend-pass';
```

Ja vastaavat kirjautumistiedot lisättiin *application/configs/application.ini*-tiedoston *production*-osioon.

```
resources.db.adapter = PDO_MYSQL
resources.db.params.host = localhost
resources.db.params.username = zend-user
resources.db.params.password = zend-pass
resources.db.params.dbname = zend
```

Seuraavaksi tietokantaan luotiin taulu kirjojen tietojen tallentamista varten.

```
CREATE TABLE `zend`.`book` (
`ID` INT NOT NULL AUTO_INCREMENT ,
`Name` VARCHAR( 255 ) NOT NULL ,
```

```
`ISBN` VARCHAR( 20 ) NOT NULL ,
`ReleasedDate` DATE NULL ,
`Pages` INT NULL ,
`Description` TEXT NULL ,
PRIMARY KEY ( `ID` )
) ENGINE = MYISAM ;
```

Tämän jälkeen *zh.sh*-työkalulla luotiin CRUD-luokka kuvaamaan tätä uutta tietokantataulua.

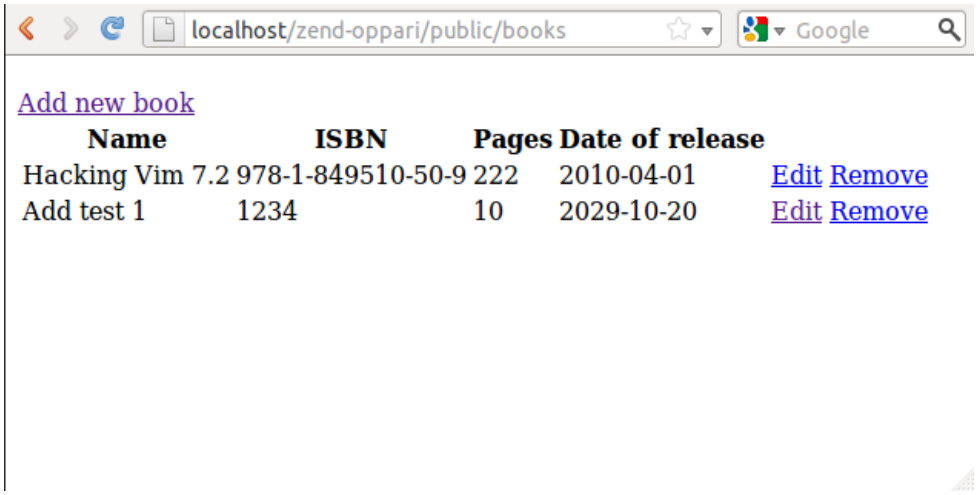
```
zf create db-table Books book
```

Komennosta syntyi sovelluksen mallia varten tiedosto *application/models/DbTable/Books.php*, johon kirjoitettiin metodit *getBook*, *addBook*, *updateBook* ja *deleteBook* CRUD-operaatioita varten. Tämä malli on nähtävissä liitteessä 2.

Tosimaailman tilanteessa mallin operaatioihin olisi vielä syytä lisätä datan validoinnit. Seuraavaksi luotiin ohjaimen oletustoiminnolle eli kirjojen listaukselle näkymä (*indexAction*), jossa listattiin kirjojen tiedot, ja näytettiin linkit muokkaus- ja poistotoimintoihin. Listaustoiminnon näkymä on nähtävissä liitteessä 3.

Uuden kirjan lisäystä ja olemassa olevien tietojen muokkausta varten tarvittiin lomakeluokka. Lomakeluokka sisältää lomakeelementit kirjan tietojen muokkaamista varten, sekä joukon suodattimia ja validointimetodeita. Näitä validointimetodeita Zendistä löytyy joka lähtöön, mutta niitä voidaan kirjoittaa myös itse. Lomakeluokka liitteenä 4.

Nyt CRUDista oli jo riittävästi osia valmiina, että selaimella pystyttiin tarkastelemaan kuviossa 6 esiteltyä kirjalistausta. Muokkaus- ja poistolinkit eivät kuitenkaan vielä toimi, koska niille ei ole vielä kirjoitettu toimintometodeita. Lisäksi näille toimintometodeille on vielä luotava omat lomakkeet.



KUVIO 6. Kirjalistaus

Lisättiin vielä ohjaimeen toiminnot kirjan lisäystä, muokkausta ja poistoa varten (ks. Liite 1. Zend CRUD -ohjaintiedosto). Toiminnoille lisättiin näkymät omiin tiedostoihinsa (ks. Liitteet 5, 6 ja 7). Nyt CRUD-liittymä oli käyttövalmis, se sen muokkausnäkymä on esitelty kuviossa 7.

KUVIO 7. Kirjojen muokkausnäkymä

Generaattoreista huolimattakin yksinkertaisen CRUD-liittymän luomiseen tarvittiin Zendissä melkoinen määrä käsin kirjoitettua koodia.

3.2.6 Dokumentaatio ja käyttäjäyhteisö

> *No matter how good the code is, lack of documentation can kill a project through lack of adoption. As Zend Framework is aimed at developers who do not want to have to dig through all the source code to get their job done, Zend Framework puts documentation on an equal footing with the code. This means that the core team will not allow new code into the framework unless it has accompanying documentation.*
>
> (*Allen, Lo & Brown*, 2008.)

Zendistä on kirjoitettu kymmeniä kirjoja. Se on myös tutkituista ohjelmointikehyksistä selvästi laajimmin käytetty. Käyttäjiä löytyy yksityishenkilöistä lähtien aina suuriin yrityksiin saakka, mainittakoon suurista yrityksistä IBM ja Fox Interactive Media.

Lähitulevaisuudessa Zendiin on luvassa suuria muutoksia, kun siitä julkaistaan versio 2.0. Uusi versio ottaa käyttöön PHP 5.3:n uusia ominaisuuksia, joista kehittäjän kannalta merkittävin lienee nimiavaruudet. Valitettavasti tämän tyyppinen harppaus tietenkin tarkoittaa myös taaksepäin yhteensopimattomuutta (O'Phinney, Dubravszky 2011).

3.2.7 Lokalisointi

Zend tarjoaa lokalisointia varten luokkia suorittamaan muunnoksia päivämäärien, valuuttojen, lämpötilojen, etäisyyksien ja tekstien eri kieli- ja kulttuuriversioiden välillä (Vaswani 2010, 272).

Käyttäjän kulttuuri voidaan tunnistaa selaimesta, tai käyttäjäprofiiliin tallennetusta asetuksesta *Zend_Locale*-luokan avulla. Kulttuuritieto suositellaan tallennettavaksi *Zend_Registry*-olioon ja tiedot haettavaksi sen staattisten metodien kautta. (Mts. 277).

```
Zend_Registry::set('locale', $value);
$value = Zend_Registry::get('locale');
```

Kielikäännöksistä huolehtii *Zend_Translate*-luokka. Se tukee käännösten hakemista PHP-taulukoista, CSV-tiedostoista, GNU Gettext –tiedostoista sekä lukuisista XML-pohjaisista formaateista (Porebski, Przystalski & Nowak 2011, 389). Tietokannasta tekstien hakemista *Zend_Translate*–luokka ei tue (mts. 391).

3.2.8 Testattavuus

Zendin kanssa suositellaan käytettäväksi PHPUnit-ohjelmaa yksikkötestausta varten. Sen käyttöönotto vaatii jonkin verran työtä, mutta konfiguraation luomisen jälkeen testien kirjoittaminen ja suorittaminen on helppoa.

Zend projektin *tests*-kansioon luotiin tiedosto *phpunit.xml*, johon kirjoitettiin seuraavanlainen PHPUnitin konfiguraatio PHPUnitin dokumentaation perusteella (phpunit-book.pdf, liite C).

```
<phpunit colors="true">
    <testsuites>
        <testsuite name="ZendinTestit">
            <directory>./</directory>
        </testsuite>
    </testsuites>
    <filter>
        <whitelist>
            <directory suffix=".php">./application/</directory>
        </whitelist>
    </filter>
    <logging>
        <log type="coverage-html" target="./log/report"
charset="UTF-8"
```

```
        yui="true" highlight="true" lowUpperBound="50"
highLowerBound="80" />
        <log type="testdox" target="./log/testdox.html" />
    </logging>
</phpunit>
```

Zendin generoidun CRUDin kenttiin syntyivät asialliset *id*-tagit XPath-hakuja helpottamaan.

3.3 Drupal

3.3.1 Yleisesti

Belgialaisen GPL-lisenssin alaisen Drupalin suunnittelun painopisteenä on yhteisöllisyys (VanDyk 2008, 1). Vaikka mistään wiki-tyyppisestä järjestelmästä ei olekaan kyse, niin tämä tavoite toteutuu esimerkiksi kaikkiin sisältötyyppeihin helposti liitettävässä kommentointimahdollisuudessa.

Kehittäjälle Drupal tarjoaa tapahtumapohjaisuutta koukkujen (engl. *hook*) muodossa. Koukkujen avulla voidaan esimerkiksi käyttäjän kirjautumisen yhteyteen kytkeä automaattisesti laukeavia tapahtumia (mts. 35-37).

Puhtaan staattista sisältöä Drupalilla on todella nopea luoda. Helppous tuo haasteen kehittäjille järjestelmän syvällisemmän ymmärtämisen suhteen, kun kaikkeen tuntuu olevan jo valmis komponentti.

Arkkitehtuurisesti Drupal poikkeaa muista tutkituista järjestelmistä toteuttamalla PAC-mallia (Presentation-Abstraction-Control) MVC-mallin (Model-View-Controller) sijasta. Huomattavin ero näiden kahden arkkitehtuurin välillä on se, että PAC-mallin näkymä (engl. *Presentation*) on tyhmä, eli siihen ei sallita tehtäväksi minkäänlaista ohjelmalogiikkaa (Buschmann, Henney, Schmidt 2007. 188-193).

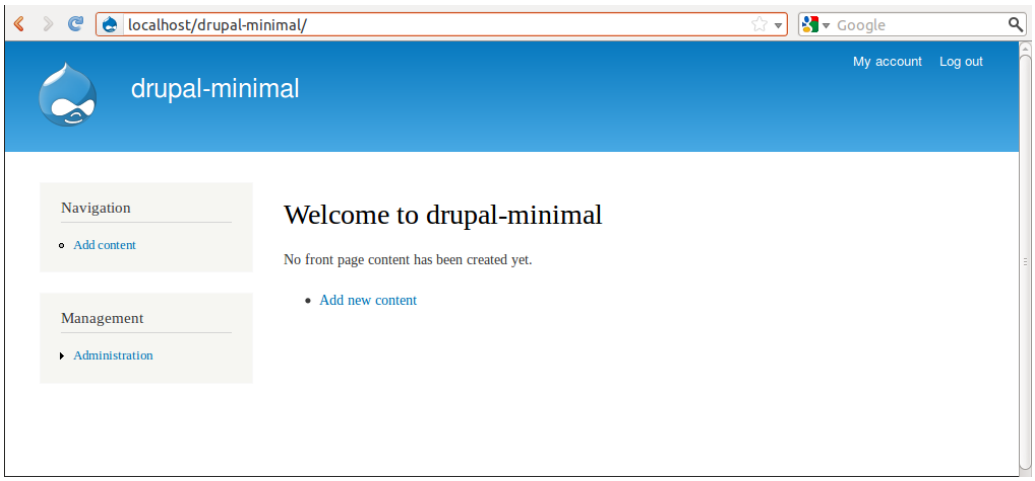
3.3.2 Asennus

Drupalin asennus sujui ongelmitta asennusvelhon ohjeita noudattaen. Asennuksen aikana tosin täytyi käydä käsityönä luomassa MySQL-tietokanta. Tietokanta luotiin seuraavilla komennoilla.

