JOOMLA! – KOMPONENTIN KEHITYS

Mira Vartiainen

Opinnäytetyö
Huhtikuu 2011
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka
Tampereen ammattikorkeakoulu

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietotekniikka
Ohjelmistotekniikka

VARTIAINEN, MIRA: Joomla!-komponentin kehitys

Opinnäytetyö 30 s.
Huhtikuu 2011

Tässä opinnäytetyössä perehdytään Joomla!-komponenttien kehitykseen. Tarkoituksena oli ymmärtää komponenttien rakenne sekä eri osien merkitykset ja tässä ohella kehittää yksinkertainen kyselykomponentti. Kyselykomponentilla voi luoda leikkimielisiä kyselyjä, jotka koostuvat useasta kysymyksestä, useasta vastausvaihtoehdosta sekä käyttäjälle annettavasta palautteesta oikeisiin ja vääriin vastauksiin.

Lisäksi työn aikana oli tarkoitus ymmärtää kunnolla MVC-arkkitehtuurin (malli-näkymä-ohjain) hyödyt ohjelmoinnissa. Joomla!:n omien luokkien oppiminen oli myös olennaista työn onnistumisen kannalta.

Opinnäytetyössä käydään vaihe vaiheelta läpi komponentin kehitys. Opinnäytetyö jakaantui luonnollisesti komponentin eri osiin eli komponentin etuosaan (eng. front end) sekä komponentin hallintaosaan (eng. back end). Lisäksi vaiheet eroteltiin MVC-arkkitehtuurin mukaisiin osiin, malliin, näkymään sekä ohjaimeen, jotta komponentin kehittäjä voi keskittyä yhteen tiedostoon aina kerrallaan. Työvaiheet on pyritty selittämään niin selkeästi, että jokainen voi hyödyntää tietoa omaa komponenttiaan kehittäessään. Jotkut vaiheet ovat lähes identtisiä kaikilla komponenteilla, joten ainakin tällaiset vaiheet käyvät muille komponentin kehittäjille selviksi.

Asiasanat: Joomla! komponentin kehitys, Joomla!, MVC-arkkitehtuuri

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Computer technology training program
Software orientation

VARTIAINEN, MIRA: Joomla! Component Development

Bachelor's thesis 30 pages
April 2011

This thesis explores component development for Joomla!. The purpose was to understand the structure of components and also create simple inquiry component. This inquiry component is meant for creating playful inquiries which include several questions, several options for answers and also feedbacks for wrong and correct answers.

As well the purpose was to understand the good qualities of MVC (model-view-controller) architecture in programming. Joomla! has own classes and it was also relevant to learn how to use those.

This thesis leads reader through component development step by step. It is divided to component's front end and back end. Also MVC architecture divides this thesis to model, view and controller so programmer can easily create one file at a time. The purpose was that stages are described so well that any other developer can benefit given information to his/her own work.

Keywords: Joomla! component development, Joomla!, MVC architecture

SISÄLLYS

LYHENTEET JA TERMISTÖ

Avoin lähdekoodi	(eng. Open source). Ohjelman lähdekoodi on kaikkien saatavilla ja kaikki voivat sitä muokata tarpeidensa mukaisesti. Myös levittäminen ja kopiointi ovat sallittuja.
Apache	HTTP-palvelin, joka perustuu avoimeen lähdekoodiin ja se on ollut vuodesta 1996 saakka internetin suosituin palvelin. (Apache HTTP Server Project).
CSS	Cascading Style Sheets on tyyliohjeiden laji, jota käytetään erityisesti WWW-sivujen kehityksessä. Sitä käytetään yhdessä (X)HTML:n ja XML:n kanssa. CSS antaa sääntöjä siitä, miten dokumentti esitetään, mutta säännöt eivät ole ehdottomia vaan ne voidaan kiertää.
HTML	Hypertext Markup Language on kuvauskieli, jota käytetään WWW-sivujen rakentamiseen.
JavaScript	Komentosarjakieli, jonka on alun perin kehittänyt Netscape Communications Corporation. Sitä käytetään esimerkiksi reagoimaan tapahtumiin sivulla tai validoimaan annettuja tietoja. (W3Schools.com)
Komponentin hallintaosa	Komponentin tausta (eng. backend), jonka avulla sivujen järjestelmänhallinnoija voi komponentista luoda instansseja sivuilleen.
Komponentin etuosa	Komponentin rakenne jakaantuu kahteen puoliskoon, etuosaan ja järjestelmänhallinnan puoleen (eng. frontend / backend). Etuosalla tarkoitetaan sitä, mikä näkyy sivustolla käyttäjälle eli lopputulosta.

MVC-arkkitehtuuri	Malli-näkymä-ohjain –arkkitehtuuri (eng. model-view-controller) on ohjelmistoarkkitehtuurityyli. Tarkoituksena on erottaa ohjelman käyttöliittymä sovellusaluetiedostoista (Tietoportti.com).
MySQL	Maailman suosituin avoimen lähdekoodin tietokantajärjestelmä, jonka hallinnointi tapahtuu komentoriviltä tai asiakasohjelmalla. Sille on saatavilla myös graafisia hallintatyökaluja, kuten MySQL Administrator tai phpMyAdmin. (Oracle MySQL)
PHP	PHP: Hypertext Preprocessor on avoimen lähdekoodin palvelinpuolen komentosarjakieli, joka soveltuu erityisen hyvin web-ohjelmointiin. PHP:tä käytetään yleisimmin upotettuna HTML-sivujen sisälle. (The PHP Group)
XML	eXtensible Markup Language on merkintäkieli. Sen tarkoitus on siirtää ja varastoida tietoa, kun HTML puolestaan esittää tietoa.

1 JOHDANTO

Työ on tehty omasta mielenkiinnosta Joomla!-komponenttien kehitystä kohtaan sekä eräältä hyväntekeväisyysyhdistykseltä tulleesta ideasta. He tarvitsivat sivuilleen komponentin, jolla voidaan tehdä leikkimielisiä kyselyjä, joissa on useampi kuin yksi kysymys sekä antaa vastauksiin palautetta.

Opinnäytetyössä perehdytään Joomla!-komponenttien kehitykseen vaihe vaiheelta sekä pyritään kirjoittamaan vaiheista niin selkeästi, että periaatteessa kuka tahansa ohjelmointia jonkin verran osaava voi työn perusteella päästä alkuun ja kehittää Joomla!-komponentteja. Työssä kehitetty komponentti on Joomla! 1.5 julkaisujärjestelmälle. Työn päätavoitteena on saada valmiiksi komponentti, jolla voidaan luoda erilaisia kyselyjä ja testejä Joomla!-sivustoille. Komponentin kehitystyön aikana kirjataan ylös eri työvaiheita, jotta perusidea kehityksestä välittyy muille kehityksestä kiinnostuneille. Jokainen komponentti on pääpiirteiltään samankaltainen, joten samankaltaisuudet ymmärtämällä voi helposti kehittää uusia komponentteja.

