LAHDEN AMMATTIKORKEAKOULU
Lahti University of Applied Sciences

ZEND FRAMEWORK 2 -POHJAINEN PALVELU

LAHDEN
AMMATTIKORKEAKOULU
Tekniikan ala
Tietotekniikka
Ohjelmistotekniikka
Opinnäytetyö
Syksy 2013
Tuominen Joona

Lahden ammattikorkeakoulu
Tietotekniikka

TUOMINEN, JOONA: Zend Framework 2 -pohjainen palvelu

Ohjelmistotekniikan opinnäytetyö, 20 sivua

Syksy 2013

TIIVISTELMÄ

Työn aiheena on Zend Framework 2-pohjainen internetpalvelu. Sen tarkoituksena on kertoa Zend-pohjaisen palvelun toimintaperiaate yleisellä tasolla. Työn toinen tavoite oli rakentaa HankintaSampo-niminen palvelu, joka perustuu Zend Framework 2:een. Palvelu rakennettiin eKeiretsu Oy:lle.

Zend Framework 2:n komponenttirakenne on ainutlaatuinen; jokaisessa komponentissa on vähän riippuvuuksia muihin komponentteihin. Zend Framework pohjautuu tietokonesovelluksissa yleisesti käytössä olevaan MVC-malliin. Zend Frameworkissa on esimerkiksi suoraan model-, view- ja controller-nimiset kansiot ja toiminta on muutenkin hyvä esimerkki MVC-mallista.

Yleensä JavaScriptiä ja jQueryä käytetään tämäntyyppisissä palveluissa. JavaScript on pääasiassa web-ympäristössä käytettävä oliopohjainen komentosarjakieli. jQuery on kaikille selaimille tarkoitettu ilmainen avoimen lähdekoodin JavaScript-kirjasto. jQueryn syntaksi on tehty mahdollisimman helposti ymmärrettäväksi, mikä tekee kirjastosta erittäin suositun.

HankintaSampo on Zend Framework 2:n pohjalle rakennettu web-palvelu. JavaScriptiä ja jQueryä on käytetty hyvin paljon palvelun toteutuksessa. HankintaSampo on rakennettu tarjouspyyntöjen ja tarjousten välittämiseen ja kuntien sekä yritysten välisen viestinnän hoitamiseen.

Asiasanat: Zend, framework, internet, javascript, php, mvc, ajax

Lahti University of Applied Sciences
Degree Programme in information technology

TUOMINEN, JOONA: Service based on Zend Framework 2

Bachelor's Thesis in software engineering, 20 pages

Autumn 2013

ABSTRACT

The theme of the thesis was a web based application based on Zend Framework 2. The aim was to tell the working principles of Zend-based applications on a general level. Another objective was to build an application called HankintaSampo, which is based on Zend Framework 2. The application was built for eKeiretsu Oy.

The component structure of Zend Framework 2 is unique; every component has only some dependencies on other components. Zend Framework is based on MVC architecture, which is commonly in use in computer applications. For example, Zend Framework has view, model and controller folders.

Normally, JavaScript and jQuery are used in this kind of application. JavaScript is an object-oriented computer programming language used in the web-environment. jQuery is a multi-browser open source and free JavaScript library. The syntax of jQuery is very easy to understand, which makes it very popular.

HankintaSampo is a web-application based on Zend Framework 2. JavaScript and jQuery are used in the implementation of the application. HankintaSampo was made for transferring requests for tenders, and tenders. It was also made to handle communication between municipalities and companies.

Key words: Zend, framework, internet, javascript, php, mvc, ajax

SISÄLLYS

1 JOHDANTO

Internet on viime vuosikymmenien keksinnöistä mahdollisesti maailmaa eniten muuttanut asia. Tiedonvälitys, multimedian levitys ja kommunikointi ovat mullistuneet internetin käytön yleistymisen myötä. Myös erilaiset internetpalvelut ovat muuttaneet ihmisten käyttäytymistä ja prosesseja, esimerkkinä Facebook.

eKeiretsu Oy on keväällä 2013 perustettu yritys. Sen perusti 4 henkilöä, jotka olivat seuranneet julkisten hankintojen ja kilpailutusten prosessia. Heidän mielessään oli kypsynyt pikku hiljaa ajatus, että nämä hankinnat voisi järjestää helpomminkin. Niinpä tämä ryhmä alkoi kehitetää internetpohjaista palvelua tätä ajatusta varten. Aluksi demon keskenään, ja keväällä, kun projekti sai positiivista palautetta, he palkkasivat yritykseen ulkopuolisia työntekijöitä ja palvelua lähdettiin kehittämään tosissaan.

Ennen palvelun avaamista markkinoilla oli se tilanne, että kuntien tekemät tarjouspyynnöt julkaistiin hyvin erilaisia kanavia pitkin, mm. sähköpostilla. Oli olemassa myös joitain alkeellisia web-palveluita, joihin pystyi lataamaan tehdyt tarjouspyynnöt pdf-muodossa. Tämä oli kuitenkin jäykkä ja huono käytäntö. Kuntien ja yritysten väliltä puuttui todellinen vuorovaikutus. Lisäksi pienten hankintojenkin tekeminen oli hankalaa ja byrokraattista. Kuntien ja paikallisten yritysten edut eivät kohdanneet.

