TEKNIIKKA JA LIIKENNE

Tietotekniikka

Ohjelmistotekniikka

INSINÖÖRITYÖ

JOOMLA-JULKAISUJÄRJESTELMÄN TYYLIPOHJASUUNNITTELU

Työn tekijä: Pasi Heiskanen
Työn ohjaajat:
Erja Nikunen, yliopettaja
Anu Leponiemi, FM, Avoine

Työ hyväksytty: ___. ___. 2010

Erja Nikunen
yliopettaja

ALKULAUSE

Käsittelen tässä insinöörityössäni Joomla-verkkojulkaisujärjestelmää ja sen laajennusmahdollisuuksia. Valitsin aiheeksi Joomlan, koska olen käyttänyt sitä laajasti jo useamman vuoden ajan. Kuitenkin vasta nyt minulla oli aikaa uppoutua siihen syvemmin.

Visaisia pähkinöitä purressani olen kokenut useita ahaa-elämyksiä. Niiden avulla olen oppinut hahmottamaan osaamiseni rajoja ja sen myötä myös sellaisia ammatillisia alueita, joilla en ole vahvimmillani. Onko oppinut enemmän, jos on hoksannut lukemattomia isoja asioita kotipihallaan, kuin jos olisi oivaltanut yhden pienen asian vieraalla maalla? Minä iloitsen eniten tämän työn kannalta varsin mitättömästä havainnostani, siitä, että Joomla on oliopohjainen sovellus. Sen ymmärtäminen saatteli minut poluille, joilla kompastelin heiveröisimpiinkin juurakoihin. Polvet verillä rämpimistähän se usein oli, mutta löysin lopulta tieni kotiin taskut täynnä hienoja opin jyväsiä.

Insinöörityöni tekemiseen osallistui iso joukko ihmisiä. Ensimmäiset kiitokset osoitan Twitter-ystävilleni Joomla-aiheisten tviittieni välittämisestä ja Facebook-kavereilleni tuskaistenkin tilapäivitysteni tykkäämisestä. Anu Leponiemeä kiitän näkemyksellisestä ohjaamisesta ja tarkkaakin tarkemmasta pilkkujen viilaamisesta. Erja Nikuselle esitän lämpimän kiitoksen kuuntelemisesta ja tinkimättömyydestä. Suurimmat kiitokset kannustamisesta ja joustamisesta kuuluvat työyhteisölleni Kokoomuksessa sekä rakkaimmalleni Jari Stenvallille.

Helsingissä 26.9.2010

Pasi Heiskanen

Metropolia

TIIVISTELMÄ

Työn tekijä: Pasi Heiskanen

Työn nimi: Joomla-julkaisujärjestelmän tyylipohjasuunnittelu.

Päivämäärä: 26.9.2010 **Sivumäärä:** 47 + 4 liitettä

Koulutusohjelma: Tietotekniikka

Suuntautumisvaihtoehto: Ohjelmistotekniikka

Työn ohjaaja: Erja Nikunen, yliopettaja

Työn ohjaaja: Anu Leponiemi, FM, Avoine

Tämä opinnäytetyö tehtiin kuvitteelliselle blogikirjoittajalle. Hän haluaa koota bloginsa yhteyteen sisältöjä useilta muilta sosiaalisen median kanaviltaan.

Opinnäytetyössä tutustuttiin Joomla-sisällönhallintajärjestelmän arkkitehtuuriin, toimintaan sekä laajennus- ja muokkausmahdollisuuksiin. Työn alussa asennettiin Joomlan kehittämiseen soveltuva ohjelmisto- ja laitteistokokoonpano ja kehitys-, testaus- ja tuotantoympäristöt.

Työn aikana tehtiin Joomla-tyylipohja, joka soveltuu blogin kirjoittamiseen ja useista lähteistä olevien sisältöjen kokoamiseen. Tyylipohjan sommittelussa hyödynnettiin valmista ruudukkopohjaa. Se toteutettiin Joomlan tyylipohjien suunnitteluun tarkoitetuilla PHP-ehtolauseilla. Tyylipohjan avulla taulukkotaittoon perustuvat Joomlan oletusasemointimallit korvattiin sellaisilla asemointimalleilla, joissa on semanttinen XTHML-rakenne.

Joomla on monipuolinen PHP-pohjainen ohjelmistokehys ja julkaisujärjestelmä. Perustoimintojensa osalta se soveltuu hyvin yksityshenkilöiden blogialustaksi vaikka sellaisenaan, mutta muokattavuutensa ansiosta se kykenee vastaamaan myös yritysten vaativimpiin verkkosivutarpeisiin.

Avainsanat: Joomla, verkkojulkaisujärjestelmä, blogi, tyylipohja, php, ohjelmistokehys

Metropolia

ABSTRACT

Name: Pasi Heiskanen

Title: Template Design For Joomla Content Management System

Date: 26.9.2010 **Number of pages:** 47 + 4 appendices

Department: Information Technology **Study Programme:** Software Engineering

Instructor: Principal Lecturer Erja Nikunen

Supervisor: M.Sc. Anu Leponiemi, Avoine

This engineering study was carried out for an imaginary person who is writing a blog. In addition to the blog he wants the website to be an aggregator of the contents produced by him on several other social media channels.

The study begins by introducing the Integrated Development Environment and hardware requirements for Joomla system. The architecture, functions and possibilities in extension development of Joomla content management system are explored as well. The development, testing and production environments were installed during the study.

A Joomla template that meets the requirements of the blogger was developed during the study. A commonly used grid system was used for the layout. The grid was generated by XHTML and PHP-statements meant for Joomla template design. The table-based design patterns used by the system were overwritten in the template using files that generate semantic XHTML structure.

Joomla is a flexible PHP-based framework and a content management system. The basic installation of a Joomla system is suitable for a simple blog and due to the wide range of customising possibilities it also meets the complex requirements of a corporate website

Keywords: Joomla, content management system, CMS, blog, template, php, framework

SISÄLLYS

SANASTO

SANASTO

API Application programming interface, ohjelmointirajapinta.

Article Joomla-järjestelmän yksittäinen sisältöartikkeli.

Article Manager
Joomla-järjestelmän artikkeleidenhallintatyökalu

Backend Verkkosivuston ylläpitotoimintoihin käytetty liittymä. Työssä käytetään termiä ylläpitoliittymä.

Category Alaryhmä, tapa järjestellä Joomlan sisältöartikkeleita hierarkkisesti.

Category Manager
Joomla-järjestelmän alaryhmien hallintatyökalu.

Component Komponentti, Joomlan laajennustyyppi.

CSS Cascading style sheets, verkkosivustojen visuaalisen ilmeen määrittelemistä varten kehitetty tyyliohjeiden laji.

Div-taitto XHTML-kielessä käytetty tapa asemoida sisältöä rakenteellisesti käärien se div-elementtien sisään.

Extension Joomlan laajennus.

Extension Manager
Joomla järjestelmän laajennustenhallintatyökalu.

Frontend Liittymä, jonka verkkosivustolla vierailija näkee. Työssä käytetään termiä julkinen liittymä.

IDE Integrated development environment, ohjelmointiympäristö.

Language Manager
Joomla-järjestelmän kieliversioidenhallintatyökalu.

Module Moduuli, Joomlan laajennustyyppi.

Module Manager
Joomla-järjestelmän moduulienhallintatyökalu.

PHP Verkkopalvelinympäristöissä dynaamisten verkkosivujen luonnissa käytetty ohjelmointikieli.

Plugin Liitännäinen, Joomlan laajennustyyppi.

Plugin Manager
Joomla-järjestelmän liitännäistenhallintatyökalu.

Section Pääryhmä, tapa ryhmitellä Joomla-järjestelmän alaryhmiä hierarkkisesti.

Section Manager
Joomla-järjestelmän pääryhmienhallintatyökalu.

Snippet Lyhyt pätkä ohjelmistokoodia, jolla on yleensä joku pieni oma tehtävä. Pätkiä ja niiden toimintoja voidaan monistaa ja yhdistää osaksi muuta koodikokonaisuutta ja näin muodostaa isompia toimivia kokonaisuuksia.

Template Joomla-järjestelmän laajennustyyppi, minkä avulla määritellään sivun ulkonäkö.

Tviitti Twitter-verkkopalveluun käyttäjän lähettämä lyhyt merkintä.

Template Manager
Joomla-järjestelmän tyylipohjienhallintatyökalu.

XHTML Extensible hypertext markup language, verkkosivujen merkintäkieli.

1 JOHDANTO

Joomla on sisällönjulkaisujärjestelmä, jonka avulla voidaan tehdä Internet-sivuja. Sivujen sisällön päivittämiseksi ei tarvitse osata verkko-ohjelmointia. Järjestelmä soveltuu hyvin sekä laajojen sivukokonaisuuksien että yksinkertaisten sivustojen ylläpitämiseen. Joomla perustuu avoimeen lähdekoodiin, ja sen lisenssi sallii ohjelmiston vapaan käytön ja muokkaamisen.

Joomla kirjoitetaan virallisesti huutomerkin kanssa muodossa Joomla!, ja tässä opinnäytetyössä käsitellään Joomla! 1.5.17 -versiota. Jatkossa käytetään yksinkertaista termiä Joomla.

Joomlaa voidaan tarkastella myös ohjelmistokehyksenä, joka tarjoa monipuolisia laajennusmahdollisuuksia. Joomlan laajentamisella tarkoitetaan lähes minkä tahansa uuden toiminnallisuuden kehittämistä Joomlan perusasennuksen ominaisuuksien lisäksi. Joomla-laajennuksia (Joomla extensions) on kolmea päätyyppiä: komponentteja (components), moduuleita (modules) ja liitännäisiä (plugins).

Työn tavoite

Tavoitteena on tutustua Joomlan laajennustyyppeihin ja tehdä tyylipohja kuvitteellisen henkilön verkkosivustolle. Henkilö kirjoittaa blogia, ja jatkossa häntä kutsutaan häntä bloggaajaksi. Oman blogin kirjoittamisen lisäksi bloggaaja tuottaa sisältöjä monille muille sosiaalisen median kanaville. Hän haluaa *Twitter*-palvelun olevan sivustolla hyvin esillä, koska hän käyttää sitä paljon. Lisäksi hän harrastaa valokuvausta ja siirtää ottamiansa kuvia *Flickr*-kuvasivustolle. Kirjanmerkit hän tallentaa *Delicio.us*-kirjanmerkkipalveluun. Bloggaaja kuuntelee myös paljon musiikkia ja jakaa kuuntelukokemuksensa *Last.fm*-sivuston kautta koko maailmalle. Tyylipohja rakennetaan niin, että sivusto toimii blogialustana ja kokoaa edellämainittujen sosiaalisten medioiden sisältöjä. Valmis tyylipohja laitetaan asennusohjeineen ladattavaksi *Joomlaportal.fi*-sivuston julkiseen tyylipohjakirjastoon.

2 JOOMLAN RAKENNE

Tässä luvussa esitellään Joomlan PHP-pohjainen ohjelmistokehys, minkä jälkeen tarkastellaan Joomlan rakennetta kolmesta eri näkökulmasta. Ensin tutkitaan, mistä Joomla-sivuston sisältö koostuu ja miten sitä jaotellaan. Sen jälkeen tutustutaan käyttöliittymiin, joiden kautta sivuston sisältöä voi tarkastella. Viimeiseksi esitellään Joomla-asennuspaketin tiedostojen ja kansioiden rakenne palvelimella.

2.1 Joomla-ohjelmistokehys

Ohjelmistokehykset ovat ohjelmistojen tekemistä nopeuttavia apuvälineitä. Ohjelmistokehys on ohjelmistojen uudelleenkäytettävä malli, ikään kuin runko, jonka päälle tietokoneohjelma rakennetaan. Samaan runkoon perustuen voidaan tehdä toisistaan paljonkin poikkeavia sovelluksia. Kehyksessä on valmiita ohjelman osia, joita ei tarvitse kirjoittaa uudelleen. Niiden toimintoja voidaan hyödyntää, kun ohjelmoidaan uusia erikoistuneita toimintoja. Joomlan ohjelmistokehys on oliopohjainen. Sen valmiit osat ovat luokkia, joiden ilmentymien yhteistyö muodostaa julkaisujärjestelmän. Joomlan perusominaisuudet riittävät Internet-sivuston rakentamiseen, minkä lisäksi kehys tarjoaa joustavia mahdollisuuksia sekä oletustoimintojen muokkaamiseen että täysin yksilöllisiin toteutuksiin.