Työ rajataan niin, että työssä ei varsinaisesti puututa Joomla!:n asentamiseen, käyttöönottoon eikä sivujen kehitykseen sen avulla. Komponenttien osalta ei puututa kielipaketteihin tai komponenttien käyttöön muuten, kuin yleisellä tasolla.

2 JOOMLA!

2.1 Joomla!:n tarkoitus

Joomla! on avoimen lähdekoodin (eng. open source) julkaisujärjestelmä. Julkaisujärjestelmä on sisällönhallintajärjestelmä, mutta se keskittyy nimenomaan julkaistavan materiaalin, eli yleiseen levitykseen tulevan materiaalin, hallintaan. Tällä hetkellä Joomla!:sta ovat käytössä versiot 1.5 ja 1.6.

Ideana Joomla!:ssa on, että kuka tahansa voi lisätä sisältöä tai päivittää olemassa olevaa sisältöä internet-sivuille, vaikka ei omaisikaan mitään ohjelmointitaitoja. Kaikki toiminnot, artikkelien lisäämisestä julkaisuun sekä esimerkiksi tapahtumien lisäämiseen, tapahtuvat internet-selaimen välityksellä. Tekstinkäsittelytaidoilla pääsee jo pitkälle, mutta sivupohjien, moduulien ja komponenttien muokkaus ja kehitys vaativat ymmärrystä ohjelmoinnista tai kokeneemman Joomla!-käyttäjän apua.

Kyseinen julkaisujärjestelmä soveltuu niin yritysten kuin yksityistenkin käyttöön ja ohjelmiston käyttö ja muokkaus ovat kaikille ilmaista. Erilaisia sivupohjia ja komponentteja löytyy valtavia määriä ilmaiseksi tai pientä maksua vastaan. Vaikka suurin osa laajennuksista onkin englanninkielellä, moniin on jo valmiit ja ilmaiseksi ladattavat kielitiedostot suomenkielelle. Ellei mieleisiä lisäosia löydy, ohjelmointitaitoisen on melko helppoa tarvitsemiaan lisäosia luoda.

2.2 Joomla!:n taustaa

”Joomla pohjautuu Mambo-ohjelmaan, ja kehitystyöstä vastaa sama tiimi, joka vastasi Mambon kehityksestä aina elokuuhun 2005 saakka.” (Wikipedia). Tuolloin kaupallistamisaikeiden seurauksena kehittäjät erosivat Mambo-projektista ja Joomla!-projekti syntyi.

Joomla! on kirjoitettu PHP:llä ja se käyttää olio-ohjelmoinnin tekniikoita. Kaikki tallennetaan MySQL tietokantaan eikä erillisiin tiedostoihin. Esimerkiksi komponentit luovat

omia taulujaan tietokantaan ja niihin tallennetaan komponentin toiminnassa vaikuttavat tiedot. Käytön aikana Joomla! hakee tarvittavat tiedot tietokannasta.

2.3 Joomla!:n käyttöönotto

Jotta Joomla! toimii, se vaatii PHP- ja MySQL-tuen toimiakseen. ”Joomla!n palvelimelle asettamat minimivaatimukset ovat:

- PHP 4.2.x tai uudempi
- MySQL 3.23.x tai uudempi
- Apache 1.13.19 tai uudempi (myös IIS käy).” (Joomlaportal.fi).

MySQL tuen lisäksi PHP:ssä on oltava tuki XML:lle ja Zlib:lle.

Alkuun päästäkseen voi etsiä internetistä jonkin vaihe vaiheelta etenevän ohjeen, jossa on suorat linkit myös vaadittaviin latauksiin ja asennuksiin. Yksi hyvä ja selkeä ohje Joomla!:n käytön aloittamiseen löytyy Joomla!:n virallisilta sivuilta (Official Joomla! site). Ohje on englanninkielinen.

2.4 Joomla!:n lisäosat

Joomla!:n parhaita puolia ovat sen laajennettavuus sekä avoimesta lähdekoodista johtuva runsas laajennusten määrä. Laajennukset jakaantuvat komponentteihin, moduuleihin ja liitännöihin. Näistä laajimpia ovat komponentit. Virallinen sivu erilaisille Joomla!-laajennuksille on http://extensions.joomla.org/extensions/ ja sinne on kasattuina maksullisia ja ilmaisia, eri aihepiireihin liittyviä laajennuksia. (Joomla! extensions)

”Komponentit ovat Joomla!:n ydintoiminnallisuutta lisääviä laajennuksia.” (Joomlaportal.fi ukk). Joomla!:an on asennettuna valmiiksi joitakin komponentteja, mutta lisää voi asentaa itse omien sivujensa tarpeiden mukaan. Esimerkkeinä suosituista komponenteista voisi toimia erilaiset kalenterit, verkkokaupat, vieraskirjat ja lomakkeet.

Moduulit ovat muualla kuin varsinaisessa sisällössä tapahtuvia laajennuksia. Ne laajentavat toimintaa pienillä sisältöalueilla ja ovat yleensä pieniä ohjelmia. Moduulipaikat ovat riippuvaisia käytettävästä sivupohjasta. Esimerkkeinä Joomla!:n moduuleista voisi toimia valikot sekä Facebookin tykkää-laatikko. (Joomlaportal.fi ukk).

Liitännät (eng. plugins) ovat sitä varten, että niiden avulla voidaan käynnistää erilaisia tapahtumia Joomla!:ssa. Esimerkkeinä liitännäisistä ovat Google Maps ja järjestelmänhallinnan puolella editorit.

3 KOMPONENTIN KEHITYKSEN ALOITUS

Periaatteessa kaikki Joomla!:n lisäosat noudattavat MVC-arkkitehtuuria eli malli – näkymä – ohjain arkkitehtuuria. Helpoin tapa ymmärtää komponentin kehitystä on tutustua MVC-arkkitehtuuriin ja rakentaa komponentti näiden osien mukaisesti. Lisäksi Joomla!:n komponentit käsittävät niin taustalla tapahtuvan hallintaosan kuin sivujen etuosan, eli sivuilla vierailevalle käyttäjälle näkyvän osan, ja myös näiden kahden puolen kehitystyö kannattaa tehdä vaiheittain.

3.1 Asennuspaketti

Kaikki laajennukset Joomla!:an asennetaan käyttäen pakattua (.zip) tiedostoa, joka sisältää kaikki, mitä tarvitaan asentamiseen ja poistamiseen. Se, kuinka paketin nimeää, ei ole merkityksellistä eikä myöskään paketin sijainti koneella. Asennuspaketin voi tehdä itse tiedosto tiedostolta, käyttäen itselle mieluisaa ohjelmointiympäristöä, tai käyttää jotakin Joomla!:n ilmaista laajennusta, joka tekee paketin valmiiksi. Ilmainen komponentin luoja löytyy esimerkiksi Not Web Designin sivuilta (Not Web Design).