HankintaSampo-nimisen palvelun ajatus on tehdä koko tästä prosessista sujuvampaa, helpompaa, nopeampaa ja ketterämpää. Yksi palvelun ajatus on myös, että yritysten ja kuntien vuorovaikutus ja yhteistyö paranee. Sen seurauksena tarjouspyynnöistä saadaan paremmin osuvia ja lisäksi kilpailu lisääntyy, kun tarjouksia on helpompi lähettää.

Ideana oli tarjota palvelu aluksi pienhankintoihin Suomessa (arvo alle 30 000 €). Tämä siitä syystä, että laeissa on omat rajoituksensa tätä suuremmille hankinnoille. Haaveissa oli kuitenkin myös toimialueen laajeneminen tästä, mahdollisesti ensin Suomessa kaikkiin hankintoihin ja sitä kautta jopa mahdollinen koko EU:n laajuinen toiminta. Nämä olivat tietenkin varsinkin alkuvaiheessa hyvin kaukaisia haaveita.

Palvelun pohjaksi valittiin Zend Framework sen takia, että se oli tekijöille tuttu. Se on myös hyvin suosittu alusta tämäntyyppisille palveluille, ja sen ominaisuuksien arveltiin vastaavan palvelun teknisiä vaatimuksia. Ajatuksissa oli myös, että tällaiselle yleisesti käytössä olevalle alustalle on helpompi löytää osaajia eli työntekijöitä yritykselle.

Opinnäytetyön tutkimusongelma on Zend Framework 2. Tarkoitus on kertoa sen toimintaperiaate ja rakenne. Työssä toteutetaan web-palvelu sen pohjalta ja kerrotaan palvelun erikoispiirteistä.

2 ZEND FRAMEWORK 2

Zend Framework on avoimeen lähdekoodiin perustuva oliopohjainen web-sovelluskehys, joka käyttää PHP 5:tä ja on lisensoitu New BSD -lisenssillä. Zend Framework 2:n ensimmäinen vakaa versio julkaistiin 5. syyskuuta 2012. Zendin kehittämisestä vastaa Zend Technologies -yhtiö, jonka ovat perustaneet Andi Gutmans ja Zeev Surarski. Sama yhtiö on myös Zend Frameworkin suurin sponsori. Teknologiakumppaneihin kuuluvat mm. IBM, Google, Microsoft, Adobe Systems ja Strikelron. Zend Framework 2 vaatii vähintään PHP-version 5.3.3. Frameworkin 2 -versio tukee myös kaikkia yleisimpiä tietokantatyyppejä. (Wikipedia 2013f.)

Zend Framework oli vuonna 2012 Google Trendsin mukaan edelleen viiden suosituimman PHP-sovelluskehyksen joukossa. Sen suosio on kuitenkin hieman hiipunut esim. vuosien 2008 - 2009 aikaisista lukemista. (Webcodepro 2012.)

2.1 Toimintaperiaate

Zend Framework 2:n komponenttirakenne on ainutlaatuinen; komponentit ovat aidosti modulaarisia eli jokaisessa komponentissa on vähän riippuvuuksia muihin komponentteihin. Peruskirjaston komponentteja voi käyttää myös erikseen. Yhdessä ne muodostavat kuitenkin tehokkaan ja laajennettavan sovelluskehyksen web-sovellukselle. Zend Framework 2 tarjoaa myös suorituskykyisen MVC-toteutuksen, helppokäyttöisen tietokanta-abstraktion ja lomakekomponentit, jotka suorittavat HTML5-pohjaisen lomakkeen renderöinnin. Muut komponentit, kuten Zend\Authentication ja Zend\Permissions\Acl tarjoavat authentikoinnin ja auktorisoinnin kaikkiin yleisimpiin tapauksiin. (Zend Framework 2013.)

```
application_root/
    config/
        application.config.php
        autoload/
            global.php
            local.php
            // etc.
    data/
    module/
    vendor/
    public/
        .htaccess
        index.php
    init_autoloader.php
```

KUVIO 1. Zend Framework -ohjelman perusrakenne (Zend Framework 2 2013b)

Kuviossa 1 on Zend-ohjelman peruskansiorakenne. Käyttäjien saapumisen sivulle käsittelee public/index.php. Zend\ModuleManagerin käyttämä konfiguraatio sijaitsee config-kansiossa. Siellä määritellään mm ohjelman moduulit ja tietokanta-asetukset. (Zend Framework 2 2013b.)

```
1  <?php
2  return array(
3      'service_manager' => array(
4          'factories' => array(
5          'Zend\Db\Adapter\Adapter' => 'Zend\Db\Adapter\AdapterServiceFactory',
6          ),
7      ),
8      'db' => array(
9         'driver'    => 'pdo',
10         'dsn'       => 'mysql:dbname=reqengine;host=127.0.0.1:3306',
11         'username'  => 'xxxx',
12         'password'  => 'xxxx',
13         'driver_options' => array(
14             PDO::MYSQL_ATTR_INIT_COMMAND => 'SET NAMES UTF8;',
15             //PDO::MYSQL_ATTR_USE_BUFFERED_QUERY => true,
16         ),
17     ),
18 );
```

KUVIO 2. Tietokanta-asetukset config/autoload-hakemistossa

Kuviossa 2 on esimerkki config/autoload-hakemiston tietokanta-asetuksista. Tässä tapauksessa ne sijaitsevat database.local.php-tiedostossa. Koodissa määritellään tietokannan tyyppi, nimi, osoite, portti, käyttäjätunnus ja salasana. Rivillä 14 varmistetaan lisäksi, että käytössä on tekstin UTF-8-koodaus.