Joomla on kolmetasoinen järjestelmä [1]. Päällimmäinen kerros on laajennuskerros [kuva 1]. Se sisältää ohjelmistokehyksen laajennuksia: moduuleita, komponentteja ja tyylipohjia. Keskimmäisenä on sovelluskerros, joka koostuu ohjelmistokehyksen *JApplication*-nimisestä luokasta periytyvistä luokista. Tällä hetkellä niitä on neljä. *JInstallation* vastaa Joomlan asentamisprosessista palvelimelle, ja se poistetaan asennuksen jälkeen. *JAdministrator*-luokka huolehtii sivuston ylläpitoon liittyvistä toiminnoista. *JSite* puolestaan vastaa sivuston sellaisista toiminnallisuuksista, jotka ovat tavalliselle kävijälle näkyviä. *XML-RPC*-luokka tarjoaa palveluita Joomla-sivuston ylläpitoon etäyhteyttä käyttäen. Alimmalla tasolla, ohjelmistokehyskerroksessa, ovat tarvittavat kirjastot, itse ohjelmistokehys ja sen luokat sekä ohjelmistokehyksen toiminnallisuuksia laajentavat liitännäiset.

Kuva 1. Joomla on kolmetasoinen järjestelmä.

2.2 Sisältö

Joomlassa sisältö tallennetaan MySQL-tietokantaan, josta sitä pyydettäessä haetaan. Sisältö järjestetään kolmetasoisesti artikkeleihin, alaryhmiin ja pääryhmiin [2, s. 68].

Artikkeli

Artikkeli on tekstimuotoista tietoa, ja se voi sisältää myös vaikkapa kuvia ja videoita. Artikkelit muodostavat suurimman osan siitä sisällöstä, mitä sivustolla näytetään. Artikkelin sisältö on täysin erillistä suhteessa siihen, miltä se sivustolla näyttää. Samaa sisältöä voidaan esittää useissa eri muodoissa ja paikoissa ja artikkeleita voidaan näyttää yhdessä tai erikseen.

Pää- ja alaryhmät

Pää- ja alaryhmät ovat tapa järjestellä artikkeleita. Pääryhmä (section) koostuu yhdestä tai useammasta alaryhmästä (category). Artikkeleita voidaan määritellä kuuluvaksi alaryhmään. Artikkeli voi kuulua vain yhteen alaryhmään, ja alaryhmä voi kuulua vain yhteen pääryhmään. Artikkeleiden järjestelemisessä Joomla toteuttaa siis hierarkkista kolmetasoista puurakennetta. Ryhmittelystä on kahdenlaista hyötyä. Ensimmäiseksi Joomlan julkisen liittymän valikosta voidaan tehdä linkki yksittäisen artikkelin lisäksi pää- tai alaryhmään. Tällöin linkin osoittamalle sivulle listataan kyseisen ryhmän kaikki artikkelit halutussa järjestyksessä. Kun ryhmään myöhemmin lisätään artikkeleita, ne näkyvät valikkokohteen osoittamalla sivulla automaattisesti. Toiseksi artikkeleita voidaan ylläpitoliittymässä nopeasti selata ja lajitella ryhmien mukaan. Näin niiden päivittäminen ja lisääminen helpottuu, etenkin jos artikkeleita on useita satoja.

2.3 Liittymät

Joomla jakautuu kahteen liittymään. Julkinen liittymä (frontend) on sivuston se puoli, jonka sivustolla kävijä havaitsee ja jota hän käyttää. Ylläpitoliittymä (backend) on tarkoitettu sivuston hallintaan ja sisältöjen ylläpitämiseen.

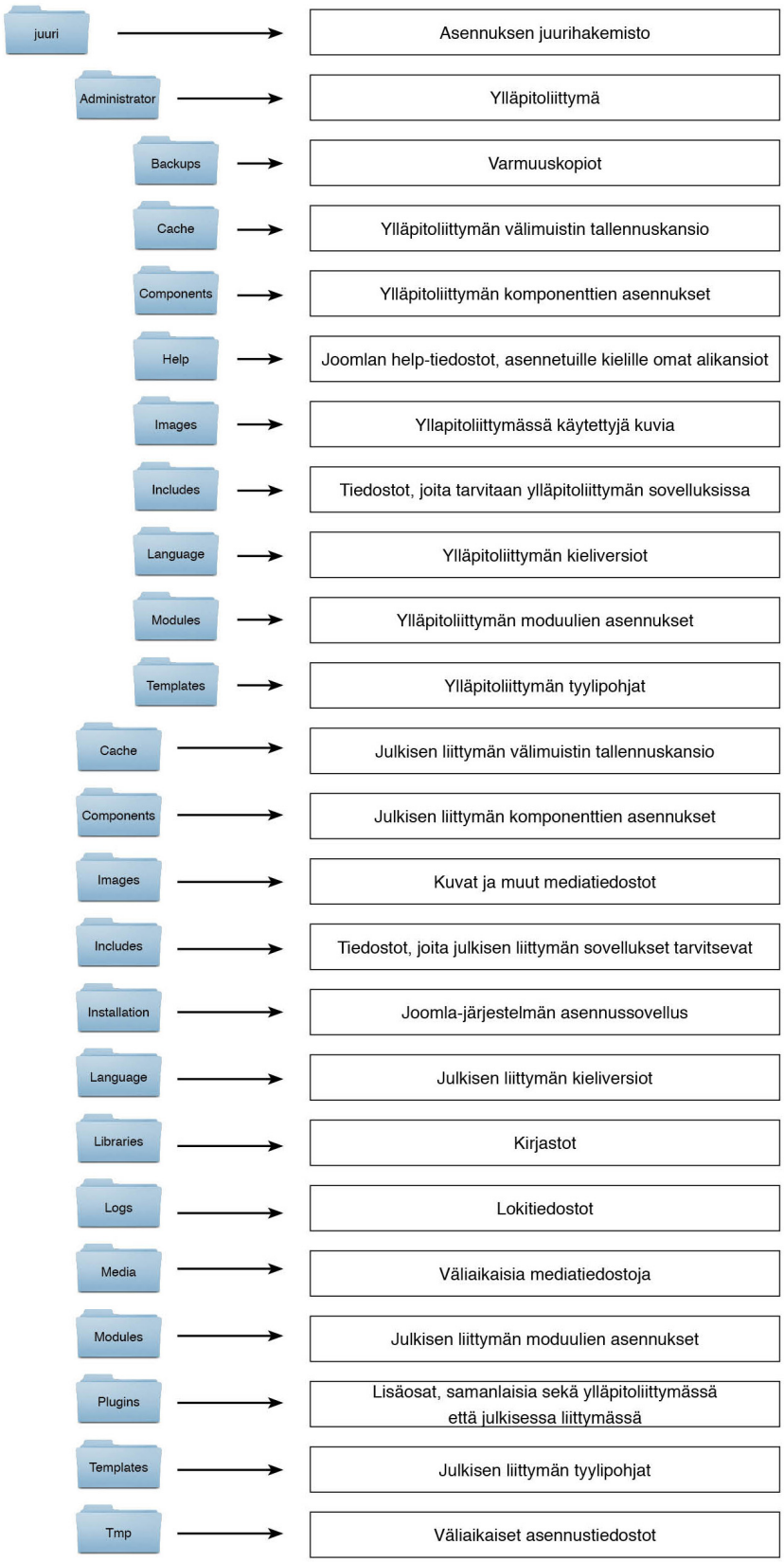
Julkisen liittymän kautta sisällöt voivat nimensä vastaisesti olla nähtävissä vain tietyille käyttäjille onnistuneen sivustolle kirjautumisen jälkeen. Julkisen liittymän kautta tietyt käyttäjät voivat myös luoda ja ylläpitää sisältöjä. Joomlalla on siis mahdollista luoda esimerkiksi täysiverinen verkkoyhteisöalusta tai yrityksen intranet. Käyttäjien oikeuksien hallintaa ei käsitellä tässä opinnäytetyössä tämän enempää.

Ylläpitäjä hallinnoi sivuston sisältöjä ylläpitoliittymän kautta. Artikkeleita sekä pää- ja alaryhmiä voi sieltä lisätä, poistaa ja muokata. Ylläpitoliittymästä asennetaan ja hallitaan myös laajennuksia.

2.4 Tiedostot

Joomla-perusasennuspaketti sisältää kuvan 2 mukaiset kansiot. Ylläpitoliittymä sijaitsee *administrato*r-alihakemistossa ja julkinen liittymä asennuksen juuressa.

Tässä opinnäytetyössä tutustutaan julkisen liittymän *components*-, *modules*-, *plugins*- ja *templates*-hakemistojen sisältöihin. *Components*-kansiossa sijaisevat julkisen liittymän komponenttien asennukset, joista tutustumme *Content*-nimisen komponentin toimintaan. *Modules*-kansiossa ovat moduuliasennukset, minne työn aikana asennetaan kolmannen osapuolen kehittämä moduuli. Blogisivustolle asennetaan myös yksi liitännäinen *plugins*-hakemistoon.



Kuva 2. Joomla-asennuksen kansiorakenne.

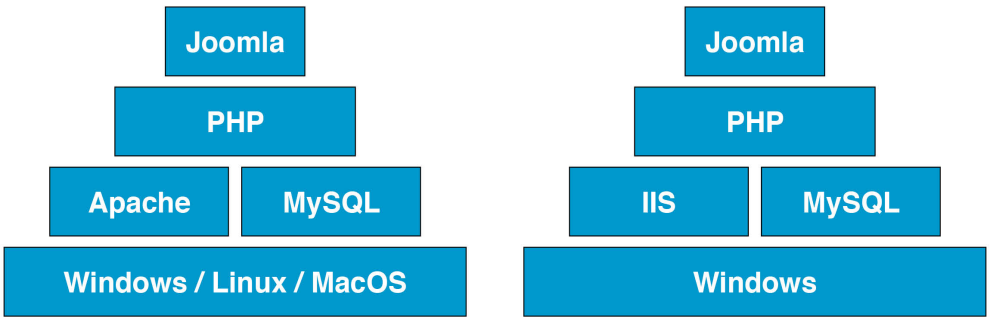
3 JOOMLAN KEHITYSYMPÄRISTÖN ASENNUS

Joomla-laajennusten kehittämiseen on tarjolla useita työkaluja ja kehitysympäristöjä. Useimmat niistä eivät ole ainoastaan Joomla-laajennusten tekemiseen tarkoitettuja, vaan ne tarjoavat myös yleisiä PHP-ohjelmoinnin työkaluja. Joomlan monikielisyyden tuki on toteutettu käyttäen yleisten kielten kaikki merkit sisältävää UTF-8-muotoista merkistökoodausta, joten myös kehitysympäristön on tuettava sitä [3, s. 34].

Joomla-sivuston palvelinympäristön kokoonpanossa on neljä toisiinsa vuorovaikutuksessa olevaa osaa. Järjestelmän ydin on HTTP-palveluita tarjoava verkkopalvelin. PHP-komentotulkki suorittaa Joomlan sovelluksia. PHP-käskyillä ohjataan MySQL-palvelinta, jonne kaikki Joomlan käsittelemä tieto tallennetaan. Vain Joomlan perusasetukset on tallennettu erilliseen *configuration.php*-tiedostoon. [4, s. 10.]

Joomla 1.5.17 -palvelinympäristön vähimmäisvaatimukset ovat

- PHP 4.3.x tai uudempi, jossa on MySQL-, XML- ja Zlib-moduulien oltava käynnissä
- MySQL 3.23.x tai uudempi
- Apache 1.13.19 tai uudempi. Windows-palvelimella myös Microsoft Internet Information Server (IIS) sopii tarkoitukseen.



Kuva 3. Joomla-palvelinympäristön tyypilliset kokoonpanovaihtoehdot.

Paikalliselle koneelle asennettiin XAMPP-palvelinympäristö. Se pitää sisällään Apache-palvelimen lisäksi MySQL-tietokantapalvelimen ja PHP:n. Ohjelmointiympäristöksi valittiin *PDT 2.1 SR-2 All In One Eclipse PHP Package* (IDE) ja siihen asennettiin Subclipse-versionhallintalisäosa (SVN). [5]

3.1 XAMPP

Avoimen lähdekoodin ohjelmistoihin perustuva ei-kaupalliseen käyttöön tarkoitettu XAMPP-asennuspaketti on ilmaiseksi tarjolla Windows-, Linux- ja MAC OS X -käyttöjärjestelmiin. Tätä opinnäytetyötä varten asennetussa *XAMPP Mac OS X 1.7.3* -versiossa on Apachen versio 2.2.14, MySQL-versio 5.1.4.4 ja PHP:n 5.3.1-versio. Paketti on yleisen GNU-lisessin alainen, joten sitä voi kopioida, muuttaa ja jakaa edelleen. XAMPP-asennuspaketin voi ladata Apache Friendsin sivuilta [6].