```
create database drupal;
```

```
grant all on drupal.* to ‘drupal’@localhost identified by
‘drupal’;
```

Asennuksen kenties keskeisin valinta on oletusasennuksen ja minimaaliasennuksen välillä. Oletusasennus tarjoaa radikaalisti minimaaliympäristöä modernimman näköisen hallintaliittymän, vaikka molemmissa asennuksissa hallintatoiminnot ovatkin samat. Toki moduuleita päälle ja pois kytkemällä molemmista asennuksista voidaan siirtyä toiseen ääripäähän, joten asennusvaiheessa tehdyllä valinnalla ei varsinaisesti ole pysyvää merkitystä. Minimiasennuksen tervehdysnäkymä on kuvattu kuviossa 8.



KUVIO 8. Drupal minimal välittömästi asennuksen jälkeen

Drupal järjestelmässä on runsaasti konfiguraatiomahdollisuuksia, mutta käyttöönotto onnistui pelkkiä perusasetuksia käyttäen muutamassa minuutissa.

3.3.3 Autentikaatio ja käyttäjien hallinta

Käyttäjien hallintaliittymä löytyy Drupalin *Administration*-valikosta *People*-välilehdeltä. Sieltä käyttäjille voidaan jakaa rooleja, joiden mukaan valtuutukset määräytyvät, tai antaa käyttäjäkohtaisia erillisoikeuksia (Geller 2011, 163). Oikeudet annetaan kuvion 9 mukai-

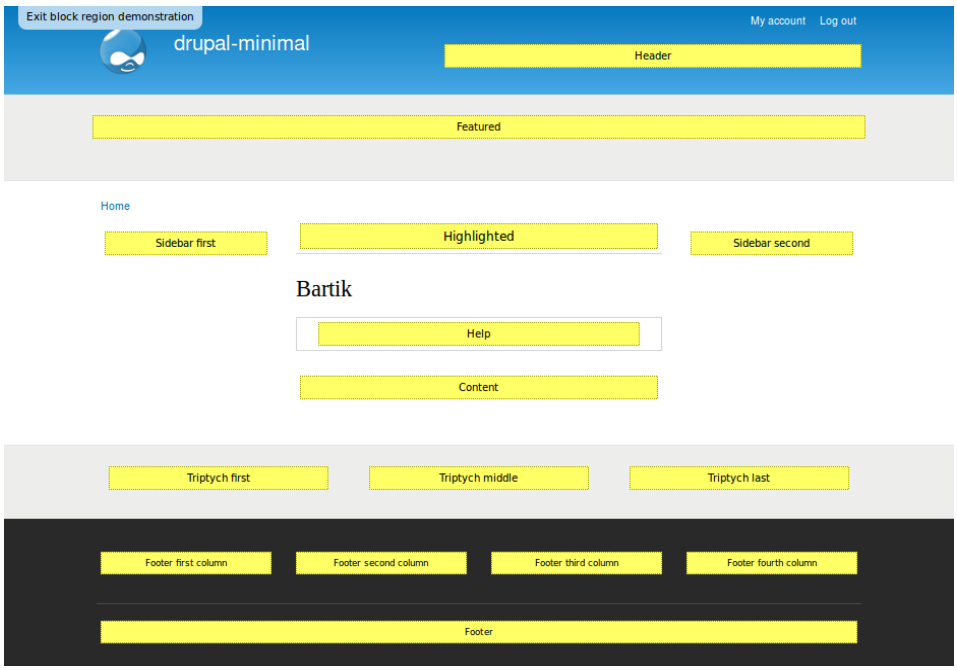
sesti toimintokohtaisesti, lisäksi moduuleilla voi myös olla omia käyttöoikeusmäärityksiään.

Permission	anonymous user	authenticated user
Book: Create new content	☐	☑
Book: Edit own content	☐	☑
Book: Edit any content	☐	☐
Book: Delete own content	☐	☑
Book: Delete any content	☐	☐
Story: Create new content	☐	☑
Story: Edit own content	☐	☑
Story: Edit any content	☐	☐
Story: Delete own content	☐	☑
Story: Delete any content	☐	☐

KUVIO 9. Drupal käyttöoikeuksien hallintaliittymä

3.3.4 Käyttökokemus

Drupalin käyttöliittymäkomponentteja sijoitetaan lohkoihin (engl. *block*), jotka puolestaan on sijoitettu ”sisältöalueisiin”. Drupalin oletusteema on Bartik. Kuviossa 10 on esitetty Bartikin mukaiset lohkot. Sisällön jakaminen lohkoihin rajoittaa jonkin verran suunnittelijoiden luovuutta, joten sivujen rakenteen vapaampaan hallintaan onkin tarjolla joitakin vaihtoehtoisia moduuleita, kuten *Panels* ja *Skinr* (Geller 2010, 146). Eräs myös mainitsemisen arvoinen ulkonäköön vaikuttava moduuli on *mobileplugin*, jonka avulla Drupalin liittymät saadaan esitettyä mobiililaitteille sopivan kokoisina (Pearce 2011, 290).



KUVIO 10. Bartik sisältöalueet

Drupal on selvästi suunniteltu ei-ohjelmoijien käyttöön. Se muistuttaakin käyttökokemukseltaan kenties enemmän Microsoft Sharepoint ympäristöä, kuin ohjelmointikehystä. Vaikutelmaa on omiaan lisäämään sekä Drupalista että Sharepointista molemmista löytyvät, toisiaan vastaavat käsitteet Field ja Content type.

Varsinaista ohjelmointia kokeiltiin luomalla Drupaliin oma moduuli. Moduulin luonti alkaa tekemällä uusi kansio /sites/all/modules -kansioon. Sijainti on eri kuin Drupalin omilla moduuleilla, mikä onkin erinomaisen hyödyllistä järjestelmän päivittämisen kannalta ajateltuna. Moduulille välttämättömiä tiedostoja ovat *.info, *.module, *.admin.inc ja *.install. Näissä *-merkillä tarkoitetaan moduulin nimeä. Info-tiedosto sisältää moduulin kuvauksen ja riippuvuudet. Admin.inc-tiedostossa on moduulin hallintaliittymä ja Install-tiedostossa asentamista ja poistamista varten vaadittavat toimenpiteet. Module-tiedostossa sijaitsee moduulin varsinainen suoritettava ohjelmakoodi (VanDyk 2008, 13-18). Näistä moduulin liittyvystä tiedostoista on esimerkit liitteissä 8-11.

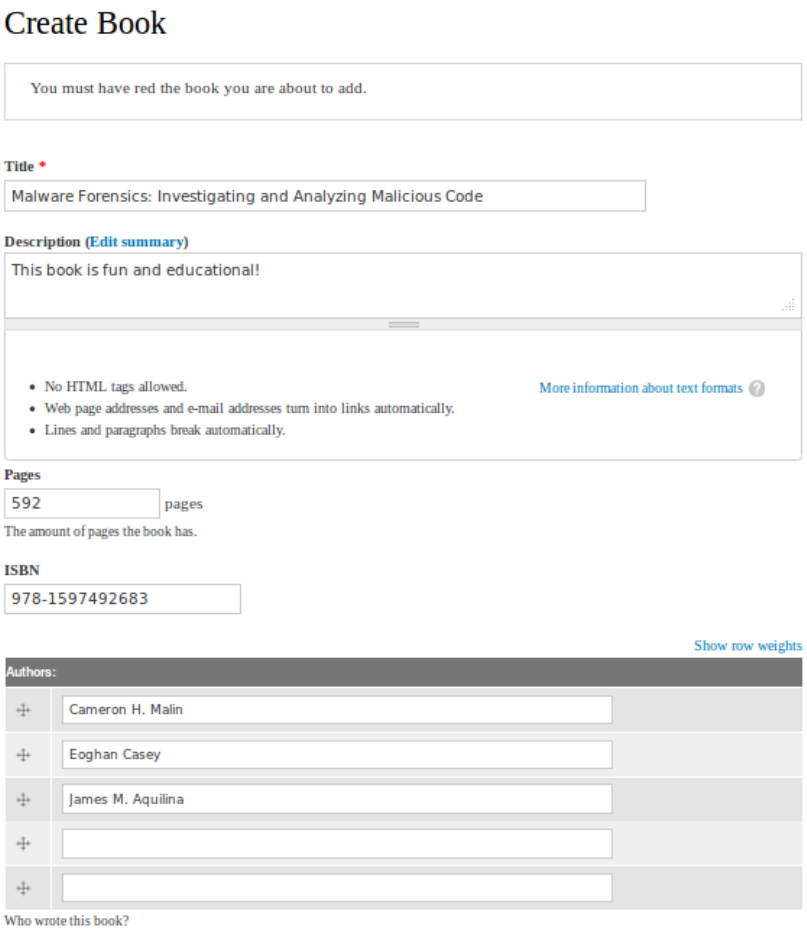
3.3.5 CRUD-liittymän toteutus

Keskeinen asia Drupalin sisällönhallinnan toiminnassa on sen perustuminen tietotyyppeihin (engl. *Content Type*). Tietotyypit puolestaan koostuvat kenttätyypeistä (engl. *Field Type*), joihin varsinainen informaatiosisältö tallennetaan. Uusia tietotyyppejä voidaan myös periä olemassa olevista, näin onkin tehty Drupalissa sen asennuksen yhteydessä luotujen tietotyyppien Pagen ja Storyn kohdalla, jotka on johdettu perustietotyypistä Node. Node itsessään on abstrakti tietotyyppi, eikä sitä siten itsenäisesti voida käyttää sivuston sisältönä. Oletustietotyyppien Pagen ja Storyn käytännön erottavana tekijänä on se, että Storyn yhteyteen voidaan tallentaa myös käyttäjien keskusteluviestit.