Vendor-hakemisto sisältää kaikki kolmansien osapuolten moduulit ja kirjastot, joita ohjelma vaatii. Se sisältää myös Zend Frameworkin. Vendor-hakemistossa

sijaitsevia kirjastoja tai moduuleita ei ole tarkoitettu muokattavaksi niiden alkuperäisestä tilasta. Module-kansio sisältää ohjelman moduulit, eli käytännössä ohjelman päätoiminnallisuuden. (Zend Framework 2 2013b.)

Moduulit ovat Zend Framework -pohjaisen palvelun peruskomponentteja. Yhdessä moduulissa on yleensä oma ohjain, malli ja näkymä-tiedostot. Moduuli vastaa usein yhtä tietokannan taulua, mutta epätavallista ei ole sekään, että yksi moduuli vastaa monen tietokantataulun käsittelystä. Tällöin jokaiselle taululle on luotu oma malli.

```
module_root<named-after-module-namespace>/
    Module.php
    autoload_classmap.php
    autoload_function.php
    autoload_register.php
    config/
        module.config.php
    public/
        images/
        css/
        js/
    src/
        <module_namespace>/
            <code files>
    test/
        phpunit.xml
        bootstrap.php
        <module_namespace>/
            <test code files>
    view/
        <dir-named-after-module-namespace>/
            <dir-named-after-a-controller>/
                <.phtml files>
```

KUVIO 3. Suositeltu moduulin rakenne (Zend Framework 2 2013b)

Autoload_*-tiedostot ovat suositeltavia, mutta eivät pakollisia. Ne tarjoavat järkevän oletusmekanismin moduulin sisältämien luokkien automaattiseen lataamiseen. Näin moduulia voidaan käyttää ilman tarvetta Zend\ModuleManagerille eli teoriassa jopa Zend Frameworkin ulkopuolella. (Zend Framework 2 2013b.)

Config-kansio sisältää kaikki moduuliin liittyvät konfiguraatiot. Siellä täytyy olla ainakin module.config.php-tiedosto, jossa on asetukset reitittimelle. (Zend Framework 2 2013b.)

```php
<?php
return array(
    'controllers' => array(
        'invokables' => array(
            'Rftender\Controller\Rftender' => 'Rftender\Controller\RftenderController',
        ),
    ),
    'router' => array(
        'routes' => array(
            'rftender' => array(
                'type'    => 'Segment',
                'options' => array(
                    // Change this to something specific to your module
                    'route'    => '/rftender[/:action][/:id][/:type]',
                    'constraints' => array(
                        'action'     => '[a-zA-Z][a-zA-Z0-9_-]*',
                        'id'     => '[a-zA-Z0-9_-][a-zA-Z0-9_-]*',
                        'type'     => '[a-zA-Z0-9_-][a-zA-Z0-9_-]*',
                    ),
                    'defaults' => array(
                        // Change this value to reflect the namespace in which
                        // the controllers for your module are found
                        '__NAMESPACE__' => 'Rftender\Controller',
                        'controller'    => 'Rftender',
                        'action'        => 'mainrftender',
                    ),
                ),
                'may_terminate' => true,
                'child_routes' => array(
                    // This route is a sane default when developing a module;
                    // as you solidify the routes for your module, however,
```

KUVIO 4. module.config.php:n sisältöä

Kuviossa 4 on config-kansion sisällä oleva module.config,php. Siinä määritellään kyseisen moduulin asetukset. Ensimmäisenä kerrotaan moduulin ohjaimen nimi. Riviltä 8 alkaen asetetaan moduulin reitittimen asetukset. Rivillä 14 kerrotaan tässä moduulissa käytössä olevien url-osoitteiden rakenne. Sen jälkeen kerrotaan, mitä merkkejä osoitteen eri osissa on mahdollista käyttää. Riviltä 20 alkaen määritellään oletusarvot, eli mikä ohjain ja toiminto suoritetaan, jos niitä ei erikseen määritellä. Tässä tapauksessa oletustoiminto on RftenderControllerin mainrftenderAction().

Src-kansiossa on moduulin lähdekoodi. Normaalisti siellä on ainakin kansio, joka on samanniminen kuin moduulin nimiavaruus. Siellä ovat controller- ja model-kansiot ja -tiedostot. (Zend Framework 2 2013b.)

Test-kansio sisältää yksikkötestit. Public-kansiota voidaan käyttää sisällölle, joka on näkyvissä myös ohjelman juurihakemistolle. Public-kansiossa voi olla esim. css- ja javascript-tiedostoja. View-kansio sisältää moduulin view-tiedostot. (Zend Framework 2013b.)

2.2 Asennus

Kehitysympäristöä ja ohjelmaa varten tarvitaan Apache web -palvelin ja siihen jokin tietokanta, tässä tapauksessa MySQL. Apache-asennuksessa pitää olla mod_rewrite-laajennus asennettuna. Täytyy myös varmistaa että Apache on määritelty tukemaan .htaccess-tiedostoja. Tämä tehdään muuttamalla httpd.conf-tiedostossa ”AllowOverride None” -arvo ”AllowOverride FileInfo”:ksi. Jos tätä ei ole asetettu, ohjelmassa ei ole mahdollista navigoida etusivulta mihinkään. (Zend Framework 2 2013a.)