Hitaampi, mutta oppimisen ja ymmärtämisen kannalta hyödyllisempi vaihtoehto, on rakentaa paketti tiedosto kerrallaan käyttäen itselle mieluisaa koodieditoria. Asennuspaketin sisältö jakautuu hallintaosaan sekä sivujen etuosaan ja lisäksi vaaditaan XML-tiedosto asennusta varten.

Kehitys on huomattavasti helpompaa, kun muutosten vaikutuksia voi seurata sivuilla, joten aluksi voi tehdä vain minimivaatimukset täyttävän paketin, jota sitten kehityksen edetessä täydennetään lisäämällä kansioita ja tiedostoja ja kirjaamalla muutokset XML-tiedostoon.

3.1.1 Komponentin XML-tiedosto

Komponentin asennustiedosto eli XML-tiedosto luodaan asennuspaketin juureen. Tiedoston sisältö on osittain tarkasti määrätty ja osa sisällöstä on vapaavalintaista. Tiedosto alkaa XML-julistuksella, jossa kerrotaan XML:n versio sekä käytettävä koodaus. Toisella rivillä kerrotaan Joomla!:lle, mitä ollaan asentamassa. Nämä esitetään kuviossa 1.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <install type="component" version="1.5.0">
```

KUVIO 1. XML-tiedoston aloitus

Seuraavaksi kirjoitetaan komponentin tiedot. Näistä pakollisia tietoja ovat ainoastaan komponentin nimi, jonka tulee olla muista komponenteista poikkeava. Esimerkki on esitetty kuviossa 2.

```
3      <name>Komponentin nimi</name>
4      <creationDate>Päivämäärä</creationDate>
5      <author>Komponentin tekijä</author>
6      <license>lisenssi, esim. GNU/GPL</license>
7      <version>1.0</version>
8      <description>Kuvaus komponentista</description>
```

KUVIO 2. XML-tiedosto, komponentin tiedot

Nimen lisäksi voi laittaa tiedon siitä, milloin komponentti on luotu, kenen toimesta ja millä lisenssillä sekä tiedon siitä, mikä versio komponentista on kyseessä sekä lyhyen kuvauksen komponentista. Tähän voi laittaa lisäksi esimerkiksi sähköpostiosoitteen, polun, josta komponentin voi ladata sekä omien sivujen osoitteen, joilla komponenttia mahdollisesti käytetään.

Seuraavaksi kirjataan tiedot siitä, mitkä tiedostot ajetaan asennettaessa ja poistettaessa. Käytännössä tässä huolehditaan siitä, että tietokantaan luodaan taulu tai tauluja, kun komponentti asennetaan ja poistetaan kaikki luodut taulut, kun komponentti poistetaan. XML-tiedostoon tarvittavat merkinnät ovat esitettyinä kuviossa 3.

```
    <install>
        <sql>
            <file driver="mysql" charset="utf8">sql/install.mysql.utf8.sql</file>
        </sql>
    </install>
    <uninstall>
        <sql>
            <file driver="mysql" charset="utf8">sql/uninstall.mysql.utf8.sql</file>
        </sql>
    </uninstall>
```

KUVIO 3. XML-tiedosto, tietokantaa varten tarvittavat tiedostot

Seuraavaksi listataan tiedostot, jotka asennetaan, sekä lisätään komponentti menuvalikoihin, jotta se voidaan sivuille lisätä. Kuvio 4 esittää minimivaatimukset täyttävän paketin tiedostot sekä menulinkkien lisäämisen.

```
    <files folder="components/com_inquirycreator">
        <filename>komponentinNimi.php</filename>
        <filename>index.html</filename>
        <filename>controller.php</filename>
    </files>

    <administration>
        <menu task="default" img="js/ThemeOffice/component.png">KomponentinNimi</menu>
        <submenu>
            <menu link="option=com_inquirycreator">KomponentinNimi</menu>
        </submenu>

        <files folder="administrator/components/com_inquirycreator">
            <filename>komponentinNimi.php</filename>
            <filename>index.html</filename>
            <filename>controller.php</filename>
            <filename>sql/index.html</filename>
            <filename>sql/install.mysql.utf8.sql</filename>
            <filename>sql/uninstall.mysql.utf8.sql</filename>
        </files>
    </administration>
</install>
```

KUVIO 4. XML-tiedosto, asennettavat tiedostot sekä menu

Jo tässä vaiheessa voi huomata, että tiedostot viedään palvelimella sivujen etuosaan ja taustan hallintaosaan, koska files folder kohtia on XML-tiedostoon merkittyinä kaksi erinimistä. Sitä mukaan, kun tiedostoja ja kansioita lisätään kehityksen myötä, on ne aina muistettava lisätä myös tähän XML-tiedostoon.

3.1.2 Komponentin etuosa

Tässä vaiheessa .xml tiedostossa on listattuina kolme tiedostoa, jotka menevät komponentin etuosaan, index.html, komponentinNimi.php ja controller.php. Index.html tiedosto on aina samanlainen ja se tuodaan jokaiseen kansioon kehityksen edetessä. Se on tavallaan tyhjä tiedosto eli siinä luodaan ainoastaan valkoinen internet-sivu, jonka tarkoituksena on estää käyttäjiä selaamasta suoraan kansioon. Tämän tiedoston sisältö on esitetty kuviossa 5.

```
1  <html>
2      <body bgcolor="#FFFFFF">
3      </body>
4  </html>
```

KUVIO 5. Index.html sisältö

KomponentinNimi.php tiedosto sisältää tässä vaiheessa kaikille komponentin PHP-tiedostoille yhteisen aloituksen, jolla estetään suora käsiksi pääsy. Sen jälkeen tuodaan Joomla!:n controller-kirjasto sekä luodaan komponentin ohjain eli controller käyttäen controller.php tiedostoa. Nämä on esitetty kuviossa 6.

```
2      defined('_JEXEC') or die('Restricted access');
3      jimport('joomla.application.component.controller');
4      require_once( JPATH_COMPONENT.DS.'controller.php' );
5      $classname = 'KomponentinNimiController'.$controller;
6      $controller = new $classname( );
7      $controller->execute( JRequest::getWord( 'task' ) );
8      $controller->redirect();
```

KUVIO 6. KomponentinNimi.php

Controller.php tiedosto alkaa samoilla riveillä, kuin KomponentinNimi.php tiedosto, joilla estetään suora käsiksi pääsy sekä tuodaan kirjasto (kuvio 6, ensimmäiset kaksi riviä). Tiedosto sisältää myös ohjain-luokan, joka laajentuu JController-luokasta. JController-luokka on Joomla!:n abstrakti luokka, joka hoitaa esimerkiksi näkymien esittämisen. Ohjain-luokka sisältää display()-funktion, joka palauttaa parent::display()-kutsun. Kehityksen edetessä näitä edellä mainittuja tiedostoja täydennetään ja pakettiin lisätään myös muita tiedostoja.

3.1.3 Komponentin hallintaosa

Hallintaosan puolelle viedään index.html tiedosto, joka on sisällöltään täysin identtinen sivuilla näkyvän puolen vastaavan tiedoston kanssa (kuvio 5), sekä tyhjä komponentin-Nimi.php tiedosto ja komponentin sql-kansioon tiedostot asennusta ja poistamista varten.

Install.mysql.utf8.sql tiedosto sisältää komponentin taulujen luonnin ja vastaavasti uninstall.mysql.utf8.sql tiedosto sisältää komponentin taulujen poiston. Taulua luodessa sille annetaan nimi sekä halutut kentät. Kyselykomponentille sopivat kentät voisivat olla id, kyselyn otsikko sekä tieto siitä, onko kysely julkaistu. Yksittäisten kyselyjen taulun tulee sisältää kaikki tiedot kyselystä, kuten kysymys, vastausvaihtoehdot, palautteet jne. Nämä .sql tiedostot ovat ulkoasultaan sql-komentoja, kuten alla kuviossa 7 esitetään. Esimerkissä varmistetaan, että aiempi versio taulusta on poistettu ennen uuden luomista.