Liittymän toteutus aloitettiin siitä, että moduulien hallintaliittymästä kytkettiin päälle omien kenttätyyppien muokkaukseen tarkoitettu moduuli Field UI, sekä usein tarvitut kenttätyypit Number ja List. Drupalin kotisivuilta löytyy suuri moduuliarkisto, johon kuuluu muun muassa runsaasti erilaisia kenttätyyppejä. Esimerkiksi vakioasennuksesta puuttuvat päivämäärä- ja sähköpostikenttä. Käyttäjien halutessa Drupal kerää statistiikkaa käytetyistä moduuleista. Eniten käytettyjen moduulien lista löytyy URL-osoitteesta http://drupal.org/project/usage, ja se onkin hyvä paikka etsiä hyödyllisimpiä Drupal laajennusosia. Eräs huomion arvoinen laajennus on Node Reference Field Type, joka nimensä mukaisesti luo viittauksen tietotyypin kentästä toiseen tietotyypin instanssiin, siten mahdollistaen listamaisen rakenteen esittämisen puu muodossa.

Admin → Content → Add Content liittymästä lisättiin uusi tietotyyppi kirjojen tallentamista varten. Kirjalle luotiin kenttätyypit sivujen, ISBN-numeron ja kirjoittajien tallentamista varten. Näille kenttätyypeille asetettiin ohjetekstit ja ISBN-numerolle 20 merkin syöttörajoite. Sivumäärän alarajaksi asetettiin 0, ja tiedon perään lisättäväksi päätteeksi teksti ”pages”.

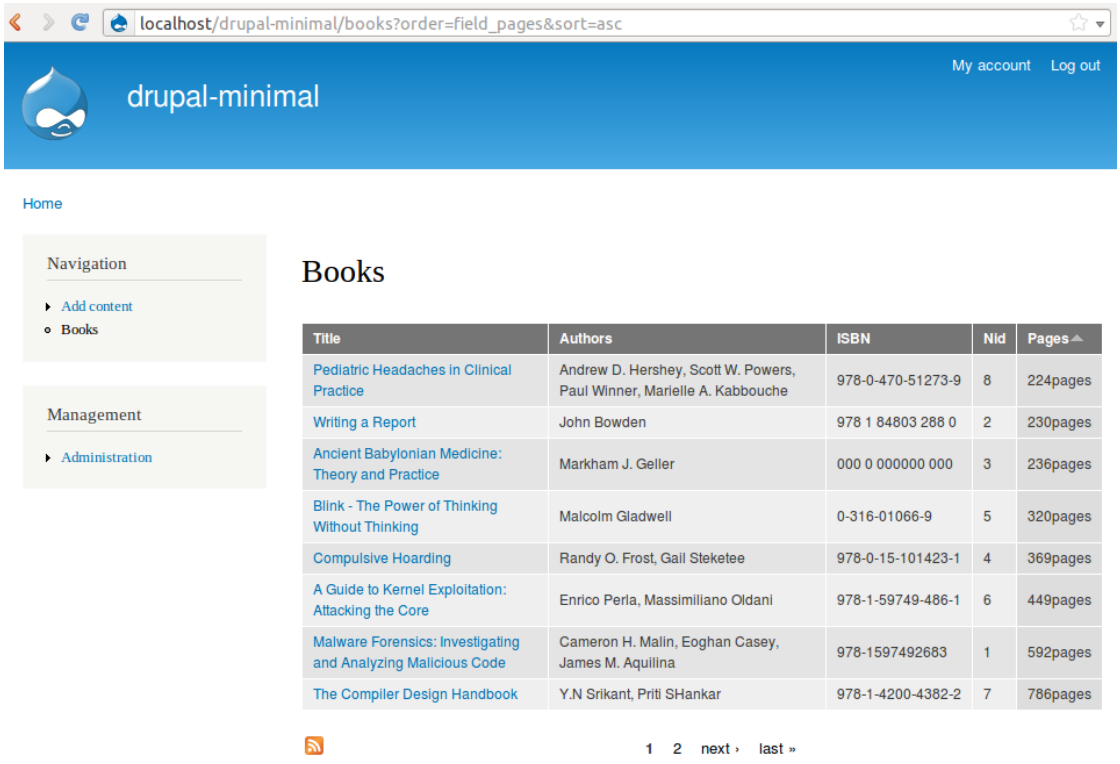
Kun tietotyyppi oli tallennettu, sitä pystyi heti käyttämään. Kuviossa 11 on esitelty uuden kirjatiedon lisäysnäkymä. Kentille asetetut syöttörajoitteet toimivat oletetulla tavalla.



KUVIO 11. Kirjan luontinäkymä

Tietotyypin luomisen jälkeen, haluttiin kokeilla saisiko järjestelmään lisättyjen kirjojen tietoja esitettyä muullakin tavalla kuin yksittäisinä Nodeina. Tätä tarkoitusta varten löydettiin suosittu Views-moduuli, jolla kirjat saatiin selkeään listamuotoon. Views-moduulin käytön esivaatimuksena oli myös Ctools-moduuli. Molemmat moduulit asennettiin yksinkertaisesti kopioimalla asennuspakettien sisällöt Drupalin *modules*-kansioon, ja sitten kytkemällä Views päälle moduulien hallintaliittymästä. Kirjalistaa varten luotiin uusi Views-näkymä nimellä *Books*. Näkymää luotaessa pystyttiin myös samalla luomaan käyttöliittymään pikalinkkejä osoittamaan luotavaan näkymään.

Kuviossa 12 on esitelty valmis kirjalistaus. Listauksen otsikoiden linkeistä pääsee siirtymään suoraan kyseisen kirjan muokkaustilaan. Views-moduulista voidaan kytkeä päälle sivutukset ja sarakkeet järjesteltäviksi. Kuviossa 12 kirjat onkin järjestetty sivumäärän mukaan nousevasti. Sivutuksen leikkausrajaksi määrätty kahdeksan kirjaa.



KUVIO 12. Valmis kirjalistaus

CRUDin luonti kirjojen hallintaa varten syntyi täysin ilman ohjelmakoodiin koskemista. Vastapainona helppoudelle syntyy toki myös ongelmia. Jos asiakas haluaa jonkin toiminnon jota moduuli ei oletuksena tarjoa, niin sen luominen voi osoittautua suhteellisen työlääksi. Drupalin tapa hajauttaa tietotyypit tietokantaan vaatisi järjestelmäintegraatioita tehtäessä suhteellisen monimutkaisten kyselyiden toteuttamista, mutta onneksi Drupal tarjoaa myös Import/Export APIn, jonka avulla tietokantaan tallennettua dataa saadaan ohjelmallisesti tuotua Drupal-tietomallista muihin järjestelmiin XML- tai CSV-muodossa.

3.3.6 Dokumentaatio ja käyttäjäyhteisö

Drupalista on kirjoitettu kymmeniä kirjoja. Suurin osa näistä kirjoista käsittelee Drupalin käyttämistä ilman ohjelmointia, mutta ohjelmointiin keskittyviäkin teoksia löytyy.

Drupalin oma dokumentaatio on eritelty kehittäjille tarkoitettuun ohjelmointipuoleen ja järjestelmää käyttöliittymän kautta käyttävien ylläpitäjien dokumentaatioksi. Varsinainen ohjelmakoodi on todella huolellisesti kommentoitu.

Monet monikansalliset yhtiöt käyttävät Drupalia, erityisesti kansallisilla sivustoillaan. Esimerkiksi McDonaldsin Australian sivusto on toteutettu Drupalilla (Burge 2010).

Drupalista on seuraavaksi luvassa kahdeksas versio. Aikataulu sen ilmestymiselle ei ole vielä tiedossa, mutta uudessa versiossa kehittäjätiimi keskittyy muuttamaan Drupalin HTML5 yhteensopivaksi (Drupal: HTML5 Initiative 2011).

3.3.7 Lokalisointi

Drupalin lokalisaatio on helppo toteuttaa. Se kattaa kielikäännösten lisäksi myös päivämäärä- ja erotinmerkkimuunnokset. Drupalin teemat on tehty tukemaan myös oikealta vasemmalle luettavia kieliä, kuten arabiaa ja hepreaa. Jos Drupalin asennuksen yhteydessä on käytetty jotain muuta kieltä kuin englantia, on lokalisaatiomoduuli kytketty päälle automaattisesti. (VanDyk 2008, 407)

Kaikki loppukäyttäjälle näkyvät tekstit tulisi koodissa esittää Drupalin *t*-funktion lävitse. Jos lokalisaatiomoduuli on kytketty päälle, osaa *t*-funktio hakea järjestelmän kielitiedostoista käännöksen ja korvata ensimmäisen parametrin merkkijonoon merkityt muuttujat funktion toisena parametrina syötetyn taulukon perusteella. (Mts. 408-409.)