Projektin hakemistorakenne kannattaa luoda Zend Studio -ohjelmalla. Voidaan myös käyttää esimerkiksi Zend Frameworkin sivulta löytyvää tutorial-ohjelmaa oman ohjelman pohjana. Kun tämä ohjelma on ladattu, pitää siihen asentaa Zend Framework 2. Tämä tehdään käyttämällä Composer-ohjelmaa, joka käsittelee projektin riippuvuudet:

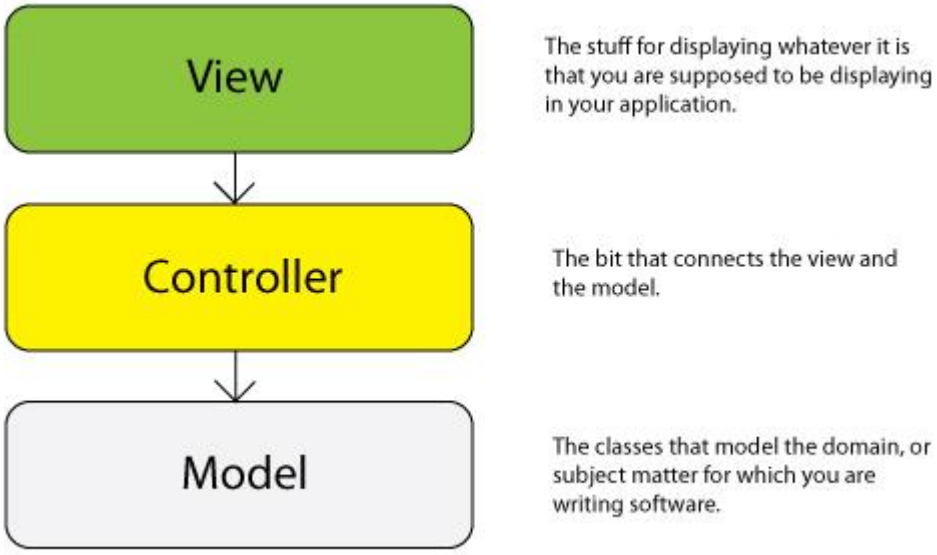
```
php composer.phar self-update
php composer.phar install
```

KUVIO 5. Zend Framework 2:n asennus projektiin (Zend Framework 2 2013a)

Kuviossa 3 asennetaan Zend Framework 2 Composerin avulla omaan projektiin. Komennot pitää käynnistää projektin juurikansiossa.

2.3 MVC-malli

Zend Framework pohjautuu tietokone- ja web-sovelluksissa yleisesti käytössä olevaan MVC-malliin. Zend Frameworkissa on suoraan model-, view- ja controller-nimiset kansiot ja sen toiminta toteuttaa muutenkin hyvin MVC-mallin periaatteita.



KUVIO 6. MVC-malli yksinkertaistettuna (How MVC Works 2009)

Kuviossa 6 on selitetty MVC-mallin toiminta yksinkertaistettuna. View on asia, jota käytetään kaiken sen näyttämiseen, jota ohjelmassa on tarkoitus näyttää. Controller on asia, joka yhdistää viewin ja modelin. Modelissa on tietokanta-asiat ja niihin liittyvät luokat.

MVC-malli tulee sanoista model, view, controller eli malli, näkymä, ohjain. Se on ohjelmistoarkkitehtuurityyli, jonka tarkoituksena on käyttöliittymän erottaminen sovellusaluetiedosta. Zend Framework noudattaa melko esimerkillisesti MVC-mallia. (Wikipedia 2013e.)

MVC-mallin etuja on mm. se, että malli voidaan suunnitella, ohjelmoida ja testata riippumatta järjestelmän muista osista. Samaan malliin voidaan lisäksi tehdä useita käyttöliittymiä. Näkymän ja ohjaimen riippuvuus toisistaan on verrattain pieni. (Wikipedia 2013e.)

Haittana on mm. se, että malli ei ohjaa kehittäjiä optimoimaan tietokantahakuja käyttöliittymän näkökulmasta. Tämän vuoksi tietokannasta haetaan helposti tarpeetonta tietoa ja tiedon hakuun käytetään tarpeeseen optimoimattomia SQL-lauseita. Tämä aiheuttaa helposti suorituskykyongelmia laajoissa sovelluksissa. Suorituskykyongelmat voidaan kustannustehokkaasti välttää kiinnittämällä huomiota tähän ongelmaan sovelluskehityksen alusta asti. (Wikipedia 2013e.)

2.3.1 Model

Model kuvaa järjestelmän tiedon tallentamisen, ylläpidon ja käsittelyn (tietokantakuvaus). Käytännössä mallin pitäisi peittää ns. tietovarastojen toiminta sekä ohjaimelta että näkymältä, eli malli on ainoa osa ohjelmistoa, joka on yhteydessä tietovarastoon. Käytännössä tämä tarkoittaa sitä, että sitä kautta suoritetaan mm. tietokantojen ja istuntojen käsittely, eli esimerkiksi näkymä ei ota suoraan yhteyttä tietokantaan vaan huolehtii asiasta mallin kautta. (Vahtolammi 2010.)

2.3.2 Näkymä (View)

Näkymä huolehtii tietojen esittämisestä. Ratkaisusta riippuen näkymä tai ohjain pyytää tiedot mallilta näkymän käytettäväksi. Web-maailmassa tieto useimmiten esitetään HTML-koodattuna, mutta näkymä voi tarvittaessa palauttaa tiedon muussakin muodossa. (Vahtolammi 2010.)

2.3.3 Ohjain (Controller)

Ohjaimen tehtävä on ottaa vastaan sovelluksen pyynnöt, eli käytännössä web-sovelluksessa se lukee tiedot http-viestistä. Sovelluksesta riippuen ohjainta voidaan ohjata erilaisilla tavoilla. (Vahtolammi 2010.)