```
1  DROP TABLE IF EXISTS `#__komponentinNimi`;
2
3  CREATE TABLE `#__komponentinNimi` (
4    `id` int(11) NOT NULL AUTO_INCREMENT,
5    `title` varchar(25) NOT NULL,
6    `published` tinyint(1) NOT NULL DEFAULT '0',
7     PRIMARY KEY  (`id`)
8  ) ENGINE=MyISAM AUTO_INCREMENT=0 DEFAULT CHARSET=utf8;
9
```

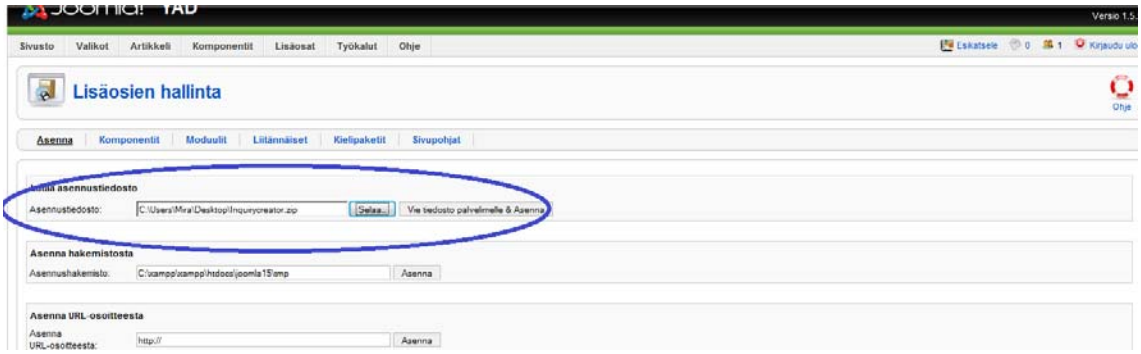
KUVIO 7. Sql-tiedoston komentoja

Tiedot on hyvä jakaa useaan tauluun ja monimutkaisissa komponenteissa saattaa olla parikymmentäkin taulua eri tiedoille. Tietokannan rakennetta kannattaa pohtia kunnolla ennen kuin komponenttia alkaa kehittää.

3.2 Komponentin asennus

Asennuspaketti viedään Joomla!-sivuille kirjautumalla sivujen ylläpitoon, menemällä ”Lisäosat”-välilehdelle ja sieltä valitsemalla ”Asenna/poista”. Aukeava sivu on nimeltään ”Lisäosien hallinta”. ”Lataa asennustiedosto” -kohdasta painetaan ”Selaa..”-nappia ja valitaan äsken tehty asennuspaketti ja painetaan ”Vie tiedosto palvelimelle & Asen-

na”. Jos paketti on kasattu oikein, kaikki tiedostot löytyvät ja ovat nimetty oikein, Joomla! antaa tiedon asennuksen onnistumisesta. Tämän jälkeen kehityksen etenemistä on helppo seurata omien Joomla!-sivujen kautta. Asennussivun näkymä on esitetty kuviossa 8.



KUVIO 8. Komponentin asennus sivuille

3.3 Komponentin sisältämät tiedostot

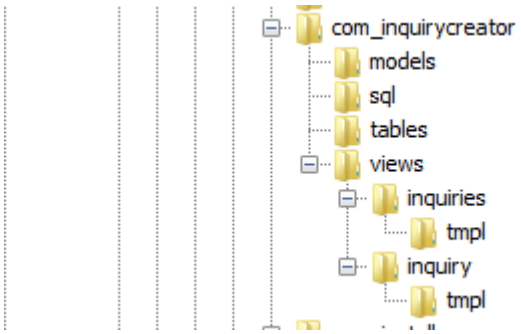
Asennuspaketin sisältämät tiedostot löytyvät asennuksen jälkeen kansioiden alta palvelimelta. Jokainen komponentti tarvitsee toimiakseen tiedostoja, jotka menevät sivujen administrator-kansioon eli hallintaosaan, components-kansioon eli etuosaan sekä mahdollisesti language-kansioon. Lisäksi tarvitaan xml-tiedosto, joka kertoo Joomla!:lle, mitä tiedostoja komponentilla on sekä tiedostojen sijainnin.

Tiedostot jakautuvat etuosan tiedostoihin, hallintaosan tiedostoihin, luokkatiedostoihin sekä asennustiedostoihin. Etuosa tarkoittaa käyttöliittymää, jonka kautta käyttäjä kommunikoi taustalla olevien tiedostojen kanssa eli sitä osaa, jonka sivujen käyttäjä komponentista näkee. Nämä etuosan tiedostot löytyvät sivujen alta components-kansiosta. Hallintaosaan liittyvät tiedostot tarkoittavat niitä tiedostoja, jotka ohjaavat sivujen järjestelmänhallinnoijan näkymää eli sitä osaa, jonka avulla sivujen hallinnoija komponenttia käyttää. Hallintaosaan liittyvät tiedostot löytyvät sivujen alta, administrator-kansiosta ja sieltä edelleen components-kansiosta. Luokkatiedostoja käytetään näkymiä varten ja niiden tiedostot löytyvätkin näkymien kansioista. Asennustiedostot tarvitaan ainoastaan asennusta varten ohjaamaan kaikki tiedostot oikeisiin kansioihin sekä luo

maan tietokantaan tauluja komponenttia varten sekä mahdollisesti komponenttia poistettaessa huolehtimaan, että kaikki oikeat tiedostot poistuvat palvelimelta sekä tietokannasta.

4 KOMPONENTIN HALLINTAOSA

Komponentin kehitystä helpottaa, jos tutustuu Joomla!:n luokkiin. Komponentista saa huomattavasti paremmin ulkonäöltään yhtenäisen muiden komponenttien kanssa, jos käyttää kehityksessä Joomla!:n CSS-luokkia, jotka ovat listattuina esimerkiksi Joomla!:n dokumenttisivuilla (Joomla! documentation, CSS classes). Lopullisen komponentin kansiorakenne hallintaosassa on kuvion 9 kaltainen. Monimutkaisemmissa komponenteissa myös kansiorakenne luonnollisesti on monimutkaisempi ja esimerkiksi ohjaimet ovat omassa controllers-kansiossaan ja lisäksi saattaa olla tarpeen mukaan paljon muitakin kansioita.



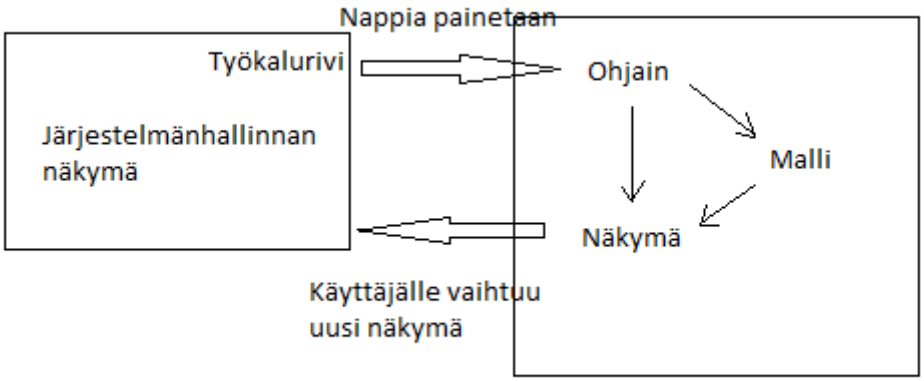
KUVIO 9. Komponentin kansiorakenne

4.1 Ohjain

Ohjain sijaitsee komponentin juuressa tai monimutkaisemmissa komponenteissa on tapana tehdä controllers-kansio komponentin juureen ja sijoittaa sinne kaikki tarvittavat ohjaimet. MVC-arkkitehtuurin ohjaimen tehtävänä on vastata sivujen käyttäjän toimintoihin. Tässä tiedostossa rekisteröidään ohjain-luokkaan kaikki tarvittavat tehtävät sekä asetetaan kullekin tehtävälle vaadittavat toiminnot. Pohjana toimii Joomla!:n JController-luokka, johon suurin osa toiminnallisuuksista on sisällytetty.

Komponentin ohjain-luokka koostuu funktioista. Display()-funktio sisältyy kaikkiin komponentteihin ja sen tehtävänä on kutsua ylemmän luokan konstruktoria. Lisäksi ohjain-luokan display()-funktion sisällä määrätään eri näkymien nimet sekä ulkoasut ja

asetetaan näkymien ulkoasut. Kuvio 10 pyrkii selvittämään, kuinka komponentti rakentuu MVC-arkkitehtuuria käyttäen, sekä ohjaimen osuutta komponentissa.



KUVIO 10. MVC-arkkitehtuuri, ohjaimen tehtävä

Eri tehtävien aktivoinnin lisäksi ohjaimen tarkoitus on suorittaa tehtävä kutsumalla oikeata mallia ja näin edelleen ohjata oikeaan näkymään. Esimerkiksi, jos järjestelmänhallinnoija haluaa tehdä uuden instanssin komponentistaan, hän painaa uusi-nappia työkaluriviltä. Napin painaminen aiheuttaa ohjain-luokassa tehtävän vastaanottamisen ja lopputuloksena yksittäisen kyselyn luomisnäkymän esittämisen.

4.2 Näkymä

Näkymä jaetaan kahteen osaan, komponentista tehtyjen instanssien listaukseen sekä yksittäisen instanssin näkymiin. Oletusnäkymänä komponentista on aina listaus tehdyistä instansseista varustettuna oletustyökalurivillä sekä sivunavigaatiolla. Näkymät vaativat kansion (views) komponentin juureen ja tämä kansio sisältää erilliset kansiot instanssien listaukselle ja yksittäiselle instanssille. Molempien näiden kansioiden rakenne on samankaltainen eli juuressa on view.html.php ja index.html tiedostot sekä kansio tmpl, joka sisältää default.php tiedoston ja mahdollisesti lisäksi yksittäisen instanssin tapauksessa näkymät eri tilanteille, kuten edit.php.

4.2.1 Työkalurivit

Joomla!:ssa on määriteltyinä omat työkalurivin toiminnot, joita voi käyttää hyvin yksinkertaisesti JToolBarHelper eli avustajaluokan avulla. Tämän avustajaluokan metodit ovat listattuina Joomla!:n dokumentaatioissa (Joomla! documentation, JToolBarHelper). Normaalisti oletusnäkymässä työkalurivillä on otsikon lisäksi napit julkaise, lopeta julkaisu, poista, muokkaa ja uusi. Muitakin nappeja voi olla käytössä komponentin tyypistä riippuen ja myös omia nappeja voi kehittää komponentin tarpeisiin. Työkalurivi luodaan oletusnäkymätiedostossa view.html.php kuvion 11 kaltaisilla komennoilla.