On huomattava, että XAMPP:in oletusasennus ei turvallisuussyistä sovellu tuotantoympäristöihin. Kehitysympäristössä avoimet asetukset ovat etu. Kehittäjä voi tehdä palvelimella lähes mitä haluaa, mutta tuotantoympäristössä ne ovat valtava tietoturvariski. XAMPP-oletusasennuksen tietoturvariskit ovat

- MySQL-ylläpitäjällä (root) ei ole salasanaa
- MySQL-taustaprosessiin on pääsy verkon kautta
- ProFTPD:ssa on käyttäjänimenä ”*nobody*” ja salasanana "*xampp*"
- PhpMyAdmin -sovellukseen päästään verkon kautta
- MySQL ja Apache toimivat samalla käyttäjätunnuksella.

Suurin osa edellä mainituista turvallisuusepäkohdista korjataan ajamalla XAMPP:in oma tietoturva-asetusten määrittelyvelho. Tämän opinnäytetyön kehitysympäristöön asennettu XAMPP toimii kuitenkin oletusasetuksilla.

Asennusvaiheessa XAMPP määrittelee palvelimen juurihakemistoksi *htdocs*-nimisen paikallisen hakemiston. Joomla-asennusten vaatimat tiedostot sijoitettiin sinne. Kun XAMPP on käynnistänyt Apache- ja MySQL-palvelimet, toimii Internet-selaimen osoitekenttään syötetty osoite *http://localhost* paikallisten verkkosivujen juurena.

3.2 Eclipse

Eclipse on avoimeen lähdekoodiin perustuva ohjelmointiympäristö (Integrated Development Environment, IDE). Eclipsen PHP-kehitysversioon haettiin Joomlan lähdekoodi (Joomla repositories) [7]. Eclipse sijoitti lähdekooditiedostot palvelimen juurihakemistoon *Joomla 1.5 Source* -nimiseen alikansioon.

Eclipseen asennettiin Subclipse-versionhallintalisäosa (Subversion, SVN) helpottamaan ohjelmiston versioiden hallintaa. Versionhallinnalla voidaan palata muokatun koodin aiempaan versioon. Tarpeen vaatiessa koko projektin lähdekoodista voidaan suorittaa sekä ohjelmiston vanhempi että uudempi versio ja vertailla niitä. Muokatusta koodista voidaan näin etsiä ja poistaa virheitä. Koska versionhallinta toimii palvelimella, voidaan sillä myös koordinoida projektia ja muutoksia silloin, kun useampi henkilö kehittää samaa ohjelmistoa eri paikoista eri aikoina. Tässä opinnäytetyössä versionhallintaa käytetään kuitenkin vain paikallisella koneella. [4, s. 138, s. 141.]

Testiympäristö

Kehitysympäristöasennuksen lisäksi tarvitaan myös testiympäristö. Joomlan identtinen asennuspaketti sijoitettiin paikallisen palvelimen juurihakemiston *test*-alikansioon, ja järjestelmä asennettiin ajamalla asennusvelho Internet-selaimen kautta [2, s. 28]. Testiympäristö toimii selaimella paikallisella palvelimella osoitteessa *http://localhost/test.* Kehitysympäristössä muokataan Joomla-laajennuksia Eclipsen koodinmuokkaustoiminnoilla. Vaikka kehitysympäristössä pystyy ajamaan Joomla-sivustoa, on tärkeää, että muokatut laajennukset pakataan zip-tiedostomuotoisiksi asennustiedostoiksi ja ne asennetaan testiympäristöön [2, s. 56]. Näin sekä laajennusten asennusprosessi että toiminta voidaan varmistaa ennen niiden jakelua ja asentamista tuotantoympäristöön.

4 TUOTANTOYMPÄRISTÖN ASENNUS

Joomlan peruspaketin viimeisimmät asennustiedostot ovat ladattavissa Joomla.org-sivustolta [8]. Tiedostopaketti puretaan paikalliselle koneelle ja tiedostot siirretään julkiselle palvelimelle. Kun verkkosivusto avataan selaimella ensimmäistä kertaa onnistuneen siirron jälkeen, avautuu Joomlan asennusvelho. Sen ohjeita noudattaen määritellään asennuksen tietokanta-asetukset ja pääkäyttäjän tiedot sekä asennetaan tarvittaessa esimerkkiartikkelit [9]. Onnistuneiden määritysten jälkeen on poistettava *installation*-kansio asennuksen juurihakemistosta.

Perusasennuksen jälkeen asennetaan kaikki tarvittavat erityyppiset laajennukset ylläpitoliittymän laajennustenhallintatyökalulla (Extension Manager), joka löytyy *Extensions*-valikon kohdasta *Install/Uninstall*. Laajennustenhallin-

tatyökalulla tehdään ainoastaan asennuksiin ja asennusten poistoihin liittyvät operaatiot. Jokaisella laajennustyypillä ja kieliversioilla on sen lisäksi oma hallintatyökalunsa: moduulienhallintatyökalu (Module Manager), liitännäistenhallintatyökalu (Plugin Manager), tyylipohjienhallintatyökalu (Template Manager) ja kieliversioidenhallintatyökalu (Language Manager). Niillä määritellään kunkin yksittäisen laajennuksen yleisiä asetuksia.

Tässä opinnäytetyössä käytetään tuotantoympäristön esimerkkinä *http://someblogger.onnidesign.net* -sivustoa, johon asennettiin Joomlan 1.5.17 -versio. Sivustolle ei asennettu esimerkkisisältöä. Ylläpitoliittymän pääryhmienhallintatyökalulla (Section Manager) tehtiin blogimerkintöjä varten pääryhmä nimeltä *blogi,* ja siihen alaryhmienhallintatyökalulla (Category Manager) ryhmät *valokuvaus* ja *musiikki*. Ryhmiin kirjoitettiin muutama esimerkkiartikkeli ylläpitoliittymän artikkeleidenhallintatyökalulla (Article Manager) [10].

4.1 Monikielisyys

Joomlan ohjelmistokehyksessä on hyvä monikielisyyden tuki. Se on käytännössä toteutettu erillisillä *ini*-muotoisilla kielitiedostoilla. Joomlan perusasennuksen laaja kielitiedostopaketti on ladattavissa *Joomlaportal.fi*-sivustolta, ja se asennetaan laajennustenhallintatyökalulla. Asennuksen jälkeen asetetaan kieliversioidenhallintatyökalulla järjestelmän oletuskieleksi suomi. Järjestelmään voidaan asentaa useita kieliä ja järjestelmän kieli voidaan määritellä myös käyttäjäkohtaisesti.

Asennetut kielitiedostot sijaitsevat Joomlan asennuksen juuren *language*-hakemistossa kielen tunnuksen mukaisesti nimetyssä alikansiossa. Järjestelmän yleiset käännökset sijaitsevat tiedostossa, joka on myöskin nimetty kielitunnuksen mukaan. Esimerkiksi järjestelmän suomenkieliset yleistermit ovat *language/fi-FI*-kansiossa sijaitsevassa *fi-FI.ini*-tiedostossa. Laajennusten käyttämät kielitiedostot nimetään laajennuksen tyypin ja nimen mukaan. Tässä opinnäytetyössä esimerkkinä käytetty *com_content*-nimisen komponentin suomenkielinen käännös on *fi-FI.com_content.ini*-tiedostossa. Luvussa 7 esiteltävän RokTwittie-moduulin käännös tehtiin *fi-FI.mod_roktwittie.ini*-tiedostoon. Myös tyylipohjat ja liitännäiset voivat käyttää kielitiedostoja. Niiden suomenkieliset versiot nimetään noudattaen vastaavasti käytäntöä *fi-FI.plg_liitännäisennimi.ini* ja *fi-FI.tpl_tyylipohjannimi.ini.*

Kielitiedostoja voidaan muokata käyttäen mitä tahansa UTF-8-merkistökoodausta tukevaa tekstieditoria.

Jos julkisen liittymän kieleksi on kieliversioidenhallintatyökalussa valittu suomi, hakevat laajennukset edellä mainitun mukaan nimettyjä ja sijoitettuja suomenkielisiä omia kielitiedostojaan. Jos suomenkielistä kielitiedostoa ei löydy, käytetään laajennuksen yhteydessä asentunutta kielitiedostoa. Laajennusten koodiin kirjoitetaan monikielisen termin kohdalle PHP-pätkä, joka hyödyntää Joomla-kirjaston *JText*-luokan tarjoamaa palvelua. Esimerkiksi RokTwittie-moduulin *default.php*-asemointimallissa on komento *<?php echo JText::_('ABOUT'); ?>*. Siinä kutsutaan *JText*-luokan _()-metodia ja välitetään sille parametrina teksti *ABOUT*. Metodi yrittää löytää parametrille kieliasetuksissa määritellyn kielen mukaisen vastineen käyttäen *JLanguage*-luokan *_()*-metodia [3, s. 417]. Jos kieli on suomi, luetaan *fi-FI.mod-.roktwittie.ini* -tiedosta rivi *ABOUT = noin*, ja palautetaan sana ”*noin*”. Jos tiedostoa tai riviä ei löydy, haetaan RokTwittien asennuksen mukana tullutta *en-GB.roktwittie.ini*-tiedostoa ja siitä vastaava *ABOUT*-avain ja palautetaan sen arvo, eli tässä tapauksessa ”*about*”. Jos sitäkään ei syystä tai toisesta löydy, palautetaan parametri sellaisenaan, eli esimerkissämme ”*ABOUT*” [3, s. 415]. Seuraavassa on esimerkki RokTwittien suomennetusta kielitiedostosta.

```
LOADING = Ladataan...
VIEWTWEET = katso Twitterissä
FROM = Lähettäjä
LESS_THAN_A_MIN = vajaa minuutti sitten
ABOUT = noin
ABOUT_A_MIN_AGO = noin minuutti sitten
MINS_AGO = minuuttia sitten
ABOUT_A_HOUR_AGO = noin tunti sitten
HOURS_AGO = tuntia sitten
ONE_DAY_AGO = eilen
```

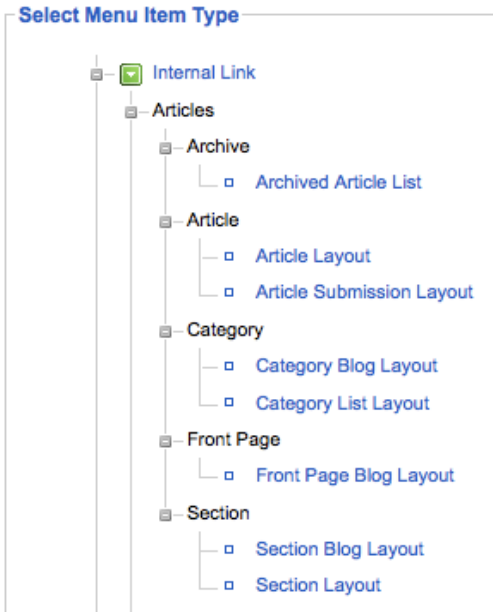
4.2 Valikot

Joomlan navigaatiojärjestelmä on erotettu artikkeleista [11, s. 8]. Sivustonavigaation lisäksi valikoilla ohjataan mitä näytetään, miten näytetään ja missä näytetään. Myös käytettävä tyylipohja voidaan määritellä valikkokohtaisesti. Tämän vuoksi valikoiden suunnittelu ja tekeminen ovat olennainen osa järjestelmän asentamista.