2.4 Lomakkeet (Formit)

Lomakkeita käytetään yleisesti web-sivuilla tekstikenttien, valintanappien ja muiden kenttien kokoelmana. Näihin kenttiin käyttäjä voi syöttää tietoja, jotka sitten lähetetään palvelimelle. Zend Frameworkissa lomakkeita käytetään siltana mallin ja näkymä-kerroksen välillä. Zend_Form yksinkertaistaa lomakkeiden luonnin web-palveluun. Siihen sisältyy elementteihin syötetyn tiedon suodatus ja muodon tarkistus, elementtien järjestely sekä elementtien ja lomakkeiden ryhmittely.

```php
<? $form = new Zend_Form;
$form->setAttrib('id', 'login');
$form->addElement('text', 'username');

//Elementistä saa helposti pakollisen kentän:

$username->setRequired(true);

//Suodattimen käyttö, muuttaa syötetyt kirjaimet pieniksi:

$username->addFilter('StringtoLower');
?>
```

KUVIO 7. Formin luonti (Zend Framework 2 2013e)

Kuviossa 7 on esimerkki Zend_Formin käytöstä: ensin luodaan lomake, sitten asetetaan sille id ja lopuksi lisätään yksi text-tyyppinen kenttä lomakkeeseen.

```php
$form->addElement('text', 'username', array(
    'validators' => array(
        'alnum',
        array('regex', false, '/^[a-z]/i')
    ),
    'required' => true,
    'filters'  => array('StringToLower'),
));
```

KUVIO 8. Elementin luonti Formiin (Zend Framework 2 2013e)

Kuviossa 8 nämä on määritelty lyhyemmin jo heti elementin luonnin yhteydessä. Tässä lisätään elementtiin myös validaattorit. (Zend Framework 2 2013e.)

2.5 Phtml-tiedostot

Zend Framework 2:n view-kansiossa olevat tiedostot ovat tavallisesti .phtml-päätteisiä. Phtml-päätettä käytetään silloin, kun tiedostossa on sekä PHP-koodia että HTML:ää. Näin palvelimen on mahdollista luoda dynaamista HTML:ää. Käytännössä phtml-päätteisessä tiedostossa oleva sisältö voidaan esittää myös .php-päätteisessä tiedostossa. (File.org 2013.)

2.6 Tunnistautuminen ja todennus

Zend\Authentication komponentti tarjoaa rajapinnan tunnistautumiseen ja sisältää tunnistautumisadaptereita yleisimpiin käyttötarkoituksiin. Jokainen tällainen adapteri implementoi Zend\Authentication\Adapter\AdapterInterface:n Tämä komponentti sisältää vain tunnistautumisen, ei auktorisointia.

```php
use Zend\Authentication\Adapter\AdapterInterface;

class My\Auth\Adapter implements AdapterInterface
{
    /**
     * Sets username and password for authentication
     *
     * @return void
     */
    public function __construct($username, $password)
    {
        // ...
    }

    /**
     * Performs an authentication attempt
     *
     * @return \Zend\Authentication\Result
     * @throws \Zend\Authentication\Adapter\Exception\ExceptionInterface
     *               If authentication cannot be performed
     */
    public function authenticate()
    {
        // ...
    }
}
```

KUVIO 9. Esimerkki tunnistautumisadapterista (Zend Framework 2 2013d)

Kuviossa 9 on esimerkki tunnistautumisadapterista. Tämä adapteri vaatii käyttäjätunnuksen ja salasanan. Adapterit palauttavat Zend\Authentication\Result:n esittääkseen tunnistautumisen tulokset.

```php
use Zend\Authentication\AuthenticationService;

// instantiate the authentication service
$auth = new AuthenticationService();

// Set up the authentication adapter
$authAdapter = new My\Auth\Adapter($username, $password);

// Attempt authentication, saving the result
$result = $auth->authenticate($authAdapter);

if (!$result->isValid()) {
    // Authentication failed; print the reasons why
    foreach ($result->getMessages() as $message) {
        echo "$message\n";
    }
} else {
    // Authentication succeeded; the identity ($username) is stored
    // in the session
    // $result->getIdentity() === $auth->getIdentity()
    // $result->getIdentity() === $username
}
```

KUVIO 10. Esimerkki tunnnistautumisen käytöstä palvelussa (Zend Framework 2 2013d)

Kuviossa 10 on esimerkki tunnistautumisen käytöstä. Ensin asetetaan tunnistautumisadapteri rivillä 7. Rivillä 10 tallennetaan tunnistautumisen tulos $result-muuttujaan. Loppuosassa tarkistetaan, onnistuiko tunnistautuminen, ja ilmoitetaan syy, jos se ei onnistunut.