```
5
6   <?php
7       JToolBarHelper::title(  JText::_('Komponentin otsikko') );
8       JToolBarHelper::publishList();
9       JToolBarHelper::unpublishList();
10      JToolBarHelper::deleteList();
11      JToolBarHelper::editListX();
12      JToolBarHelper::addNewX();
13  ?>
```

KUVIO 11. Työkalurivin luominen

Yksittäisen instanssin näkymässä, esimerkiksi luomis- tai muokkausnäkymässä, työkalurivillä on otsikon lisäksi napit käytä, tallenna, peru sekä esikatsele.

4.2.2 Instanssien listaus

Työkalurivin jälkeen oletusnäkymässä esitetään instanssien listaus. Tiedot tallennetuista instansseista haetaan mallin puolelta kutsumalla view.html.php tiedostossa getData()-funtiota kuvion 12 esimerkin kaltaisesti.

```
33          $rows =& $this->get('Data');
34          $this->assignRef('items', $rows);
35          parent::display($tpl);
```

KUVIO 12. Haetaan tiedot mallin puolelta ja luovutetaan tiedot näkymälle

Jotta listauksen ulkoasu olisi muiden komponenttien kaltainen, käytetään Joomla!:n CSS-luokkia. Listaus tehdään default.php tiedostossa lomakkeen sisään, taulukkona,

joka on jaettu HTML:n thead-, tfoot- ja tbody-tagien väliin. Thead-tagien väliin tuodaan taulukon otsikkorivi eli selitteet sen alapuolelle listattaville asioille. Tapana on, että tässä järjestyksenä on id, joka on instanssin yksilöivä tunniste, valintaruutu, otsikko ja muut listauksessa näytettävät tiedot, kuten onko julkaistu vai ei. Id merkitään usein ’#’-merkillä ja valintaruutuun lisätään toiminto, että sen ollessa valittuna kaikki listatut instanssit ovat valittuina. Kuvio 13 selventää otsikkorivin id:n ja valintaruudun käyttöönottoa.