Valikoita ylläpidetään ylläpitoliittymän valikoidenhallintatyökalulla (Menu Manager). Vaikka esimerkkisivulle ei asennettu mitään sisältöä, muodostui järjestelmään yksi valikko, päävalikko (Main Menu), ja siihen yksi valikkokohde,

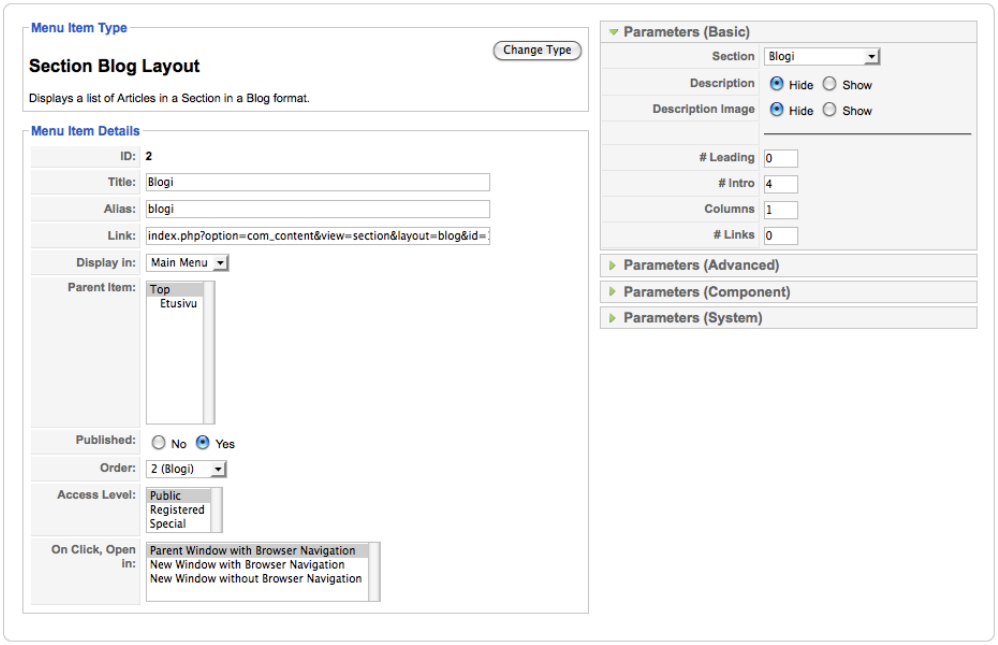
etusivu (Home). Järjestelmään voi tehdä useita valikoita ja jokaiseen valikkoon voi tehdä useita puuhierarkkisia valikkokohteita. Jokaisella valikolla on ainakin yksi esiintymä, moduuli. Valikot ja valikkokohteet ovat tietokannassa sijaitsevaa sisältöä, ja moduuli on paikka, jossa se näytetään.

Valikkokohde voi osoittaa mihin tahansa verkko-osoitteeseen, mutta sivuston sisällä se osoittaa järjestelmään asennettuun komponenttiin tai sen sisäiseen kohteeseen. Sisältökomponenttiin, eli tarkemmin sanottuna artikkeleihin osoitettaessa on ensin määriteltävä, missä näkymässä artikkelit halutaan tulostaa sivulle [kuva 4]. Näytetäänkö vain yksi artikkeli vai useita samaan ryhmään kuuluvia artikkeleita? Vasta tämän jälkeen voidaan määritellä, mikä tai mitkä artikkelit näytetään. Näkymistä kerrotaan lisää luvussa 5.2.3.



Kuva 4. Valikkokohteella voidaan osoittaa artikkeliin tai artikkeleihin monin eri tavoin.

Kuvassa 5 nähdään bloggaajan sivujen päävalikon kohde ”*Blogi*”. Jokaisella valikkokohteella on yksilöllinen tunniste, *ItemID*, eikä sitä voi muuttaa. Valikkokohteelle annettu nimi, *title*, näkyy valikossa. *Alias*-kenttään syötetty teksti näkyy käyttäjäystävällisissä URL-osoitteissa valikkokohteen nimen tilalla. Parametreissa määritellään tarkemmin, miten artikkelit ladotaan sivulle. Tässä tapauksessa artikkeleita ladotaan neljä [kuva 5, kohta *#Intro*], ja ne asetellaan yhdelle palstalle [kuva 5, kohta *Columns*]. Edistyneistä asetuksista [kuva 5, kohta *Parameters (Advanced)*] voidaan määritellä, missä järjestyksessä artikkelit ladotaan. Oletusarvoisesti ne asetellaan julkaisupäivämäärän mukaisessa laskevassa järjestyksessä blogille tyypilliseen tapaan.



Kuva 5. Valikkokohteen asetuksia.

Esimerkkisivuston päävalikkoon tehtiin valikoidenhallintatyökalulla kaikkiaan neljä valikkokohdetta: Etusivu, Blogi, Valokuvaus ja Musiikki. Ne laitettiin osoittamaan vastaavasti sivuston etusivulle, blogi-pääryhmän alaryhmien sisältämiin artikkeleihin, valokuvaus-alaryhmän artikkeleihin ja musiikki-alaryhmän artikkeleihin. Artikkelit määriteltiin näkymään *Category Blog Layout* -määrityksen mukaisesti. Se tarkoittaa, että kyseisen ryhmän artikkelit listataan allekkain kunkin artikkelin *lue lisää* -merkintään saakka, jos sellainen on luontivaiheessa artikkeliin tehty.

Kun sivuston valikot ja valikkokohteet on tehty, on jokaisesta valikosta muodostunut yksi esiintymä, moduuli. Niiden asetuksia määritellään moduulienhallintatyökalulla. Siellä voidaan valita se tai ne valikkokohteet, joita klikatessa moduuli on ylipäätään näkyvillä. Käytettävä tyylipohja voidaan määritellä samalla tavalla tyylipohjienhallintatyökalulla valikkokohteen mukaan. Näin esimerkiksi tietty sivuston osio voidaan suunnitella ulkonäöltään ja asemoinniltaan täysin muusta sivustosta poikkeavaksi. Toisin sanoen moduulin näkyvyys ja tyylipohjat sidotaan valikkokohteiden *ItemID*-tunnisteisiin. [2, s. 109-134].

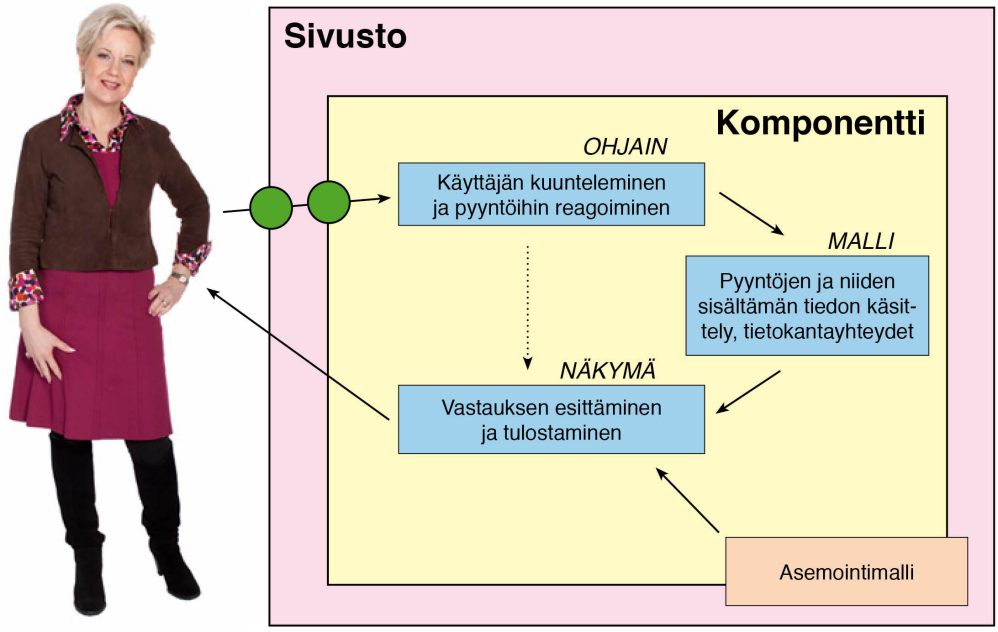
5 TOTEUTETTAVAT KOMPONENTTIMUUTOKSET

Kuten aiemmin jo todettiin, komponentit ovat Joomlan olennaisin osa. Aina kun käyttäjä tekee Joomlassa jotain, kutsutaan komponenttia [3, s. 10].

Komponentit ova monimutkaisimpia laajennuksia. Komponentteja voidaan käyttää sekä julkisen että ylläpitoliittymän kautta. Ylläpitäjä määrittelee komponentin asetuksia ylläpitoliittymästä, ja sivustolla vierailija käyttää komponentin tarjoamia palveluita julkisesta liittymästä. On myös pelkästään ylläpitoon tarkoitettuja komponentteja, kuten pää- ja alaryhmien hallintakomponentit, joissa ei ole julkisen liittymän toimintoja.

5.1 MVC-arkkitehtuuri

Komponenttisuunnittelussa pyritään toteuttamaan kuvassa 6 havainnollistettua MVC-arkkitehtuuria. Siinä komponentin toiminta on jaettu kolmeen vastuualueeseen: käyttäjän pyyntöjen kuuntelemiseen ja niihin reagoimiseen, pyyntöön liittyvien tietojen käsittelemiseen ja vastaukseen liittyvien tietojen näyttämiseen. Ohjain (controller) vastaanottaa käyttäjältä tulevia käskyjä ja käsittelee tapahtumia. Tapahtumien kuuntelemisessa komponentti hyödyntää usein lisäosien tarjoamia palveluita. Malli (model) on komponentin loogi-nen tai suorittava koodi. Se myös huolehtii tarvittavista tietokantayhteyksistä. Näkymä (view) vastaa komponentin tuottaman datan esittämisestä halutunlaisessa muodossa. Jos MVC-arkkitehtuuria havainnollistetaan yleisellä esimerkillä, voisi ohjain olla esimerkiksi sivulla oleva hakutoiminto, jolla käyttäjä lähettää järjestelmään haluamansa hakutermin. Malli olisi tällöin palvelimella suoritettava haun käsittelevä PHP- ja tietokantakoodi. Näkymä olisi puolestaan sapluuna, jonka mukaan hakutulokset asemoidaan tulossivulle. Näkymä käyttää asemoinnissa ja ulkonäön rakentamisessa apunaan asemointimalleja. [4, s. 186].



Kuva 6. MVC-arkkitehtuurissa komponentin toiminta jaetaan kolmeen vastuualueeseen.

Jokainen MVC-arkkitehtuurin mukaan suunnitellun komponentin kolmesta osasta ovat itsenäisiä sovelluksen palasia, joita voidaan muokata ja kehittää toisistaan erillisinä. Yhteen osaan tehdyt muutokset eivät vaikuta kahden muun osa-alueen toimintoihin. Tämä helpottaa komponentin kehittämistä ja ylläpitämistä. [3, s. 68.]

5.2 Toiminta ja rakenne

Tässä luvussa tutustutaan *com_content*-nimisen komponentin rakenteeseen ja toimintaan. Sitä havainnollistetaan niiltä osin, joilta sen toimintoja muutetaan tyylipohjaa tehdessä. Komponentista käytetään jatkossa nimitystä sisältökomponentti.

5.2.1 Tulopiste ja komponentin kutsuminen

Molemmissa Joomla-järjestelmän liittymissä on yksi piste, jonka kautta niihin otetaan yhteyttä [12]. Julkisen liittymän sovelluksissa se on juurihakemiston *index.php*-tiedosto ja ylläpitoliittymän sovelluksissa *administrator*-kansiossa sijaitseva *index.php*-tiedosto. Käyttäjä pystyy lähestymään järjestelmää vain näiden pisteiden kautta. Kun käyttäjä on lähettänyt kutsun Joomlaan, lähestyy Joomla puolestaan tarvittavia komponentteja niiden omien tulopisteiden kautta. Komponentin tulopiste on nimetty samoin kuin komponentti itse, eli sisältökomponentin tulopiste on *content.php*. Käyttäjällä ei siis ole suoraa

yhteyttä komponentteihin. Julkisen liittymän ja komponentin tulopisteet on merkitty kuvaan 6 vihreillä palloilla.

Joomla käyttää *_JEXEC*-nimistä vakiota tulopisteiden kontrolloimisessa. Järjestelmä asettaa sen arvoksi 1, kun se kutsuu komponenttia. Pahaaikeinen käyttäjä voi kutsua komponenttia suoraan kirjoittamalla komponentin tulopisteen osoitteen Internet-selaimen osoitekenttään. Tällöin *_JEXEC* ei saa arvoa ollenkaan tai sen arvo on negatiivinen. Koodin suorittamisessa edetään vain, jos *_JEXEC*-vakiolla on arvo ja se on positiivinen. Tarkistuksen suorittava komento kirjoitetaan aina komponentin tulopisteen koodin alkuun. Ilman sitä kuka tahansa voi murtautua sisään järjestelmään. Tarkistuskoodi kirjoitetaan käytännössä kaikkien Joomla-laajennusten tulopisteiden koodin ensimmäiselle riville. [13], [3, s. 32.]