2.7 Auktorisointi

Zend\Permissions\Acl-komponentti tarjoaa kevyen ja joustavan ACL (access control list) -toteutuksen oikeuksien hallintaan. Tätä komponenttia käytettäessä määritellään resurssi, joka on objekti, johon liittyviä oikeuksia tarkistetaan. Lisäksi määritellään rooli, joka on objekti, joka pyytää oikeuksia tiettyyn resurssiin. Rooli voi periytyä yhdestä tai useammasta muusta roolista.

```
1  use Zend\Permissions\Acl\Acl;
2  use Zend\Permissions\Acl\Role\GenericRole as Role;
3  use Zend\Permissions\Acl\Resource\GenericResource as Resource;
4
5  $acl = new Acl();
6
7  $acl->addRole(new Role('guest'))
8      ->addRole(new Role('member'))
9      ->addRole(new Role('admin'));
10
11 $parents = array('guest', 'member', 'admin');
12 $acl->addRole(new Role('someUser'), $parents);
13
14 $acl->addResource(new Resource('someResource'));
15
16 $acl->deny('guest', 'someResource');
17 $acl->allow('member', 'someResource');
18
19 echo $acl->isAllowed('someUser', 'someResource') ? 'allowed' : 'denied';
```

KUVIO 11. Esimerkki Zend\Permission\Acl:n käytöstä (Zend Framework 2 2013c)

Kuviossa 11 luodaan ensin kolme perusroolia: guest, member ja admin. Näistä kolmesta perusroolista muut roolit voivat periytyä. Järjestys, jossa roolit on määritetty $parents-arrayhin, on tärkeä, koska se määrittää niiden periytymisjärjestyksen. Rivillä 19 kysytään, onko someUserilla oikeutta someResourceen. SomeUserille ei ole määritetty erikseen, onko hänellä oikeutta kyseiseen resourceen, joten on tarkistettava, mitä oikeuksia hänelle periytyy. Ensin katsotaan, onko adminille määritetty oikeuksia. Sille ei ole, joten seuraavaksi tarkistetaan memberin oikeudet. Memberille on myönnetty oikeus someResourceen, joten myös someUserilla on oikeus tähän resourceen. (Zend Framework 2 2013c.)

2.8 Kehitystyökalut

Luonnollisin tapa kehittää ohjelmia Zend Frameworkin pohjalle on käyttää Zend Studio- ja Zend Server-ohjelmia. Muitakin vaihtoehtoja on, mutta nämä ohjelmat on suunniteltu Zend Technologiesin toimesta juuri tähän tarkoitukseen.

2.8.1 Zend Studio

Zend Studio on kaupallinen, PHP-ohjelmointia varten kehitetty ohjelmointiympäristö. Se perustuu PDT-lisäosaan Eclipselle. Sitä kehittää ja ylläpitää Zend Technologies. Ensimmäinen julkinen versio ohjelmasta julkaistiin 20. helmikuuta 2002. Kritiikkiä ohjelma on saanut lähinnä hitaudestaan. Zend Studio on vahvasti liitetty Zend Frameworkiin. Se esimerkiksi tarjoaa MVC-näkymän helpottamaan koodissa navigointia. Se sisältää myös Zend_Toolin automaattiseen koodin luontiin. (Wikipedia 2013h.)

2.8.2 Zend Server

Zend Server on Zend Technologiesin kehittämä palvelinohjelmisto. Sen ensimmäinen versio julkaistiin 2009. Se on saatavilla kahtena eri versiona: Zend Server ja hieman karsitumpi Zend Server Community Edition. Zend Server tarjoaa Community Editionin lisäksi mm. ohjelmien monitoroinnin. Molemmat versiot tukevat Windowsia, Linuxia ja MacOS:ää. (Wikipedia 2013g.)

2.9 Zend Framework ja jQuery

Zend Framework sisältää jQuery view -avustajan (Helper). Se yksinkertaistaa jQuery-ympäristön asennusta ohjelmaan. Se huolehtii tarvittaessa jQuery-ytimen ja UI-kirjaston riippuvuuksien lataamisesta. (Zend Framework 2 2013f.)

```
1. $view->addHelperPath('ZendX/JQuery/View/Helper/', 'ZendX_JQuery_View_Helper');
```

KUVIO 13. Helperin lisäys (Zend Framework 2 2013f)

```
1. <?php echo $this->ajaxLink("Show me something",
2.                     "/hello/world",
3.                     array('update' => '#content'));?>
4.
5. <div id="content"></div>
6.
7. <form method="post" action="/hello/world">
8. Pick your Date: <?php echo $this->datePicker("dp1",
9.                                              '',
10.                                             array(
11.                                                'defaultDate' =>
12.                                                    date('Y/m/d', time())));?>
13. <input type="submit" value="Submit" />
14. </form>
```

KUVIO 14. jQuery datepickerin ja ajax-linkin asetus (Zend Framework 2 2013f)

Kuviossa 13 lisätään tiedostoon helper. Tässä lisäyksessä määritellään myös sen polku. Kuvioissa 13 ja 14 on perusesimerkki ZendX_Jquery:n käytöstä. Siinä tulostetaan ajax-linkki ja jQuery datepicker. Kuviossa 15 vielä tulostetaan jQuery-ympäristö.

```
1. <html>
2.     <head>
3.         <title>A jQuery View Helper Example</title>
4.         <?php echo $this->jQuery(); ?>
5.     </head>
6.
7.     <body>
8.         <?php echo $this->layout()->content; ?>
9.     </body>
10. </html>
```

KUVIO 15. jQuery-ympäristön tulostus (Zend Framework 2 2013f)

3 JAVASCRIPT

JavaScript on alun perin Netscape Communications Corporationin kehittämä pääasiassa web-ympäristössä käytettävä oliopohjainen komentosarjakieli. JavaScriptin tärkein sovellus on mahdollisuus lisätä web-sivuille dynaamista toiminnallisuutta. JavaScriptiä ei tule sekoittaa Javaan. Javascriptin syntaksi perustuu löyhästi C-ohjelmointikieleen. (Wikipedia 2013c.)