```
<tr>
    <th width="5">
        <?php echo JText::_('NUM'); ?>
    </th>
    <th width="20">
        <input type="checkbox" name="toggle" value="" onclick="checkAll(<?php echo count($this->items); ?>)" />
    </th>
```

KUVIO 13. Otsikkorivin id:n merkintä sekä valintaruudun toiminnallisuus

Tbody-tagien väliin listataan komponentin instanssit. Yksinkertaisin tapa on tehdä for-silmukka, joka tulostaa jokaisen tietokannasta haetun rivin elementti kerrallaan. Tässä ei varsinaisesti tarvitse mitään erikoistaitoja, mutta ennen tulostusta kannattaa esimerkiksi otsikosta tehdä linkki kyseisen instanssin muokkaussivuun, jotta listan käyttäminen helpottuu. Linkki tehdään kuvion 14 komennolla.

```
<?php
    $k = 0;
    for (0, $n=count($this->items); $i < $n; $i++)
    {
        $row = &$this->items[$i];
        $link = JRoute::_( 'index.php?option=com_inquirycreator&view=inquiry&task=edit&cid[]='. $row->id );
        $checked    = JHTML::_('grid.checkedout', $row, $i );
        $published  = JHTML::_('grid.published', $row, $i );
```

KUVIO 14. For-silmukan aloitus, josta käy ilmi linkin muodostus

Tfoot-tagien sisällä tulostetaan listauksen jälkeen tuleva näyttö-valikko, jonka tarkoituksena on muuttaa listausta siten, että sivulla voidaan näyttää esimerkiksi vain viisi tai sitten 100 instanssia ja samalla jakaa listaus useammalle sivulle. Tämä Joomla!:n JPagination-luokkaa käyttävä taitto-toiminto luodaan view.html.php tiedostossa antamalla sille tieto tallennettujen instanssien kokonaismäärästä, aloitettavasta instanssista sekä sivulla näytettävien instanssien määrästä kuvion 15 mukaisesti.

```
$total =& $this->get('Total');
$pagination = new JPagination( $total, $pagination->limitstart, $pagination->limit);
$this->assignRef('pagination',  $pagination);
```

KUVIO 15. Taitto-toiminto

4.2.3 Yksittäisen instanssin luomis- tai muokkausnäkymä

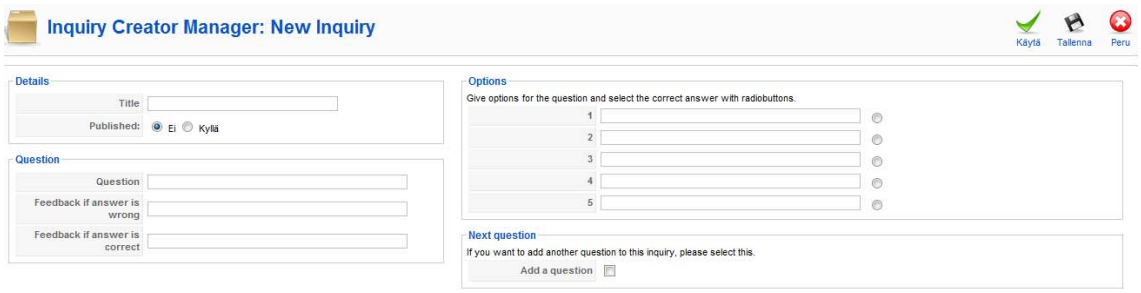
Yleensä uuden komponentin esiintymän luomissivu on samanlainen kuin muokkaussivu, jolloin ne voi tehdä yhdeksi default.php sivuksi. Kuten listaussivuakin tehdessä, myös tässä käytetään ulkoasun yhteneväisyyden vuoksi Joomla!:n omia CSS-luokkia. Näkymä tehdään listauksen tavoin lomakkeen sisään taulukkona. Kuitenkin ennen lomaketta kirjoitetaan JavaScriptillä toiminnot työkalurivin nappien painamiselle, työkalurivin luominen käsiteltiin jo edellä.

Kyseessä on lomake, joten lomake lähetetään jonnekin ja siinä pitää olla tarvittavat tiedot täytettyinä ennen lähetystä. JavaScriptillä kirjoitetaan yksinkertainen lomakkeen tarkistus ja määrätään näin kentät, jotka eivät saa olla tyhjiä tallennettaessa. Normaalisti jokaisella komponentin instanssilla tulee olla ainakin otsikko ja tässä kyselykomponentin tapauksessa myös kysymys. Tarkistukset tehdään kuvion 16 kaltaisella if-else-rakenteella.

```
<script language="javascript" type="text/javascript">
    function submitbutton(pressbutton) {
        var form = document.adminForm;
        if (pressbutton == 'cancel') {
            submitform( pressbutton );
            return;
        }
        // do field validation
        if (form.title.value == "") {
            alert( "<?php echo JText::_( 'Inquiry must have a title', true ); ?>" );
        } else if (form.question.value == "") {
            alert("<?php echo JText::_('Inquiry must have a question', true); ?>");
        } else {
            submitform( pressbutton );
        }
    }
</script>
```

KUVIO 16. Lomakkeen validointi

JavaScript-osion jälkeen näkymä rakennetaan komponentin tarpeiden mukaisesti eri osioihin. Listauksen tavoin tiedot haetaan aina view.html.php tiedoston avulla ja tiedot näytetään tai ne kerätään edelleen lähetettäviksi PHP-tiedoston avulla, joka sijaitsee yksittäisen näkymän kansiossa tmpl kansion alla. Tämän tiedoston nimeämisellä ei ole merkitystä, mutta usein käytetään nimeä form.php. Lähes kaikilta ellei jopa kaikilta komponenteilta löytyy Details-osio, johon instanssia luova henkilö voi antaa tiedot otsikosta, mahdollisesti antaa jonkin aliaksen sekä määrätä, onko instanssi julkaistu vai ei. Muita osioita kyselykomponentissa voi olla tiedot kysymyksestä, joka sisältää itse kysymyksen sekä mahdolliset palautteet käyttäjän antamasta vastauksesta sekä osio vastausvaihtoehdoista ja vielä osio, jossa luodaan linkki mahdolliseen seuraavaan kysymykseen. Kuitenkin ideana on, että tässä näkymässä kysytään kaikki tiedot, jotka tarvitaan instanssin tallentamiseen tietokantaan. Jokainen osio luodaan omana taulukkonaan, jotta tietojen käsittely helpottuu. Lopullinen näkymä uutta instanssia luodessa voi näyttää esimerkiksi kuvion 17 kaltaiselta. Samaa näkymää voi käyttää instanssia muokatessa muuttamalla otsikon.



KUVIO 17. Yksittäisen instanssin luomisnäkymä

Instanssia muokatessa näkymä voi siis olla samankaltainen, mutta kentät eivät ole tyhjiä vaan niihin on haettu jo tallennetut tiedot. Tallennettujen tietojen haku tapahtuu mallissa sen jälkeen, kun view.html.php tiedostossa tehdään niistä kysely. Tämä tapahtuu samaan tapaan kuin instanssien listausta tehtäessä. Ainoa ero on, että nyt tietoa haetaan useammasta taulusta, jos komponentista tallennettavat tiedot ovat hajautettuina useampaan tauluun. Esimerkin kyselykomponentissa kyselyä muokattaessa view.html.php tiedostoon tuodaan kyselyn tiedot kyselyn id:n perusteella. Tästä tiedot välitetään lomakkeelle ja kaikki tallennetut tiedot näkyvät omissa kentissään ja ovat käyttäjän muokattavissa.

4.3 Malli

Joomla! on rakennettu niin, että ohjain lataa automaattisesti mallin, jolla on sama nimi kuin näkymällä ja laittaa sen näkymään. (Joomla! Developing a Model-View-Controller Component, part2). Kun tiedostot ja luokat ovat oikein nimettyinä, ei niiden yhdistämisestä tarvitse välittää.

Mallin tehtävä on hakea tarvittavat tiedot tietokannasta. Näkymää luodessa kutsumme näkymän puolella sitä funktiota mallissa, joka palauttaa meille tiedot, jotka haluamme näyttää. Esimerkiksi aiemmin esillä oli funtiokutsu, joka palauttaa tiedot kaikista tallennetuista instansseista listausta varten. Malli tekee kutsun saatuaan haun tietokantaan ja hakee kaikki tarvittavat tiedot, tekee niistä listan ja palauttaa listan näkymälle. Vastaavasti mallista voidaan hakea tieto tallennettujen instanssien kokonaismäärästä tai yksittäisen instanssin tiedot muokkausta varten. Listauksen tietojen haku tapahtuu mallissa kuvion 18 tapaan.