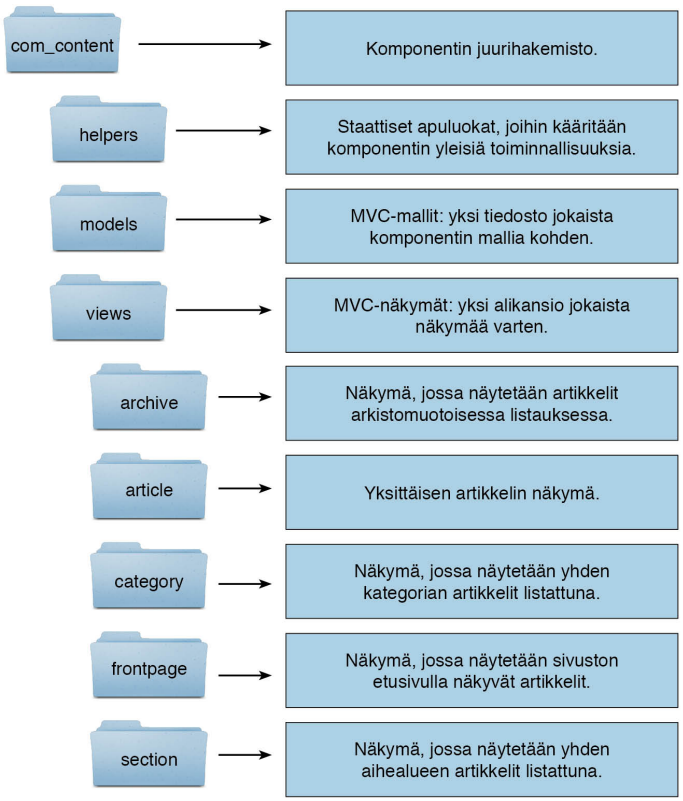
Käytetään esimerkkinä sivustolla sijaitsevaa linkkiä, joka on määritelty osoittamaan osoitteeseen *index.php?option=com_content.* Sivuston julkisen liittymän tulopiste on *index.php,* ja käyttäjä voi sitä kautta lähestyä järjestelmää. Osoitteessa välitetään *option*-avaimen arvona *com_content*, jonka avulla Joomla kutsuu sisältökomponenttia. Sisältökomponentin tulopiste on siis *content.php*, ja Joomlalla on oikeudet kutsua sitä.

5.2.2 Sisältökomponentin kansiorakenne

Ylläpitoliittymän ja julkisen liittymän komponenttien kansiorakenteet ovat hyvin samanlaisia. Ylläpitoliittymän komponenttiin kuuluvat julkisen liittymän komponentin kansioiden lisäksi *elements*-, *help*- ja *tables*-kansiot. *Elements*-kansiossa on luokkia, joilla tehdään komponentin ylläpitoon liittyviä erikoislomekekenttiä, *help*-kansiossa on ylläpitäjälle hyödyllisiä ohjetiedostoja sekä niiden kieliversioita, ja *tables*-kansiossa on tietokantayhteyksiin tarvittavia luokkia. Julkisen liittymän komponenttien asennuskansiot sijaitsevat Joomlan kansiorakenteen juuressa sijaitsevassa *components*-kansiossa ja ylläpitoliittymän vastaavat kansiot *administrator*-kansion *components*-nimisessä alihakemistossa [kuva 2]. Kansiot nimetään komponentin nimen mukaan niin, että komponentin nimeen lisätään *com_*-etuliite.

Seuraavana on havainnekuva julkisen liittymän sisältökomponentin kansiorakenteesta. Sisältökomponentti on Joomlan eräs olennaisimmista komponenteista. Se käsittelee Joomla-sivun sisältöartikkeleihin liittyviä julkisen

liittymän kautta tapahtuvia pyyntöjä, muokkauksia ja artikkeleiden näyttämistä.



Kuva 7. Sisältökomponentin kansiorakenne

Sisältökomponentin juurihakemistossa sijaitsevat seuraavat tiedostot. Listauksessa selitetään pääpiirteittäin kunkin tiedoston tarkoitus.

- *Content.php* on komponentin tulopiste. Jokaisen komponentin tulopiste on nimetty samoin kuin komponentti itse.
- *Controller.php* on komponentin ohjain. Jos ohjaimia on useampia, sijaitsevat ne controllers-kansiossa. Tässä tapauksessa ohjaimia on vain yksi, jolloin se voidaan sijoittaa komponentin juurihakemistoon.
- *Index.html* estää käyttäjiltä pääsyn selaamaan kansion sisältöä. Käytännössä tiedosto on tyhjä html-sivu, ja vastaava tiedosto sijaitsee jokaisessa Joomlan alikansiossa [3, s. 68].
- *Metadata.xml*-tiedostoa tarvitaan komponentin asennuksessa. Sitä kutsutaan nimellä manifestitiedosto ja siinä kuvataan komponentin tiedosto- ja kansiorakenne [3, s. 103].
- *Router.php* vastaanottaa osoiterivillä välitettyjä parametreja ja ohjaa komponentin toimintoja niiden perusteella [3, s. 101].

- *View.php* sisältää *ViewContent*-luokan, jonka kaikki sisältökomponentin näkymien luokat perivät. *ViewContent*-luokka puolestaan perii Joomlan kirjaston */libraries/joomla/application/component*-kansion *view.php*-tiedostossa sijaitsevan *JView*-luokan, jossa ovat näkymien ydintoiminnot.

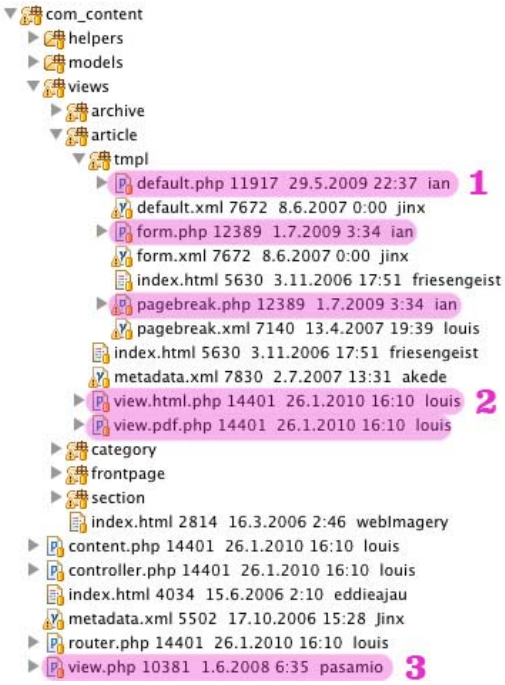
5.2.3 Näkymät ja niiden asemointimallit

Sisältökomponentilla on *views*-hakemiston alikansioissa viisi erilaista näkymää seuraavasti nimettynä: *article*, *archive*, *category*, *frontpage* ja *section*. *Article*-näkymää käytetään tulostettaessa yksittäinen artikkeli kokonaisuudessaan. *Archive*-näkymällä puolestaan muodostetaan lista joko pää- tai aliryhmän kaikkien artikkeleiden otsikoista. Otsikot ovat linkkejä, ja niitä klikkaamalla näytetään kyseisen artikkelin *article*-näkymä. *Category*-näkymässä listataan ryhmän artikkelit ja vastaavasti *section*-näkymässä pääryhmän sisältämät artikkelit halutussa järjestyksessä. *Frontpage*-näkymä tulostaa sivulle ne artikkelit, jotka on määritelty näkymään etusivulla.

Käytetään esimerkkinä sivulla olevaa linkkiä, joka tällä kertaa viittaa osoitteeseen *index.php?option=com_content&view=article&id=10.* Tulopisteen kautta välitetään Joomlalle pyyntö kutsua Sisältökomponenttia. Lisäksi *view*-avaimella välitetään arvo *article*, jonka perusteella komponentti näyttää pyydetyn artikkelin *article*-näkymässä. Esimerkissä välitetään *id*-avaimella myös tieto siitä, mikä nimenomainen artikkeli halutaan katsoa.

Jokaisen näkymän kansiossa on luokkatiedostot niiden dokumenttityyppien muodostamiseksi, mitä komponentti voi tulostaa näyttöruudulle [3, s. 75, s. 87]. Sisältökomponentti voi tulostaa *article*-näkymässä XHTML- ja PDF-dokumentteja. Niitä varten *article*-hakemiston *view*-kansiossa on *view.html.php*- ja *view.pdf.php*-luokkatiedostot. *View.html.php* käyttää XHTML-sivun generoimiseen *tmpl*-alikansiossa sijaitsevia lyhyistä PHP- ja XHTML-kielisistä pätkistä koostuvia asemointimalleja. *View.html.php*-luokalla on käytettävissään asemointimallit yksisivuisen artikkelin näyttämiseksi (*default.php*), monisivuisen artikkelin asemoimiseksi (*pagebreak.php*) sekä yksi- tai monisivuisen artikkelin muokkaamistilan asemoimiseksi (*form.php*). Kun kävijä napsauttaa tutun esimerkin mukaisesti kohteeseen *index.php?option=com_content&view=article&format=html&layout=default osoittavaa linkkiä,* hän välittää julkisen liittymän tulopisteen kautta sisältökomponentille pyynnön näyttää yksittäinen artikkeli XHTML-dokumenttityypisenä ja asetella se käyttäen *default.php*-asemointimallia. To-

sielämän tilanteessa osoitteessa välitetään luonnollisesti myös *id*-avaimen arvolla tieto, mikä artikkeli halutaan katsoa. Tässä kappaleessa mainittujen PHP-tiedostojen sijainti toisiinsa nähden selviää parhaiten kuvan 6 kansiorakenteesta. [3, s. 77.]



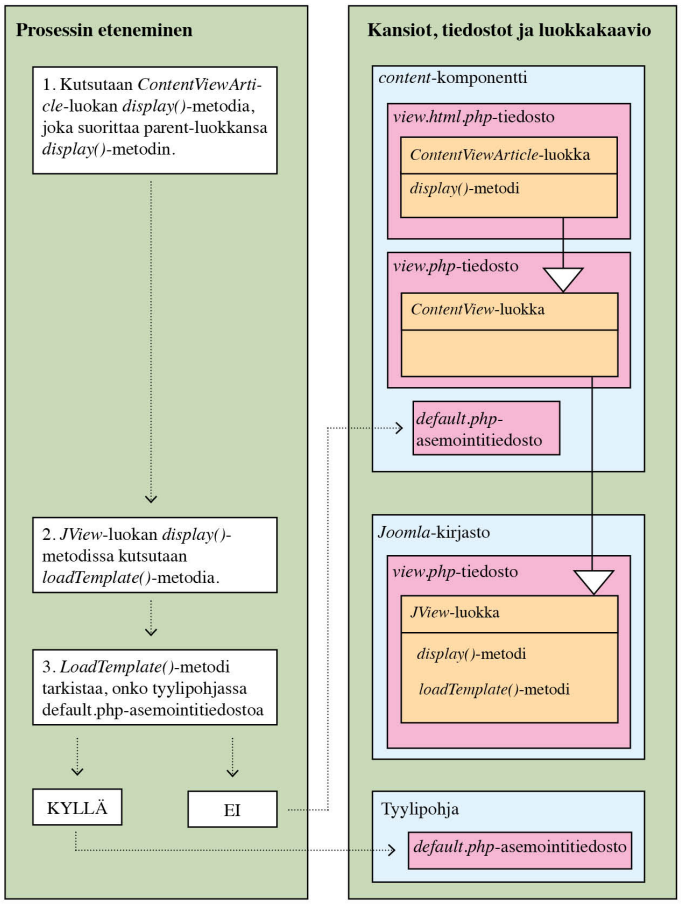
Kuva 8. Sisältökomponentin article-näkymän asemointimallikansio avattuna. Kuvassa on numeroituna ne tiedostot, joiden toimintaa esitetään luvussa 5.2.4.