```
t('Hauska tavata %nimi.', array('%nimi', => 'Jorma'));
```

Eli tämä esimerkki voisi tulostaa englanninkielisessä kontekstissa vaikkapa “Nice to meet you Jorma.”.

3.3.8 Testattavuus

Käyttöliittymän HTML-elementeissä on käytetty kaikkialla *id*- ja *class*-attribuutteja, joten automaattitesteissä tärkeät XPath-haut on helppo toteuttaa. Myöskin sivujen HTML-rakenne on selkeä.

Testausalustaksi on valittu SimpleTest. Moduulikohtaisen testin kirjoittaminen alkaa lisäämällä *.test*-tiedosto moduulin *.info*-määrittelytiedoston *files[]*-listaan. Tällaisesta *.info*-tiedostosta on esimerkki liitteessä 8. Tämän jälkeen voidaan alkaa kirjoittaa moduulille yksikkötestejä.

```
class EsimerkkiTestiCase extends DrupalWebTestCase {
  public function setUp() { }

  public function testEsimerkkiTesti(){
    $this->assertTrue(true);
  }
}
```

Testien kantaluokka DrupalWebTestCase tarjoaa jonkin verran Drupaliin itseensä liittyviä apufunktioita testitapahtumien simuloimiseen. Esimerkiksi *drupalPost($polku, $kentät, $submitNapinNimi)* -metodin avulla voidaan testata miten lomakkeen käsittely toteutuu.

3.4 Yii

3.4.1 Yleisesti

Yii on Prado-ohjelmistokehyksen tekijöiden uusi BSD-lisenssin alainen ohjelmistokehys, jossa on Pradoon verrattuna paranneltu tehokkuutta ja käytettävyyttä.

Yiin suunnittelun lähtökohtina ovat suorituskykyisyys ja helppokäyttöisyys (Winesett 2010, 8-10). Web-sovelluksen selkeä eriyttäminen ohjelmointikehyksen tiedostoista varmistaa, että kun kehystä halutaan päivittää, ei synny konflikteja tiedostojen välille.

3.4.2 Asennus

Kun Yiin asennuspaketti oli purettu julkiseen sijaintiin, täytyi tarkistaa, että järjestelmävaatimukset toteutuvat. Tämä onnistui menemällä selaimella */requirements/*-kansioon, jolloin saatiin kuvion 13 mukainen listausnäkymä järjestelmävaatimuksista. Asennuspaketin mukana toimitetut demot toimivat suoraan, kunhan ne ensin asetettiin julkisesti näkyville.

Details

Name	Result	Required By	Memo
PHP version	Passed	Yii Framework	PHP 5.1.0 or higher is required.
$_SERVER variable	Passed	Yii Framework	
Reflection extension	Passed	Yii Framework	
PCRE extension	Passed	Yii Framework	
SPL extension	Passed	Yii Framework	
DOM extension	Passed	CHtmlPurifier, CWsdlGenerator	
PDO extension	Passed	All DB-related classes	
PDO SQLite extension	Passed	All DB-related classes	This is required if you are using SQLite database.
PDO MySQL extension	Passed	All DB-related classes	This is required if you are using MySQL database.
PDO PostgreSQL extension	Warning	All DB-related classes	This is required if you are using PostgreSQL database.
Memcache extension	Warning	CMemCache	
APC extension	Passed	CApcCache	
Mcrypt extension	Passed	CSecurityManager	This is required by encrypt and decrypt methods.
SOAP extension	Passed	CWebService, CWebServiceAction	
GD extension with FreeType support	Passed	CCaptchaAction	
Ctype extension	Passed	CDateFormatter, CDateTimeParser, CTextHighlighter, CHtmlPurifier	

passed failed warning

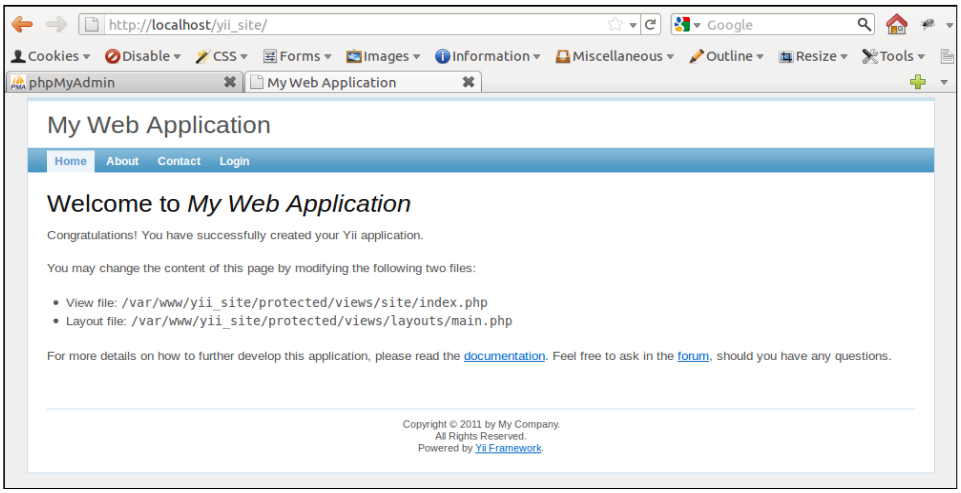
Apache/2.2.17 (Ubuntu) Yii Framework/1.1.8 2011-09-20 13:32

KUVIO 13. Yii vaatimukset

Varsinainen web-sovellus luotiin *yiic*-komentorivityökalun avulla.

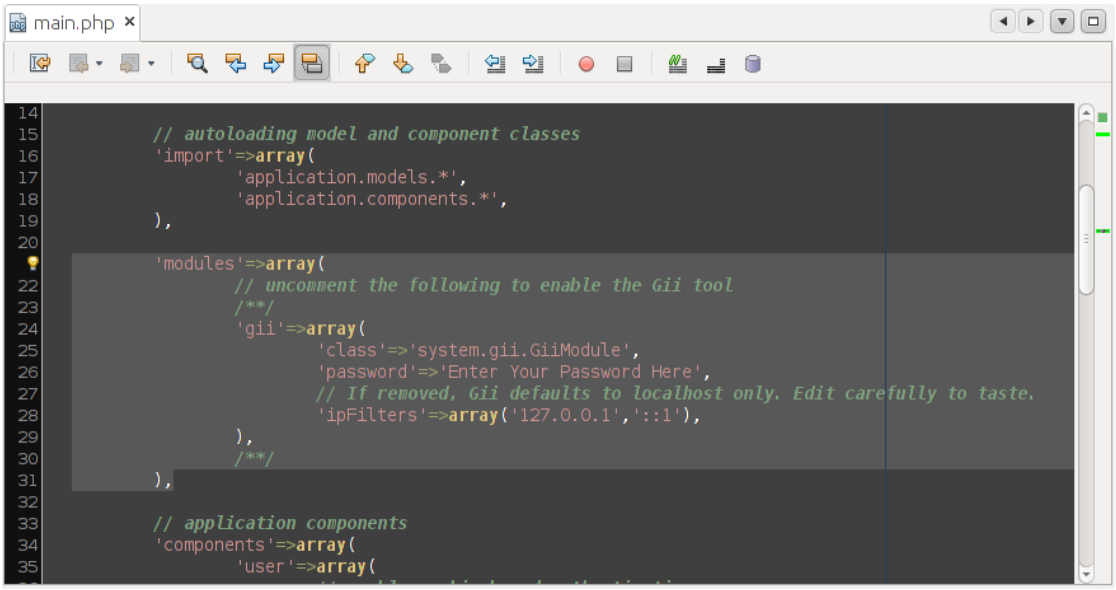
```
./<polku>/framework/yiic webapp <applikaatio>
```

Komento luo Yii web -sovelluksen perusrungon suoritussijaintiinsa, kansioon nimellä <*applikaatio*>. Komennon <*polku*> puolestaan on polku johon Yii on asennettu. Kuviosta 14 nähdään Yii-sovelluksen tervehdysruutu.



KUVIO 14. Yii välittömästi asennuksen jälkeen

Tämän jälkeen käytiin kytkemässä päälle Yiin Gii-moduuli *protected/config/main.php*–tiedostosta kuvion 15 mukaisesti. Samalla asetettiin moduulille oma salasana.



KUVIO 15. Yii moduulien konfigurointi

Gii on Yiin koodigenerointityökalu, jolla uusien sivujen luominen sovellukseen on nopeaa.

Ennen kuin ohjaimen generointityökalua voidaan käyttää, tulee varmistua siitä, että käyttäjällä jolla Apachea (ja siten myös PHP:tä) suoritetaan, on kirjoitusoikeudet *controller*- ja *view*-kansioihin.

3.4.3 Autentikaatio ja käyttäjien hallinta

Yiin konsolityökalu luo sovelluksen luonnin yhteydessä pohjan kirjautumislomakkeelle. Sen päälle kehittäjän on helppo lähteä laajentamaan omaa toteutustaan. Kirjautumislomake löytyy tiedostosta */protected/models/LoginForm.php* ja varsinainen todennuslogiikka tiedostosta */protected/components/UserIdentity.php*.

Yii tarjoaa käyttöoikeuksien valtuutusta varten joko yksinkertaisen suoraan ohjainten yhteyteen kirjoitetun käyttäjäkohtaisen ACL-valtuutuksen, tai hivenen monimutkaisemman käyttäjärooleihin perustuvan RBAC-mallin (Winesett 2010, 171).

3.4.4 Käyttökokemus

Suorituskykyisyys on ollut Yiin suunnittelun lähtökohtana. Tämä näkyy siinä, että kaikkialla kehyksessä on noudatettu tarvepohjaisen latauksen periaatteita (engl. *lazy loading*). Tarvepohjaisella latauksella tarkoitetaan, että tiedostoja ei ladata ennen kuin niiden sisältämiä luokkia kutsutaan. Ja sitä että oliot luodaan vasta niiden ensimmäisen käyttökerran yhteydessä. Menetelmä tuo Yiille sukkeluutta verrattuna sellaisiin järjestelmiin, joissa kaikki komponentit ladataan ja luodaan jokaisen pyynnön yhteydessä. Tästä on hyötyä varsinkin Ajax-pyyntöjä käsitellessä, jolloin harvoin tarvitaan kaikkia web-sovelluksen toimintoja.

Yii sivuasettelun perustana toimivat */protected/views/layout* -kansiosta löytyvät PHP-tiedostot. Erillistä mallipohjamoottoria Yii ei oletuksena käytä, vaan sen mallipohjat toteutetaan HTML-kielen sekaan kirjoitetutulla PHP-koodilla.

3.4.5 CRUD-liittymän toteutus

Yii tukee CRUD-liittymien generointia suoraan ilman lisäosia. CRUDin tekemiseen kuuluu neljä vaihetta:

- tietokannan luominen,
- tietokannan konfigurointi main.php asetustiedostoon,
- tietomallin ja
- CRUDin luonti Gii:llä.

Aluksi MySQL:ään luotiin uusi tietokanta nimellä yii ja sille käyttäjätunnukset.

```
create database yii;
grant all on yii-user.* to 'yii'@localhost identified by 'yii-
pass';
```

Ja tähän tietokantaan yksi uusi taulu.

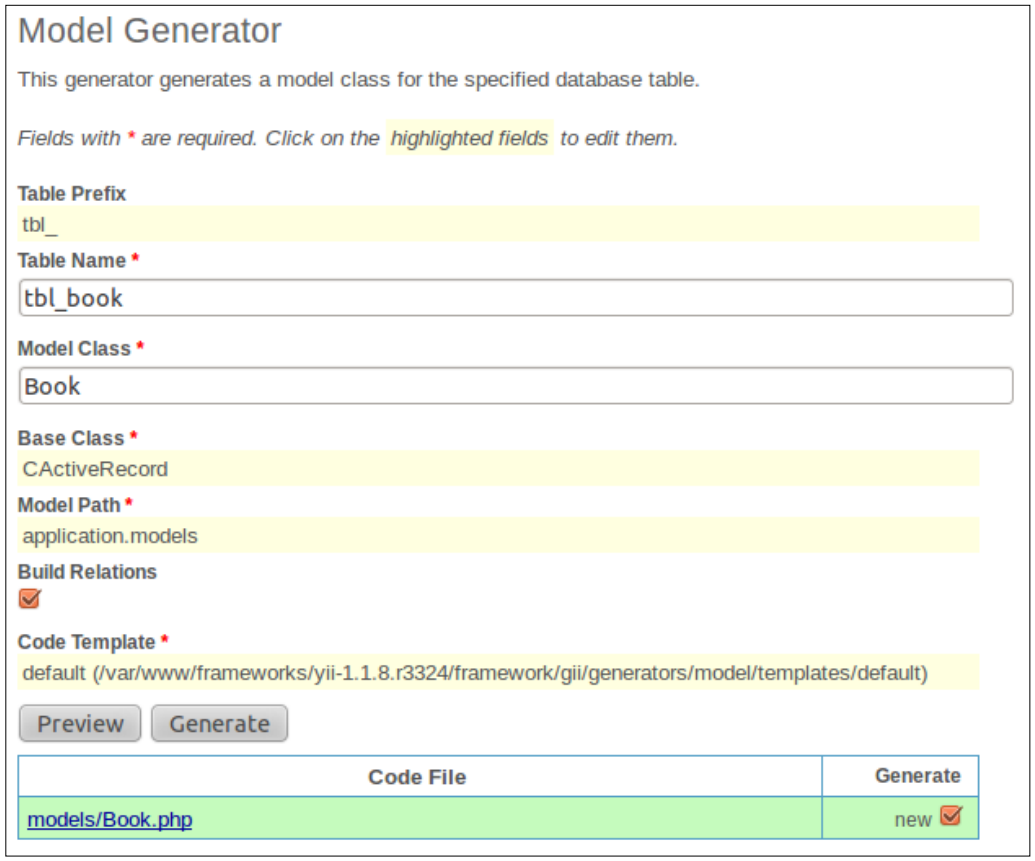
```
CREATE TABLE `yii`.`tbl_book` (
`ID` INT NOT NULL AUTO_INCREMENT ,
`Name` VARCHAR( 40 ) NOT NULL ,
`ISBN` VARCHAR( 20 ) NOT NULL ,
`ReleasedDate` DATE NULL ,
`Pages` INT NULL ,
`Description` TEXT NULL ,
PRIMARY KEY ( `ID` )
) ENGINE = MYISAM ;
```

Uuden tietokannan yhteystiedot lisättiin *protected/config/main.php*-konfiguraatiotiedostoon.