```
class InquiryCreatorModelInquiries extends JModel
{
    function getData()
    {
        $db =& JFactory::getDBO();

        $query = 'SELECT m.*'
        . ' FROM #__inquirycreator AS m'
        . $where
        . ' GROUP BY m.id'
        . $orderby
        ;
        $db->setQuery( $query);
        $rows = $db->loadObjectList();
        return $rows;
    }
```

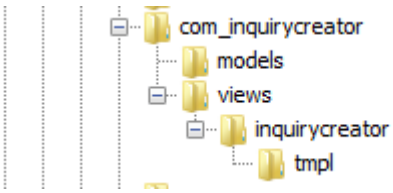
KUVIO 18. Tallennettujen kyselyjen haku ja kyselyjen vienti näkymälle

5 KOMPONENTIN ETUOSA

Taustan tavoin komponentin etuosa, eli sivuilla näkyvä puoli, jaetaan malliin, näkymään ja ohjaimeen. Lisäksi etuosasta löytyvät kansion juuresta tiedostot ”komponentinNimi.php” sekä index.html, joiden pohjat luotiin jo asennuspakettia tehdessä. Komponentin etuosan sisältö on huomattavasti suppeampi kuin hallintaosan, koska täällä ei enää luoda uutta, vaan esitetään jo luotu tieto sivujen käyttäjälle. Tässä luvussa keskitytään komponentin etuosan rakenteeseen ja sisältöön, rajaamalla kuitenkin pois tarkemman käsittelyn asioista, jotka liittyvät sivujen tekemiseen Joomla!:lla.

5.1 Etuosan kansiorakenne ja sivuille liittäminen

Kyselykomponentti on pohjimmiltaan hyvin yksinkertainen komponentti ja siksi kansiorakenne etuosan puolella on vain minimivaatimukset täyttävä. Komponentin kansion juuressa sijaitsevat ohjain, ”komponentinNimi”.php sekä index.html. Etuosan kansiorakenne on kuvattuna kuviossa 20.



KUVIO 20. Etuosan kansiorakenne

Jotta komponentin instansseja voidaan liittää sivuille, on komponentille luotava valikoihin linkki. Tämä tehtiin jo asennuspakettia luotaessa XML-tiedostoon. Kun komponentille luotiin linkki valikoihin, voidaan sivuille tehtävälle näkymälle antaa tyypiksi komponentti. Tällä tarkoitetaan sitä, että sivuilla on valikko ja valikon linkit ohjaavat eri paikkoihin, toiset artikkeleihin, toiset verkkokauppaan tai valokuvagalleriaan, nyt valikon linkki voi ohjata myös tehdyn uuden komponentin näkymään.

Mikäli komponentista on tehty useita instansseja, on valikkoja tehtäessä myös oltava mahdollisuus valita juuri oikean instanssin näkymä linkiksi. Tätä varten valikon linkkiä valittaessa, parametrit kohdassa on oltava mahdollisuus valita kaikista instansseista yk-

si. Tässä tarvitaan ymmärrystä Joomla!:n käyttämisestä sivujen tekemiseen, joten tässä työssä ei puututa tähän sen enempää.

5.2 Ohjain ja muut tiedostot kansion juuressa

Kansion juuressa sijaitsee jo useampaan kertaan mainittu index.html, jonka sisältö on täysin vastaava aiemmin mainittujen kanssa. Tätä ei tässä sen enempää käsitellä.

Kansion juuresta löytyy myös aiemmin mainittu ”komponentinNimi.php” tiedosto, jonka tarkoitus on toimia niin sanotusti komponentin sisäänkäyntinä. Kyseisen tiedoston sisältö on hyvin samankaltainen kaikille komponenteille ja mallina voi käyttää lähteen versiota omaa luodessa. Tiedoston tarkoituksena on ladata sekä luoda ohjain. (Joomla! Developing a Model-View-Controller Component, part1).

Ohjain huolehtii, taustan ohjaimen tavoin, suoritettavista tehtävistä. Yksinkertaisimmissa komponenteissa tässä tiedostossa ei siis huolehdita muusta kuin näkymän lataamisesta, mutta monimutkaisemmissa komponenteissa täällä huolehditaan eri tehtävien ohjaamisesta oikeisiin paikkoihin tai uusien näkymien lataamisesta.

5.3 Malli

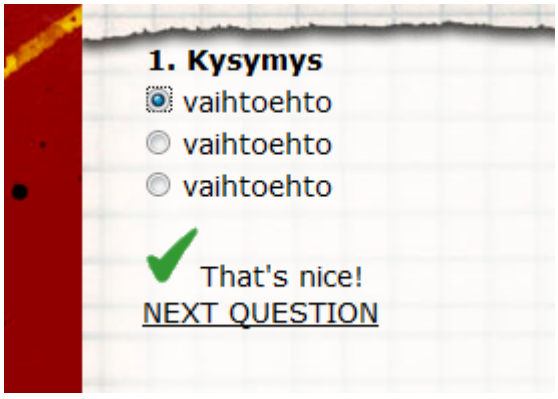
Mallin tehtävänä on hallintaosan mallin tavoin huolehtia tietojen hakemisesta tietokannasta sekä edelleen välittää nämä haetut tiedot näkymälle. Tiedot haetaan täysin vastaavilla komennoilla kuin hallintaosan puolellakin. Kyselykomponentin tapauksessa tietokannasta haetaan tieto kysymyksestä sekä vastausvaihtoehdot ja mahdolliset palautteet niihin ja vielä mahdollisen seuraavan kysymyksen tiedot.

5.4 Näkymä

Näkymän kansiorakenne on samanlainen kuin järjestelmänhallinnan puolellakin. View-kansio sisältää index.html ja view.html.php tiedostot sekä tmpl-kansion. Tmpl-kansiosta löytyy default.php sekä index.html tiedostot.

View.html.php tiedostoon tuodaan mallin puolella tietokannasta haetut tiedot, jotka puolestaan välitetään sivujen todellisen näkymän rakentavalle default.php tiedostolle, joka sijaitsee tmpl-kansiossa. Kyselykomponentin tapauksessa näkymä tarvitsee tiedot kysymyksestä, vastausvaihtoehdot sekä tiedot mahdollisista jatkokysymyksistä ja mahdollisista vastauksiin annettavista palautteista.