5.2.4 Asemointimallien ylikirjoitus

Komponenttien asemointimallit voidaan ylikirjoittaa tekemällä tyylipohjakansioon vastaavat tiedostot [2, s. 293]. Esimerkiksi sisältökomponentin *article*-näkymän *default.php*-asemoinnin ylikirjoittava tiedosto sijoitetaan tyylipohjan tiedostokansioon *html/com_content/article* ja nimetään se vastaavasti *default.php*:ksi. Tyylipohjaa käsitellään tarkemmin luvussa 6. Tässä luvussa esitellään, miten Joomla valitsee näkymän yhteydessä käytettävän asemointimallin. [14]

Luvun 5.2.3 viimeisen kappaleen esimerkissä komponentille välitettiin osoiterivillä tieto, että sisältö halutaan katsoa article-näkymässä, XHTML-muodossa ja asetella se *default.php*-asemointimallin mukaisesti [kuva 8, kohta 1]. XHTML-muotoisen dokumentin muodostava *view.html.php*-tiedoston *ContentViewArticle*-luokka perii sisältökomponentin juurihakemiston *view.php*-tiedostossa sijaitsevan *ContentView*-luokan [kuva 8, kohdat 2 ja 3]. Kuten luvun 5.2.2 neljällä viimeisellä rivillä kerrotaan, *ContentView*-

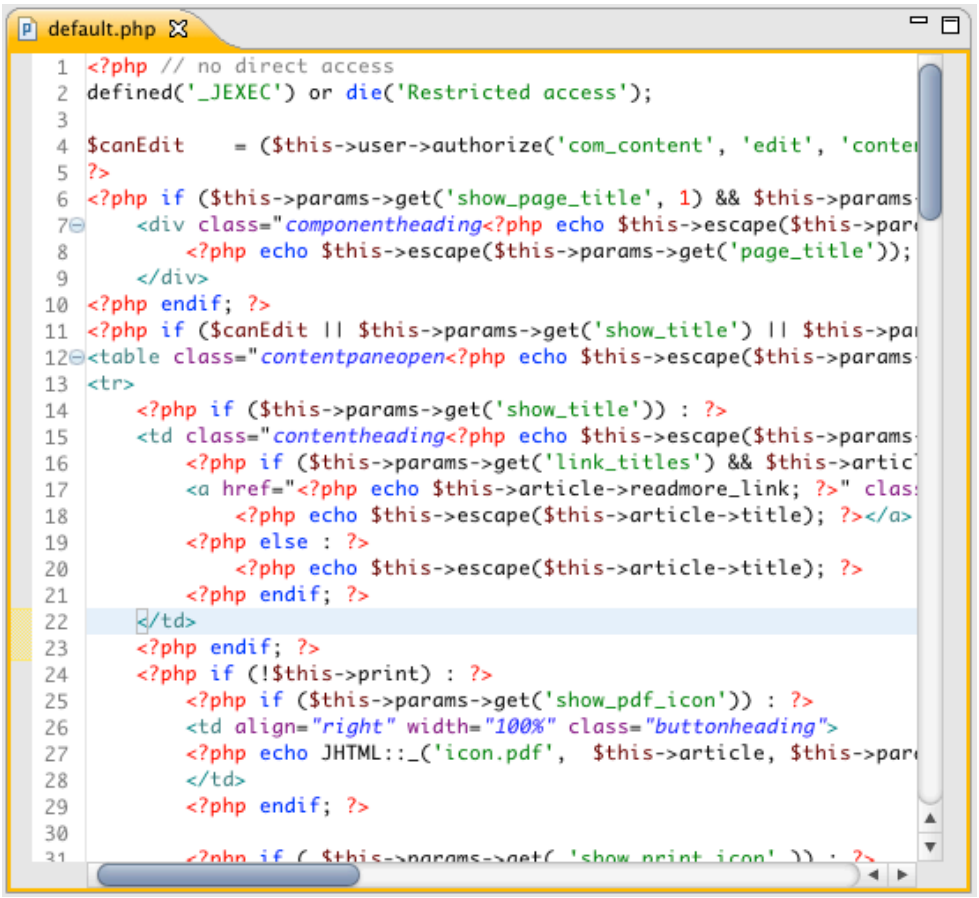
luokka puolestaan perii Joomlan kirjastosta *JView*-luokan. *ContentViewArticle*-luokka ylikirjoittaa *JView*-luokan *display()*-metodin. Oman *display()*-metodinsa lopussa se kuitenkin kutsuu *JView*-luokan *display()*-metodia [3, s. 77]. Se puolestaan suorittaa *loadTemplate()*-metodin, joka tarkistaa, löytyykö käytettävän tyylipohjan kansiosta *default.php*-asemointimallin ylikirjoittavaa tiedostoa. Jos se löytyy, asemoidaan sivu sen mukaan. Jos sellaista ei ole saatavilla, käytetään komponentin sisäistä asemointimallia. Perimisellä ja parent-luokan *display()*-metodia kutsumalla varmistetaan, että tarkistus tapahtuu, jolloin tyylipohjassa voidaan käyttää omia asemointimalleja. Sisältökomponentin *view.php*-, *view.html.php*- ja Joomlan kirjaston *view.php*-koodit ovat liitteessä 2, johon edellämainitut luokat ja metodit on kommentoitu. Tässä kappaleessa esitelty prosessi on havainnollistettu seuraavassa kuvassa.



Kuva 9. Havainnekaavio sisältökomponentin oletusasemointimallin mahdollisesta ylikirjoittamisesta.

Sisältökomponentin *article*-näkymän *default.php*-koodista nähdään, että XHTML-sivun generoimiseen käytetään XHTML-merkintöjen lomassa lyhyitä PHP-pätkiä (PHP snippets) [kuva 10]. Joomlan laajennusten kehittämisessä

on käytäntönä erottaa XHTML ja PHP-koodit toisistaan. PHP:lla ei siis kirjoiteta XTHML-merkintöjä PHP:n *echo*- tai *print*-käskyillä. Huomataan myös, että oletusarvoisesti Joomla käärii artikkelin sisällön taulukon sisään [kuva 10, rivi 12]. Yksittäinen artikkeli vaatii kuitenkin vain harvoin, jos koskaan, että se täytyisi taittaa käyttäen XHTML-taulukoita. Taulukoilla on tarkoituksenmukaista näyttää taulukkomuotoista sisältöä, mutta yksittäisen artikkelin XHTML-taittamisessa parempi vaihtoehto on kääriä sisältö rakenteellisesti *div*-elementtien sisään [15]. Siksi tässä työssä tehtävän tyylipohjan kaikkien sisältökomponentin näkymien taulukoita käyttävät asemointimallit korvataan rakenteellista div-taittoa käyttävillä asemointimalleilla.



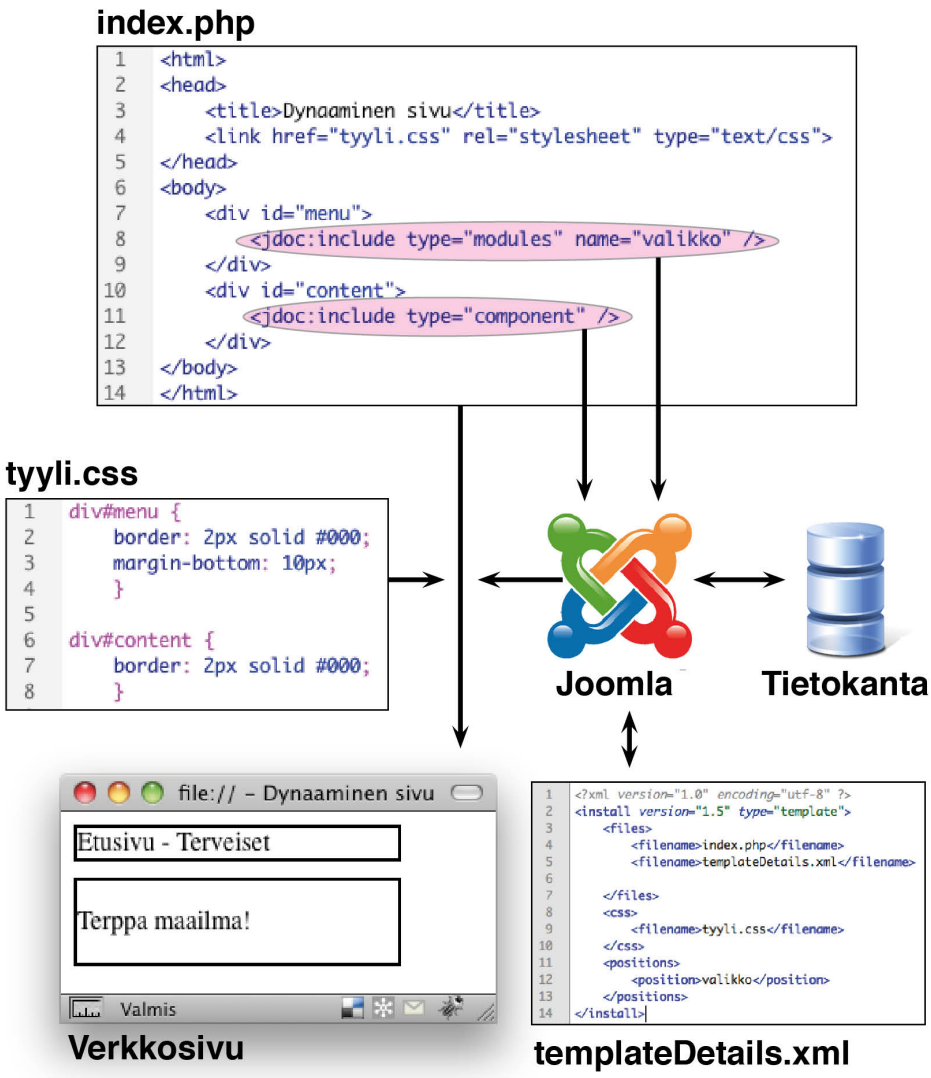
Kuva 10. Yksittäisen artikkelin oletus-XHTML-asemointi rakennetaan PHP-pätkillä ja XHTML-merkinnöillä, ja se kääritään taulukkoon.

6 TYYLIPOHJAN TOTEUTUS

Sisällöntuottaja määrittelee sisällön, eikä hänen tarvitse tietää teknisistä asioista käytännössä mitään. Komponentit, moduulit ja liitännäiset määrittelevät Joomla-sivuston toiminnallisuuden. Tyylipohjilla puolestaan määritellään, miltä sivusto näyttää. *Index.php* [luku 6.1] ja *templateDetails.xml* [luku 6.4]

ovat tyylipohjan olennaisimmat tiedostot, eikä tyylipohja välttämättä tarvitse muita tiedostoja toimiakseen.

Kuva 11 yksinkertaistaa ja havainnollistaa tyylipohjan toimintaa. *Index.php*-tiedostossa kutsutaan moduulia ja komponenttia, joiden sisällön Joomla hakee ja sijoittaa merkatuille paikoille. *Tyyli.css*-tiedostossa määritellään XHTML-elementtien ulkonäköä ja asemointia, ja lopuksi sivu näytetään selaimessa.



Kuva 11 - Joomlan tyylipohjan toiminta yksinkertaistetusti.

Tyylipohjasuunnittelu on pääasiallisesti raamien piirtämistä XHTML-kielellä, CSS-tyylimäärittelyitä ja komponentin sekä moduulien paikkojen merkkaamista PHP-pätkillä. Kuten aiemmin todettiin, voidaan tyylipohjilla myös ylikirjoittaa komponenttien ja moduulien MVC-mallin näkymien asetteluita ja vaikuttaa siten tarkemmin siihen, miten tieto sivulla näytetään. Näin Joomla-

sivuston ulkonäkösuunnittelu voidaan toteuttaa täysin yhden kansion sisällä kajoamatta Joomlan ytimen komponenttien asennustiedostoihin. Tyylipohja-pakettia voidaan helposti myös jakaa ja asentaa muihin Joomla-järjestelmiin.

Sana tietoturvasta

Julkisessa jaossa olevia komponentteja kehitetään jatkuvasti, ja järjestelmän tietoturvaa parannetaan. Joskus kehitys- ja tietoturvaparannukset saattavat kohdistua komponentin asettelumalleihin. Tällöin tarvittavat muutokset on muistettava tarkistaa ja tehdä myös räätälöityihin asettelutiedostoihin, koska ne eivät komponentin päivittämisen yhteydessä muutu.

6.1 Joomla-tyylipohjan index.php-tiedoston perusrakenne

Tyylipohjan *index.php*-tiedosto on tyylipohjan tulopiste, ja vain Joomla-järjestelmällä on lupa sitä lähestyä. Se on myös se index-tiedosto, jonka Joomla näyttää käyttäjälle jokaisen sivulatauksen yhteydessä. Tiedosto ei yleensä sisällä mitään tietoa, vaan se kertoo, missä paikoissa Joomlan on tietoa näytettävä. Kuvassa 12 esitetään yksinkertaisen tyylipohjatiedoston koodi, ja tässä luvussa käymme sen lävitse rivi riviltä.

*index.php

```
1  <?php defined( '_JEXEC' ) or die( 'Restricted access' ); ?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="<?php echo $this->language; ?>"
5  lang="<?php echo $this->language; ?>" >
6  <head>
7  <jdoc:include type="head" />
8  <link rel="stylesheet" href="<?php echo $this->baseurl ?>templates/
9  <?php echo $this->template ?>/css/style.css" type="text/css" />
10 </head>
11 <body>
12 <jdoc:include type="module" name="valikko" />
13 <jdoc:include type="component" />
14 <jdoc:include type="modules" name="alapalkki" />
15 </body>
16 </html>
```

Kuva 12. Yksinkertainen tyylipohjan index.php-tiedosto.