```
'db'=>array(
'connectionString' => 'mysql:host=localhost;dbname=yii',
'emulatePrepare' => true,
'username' => 'yii',
'password' => 'yii',
'charset' => 'utf8',
),
```

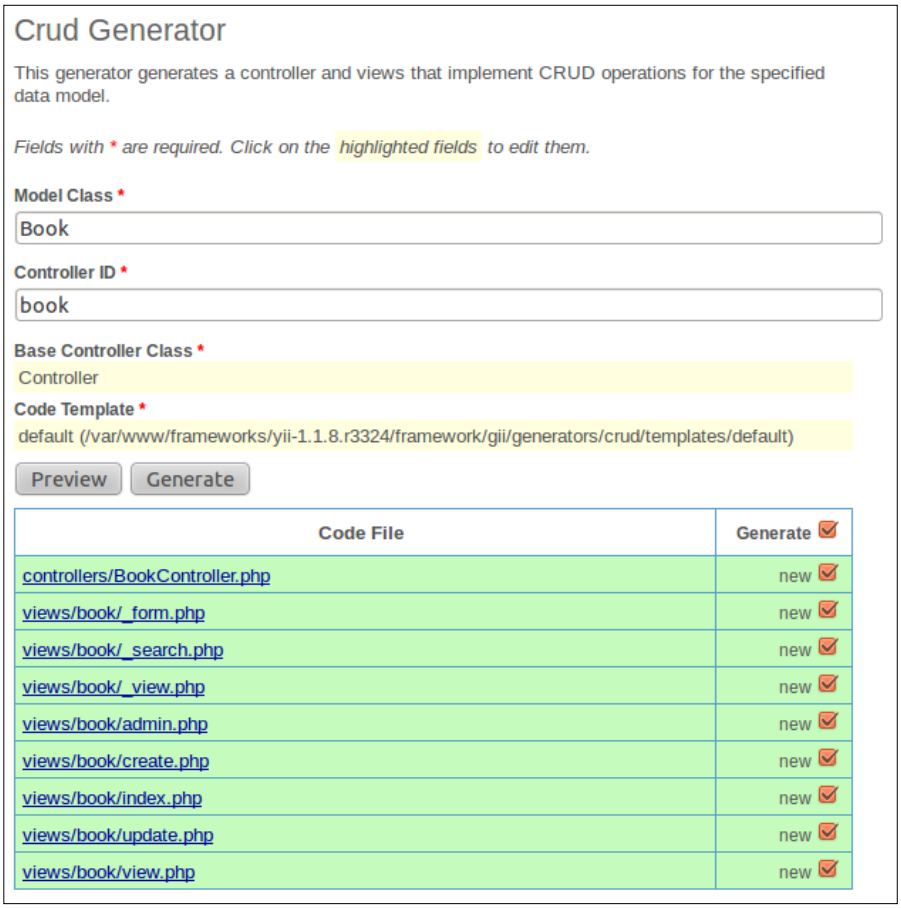
Kun Gii on kytketty päälle, sen voi löytää antamalla index.php:lle request parametriksi *r* arvo *gii*. Giin salasana oli asetettu jo järjestelmän asennuksen yhteydessä luvussa 3.4.2.

Seuraavaksi siirryttiin käyttämään Gii Model Generatoria, sen käyttöliittymä on esitelty kuviossa 16. Gii Model Generator loi meille CActiveRecordia perivän Book-luokan.



KUVIO 16. Yii tietomallin luontivelho

Nyt tuoreelle mallille luotiin Gii Crud Generatorilla käyttöliittymälomakkeet. Kun kuviossa 17 esitellyn CRUDin luontivelhon Generate-painiketta painettiin, saatiin linkki josta luotuja CRUD-liittymiä pääsi välittömästi kokeilemaan.



KUVIO 17. Yii CRUDin luontivelho

Operaation tulos ei ollut täysin toivotun kaltainen; päivämääräkenttä olikin pelkkä tekstikenttä, jota ei edes ollut lokalisoitu suomalaisiin päivämääriin. VARCHAR(40) tyyppinen *Nimi*-kenttä kuitenkin toteutti sille asetetun 40 merkin syöttörajoitteen asiallisesti

Yiillä CRUDien toteuttaminen oli luontevaa ja helppoa, mutta muiden kuin tavallisten tekstimuotoisten tietotyyppien tukemisen suhteen sen ominaisuudet olivat varsin vaatimattomat. Ainakin verrattuna vertailun muiden ohjelmointikehysten tuottamiin CRUDeihin.

3.4.6 Dokumentaatio ja käyttäjäyhteisö

Yii oli tutkitusta joukosta se järjestelmä, jolle oli haastavinta löytää dokumentaatiota. Aiheesta oli kirjoitettu vain muutamia kirjoja, ja nekin yleensä käsittelivät useita ohjelmointikehyksiä kerralla. Käytännössä ainoaksi paikaksi tiedon hakuun jäikin Yiin oma verk-

kosivusto. Myöskään mitään kovin suuria toimijoita ei Yiin tueksi löydetty.

3.4.7 Lokalisointi

Yii tarjoaa työkalut merkkijonojen, päivämäärien ja lukujen lokalisointia varten. Lisäksi mukana on CLDR (Common Locale Data Repository) komponentti, jolta voidaan maatunnuksen perusteella pyytää aakkosjärjestykset, translitterointisäännöt ja valuuttatunnukset, sekä kielikäännökset muun muassa viikonpäivien ja kuukausien nimille. (Agile Web Application Development with Yii 1.1 and PHP5 s. 279–281)

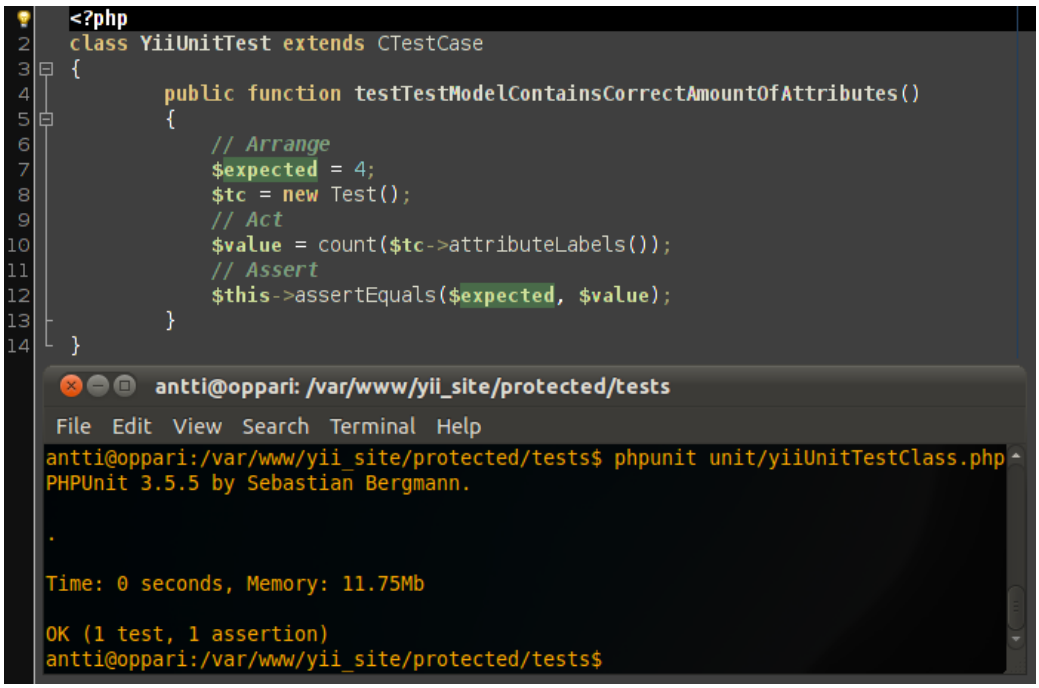
Merkkijonon kielikäännös onnistuu Yii-sovelluksen staattisen *t* metodin avulla.

```
Yii::app()->language='fi';
echo Yii::t('category', 'Hello {Name}!', array('{Name}' =>
$name));
```

Käännösmetodi etsii järjestelmän tietovarastoista toisen parametrin viestiä vastaavan käännöksen, korvaten siitä merkityt merkkijonot kolmannen parametrin taulukon mukaisesti. Käännöstietovarastona voi toimia PHP-taulukko, GNU Gettext -tiedosto tai tietokannan taulu. Kategorioiden avulla käännökset voidaan tarjota kontekstikohtaisesti (Agile Web Application Development with Yii 1.1 and PHP5).

3.4.8 Testattavuus

Monien muiden tutkittujen ohjelmointikehysten tapaan, myös Yii on valinnut yksikkötestausalustakseen suositun PHPUnit-testikehyksen. Yii-sovelluksessa on mukana suoraan oletusasetuksillaan valmiit rakenteet yksikkötestejä varten, joten testejä voi yksinkertaisesti alkaa kirjoittamaan heti sovelluksen luomisen jälkeen. Kuviossa 18 esitellään kokeilutesti PHPUnit-testikehyksellä suoritettuna.



KUVIO 18. PHPUnit yksikkötesti suoritettu

Yiissä PHPUnitiin on integroitu myös tuki Selenium-automatisoitujen hyväksymistestien suorittamiseen PHP-koodista käsin (Winesett 2010, 45–47). Ohessa esimerkki tälläistä Selenium-automaatiota hyödyntävästä hyväksymistestistä.

```
class EsimerkkiWebTestCase extends CWebTestCase {
  protected function setUp() {
    parent::setUp();
    $this->setBrowserUrl(
      'http://localhost/yii_site/books/');
  }
  public function testEsimerkki() {
    $this->open(view/1');
    $this->assertTextPresent('View Book #1');
  }
}
```

Giillä generoiduissa CRUD-näkymissä on vain minimaalisesti id ja class tageja, ja niistä osa on kryptisesti nimettyjä. Tämä vaikeuttaa XPath-kyselyiden kirjoittamista.

3.5 Joomla!

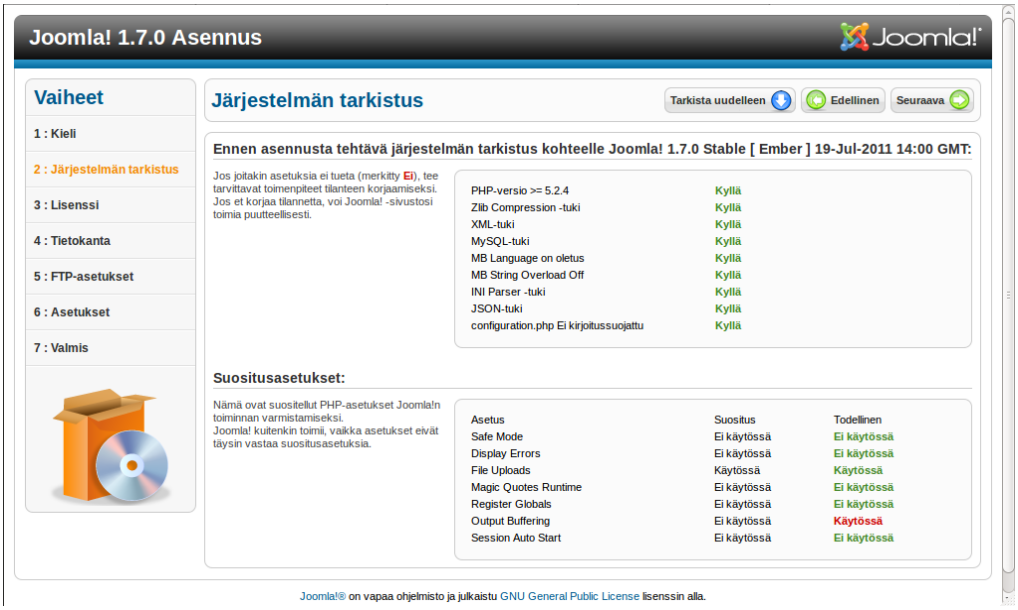
3.5.1 Yleisesti

Joomla on GPL-lisenssin alainen sisällönhallintajärjestelmä. Se on tutkituista järjestelmistä selvästi laajimmin käytetty.

Lähdekoodin lukuisat TODO-merkinnät eivät antaneet kovinkaan vakuuttavaa kuvaa Joomlan laadusta. Lisäksi Joomlan asennuksen jälkeen kaikki järjestelmän tiedostot olivat oletuksena julkisesti saatavilla siten, että boostrap tiedosto sijaitsi järjestelmän juuressa, ja järjestelmän omat kansiot tämän alapuolella. Kyseinen ratkaisu tuntui hivenen huolimattomalta. Useimmissa muissa tutkituissa järjestelmissä oli vain yksi julkinen kansio, jossa järjestelmän bootstrap-tiedosto sijaitsi. Tai sitten ainakin Internet-selainten pääsy järjestelmäkansioihin oli estetty apachen *.htaccess*-tiedostolla.

3.5.2 Asennus

Joomlan asennus suoritettiin asiallisen kuviossa 19 esitellyn asennusvelhon avulla. Asennusvelhon käyttöliittymä on jopa käännetty useille kielille, mukaan lukien suomi.



KUVIO 19. Joomla! asennusvelho

MySQL-tietokanta on Joomlan toiminnalle välttämätön vaatimus. Tietokanta käytiin luomassa MySQL-komentoriviltä seuraavilla käskyillä.