JavaScriptin oliomallin perusyksikköjä ovat object- ja function-oliot. Oliomallin ja sen kapselointi ja periytyvyys pohjautuu prototyyppeihin, ei luokkiin, kuten useimmissa olio-ohjelmointikielissä (C++, Java, Ruby). (Wikipedia 2013c.)

```
<html>
 <head>
  <title>JavaScript-esimerkki 2</title>
 </head>
 <body>
  <script>
   function SayHello()
    {
     alert("Hei!")
     setTimeout("SayHello()", 10000);
    }
  </script>
  <input type="button" value="Paina" onclick="SayHello()"/>
 </body>
</html>
```

KUVIO 16. Itseään toistava esimerkki upotettuna HTML-dokumenttiin (Wikipedia 2013c)

Kuviossa 16 normaaliin, painikkeen sisältävään HTML-dokumenttiin on upotettu lyhyt JavaScript-funktio. Kun painiketta painetaan, funktion suoritus käynnistyy. Ponnahdusikkuna, joka sisältää tekstin ”Hei!”, luodaan 10 sekunnin välein näytölle. Javascriptiä voidaan upottaa samaan tiedostoon HTML:n kanssa tai luoda erillisiä .js-päätteisiä tiedostoja, jotka sisältävät JavaScript-koodin.

3.1 jQuery

jQuery on kaikille selaimille tarkoitettu ilmainen avoimen lähdekoodin JavaScript-kirjasto. jQueryn syntaksi on tehty mahdollisimman helposti

ymmärrettäväksi, mikä tekee kirjastosta erittäin suositun. jQuerya voi käyttää toimintojen käsittelyyn, animaatioiden tekemiseen, DOM-elementtien valitsemiseseen ja Ajax-sovelluksien toteutukseen. jQuery julkaistiin vuonna 2006, ja se on nykyään maailman suosituin JavaScript-kirjasto. jQuery on erillinen JavaScript-tiedosto sisältäen kaiken tarvittavan. Se voidaan sisällyttää www-sivuun linkittämällä paikalliseen kopioon tai johonkin julkisten palvelimien, kuten Googlen tai Microsoftin, kopioihin. (Wikipedia 2013c.)

```
$(document).ready(function () {
    $('#textarea_1').click(function() {
       $(this).css({"height":"200px"});
    });
});
```

KUVIO 17. Esimerkki jQuery-koodista (Wikipedia 2013c)

Kuviossa 17 on esimerkki jQuery-koodista. Siinä muutetaan elementin, jonka id on textarea_1, korkeuden 200 pikseliin sitä painettaessa.

3.2 Ajax

Ajax (Asynchronous Javascript And XML) on joukko web-sovelluskehityksen tekniikoita, joiden avulla web-sovelluksista voi tehdä vuorovaikutteisempia. Alkuperäisessä merkityksessään AJAX:lla on alun perin viitattu tekniikkaan, jossa verkkosivulla JavaScript:llä asynkronisesti tehtävistä HTTP-pyynnöistä palautetaan XML-merkkausta, mutta nykyään käytetään laajasti yksinkertaisempaa JSON-merkkausta. Ajaxissa selainohjelma vaihtaa pieniä määriä dataa palvelimen kanssa taustalla niin, ettei koko verkkosivua tarvitse ladata uudelleen joka kerta käyttäjän tehdessä muutoksen. Tekniikan päämääränä on siis lisätä verkkopalvelun vuorovaikutteisuutta, nopeutta ja käytettävyyttä. (Wikipedia 2013a.)

```
$.ajax({
  type: "POST",
  url: "some.php",
  data: { name: "John", location: "Boston" }
})
  .done(function( msg ) {
    alert( "Data Saved: " + msg );
  });
```

KUVIO 18. Esimerkki Ajaxin käytöstä (jQuery API 2013)

Kuviossa 18 lähetetään jQueryn ajax-metodilla tietoa (name: ”John”, location: ”Boston”) tiettyyn paikkaan (some.php). Lähetystavaksi on valittu post. Kun vastaus on tullut, tehdään alert-ponnahdusikkuna, jossa ilmoitetaan tiedon tallennuksesta.

Ajaxin haittapuolena ennen HTML5:tä oli se, että ajax-kutsut eivät automaattisesti rekisteröi itseään selaimen historiamoottoriin, joten selaimen Takaisin-painikkeen painaminen palauttaa selaimen takaisin edelliselle sivulle, ei tilanteeseen ennen ajax-kutsua. HTML5 tarjoaa kuitenkin hyvät työkalut työskentelyyn selaimen historiamoottorin kanssa. Myös kirjanmerkin luominen sivun tiettyyn tilaan liittyy tähän ja siihen voi käyttää samaa ratkaisua. (Wikipedia 2013b.)

Myös hakukoneiden indexointibotit voivat olla vaikeuksissa ajax-pohjaisien sivujen kanssa, koska botit eivät yleensä suorita javascript-koodia. Tällöin ne eivät myöskään pääse käsiksi ajaxilla ladattuun sisältöön. Tämä pitää ottaa huomioon ajax-pohjaista sivustoa luotaessa, jos halutaan hakukoneiden indexoivan sivuston kunnolla. (Wikipedia 2013b.)

4 YHTEENVETO

Ennen palvelun avaamista markkinoilla oli se tilanne, että kuntien tekemät tarjouspyynnöt julkaistiin hyvin erilaisia kanavia pitkin, mm. sähköpostilla. Oli olemassa myös joitain alkeellisia web-palveuita, joihin pystyi lataamaan tehdyt tarjouspyynnöt pdf-muodossa.