Ensimmäisen rivin koodi tarkistaa, mistä tiedostoa kutsutaan. _JEXEC-vakion käyttö käsiteltiin luvussa 5.2.1. Toisella rivillä kerrotaan Internet-selaimelle millä kielellä ja kieliversiolla sivu on kirjoitettu [16], jotta se osaa generoida sisällön oikein. Sen sijaan, että käytettäisiin XHTML-elementit täsmälleen standardien mukaan tulkitsevaa *Strict*-määrittelyä, määritellään Joomlan XHMTL-elementit *DTD XHTML 1.0 Transitional* -määrittelyiden mukaisesti. *Transitional*-määrittely sisältää muutamia poikkeuksia standardeista ja kaikki XHTML-elementit lukuunottamatta *frameset*-elementtiä [2, s. 250].

Neljännellä ja viidennellä rivillä määritellään sivun käyttämä nimiavaruus, jonka avulla niputetaan käytettävä XHMTL-sovelluksen sanasto erilliseksi kokonaisuudeksi. Näin käytettävä sanasto ei mene sekaisin mahdollisten muiden käytettävien sovellusten samojen sanojen kanssa. Neljännellä ja viidennellä rivillä määritellään myös sivun XML-tiedostoissa ja sisällöissä käytettävä kieli. Sivuston yleisiä asetuksia määritellään Joomlan ylläpitoliittymässä *Global Configuration* -sovelluksella. Jos siellä on sivuston kieleksi määritelty suomi, niin kolmannen rivin *<?php echo $this->language; ?>* -PHP-pätkät generoivat sen mukaisesti tekstin ”*fi-fi*”.

Sivun *head*-osio alkaa riviltä kuusi, ja seisemännellä rivillä sen sisältö noudetaan dynaamisesti *<jdoc:include type=”head” />* -käskyllä. Komento noutaa *Global Configuration* -asetuksissa määritellyt oletusarvoisen otsikon, avainsanat sekä sivun kuvauksen ja käärii ne *title*- ja *meta*-elementtien sisään. Niiden arvot voidaan määritellä jokaisen artikkelin yhteydessä erikseen, jolloin ne korvaavat kenttien oletusarvot. Myös monet Joomla-laajennukset kirjoittavat toimintoihinsa tarvittavaa sisältöä *head*-osioon. Se voi olla esimerkiksi Javascript-koodia tai vaikkapa linkkejä laajennusten omiin CSS-tyylitiedostoihin.

Kahdeksannella ja yhdeksännellä rivillä luodaan linkki tyylipohjan käyttämään CSS-tyylitiedostoon. *<?php echo $this->baseurl ?>* -koodi kirjoittaa sivuston verkko-osoitteen juuren ja *<?php echo $this->template ?>* tyylipohjatiedoston nimen. Tyylipohjan nimi on oltava sama kuin sen kansion nimi, missä tyylipohjan tiedostot sijaitsevat.

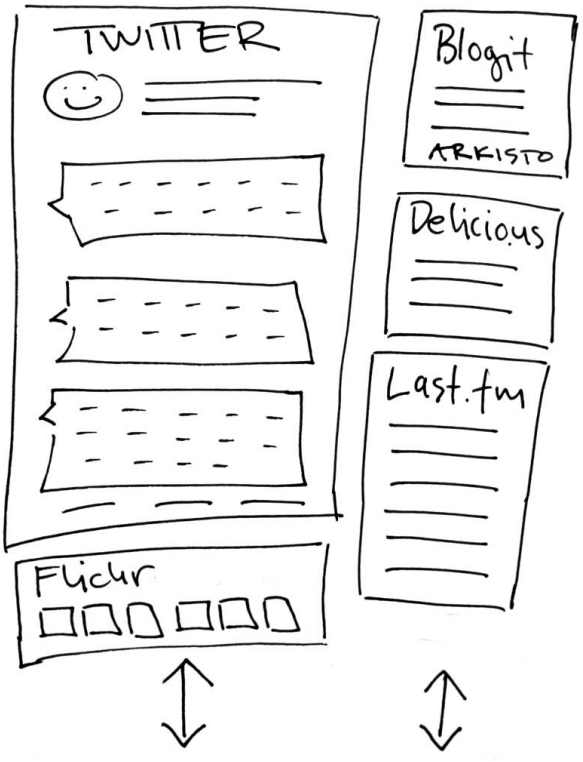
Body-elementissä on kolme kutsua. Ensin määritellään moduulipaikka ja nimetään se *valikoksi*: *<jdoc:include type="modules" name="valikko" />*. Kohdassa kutsutaan niitä moduuleita, jotka ovat ylläpitoliittymän *Module Managerissa* määritelty näkymään moduulipaikassa nimeltä *valikko*. Rivillä kolmetoista kutsutaan komponenttia: *<jdoc:include type="component" />*, ja neljännellätoista rivillä uudelleen moduuleita. Tällä kertaa kutsutaan niitä, jotka ovat määritelty näkyväksi moduulipaikassa nimeltä *alapalkki*. Moduulipaikkoja voidaan siis merkata tyylipohjaan useita ja komponenttia kutsutaan vain kerran. Riveillä 15 ja 16 suljetaan *body*- ja *html*-elementit.

6.2 Blogisivuston graafinen ilme ja sen taitto tyylipohjaksi

Blogisivustolle suunniteltiin kynä ja paperi -tekniikalla kolme erilaista asettelua. Ideatasolla tehtyjen suunnitelmien perusteella tehdyt piirrokset eivät ota kantaa julkaisujärjestelmään eivätkä sen käyttämiin tekniikoihin. Piirrosten pohjalta suunniteltiin rautalankamalli, jossa voitiin jo määritellä teknisiä ominaisuuksia. Rautalankamallin ja piirustusten avulla pystyttiin tekemään tyylipohjan *index-php*-tiedosto.

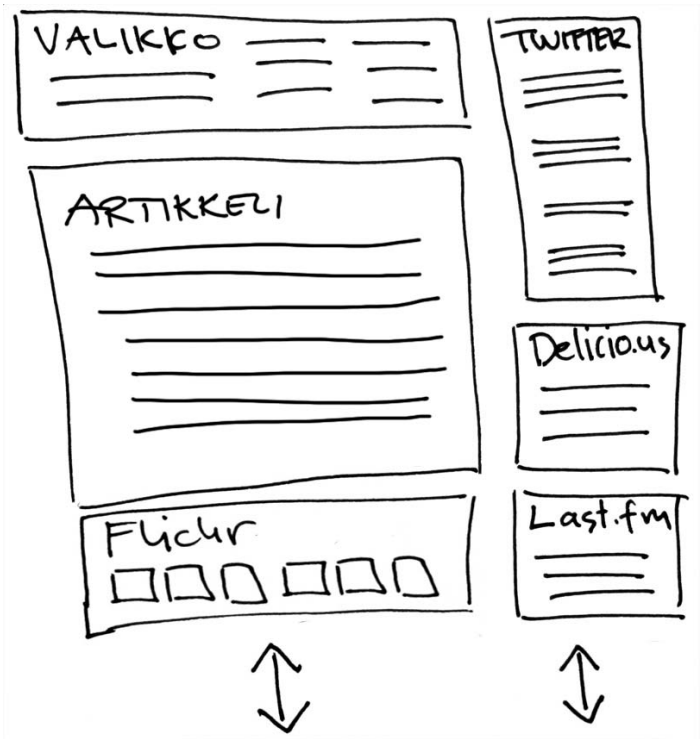
6.2.1 Hahmotelmat

Kuvassa 13 näkyvä etusivun asettelu kokoaa sisältöjä ja RSS-syötteitä niistä sosiaalisista medioista, joihin bloggaaja tuottaa sisältöjä aktiivisesti: *Twitter*-mikroblogista, *Delicio.us*-kirjanmerkkipalvelusta, *Flickr*-kuvapalvelusta ja *Last.fm*-musiikkisivustolta. Etusivulla näytetään myös blogin tuoreimpien artikkeleiden otsikot. Sivu jaettiin kahteen palstaan, joiden korkeus vaihtelee toisistaan riippumatta niissä näkyvän sisällön määrän mukaan. Blogit-valikossa näytetään tuoreimpien otsikkolinkkien lisäksi myös linkki blogiarkistoon: sivulle, jolla näytetään blogimerkinnät aihepiirien mukaan järjesteltynä.



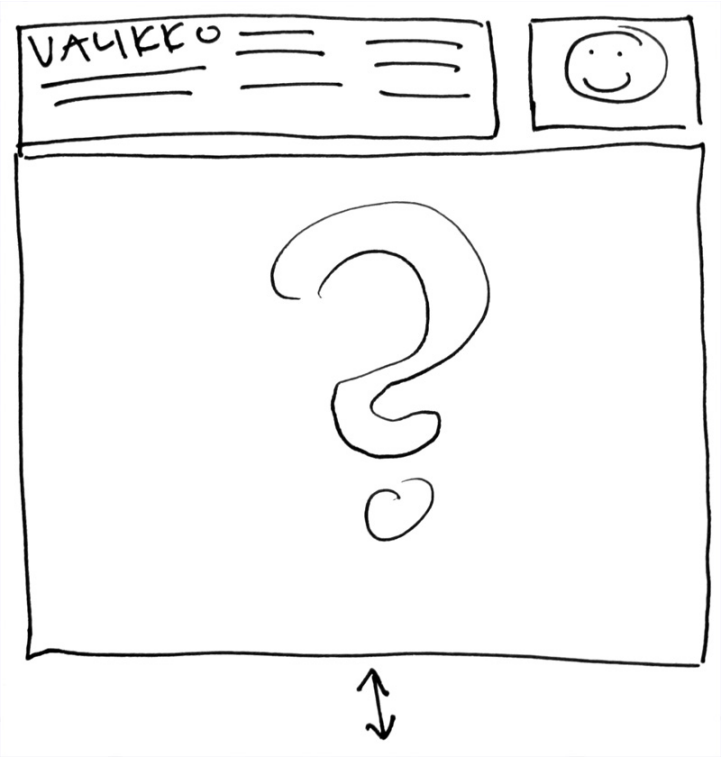
Kuva 13. Blogisivuston etusivu asetellaan kaksipalstaiseksi, ja siinä näytetään dynaamisesti bloggaajan tuottamia sisältöjä ja syötteitä sosiaalisen median sivustoilta

Sosiaalisen median sivustoilta noudettujen sisältöjen linkit johtavat blogisivustolta pois, ja ne avautuvat samaan selainikkunaan. Blogimerkintöihin johtavat linkit avaavat kuvan 14 mukaisen bloginäkymän. Myös tällöin sisältö asemoidaan kahdelle palstalle. Vasemman palstan yläreunassa näytetään valikko, jossa on blogin valikkorakenne. Artikkeleiden sisältö näytetään vasemmalla palstalla valikon alla. Tässäkin näkymässä vasemman ja oikean palstan korkeudet vaihtelevat sisällön määrän mukaan, joten alareuna ei ole tasattu.



Kuva 14. Bloginäkymässä näytetään selkeä blogivalikko, josta blogimerkintöjä ja -aiheita voi selata.

Toisinaan saatetaan tarvita kuvan 15 mukaista yksipalstaista täysleveää näkymää, esimerkiksi laajaa kuvagalleriaa esitettäessä. Myös sen korkeus vaihtelee sisällön määrän mukaan. Valikon oikealle puolelle halutaan toisinaan alue, johon bloggaaja voi määritellä sisällön sivukohtaisesti. Sen on aina oltava saman korkuinen kuin valikko. Jos bloggaaja ei laita pienelle alueelle mitään sisältöä, laajenee valikko täysleveäksi. Siitä ei ole erillistä piirrosta.