```
create database joomla;
grant all on joomla.* to 'joomla'@localhost identified by
'joomla';
```

Lopuksi täytyi vielä poistaa asennuskansio. Tämän jälkeen Joomla oli valmis käytettäväksi.

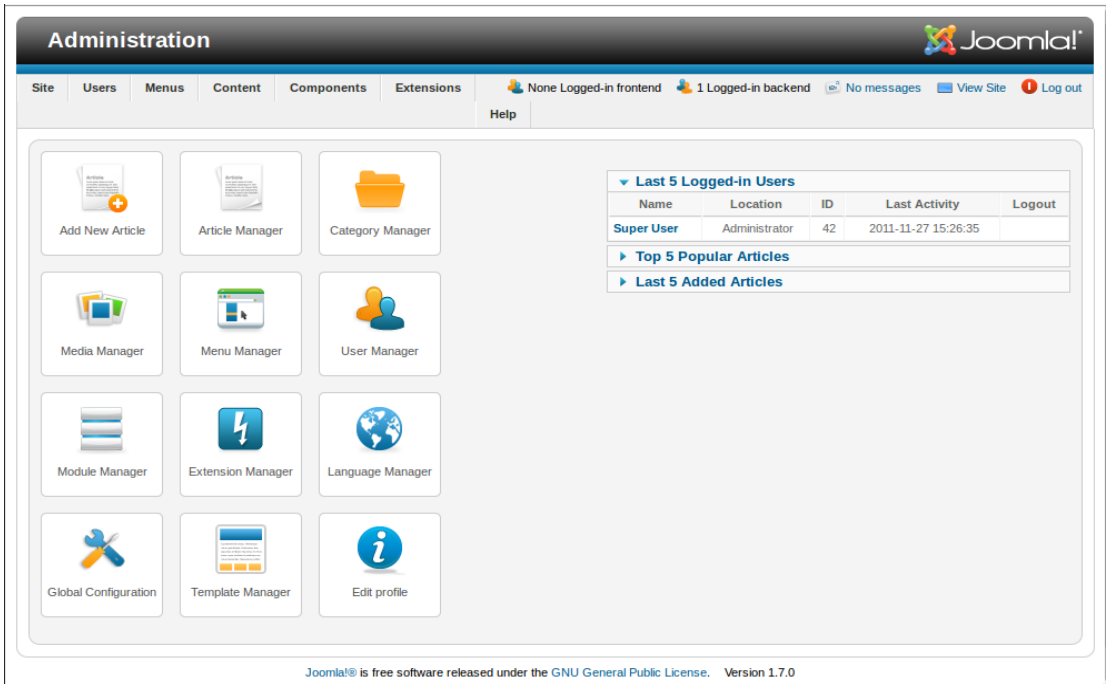
3.5.3 Autentikaatio ja käyttäjien hallinta

Joomlan käyttöoikeuksien hallinta on toteutettu kolmitasoisesti, siten että yksittäiset käyttäjät kuuluvat käyttäjäryhmiin, ja näille käyttäjäryhmille on määritelty käyttöoikeustaso, jonka mukaan käyttäjät sitten saavat pääsyn eri alasivuille ja sivuryhmiin (engl. *category*). Toki Joomlan sivut voidaan myös määrätä julkisiksi, jolloin kuka hyvänsä voi nähdä ne ilman sisään kirjautumista (North 2011, 357). Julkisuusmääräys tehdään kategorian tai artikkelin *Access*-ominaisuudella, joka määrää onko artikkeli täysin julkinen,

vain rekisteröityneiden käyttäjien nähtävissä, vai pelkästään järjestelmänvalvojille tarkoitettu.

3.5.4 Käyttökokemus

Joomlan hallinnointi tapahtuu ylläpitoliittymän kautta, jota Joomla itse kutsuu nimellä *backend* . Sisältöjä voi jonkin verran hallita myös julkisen liittymän kautta (vastaavasti frontend). Backend löytyy */administrator/*-kansiosta. Kuvion 20 mukaisesta hallintaliittymästä löytyy kattavasti toimintoja valikoiden, kielien, laajennusten, käyttäjien ja sisällön hallintaan.



KUVIO 20. Joomlan hallintaliittymä

Joomlaan on kirjoitettu tuhansia toiminnallisuuksia lisääviä ja korjaavia laajennuksia. Laajennushakemisto löytyy osoitteesta http://extensions.joomla.org/.

Rahmel (2007, 376) listaa Joomlan heikkouksiksi puuttuvan dokumenttien versionhallinnan, puuttuvat datan tuonti -toiminnot, rajoittuneen käyttöoikeuksien hallinnan, puuttuvan kuormantasauksen ja

käyttöliittymän ulkonäön muokkauksen vaikeuden. Näin vanhaan tietoon on kuitenkin syytä suhtautua varauksella.

Joomlan sivuasettelua voidaan muokata luomalla omat mallipohjatiedostot *templates*-kansioon. Pakolliset tiedostot tässä kansiossa ovat *index.php* ja *templateDetails.xml*. XML-tiedosto sisältää yleisiä tietoja mallipohjasta ja listaa mallipohjaan liittyvät sisältöalueet. Sisältöalueisiin voidaan sitten hallintaliittymästä sijoittaa komponentteja. *Index.php*-tiedostossa puolestaan on varsinainen mallipohjan HTML-rakenne.

3.5.5 CRUD-liittymän toteutus

CRUDien tekemiseen Joomla ei tarjoa valmista työkalua. Omaa CRUD-moduulin toteutusta ei ryhdytty ohjelmoimaan, koska työn toimeksiantajalle on hyötyä vain sellaisiin ominaisuuksiin tutustumisesta, jotka ovat jo valmiiksi olemassa olevia.

3.5.6 Dokumentaatio ja käyttäjäyhteisö

Kun Joomla on tutkituista järjestelmistä eniten käytetyin, ei ole lainkaan yllättävää, että sillä on myös aktiivisimmat käyttäjäfoorumit, ja että siitä on kirjoitettu kymmeniä kirjoja. Valitettavasti suurin osa kirjallisuudesta käsittelee järjestelmän käyttöliittymän kautta tapahtuvaa käyttöä. Vain muutamassa kirjassa opastetaan omien moduulien kirjoittamisen saloihin.

Monet monikansalliset yhtiöt käyttävät Joomlaa, varsinkin kansallisilla sivustoillaan, esimerkiksi McDonaldsin Arabialaisille suunnatut kotisivut on toteutettu Joomlalla (Burge 2010).

Joomlan Wiki-dokumentaatiossa ilahdutti artikkelien jaottelu käyttäjäroolien mukaan, tiedon selailua helpottaa, kun voi suodattaa artikkelit riippuen siitä onko lukija suunnittelija, ylläpitäjä tai kehittäjä.

3.5.7 Lokalisointi

Joomlan käyttäjälle näkyvät lokalisoitavat tekstit tulee esittää staattisen *JText::_*-metodin avulla. Esimerkiksi näin:

```
echo JText::_('MOD_SEARCH_LABEL_TEXT'));
```

Joomlan ydintoiminnot on käännetty peräti 40 kielelle (North 2011, 58). Näiden perustoimintojen kielikäännökset, sekä lukujen ja päivämäärien muodot, voidaan määrittää locale-kohtaisesti ini-tiedostoihin (Muehleisen, 2011). Esimerkiksi suomelle tällainen tiedosto löytyy sijainnista *language/fi-FI/fi-FI.ini*.

3.5.8 Testattavuus

Selainautomatisoituja hyväksymistestejä ajatellen, Joomla tarjoaa mukavan selkeän HTML-rakenteen ja sen elementeissä on kattavasti hyvin nimettyjä *id*- ja *class*-attribuutteja.

Ohjelmistokehykseen itsensä liittyvät – PHPUnit-testausalustalla toteutetut - testit haettiin Subversion-varastosta komennolla:

```
svn checkout http://joomlacode.org/svn/joomla/testing/trunk/
--username anonymous
```

Omien moduulien yksikkötestausta varten valmista alustaa ei valitettavasti Joomlan mukana toimiteta. Joomlan dokumentaatiosta (http://docs.joomla.org) kuitenkin löytyy jonkin verran ohjeita jatkuvan integraation toteuttamiseksi.

4 TULOKSET

4.1 Symfony

Kaikista tutkituista järjestelmistä Symfonyyn tutustumiseen jouduttiin käyttämään suhteessa eniten aikaa. Joten se ei varmastikaan ollut ainakaan vertailun helppokäyttöisin järjestelmä. Vaikeutta aiheutti varsinkin Symfonyn siirtyminen äskettäin toiseen versioonsa. Ensimmäiseen versioon verrattuna Symfony2 on lähes kokonaan uusiksi kirjoitettu. Siitä johtuen myöskään suurin osa Symfonyä käsittelevästä kirjallisuudesta tai Internetistä löydetyistä oppaista ei ollut enää ajantasaista. Uusi versio sai aikaan myös epäilyksen siitä, onko järjestelmä vielä riittävän valmis oikeaan käyttöön.

Oikean dokumentaation löytämisen jälkeen, CRUDin generoiminen Symfonylle lopulta onnistui. Generoitu CRUD oli kuitenkin turhan yksinkertainen ominaisuuksiltaan, ja täysin vailla muotoiluja.

Symfonyn tietotyypin luonnissa törmättiin monta kertaa erilaisiin käyttöoikeusongelmiin, nämä ovat toki ohjelmistokehityksessä arkipäivää, mutta kuitenkin aiheuttivat ihmetystä, kun perusasiat eivät toimineetkaan oletusasennuksella.

4.2 Zend

Kenties eniten Zendin uskottavuutta lisäävä tekijä, on sen kirjoittajajoukko, joka työskentelee myös PHP:n ytimen, Zend Enginen parissa.

Oman moduulin kirjoittaminen Zendille oli kiinnostavaa, mutta työlästä. Zendin periaatteita noudattaen yksinkertaisenkin asian tekeminen jakautui valitettavan moneen, erillään sijaitsevaan tiedostoon.

Koska kaikki Zendin toiminnot toteutetaan sen omia luokkia laajentamalla, ja Zend komponenttien väliset kytkökset on tehty tarkoituksella väljiksi, on helppoa käyttää Zendistä pelkästään niitä osia mitkä koetaan vahvimmiksi. Tämä mahdollistaa myös Zendin käyttämisen jossakin toisessa kehysjärjestelmässä komponenttikirjaston ominaisuudessa.

Eräs Zendin vahva puoli on myös ajantasaisen kirjallisuuden runsas määrä. Painotuotteiden ongelmana on toki alalle tyypillinen nopea muutos, joten tilanne saattaa muuttua yllättäen.

4.3 Drupal

Käyttökokemus oli oman moduulin ohjelmointia lukuun ottamatta niin hyvä, että Drupal muodostui eräänlaiseksi henkilökohtaiseksi suosikiksi. Siinä kaikki vain tuntui toimivan, ja ilman joka lähtöön löytyviä moduuleitakin järjestelmässä oli mukavasti hyviä ominaisuuksia. Tuotantojärjestelmän käyttöä nopeuttavat välimuistitiedostot aiheuttivat hieman hämmennystä omaa moduulia kirjoittaessa, kun omat muutokset eivät näkyneetkään järjestelmässä, ennen kuin välimuisti käytiin tyhjentämässä.

Eräs huolestuttavimpia seikkoja Drupalille kehittämisen suhteen on, että se ei lupaa minkäänlaista taaksepäin yhteensopivuutta. Tietoturvan vuoksi päivityksistä huolehtiminen on kuitenkin välttämätöntä. Kolmansien osapuolien toteuttamat moduulit tuottavat oman ongelmansa, sillä vaikka Drupal itsessään olisikin päivityksissä ajan tasalla, moduulit eivät välttämättä ole olleet turvallisia alun perinkään.

4.4 Yii

Muiden tutkittujen ohjelmointikehysten tapaan, myöskään Yiissä ei ole kovin paljoa valmista toiminnallisuutta, vaan perustoimintojenkin toteuttamiseen tarvitaan kehittäjältä paljon työtä. CRUD työkalut olivat helppokäyttöiset, mutta niiden tuotokset olivat käytettävyydeltään ja syötteen validoinnin suhteen heikkoja.

Yiin vakavin puute on varmastikin sen käyttäjäyhteisön pienuus. Dokumentaatiota kyllä löytyy, mutta käyttäjäfoorumit olivat varsin hiljaisia.

Tarvepohjainen lataus (engl. *lazy loading*) on Yiin parhaita puolia. Erityisesti Ajax-kutsujen vasteajat paranevat varmasti, kun kaikkia järjestelmän riippuvuuksia ei ladata jokaisen suorituksen yhteydessä.

4.5 Joomla!

Joomla oli tutkituista järjestelmistä selvästi eniten käytetty, ja sen käyttäjäyhteisö on hyvin aktiivista. Valitettavasti pitkästä historiasta ja kirjavasta kehittäjäjoukosta syntyy myös ongelmia, arkkitehtuurin ja lähdekoodien rämettyessä ajan saatossa.

Verrattuna Drupalin hallintatoimintoihin Joomla tuntui karulta ja vanhanaikaiselta. Myöskin käytettävyyden suhteen hallintaliittymässä oli epäintuitiivisia ratkaisuja. Esimerkiksi katseluoikeuksien erottaminen muokkausoikeuksista tuntui täysin tarpeettomalta.

Valtavan käyttäjäyhteisönsä ansiosta Joomla on kuitenkin houkutteleva vaihtoehto, yhteisön tukea kun sillä riittää varmasti pitkälle tulevaisuuteen.

4.6 Vertailu

Opinnäytetyössä tarkasteltujen ohjelmointikehysjärjestelmien ominaisuudet on pisteytettynä taulukossa 2 asteikolla 0-5. Taulukkoon on laskettu, toimeksiantajan esittämien painoarvojen perusteella, jokaiselle järjestelmälle yleinen arvosana. Arvosanojen perusteella järjestelmät voidaan asettaa järjestykseen.

TAULUKKO 2. Numeerinen arviointi

	Paino-arvo	Symfony	Zend	Drupal	Yii	Joomla!
Asennus	10 %	4	4	4	3	4
Autentikaatio ja käyttäjien hallinta	10 %	2	2	4	3	3
Käyttökokemus	30 %	2	3	4	2	3
CRUD-toteutus	10 %	3	3	4	3	0
Dokumentaatio ja yhteisö	20 %	2	4	5	2	4
Lokalisaatio	10 %	3	4	4	4	4
Testattavuus	10 %	5	3	4	4	3
Yhteensä	100 %	**54 %**	**66 %**	**84 %**	**54 %**	**62 %**

Vaikka tutkitut järjestelmät saivatkin tässä tutkimuksessa keskinäisen järjestyksen, ei numeroita voida tuijottaa liikaa, vaan ohjelmointikehystä valitessa tulee miettiä enemmän sillä hetkellä toteutettavan ohjelmistotuotteen keskeisiä tarpeita. Kaikista tutkituista kehysjärjestelmissä kuitenkin löytyi joitakin ominaisuuksia, joissa ne olivat muita vahvempia. Erityisesti tulee pohtia sitä, miten rakennettavan ohjelmistotuotteen vaatimukset voisivat toteutua missäkin kehysjärjestelmässä.

On syytä huomioida myös, että varsinkin käyttökokemukseen liittyvät arviot ovat opinnäytetyön kirjoittajan henkilökohtaisia mielipiteitä, ja joku toinen arvioitsija saattaa painottaa täysin eri asioita tärkeinä. Lisäksi on syytä huomioida, että kaikki tutkittavat järjestelmät elävät jatkuvan muutoksen alaisina. Joten nyt järjestelmistä kerätyt ajantasaiset tiedot saattavatkin muutaman vuoden kuluttua olla jo auttamattomasti vanhentuneita.

5 POHDINTA

Sekä julkaisujärjestelmien että ohjelmointikehysten ottaminen yhdessä mukaan samaan vertailuun aiheutti haasteita arvioinnille. Kuinka voidaan järkevästi verrata ominaisuuksia, jotka toisista järjestelmistä löytyvät valmiina, mutta toisissa joutuu itse toteuttamaan? Kenties onkin järkevämpää verrata tätä eroa käytännön tasolla. Kun asiakas ostaa ohjelmistoonsa ominaisuuksia pyrähdyksittäin, pitäisi jokaisen pyrähdyksen lopussa voida esitellä jotain valmista. Ilman muuta nopeasti näkyvää sisältöä tuottavat julkaisujärjestelmät tarjoavat pyrähdyksen lopussa tehtävään demonstraatioon asiakkaalle enemmän pureskeltavaa. Toisaalta taas valmiista komponenteista saadut edut kuitenkin karisevat nopeasti, jos niistä sattuukin löytymään tietoturva-aukkoja. Ohjelmointikehysten eduksi voidaan katsoa myös se, että niiden päälle järjestelmän rakentaneet ohjelmoijat varmasti tuntevat itse toteuttamansa järjestelmät läpikotaisin, ja ovat siten valmiimpia toteuttamaan muutoksia.

Opinnäytetyön alussa tehty ohjelmointikehysjärjestelmien rajaaminen laajasta joukosta viiteen kiinnostavimpaan, perustui kenties turhan paljon järjestelmien tunnetuimuuteen ja käyttäjäyhteisön kokoon niiden teknisten ominaisuuksien sijasta. Koska yhteisön tuki koettiin toimeksiantajan puolesta suhteessa erääksi painavimmista vertailukriteereistä, on näillä perusteilla tehty tutkittavien järjestelmien rajaaminen kuitenkin varsin perusteltua.

Vuosikausia kehitetyt sisällönhallintajärjestelmät Drupal ja Joomla kantavat mukanaan historian painolastia. Jokin vähemmän valmiita ominaisuuksia sisältävä ohjelmointikehys saattaakin siten olla sijoitus tulevaisuuden ylläpidettävyyteen.

Kaikki tutkitut järjestelmät ovat sen verran laajoja kokonaisuuksia, että ne vaativat kehittäjiltä paljon opettelemista. Yiitä lukuun ottamatta tutkituille järjestelmille on tarjolla jopa jonkin verran kurssi-

muotoista koulutusta. Zend, Drupal ja Symfony koulutusta suomessa tarjoaa Brain Alliance Oy. Joomlakursseja puolestaan voi löytää www.joomlaportal.fi -sivustolta.

Järjestelmille annetut lopulliset arvosanat ovat kenties enemmänkin suuntaa antavia, kuin tarkkoja arvoja, mutta vertailukelpoisia suhteessa toisiinsa. Yhden oikean valinnan löytäminen ei opinnäytetyön tavoitteena ollutkaan, vaan järjestelmien keskinäisten eroavaisuuksien paikallistaminen. Merkittävimmät erot ja yhtäläisyydet onnistuttiin mielestäni opinnäytetyön työstämisen aikana selvittämään varsin hyvin.

LÄHTEET

Allen, R. Lo, N. Brown, S. 2008 Zend Framework in Action. USA: Manning.

Allen, R. Getting started with Zend framework. 2010. Verkkodokumentti. Viitattu 29.10.2011. http://akrabat.com/wp-content/uploads/Getting-Started-with-Zend-Framework.pdf

Bergmann, S. PHPUnit Manual. 2011. Verkkodokumentti. Viitattu 5.10.2011. www.phpunit.de/manual/3.6/en/phpunit-book.pdf.

Bergmann, S. Priebsch, S. 2011. Real-World Solutions for Developing Hiqh-Quality PHP Frameworks and Applications. USA. Wiley Publishing.

Burge, S. 2010. Joomla Community Portal. McDonald's uses Joomla. Verkkodokumentti. Viitattu 25.11.2011. http://community.joomla.org/blogs/community/1272-mcdonalds-uses-joomla.html

Buschmann, F. Henney, K. Schmidt, D. 2007. Pattern-Oriented Software Architecture Volume 4. Englanti. Wiley Publishing.

Describing Your Database as XML Schema. Verkkodokumentti. Viitattu 10.10.2011. http://www.propelorm.org/documentation/02-buildtime.html

Drupal: HTML5 Initiative. 2011. Verkkodokumentti. Viitattu 25.11.2011. http://drupal.org/community-initiatives/drupal-core/html5

Geller, T. 2011 Drupal 7 Visual QuickStart Guide. USA. Peachpit Press.

Lyman, F. 2009. Pro Zend Framework Techniques: Build Full CMS Project. USA. Apress.

Muehleisen, J. 2011. Welcome to Joomla! Verkkodokumentti. Viitattu 25.11.2011. http://welcometojoomla.com/how-do-i-articles/31-configuration/116-how-to-change-the-joomla-date-format

Mundy, S. 2007. Zend_Acl / Zend_Auth example scenario. Verkko-dokumentti. Viitattu 12.11.2011.
http://devzone.zend.com/article/1665

North, B. 2011. Joomla! 1.6: A User's Guide: Building a Successful Joomla! Powered Website. USA. Prentice Hall.

O'Phinney, M. Dubravszky, J. 2011. Zend Framework 2.0 Roadmap. Verkkodokumentti. Viitattu 26.11.2011.
http://framework.zend.com/wiki/display/ZFDEV2/Zend+Framework+2.0+Roadmap

Pearce, J. 2011. Mobile Web Development with WordPress, Joomla!, and Drupal. USA. Wiley Publishing.

Porebski, B. Przystalski, K. Nowak, L. 2011. Building PHP Applications with Symfony, CakePHP, and Zend Framework. USA. Wiley Publishing.

Protacon. 2011. Konsernin verkkosivusto. Viitattu 6.9.2011.
http://www.protacon.fi/

Symfony Blog. 2011. Viitattu 31.10.2011.
http://symfony.com/blog/.

Symfony Cookbook. 2011. Viitattu 10.11.2011.
http://symfony.com/doc/2.0/cookbook/, Doctrine.

Symfony käyttöopas. 2011. Verkkodokumentti. Viitattu 30.10.2011.
http://symfony.com/doc/current/, Translations.

Symfonians.net. 2011. Viitattu 31.10.2011.
http://symfonians.net/applications/country/FI/page.

Särkkä, T. 2011. Open source framework –selvitys. Protacon Oy.

VanDyk, J. 2008. Pro Drupal Development, Second Edition. USA. Apress.

Vaswani, V. 2010 Zend Framework: A Beginner's Guide. USA. McGraw-Hill.

Winesett, J. 2010. Agile Web Application Development with Yii 1.1 and PHP5. USA. Packt Publishing.

LIITTEET

Liite 1. Zend CRUD -ohjain.

(application/controllers/BooksController.php)

```
class BooksController extends Zend_Controller_Action {
  public function init() { }

  public function indexAction() {
    $books = new Application_Model_DbTable_Books();
    $this->view->books = $books->fetchAll();
  }

  public function addAction() {
    $form = new Application_Form_Book();
    $form->submit->setLabel('Add');
    $this->view->form = $form;

    if($this->getRequest()->isPost()) {
      $formData = $this->getRequest()->getPost();
      if($form->isValid($formData)){
        $name = $form->getValue('Name');
        $isbn = $form->getValue('ISBN');
        $pages = $form->getValue('Pages');
        $date = $form->getValue('ReleasedDate');
        $description = $form->getValue('Description');