HankintaSampo-palvelun myötä tähän saatiin muutosta. Tarjouspyyntöjen tekemisestä ja jättämisestä tuli huomattavasti ketterämpää ja nopeampaa. Monet hankinnoista vastaavat ihmiset ottivat palvelun ilolla vastaan.

Ohjelman tekemisessä valittu alusta osoittautui hyväksi. Tekijöiden tiedot kyseisestä alustasta kasvoivat projektin edetessä. Kaikkia Zend Framework 2:n tarjoamia mahdollisuuksia ei hyödynnetty, vaan osa toiminnoita tehtiin tee-se-itse-periaatteella. Kuitenkin ne Zend Frameworkin komponentit, joita käytettiin, helpottivat ja yksinkertaistivat työtä huomattavasti.

Tällä hetkellä palvelun suhteen ollaan siinä tilanteessa, että se on ollut julkisesti avoinna noin viikon. Palvelua ei kuitenkaan ole vielä suuremmin mainostettu, vaan sen käyttöä on vielä testattu pienellä joukolla oikeita käyttäjiä. Laajempi käyttö on tarkoitus aloittaa 2 viikon kuluttua. Käyttäjät ovat olleet pääasiassa innostuneita uudesta palvelusta. Se vaatii kuitenkin totuttelua ja pientä opettelua vanhaan järjestelmään tottuneilta, koska enää tarjouspyyntöjä ei lähetetä pdf-muodossa sähköpostin välityksellä vastaanottajille.

Palvelun kehitys jatkuu koko ajan virheiden korjauksilla. Palvelua päivitetään tällä hetkellä parin päivän välein virhekorjauksilla ja joillain uusilla ominaisuuksilla.

LÄHTEET

File.org. 2013. Opening PHTML files [viitattu 9.11.2013.] Saatavissa: http://file.org/extension/phtml

How MVC Works. 2009. What is the model-view-controller pattern? [viitattu 25.10.2013]. Saatavissa: http://www.howmvcworks.net/GettingStarted/WhatIsTheModelViewControllerPattern

jQuery API 2013. jQuery.ajax() [viitattu 11.10.2013]. Saatavissa: http://api.jquery.com/jQuery.ajax/

Model View Controller [viitattu 18.10.2013]. Saatavissa: http://c2.com/cgi/wiki?ModelViewController

Vahtolammi K. 2010. MVC- Malli, peruskauraa frameworkkien käyttäjille [viitattu 11.10.2013]. Saatavissa: http://vahtolam.wordpress.com/2010/09/23/mvc-malli-peruskauraaframeworkkien-kayttajille/

Webcodepro. 2012. TOP 5 most popular PHP frameworks of 2012 [viitattu 11.10.2013]. Saatavissa: http://webcoderpro.com/blog/top-5-most-popular-php-frameworks-of-2012/

Wikipedia. 2013a. Ajax (ohjelmointi) [viitattu 11.10.2013]. Saatavissa: http://fi.wikipedia.org/wiki/Ajax_%28ohjelmointi%29

Wikipedia. 2013b. Ajax (programming) [viitattu 26.10.2013]. Saatavissa: http://en.wikipedia.org/wiki/Ajax_%28programming%29

Wikipedia. 2013c. JavaScript [viitattu 6.11.2013]. Saatavissa: http://fi.wikipedia.org/wiki/JavaScript

Wikipdedia. 2013d. jQuery [viitattu 11.10.2013]. Saatavissa: http://fi.wikipedia.org/wiki/JQuery

Wikipedia. 2013e. MVC-arkkitehtuuri [viitattu 11.10.2013]. Saatavissa:

http://fi.wikipedia.org/wiki/MVC-arkkitehtuuri

Wikipedia. 2013f. Zend Framework [viitattu 11.10.2013]. Saatavissa:

http://en.wikipedia.org/wiki/Zend_Framework

Wikipedia. 2013g. Zend Server [viitattu 11.10.2013]. Saatavissa: http://en.wikipedia.org/wiki/Zend_Server

Wikipedia. 2013h. Zend Studio [viitattu 11.10.2013]. Saatavissa: http://en.wikipedia.org/wiki/Zend_Studio

Zend Framework. About [viitattu 24.10.2013]. Saatavissa: http://framework.zend.com/about/

Zend Framework 2. 2013a. Getting started with Zend Framework 2 [viitattu 6.11.2013]. Saatavissa: http://framework.zend.com/manual/2.2/en/user-guide/overview.html

Zend Framework 2. 2013b. Introduction to the MVC Layer [viitattu 6.11.2013]. Saatavissa: http://framework.zend.com/manual/2.1/en/modules/zend.mvc.intro.html

Zend Framework 2. 2013c. Introduction to Zend\Permissions\Acl [viitattu 25.10.2013]. Saatavissa: http://framework.zend.com/manual/2.2/en/modules/zend.permissions.acl.intro.html

Zend Framework 2. 2013d. Zend\Authentication [viitattu 24.10.2013]. Saatavissa: http://framework.zend.com/manual/2.0/en/modules/zend.authentication.intro.html

Zend Framework 2. 2013e. Zend_Form Quick Start [viitattu 18.10.2013]. Saatavissa: http://framework.zend.com/manual/1.12/en/zend.form.quickstart.html

Zend Framework 2. 2013f. ZendX_JQuery View Helpers [viitattu 24.10.2013]. Saatavissa: http://framework.zend.com/manual/1.12/en/zendx.jquery.view.html