Default.php tiedostossa määrätään se, miltä komponentin tulee sivuilla vierailevalle käyttäjälle näkyä. Kyselykomponentin tarkoitus on näyttää riveittään ensin kysymys ja sen perään vastausvaihtoehdot varustettuna valintaruudulla. Kun vastaaja valitsee jonkin vastausvaihtoehdoista, saa hän järjestelmänhallinnan puolella mahdollisesti (vapaaehtoinen ominaisuus, voi jättää kyselyä tehtäessä myös tyhjäksi) määrätyn palautteen vastaukselleen eli oliko vastaus oikein vai väärin. Lisäksi oikean vastauksen yhteydessä ilmestyy näkyville linkki mahdolliseen seuraavaan kysymykseen. Kuviossa 19 esitetään kyselykomponentin sivuilla näkyvä puoli.



KUVIO 19. Kysymyksen ja palautteen esittäminen sivuilla

Kyselyn tiedot esitetään html-tietojen sisään rakennetun for-silmukan avulla ja varsinainen toiminnallisuus eli palautteen antaminen ja seuraavaan kysymykseen ohjaaminen suoritetaan JavaScriptillä html-tiedoston thead-tagien välissä. Komponentin ulkoista

asemointia voi ja kannattaa muokata sivujen CSS-tiedostossa tai laajassa komponentissa voi tehdä komponentille oman CSS-tiedoston.

6 YHTEENVETO

Komponenttien kehittäminen Joomla!:an on monivaiheinen prosessi, joka vaatii hyvää ymmärrystä MVC-arkkitehtuurista. Lisäksi kehittäjän tulee tutustua Joomla!:n olemassa oleviin komponentteihin melko laajasti, jotta ymmärtää täysin eron komponentin hallintaosan ja etuosan tarkoituksilla sekä pystyy hyödyntämään Joomla!:n omia luokkia saadakseen komponentista yhtenäisen näköisen jo olemassa olevien komponenttien kanssa.

Pelkkä sivujen tekeminen Joomla!:lla ei valmista komponenttien kehitykseen. Kuitenkin suuren sivuprojektin edetessä yleensä tieto ilmaiseksi löytyvistä komponenteista lisääntyy ja tällöin myös mahdollisesti huomaa, että jotakin tarpeellista komponenttia ei löydykään. Myös olemassa olevien komponenttien muokkaus omien sivujen tarpeisiin, auttaa ymmärtämään komponenttien rakennetta jonkin verran.

Komponenttia kehitettäessä työvaiheet voidaan jakaa komponentin etuosaan ja hallintaosaan. Etuosa ja hallintaosa jaetaan MVC-arkkitehtuurin mukaisesti malliin, näkymään ja ohjaimeen. Lisäksi hallintaosan puolella on huomioitava, että erotetaan yksittäisen instanssin esitys kaikkien luotujen instanssien esityksestä. Ohjaimen tehtävänä on vastata sivujen käyttäjän toimintoihin. Ohjain kutsuu mallia, joka hakee tarvittavat tiedot tietokannasta, ja välittää mallilta tulleet tiedot näkymälle. Jokainen komponentti vaatii myös asennusta varten ”komponentinNimi”.xml tiedoston, johon kaikki tiedostot ovat listattuina. Joomla!:an asennusta varten kaikki komponentin tiedostot pakataan .zip paketiksi ja Joomla! ohjaa kaikki tiedostot ”komponentinNimi”.xml tiedoston osoittamiin oikeisiin paikkoihin palvelimella.

Työssä keskityttiin eri työvaiheisiin komponenttia luotaessa, mutta jotain jätettiin myös pois. Jos omaa komponenttiaan haluaa edelleen kehittää ja tehdä siitä yhä useammalle käyttökelpoisen, kannattaa paneutua komponenttien kielitiedostoihin. Kielitiedostojen tarkoitus on helpottaa komponentin kääntämistä eri kielille ja näin tarjota sama komponentti niin suomalaisen kuin saksalaisenkin Joomla! sivujen tekijän käyttöön.

Tässä työssä ei myöskään edetty komponentin julkaisuprosessiin saakka. Tästä prosessista saa lisätietoa Joomla!:n laajennuksille tarkoitetulta sivulta extensions.joomla.org.

LÄHTEET

Sähköiset lähteet

Apache HTTP Server Project. Luettu 03.02.2011. http://httpd.apache.org/

Joomla! Developing a Model-View-Controller Component, part1. Luettu 10.04.2011. http://docs.joomla.org/Developing_a_Model-View-Controller_Component_-_Part_1

Joomla! Developing a Model-View-Controller Component, part2. Luettu 03.04.2011. http://docs.joomla.org/Developing_a_Model-View-Controller_Component_-_Part_2_-_Adding_a_Model

Joomla! Documentation, CSS classes. Luettu 21.03.2011. http://docs.joomla.org/List_of_Joomla!_generated_core_CSS_classes

Joomla! Documentation, JToolBarHelper. Luettu 21.03.2011. http://docs.joomla.org/JToolBarHelper

Joomla! extensions. Luettu 11.03.2011. http://extensions.joomla.org/extensions/

Joomlaportal.fi. Luettu 27.01.2011. http://www.joomlaportal.fi/content/view/93/39/

Joomlaportal.fi ukk. Luettu 11.03.2011. http://www.joomlaportal.fi/component/option,com_smf/Itemid,51/board,3.0

Not Web Design. Luettu 30.01.2011. http://www.notwebdesign.com/joomla-component-creator/

Official Joomla! site. Luettu 27.01.2011. http://help.joomla.org/ghop/feb2008/task048/joomla_15_quickstart.pdf

Oracle MySQL. Luettu 03.02.2011. http://www.mysql.com/about/

The PHP Group. Luettu 03.02.2011. http://www.php.net/

Tietoportti.com. Luettu 04.03.2011. http://www.tietoportti.com/MVC-arkkitehtuuri.html

Wikipedia. Luettu 11.03.2011. http://fi.wikipedia.org/wiki/Joomla

W3Schools.com. Luettu 03.02.2011. http://www.w3schools.com/js/js_intro.asp