        $books = new Application_Model_DbTable_Books();
        $books->addBook($name, $isbn, $date, $pages,
$description);

        $this->_helper->redirector('index');
      } else {
        $form->populate($formData);
      }
    }
  }

  public function editAction() {
    $form = new Application_Form_Book();
    $form->submit->setLabel('Edit');
    $this->view->form = $form;

    if($this->getRequest()->isPost()) {
      $formData = $this->getRequest()->getPost();
      if($form->isValid($formData)) {
        $id = (int)$form->getValue('ID');
        $name = $form->getValue('Name');
```

```
        $isbn = $form->getValue('ISBN');
        $pages = $form->getValue('Pages');
        $date = $form->getValue('ReleasedDate');
        $description = $form->getValue('Description');

        $books = new Application_Model_DbTable_Books();
        $books->updateBook($id, $name, $isbn, $date, $pages,
$description);

        $this->_helper->redirector('index');
      } else {
        $form->populate($formData);
      }
    } else {
      $id = $this->_getParam('id', 0);
      if($id > 0) {
        $books = new Application_Model_DbTable_Books();
        $form->populate($books->getBook($id));
      }
    }
  }

  public function deleteAction() {
    if($this->getRequest()->isPost()) {
      $del = $this->getRequest()->getPost('del');
      if($del == 'Yes') {
        $id = $this->getRequest()->getPost('id');
        $books = new Application_Model_DbTable_Books();
        $books->deleteBook($id);
      }
      $this->_helper->redirector('index');
    } else {
      $id = $this->_getParam('id', 0);
      $books = new Application_Model_DbTable_Books();
      $this->view->books = $books->getBook($id);
    }
  }
}
```

Liite 2. Zend CRUD -malli.

(application/models/DbTable/Books.php)

```
class Application_Model_DbTable_Books extends
Zend_Db_Table_Abstract {
  protected $_name = 'book';

  public function getBook($id) {
    $row = $this->fetchRow('id = '.$id);
    if(!$row)
      throw new Exception ("Can't find row ".$id);
```

```
    return $row->toArray();
  }

  public function addBook($name, $isbn, $releaseDate,
$pageCount, $description) {
    $this->insert(array(
      'Name' => $name,
      'ISBN' => $isbn,
      'ReleasedDate' => $releaseDate,
      'Pages' => $pageCount,
      'Description' => $description,
    ));
  }

  public function updateBook($id, $name, $isbn, $releaseDate,
$pageCount, $description) {
    $this->update(
      array(
        'Name' => $name,
        'ISBN' => $isbn,
        'ReleasedDate' => $releaseDate,
        'Pages' => $pageCount,
        'Description' => $description,
      ),
      'id ='.$id
    );
  }

  public function deleteBook($id) {
    $this->delete('id = '.$id);
  }
}
```

Liite 3. Zend CRUD -listausnäkymä.

(application/views/scripts/books/index.phtml)

```
<div id="view-content">
  <div>
    <a href="<?php
      echo $this->url(
        array('controller' => 'books', 'action' => 'add')
      );
    ?>" >Add new book</a>
  </div>
  <table>
    <tr>
      <th>Name</th>
      <th>ISBN</th>
      <th>Pages</th>
```

```
      <th>Date of release</th>
    </tr>
  <?php foreach($this->books as $book) : ?>
  <tr>
    <td><?php echo $this->escape($book->Name) ?></td>
    <td><?php echo $this->escape($book->ISBN) ?></td>
    <td><?php echo $this->escape($book->Pages) ?></td>
    <td><?php echo $this->escape($book->ReleasedDate) ?></td>
    <td>
      <a href="<?php echo $this->url(array(
        'controller' => 'books',
        'action' => 'edit',
        'id' => $book->ID
      )); ?>">Edit</a>
      <a href="<?php echo $this->url(array(
        'controller' => 'books',
        'action' => 'delete',
        'id' => $book->ID
      )); ?>">Remove</a>
    </td>
  </tr>
  <?php endforeach; ?>
  </table>
</div>
```

Liite 4. Zend CRUD-lomake -luokka.

(application/forms/Book.php)

```
class Application_Form_Book extends Zend_Form {

  public function init() {
    $this->setName('book');

    $id = new Zend_Form_Element_Hidden('ID');
    $id->addFilter('Int');

    $name = new Zend_Form_Element_Text('Name');
    $name->setLabel('Name')
       ->setRequired(true)
       ->addFilter('StripTags')
       ->addFilter('StringTrim')
       ->addValidator('NotEmpty');

    $isbn = new Zend_Form_Element_Text('ISBN');
    $isbn->setLabel('ISBN')
       ->setRequired(false)
       ->addFilter('StripTags')
       ->addFilter('StringTrim');

    $pages = new Zend_Form_Element_Text('Pages');
```

```
    $pages->setLabel('Pages')
        ->setRequired(true)
        ->addFilter('Int');

    $date = new
Zend_Dojo_Form_Element_DateTextBox('ReleasedDate');
    $date->setLabel('Release Date');

    $description = new Zend_Form_Element_Text('Description');
    $description->setLabel('Description')
       ->setRequired(false)
       ->addFilter('StripTags')
       ->addFilter('StringTrim');

    $submit = new Zend_Form_Element_Submit('submit');
    $submit->setAttrib('id', 'submitbutton');

    $this->addElements(array($id, $name, $isbn, $pages, $date,
$description, $submit));
  }
}
```

Liite 5. Zend CRUD -lisäysnäkymä.

(application/views/scripts/books/add.phtml)

```
<?php
$this->title = "Add new book";
$this->headTitle($this->title);
echo $this->form;
```

Liite 6. Zend CRUD -muokkausnäkymä.

(application/views/scripts/books/edit.phtml)

```
<?php
$this->title = 'Edit book';
$this->headTitle($this->title);
echo $this->form;
```

Liite 7. Zend CRUD -poistonäkymä.

(application/views/scripts/books/delete.phtml)

```
<?php
$this->title = "Delete album";
$this->headTitle($this->title);
?>
<p>Are you sure that you want to delete '<?php echo $this-
>escape($this->books['Name'])?>' ?</p>
```

```
<form action="<?php echo $this->url(array('action' =>
'delete')); ?>" method="post">
  <div>
    <input type="hidden" name="id" value="<?php echo $this-
>books['ID'] ?>" />
    <input type="submit" name="del" value="Yes" />
    <input type="submit" name="del" value="No" />
  </div>
</form>
```

Liite 8. Drupal moduulin info -tiedosto

(sites/all/modules/oppari/oppari.info)

```
name = oppari
description = Tervehtii käyttäjää ystävällisesti.
core = 7.x
files[] = oppari.module
files[] = oppari.install
files[] = oppari.admin.inc
dependencies[] = field
package = "Antin oppari"
php = 5.2
```

Liite 9. Drupal moduulin install -tiedosto

(sites/all/modules/oppari/oppari.install)

```
<?php

/**
 * Install hook.
 * Creates oppari field.
 */
function oppari_install() {
  $field = field_info_field('oppari');
  if(empty($field)) {
    $field = array(
      'field_name' => 'oppari',
      'type' => 'text_with_summary',
      'entity_types' => array('node'),
      'translatable' => true,
    );
    $field = field_create_field($field);
  }
}

/**
 * Uninstall hook.
 * Removes oppari field-
```

```
 */
function oppari_uninstall() {
  watchdog('oppari module', 'Uninstalling module and deleting
oppari field.');
  foreach(node_type_get_types() as $type)
  {
    $instance = field_info_instance('node', 'oppari', $type-
>type);
    if($instance)
      field_delete_instance($instance);
  }

  if(field_info_field('oppari'))
    field_delete_field('oppari', false);
}
```

Liite 10. Drupal moduuli -tiedosto.

(sites/all/modules/oppari/oppari.module)

```
<?php
/**
 * @file
 * Oppari module
 */
/**
 * Implementation of hook_menu().
 */
function oppari_menu() {
  $items = array();
  $items['admin/config/oppari'] = array(
    'title' => 'Oppari example.',
    'description' => 'Change oppari options.',
    'position' => 'right',
    'weight' => -5,
    'page callback' => 'system_admin_menu_block_page',
    'access arguments' => array('registered'),
    'file' => 'system.admin.inc',
    'file path' => drupal_get_path('module', 'system'),
  );
  $items['admin/config/oppari/settings'] = array(
    'title' => 'Oppari example module settings.',
    'description' => 'Change how oppari example behaves.',
    'page callback' => 'drupal_get_form',
    'page arguments' => array('oppari_admin_settings'),
    'access arguments' => array('registered'),
    'type' => MENU_NORMAL_ITEM,
    'file' => 'oppari.admin.inc',
  );
  return $items;
}
```

```
function oppari_node_load($nodes, $types) {
  // Things to do whenever our module is loaded.
  // For example filtering content by user.
}
```

Liite 11. Drupal admin.inc -tiedosto.

(sites/all/modules/oppari/oppari.admin.inc)

```
<?php
/**
* @file
* Administration page callbacks for the oppari module.
*/
/**
* Form builder. Configure oppari module.
*/
function oppari_admin_settings() {
  foreach(node_type_get_types() as $nodeType)
      $options[$nodeType->type] = $nodeType->name;

  $form['oppari_node_types'] = array(
    '#type' => 'checkboxes',
    '#title' => t('Users may add oppari module to these content
types'),
    '#options' => $options,
    '#default_value' => variable_get('oppari_node_types',
array('story', 'book')),
    '#description' => t('You get new oppari field to node
types.'),
  );
  $form['#submit'][] = 'oppari_admin_settings_submit';

  return system_settings_form($form);
}

/**
* Process oppari settings submission.
*/
function oppari_admin_settings_submit($form, $form_state) {
  foreach ($form_state['values']['oppari_node_types'] as $key
=> $value) {
    if (!$value) {
      // Remove oppari field from node type
      $instance = field_info_instance('node', 'oppari', $key);
      if (!empty($instance)) {
        field_delete_instance($instance);
        watchdog("oppari", 'Deleted oppari field from content
type:
          %key', array('%key' => $key));
```

```
      }
    } else {
      // Add oppari field to node type
      $instance = field_info_instance('node', 'oppari', $key);
      if (empty($instance)) {
        $instance = array(
          'field_name' => 'oppari',
          'entity_type' => 'node',
          'bundle' => $key,
          'label' => t('oppari'),
          'widget_type' => 'text_textarea_with_summary',
          'settings' => array('display_summary' => TRUE),
          'display' => array(
            'default' => array(
              'type' => 'text_default',
            ),
            'teaser' => array(
              'type' => 'text_summary_or_trimmed',
            ),
        ),
      );
      $instance = field_create_instance($instance);
      watchdog('oppari', 'Added oppari field to content type:
%key',
        array('%key' => $key));
      }
    }
  }
}
```

Liite 12. Zend boostrap -tiedosto

(application/boostrap.php)

```
<?php
class Bootstrap extends Zend_Application_Bootstrap_Bootstrap
{
  protected function _initViewHelpers()
  {
    $this->bootstrap('layout');
    $layout = $this->getResource('layout');
    $view = $layout->getView();
    $view->headMeta()->appendHttpEquiv('Content-Type',
'text/html;charset=utf-8');
    $view->headTitle()->setSeparator(' - ');
    $view->headTitle('Oppari');
    $view->headLink()->setStylesheet('css/oppari.css');
  }
}
```