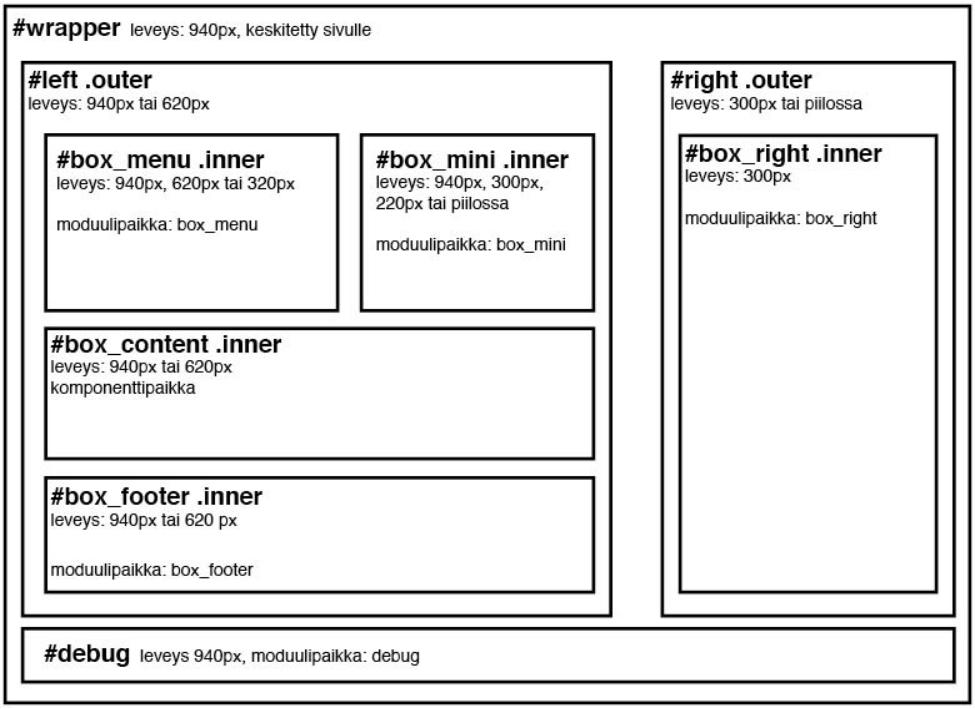


Kuva 15. Tulevaisuuden tarpeita varten tyylipohjan rakenteeseen suunnitellaan mahdollisuus näyttää sisältö täysleveässä palstassa.

Piirrosten ruudut eivät tässä vaiheessa kerro, tehdäänkö sivun taitto käyttäen taulukoita vai *div*-rakennetta. Kuvausten pohjalta todettiin kuitenkin, että Joomla sopii erittäin hyvin sivuston rakentamiseen. Hahmotelmissa näkyvät valikot, sosiaalisten medioiden ruudut ja sisältöalue ovat toteutettavissa moduuleita ja komponenttia hyödyntäen. Joomla tarjoaakin joustavia mahdollisuuksia sekä ruutujen paikkojen ja että ruudun sisäisen asettelun vaihtamiseen sivukohtaisesti. Sisältöalueen vaihteleva leveys on helposti toteutettavissa ehdollisten PHP-lausekkeiden avulla.

6.2.2 Rautalankamalli

Piirrosten pohjalta suunniteltiin tyylipohjan rautalankamalli. Siihen merkattiin komponentin ja moduulien paikat ja ne nimettiin. Malliin merkattiin myös palstojen leveydet pikseleinä.



Kuva 16. Blogisivuston asemointipohjan malli.

Kuvan 16 rakenne tehdään käyttäen ainoastaan *div*-elementtejä, jotka ladotaan 12-palstaiseen *960 Grid System* -ruudukkoon [17]. Nykyisten tietokoneiden näyttöjen resoluutio on lähes aina vähintään 1024 pikselin levyinen lukuunottamatta mobiililaitteita, joiden näyttöresoluutiot ovat alhaisemmat. 960 pikselin levyinen alue mahtuu hyvin 1024 pikselin levyisille näyttölaitteille myös vierityspalkin kanssa. Joomlan ohjelmistokehys tarjoaa työkaluja mobiililaitteiden tunnistamiseen. Niitä hyödyntäen voidaan suunnitella oma tyylipohja vaikka jokaiselle laitteelle erikseen, mutta tässä työssä tehdään tyylipohja vain nykyaikaisille vähintään 1024 pikselin levyisen resoluution toistoon kykeneville näyttölaitteille. 960 pikseliä on erinomainen leveys myös sen vuoksi, että se on jaollinen luvuilla 2, 3, 4, 5, 6, 8, 10, 12, 15, 16, 20, 24, 30, 32, 40, 48, 60, 64, 80, 96, 120, 160, 192, 240, 320 ja 480. Siitä johtuen se sopii hyvin pikseliruudukkoon pohjautuvaan verkkosivusuunnitteluun. Pienillä muutoksilla seuraavassa luvussa esiteltyihin PHP-ehtolauseisiin voidaan palstojen leveyksiä helposti muuttaa ilman, että niiden arvoja tarvitsisi työläästi muuttaa useisiin kohtiin CSS-tiedostossa.

6.2.3 Someblogger-tyylipohjan index.php-tiedoston koodi pala palalta

Tässä luvussa käymme lävitse olennaiset kohdat suunnitelmien perusteella tehdyn tyylipohjan *index.php*-tiedostosta [liite 1]. Tiedoston alussa on samat määritykset kuin kuvan 12 alussa, eli *_JEXEC*-tarkistus sekä dokumentti-

tyyppi ja -kielimääritykset. *Head*-osion alussa kutsutaan Joomlan sisäistä metodia, joka generoi tarvittavat tiedot. Sen jälkeen kutsutaan CSS-tiedostoja. Ensin kutsutaan Joomla-järjestelmän käyttämiä yleisiä tyylejä. Nämä kutsut on oltava kaikissa Joomla-tyylipohjissa. Tässä työssä käytetyt CSS-tyylit jaettiin kolmeen eri tiedostoon. Http://960.gs-sivuilta ladatussa *960.css*-tiedostossa määritellään elementtien asemointia *960 Grid Systemin* mukaiseen ruudukkoon, reset.css-tiedostossa nollataan selainten oletusarvoiset keskenään erilaiset tyylimääritykset ja *tyyli.css* -tiedostossa määritellään omia tyylejä [17] , [18], [liite 3].

```
<link rel="stylesheet" href="templates/system/css/system.css" ty-
pe="text/css" />
<link rel="stylesheet" href="templates/system/css/general.css" ty-
pe="text/css" />
<link href="templates/<?php echo $this->template?>/css/reset.css"
rel="stylesheet" type="text/css" />
<link href="templates/<?php echo $this- >template?>/css/960.css"
rel="stylesheet" type="text/css" />
<link href="templates/<?php echo $this->template?>/css/tyyli.css"
rel="stylesheet" type="text/css" />
```

Head-osiossa määritellään PHP:lla muutamia apumuuttujia, joihin viitataan myöhemmin tässä luvussa. *Body*-osiossa on ulommaisena *#wrapper*-elementti, joka toimii 12-palstaisena ruudukkona. Se on käytännössä *div*-elementti, jonka sisään muut div-elementit kääritään. *#wrapper*-elementti noudattaa *960 Grid Systemin container_12*-luokkaa, joka määrittelee elementin leveydeksi 960 pikseliä ja keskittää alueen ruudun keskelle. Kaikki sen sisällä olevat asemointiin vaikuttavat *div*-elementit noudattelevat *960.css*-tyylitiedoston *grid_1*, *grid_2*, *grid_3* … ja *grid_12*-luokkia sen mukaan, kuinka leveä elementti on. *Grid_1* on 60 pikseliä leveä, josta leveydet nousevat 80 pikselin välein. Näin *grid_12* on 940 pikselin levyinen. *Grid*-luokilla on 10 pikselin marginaali sekä vasemmalla että oikealla puolella. *Grid_12*-luokkaa noudattelevan *div*-elementin todellinen leveys on siis 940+10+10 eli 960 pikseliä ja täyttää siten *container_12*-luokan mukaisen elementin kokonaan. *#wrapper*-elementti on seuraavanlainen:

```
<div id="wrapper" class="container_12">
Muut elementit kääritään wrapperin sisään.
</div>
```

#left-elementin leveys riippuu siitä, ladataanko *#right*-elementissä sijaitsevaan *box_right*-moduulipaikkaan sisältöä vai ei. *#left*-elementin leveydeksi määritellään joko 620 pikseliä tai täydet 940 pikseliä, ja *960 Grid Systemin* luokkien mukaan tämä tarkoittaa vastaavasti joko luokan *grid_8* tai *grid_12*-luokan noudattamista. Vaihtoehtoinen leveys toteutettiin Joomlan ehdollisia

lauseita käyttäen [19]. Otsikossa määritelty apumuuttuja *$cols* saa oletusarvokseen 8. Sen jälkeen tarkistetaan, ladataanko *box_right*-moduulipaikkaan moduulia. Ellei ladata, saa *$cols* arvon 12. Muuttujaa käytetään *grid*-luokan tarkenteena, kun *#left*-elementtiä kirjoitetaan. []

```
<head> ---
<?php
$cols=8;
if(!$this->countModules('box_right')) { $cols = 12; };
?> ---
</head>
<body> ---
<div id="left" class="grid_<?php echo $cols; ?> outer">
---
</div>
```

Tilanne on hieman monimutkaisempi *#box_menu* ja *#box_mini* -elementtien kohdalla. Edellisen esimerkin mukaan #left-elementti on siis joko kahdeksan tai kahdentoista palstan levyinen. Tämä tila jakaantuu *#box_menun* ja *#box_minin* välillä riippuen siitä, onko niiden sisäisissä moduulipaikoissa sisältöä. *#box_menu* on joko viiden, kahdeksan tai kahdentoista ja vastaavasti *#box_mini* kolmen, kahdeksan tai kahdentoista palstan levyinen. Myös tässä käytetään sivun *head*-osiossa määriteltyjä apumuuttujia.

```
<?php
// Annetaan apumuuttujille oletusarvot
$menucols=5;
$minicols=3;

// Tarkistetaan, onko  box_right-, box_mini- ja
// box_menu-elementeissä sisältöä ja muutetaan
// arvoja tilanteen mukaan sopivaksi.
if($this->countModules('box_right')
|| $this->countModules('box_mini'))
      { $menucols = 8; };
if($this->countModules('box_right')
&& !$this->countModules('box_menu')
      { $minicols = 8; };
if(!$this->countModules('box_right')
&& $this->countModules('box_menu')
&& $this->countModules('box_mini'))
      { $menucols = 8; $minicols = 4; };
if(!$this->countModules('box_right')
&& !$this->countModules('box_mini'))
      { $menucols = 12; };
if(!$this->countModules('box_right')
&& !$this->countModules('box_menu')
     { $minicols = 12; };
?>
```

Kun apumuuttujat on määritelty, kirjoitetaan div-elementit, ja niiden grid-luokkavalitsimiin kirjoitetaan PHP:n avulla tarvittava leveys palstojen lukumäärän kertovalla numerolla. *960 Grid Systemissä* elementtien marginaalit ovat oletusarvoisesti 10 pikseliä sekä vasemmalla että oikealla reunalla. Lapsielementtien marginaalit määräytyvät siten, että ensimmäisen lapsielementin vasen marginaali onkin nolla. Jos ensimmäisellä lapsielementillä on

sisaruksia, niiden molemmat marginaalit ovat normaalit 10 pikseliä lukuunottamatta viimeistä, jonka oikea marginaali on nolla. Jos lapsielementti on ainoa, ovat sen molemmat marginaalit nolla. Sisäkkäisten elementtien marginaalit määritellään käytännössä siten, että ensimmäinen lapsielementti noudattelee luokkaa *alpha* ja viimeinen luokkaa *omega*. Koska asemoinnissa käytetään sisäkkäisiä kelluvia elementtejä, on käytettävä myös ns. *clearfix*-CSS-tyylimäärittelyä [20]. Sen avulla sisäkkäiset elementit näkyvät oikein kaikilla selaimilla.

```
// Tarkistetaan, tarvitseeko #box_menu-elementtiä edes kirjoittaa.
<?php if( $this->countModules('box_menu')) : ?>

// Kirjoitetaan elementti ja siihen oikea leveys.
// Määritellään myös tarvittavat marginaalit
// alpha ja omega-luokilla sekä elementtien
// näkyminen clearfix-luokkaa käyttäen.
<div id="box_menu" class="grid_<?php echo $menucols; ?> clearfix
alpha <?php if( !$this->countModules('box_mini')) : ?>omega<?php en-
dif; ?> inner">
<jdoc:include type="modules" name="box_menu" style="xhtml" />
</div>
<?php endif; ?>

// box_mini määritellään samoin kuin box_menu.
<?php if( $this->countModules('box_mini')) : ?>
<div id="box_mini" class="grid_<?php echo $minicols; ?> clearfix
<?php if( !$this->countModules('box_menu')) : ?>alpha <?php endif; ?>
omega inner">
<jdoc:include type="modules" name="box_mini" style="xhtml" />
</div>
<?php endif; ?>
```

Komponenttia kutsutaan *#box_content*-alueelle oman *#sisalto*-elementin sisään. Koska myös sen elementin sisältö kääriytyy näin vähintään kahden *div*-elementin sisään, ovat kaikkien sisältöalueiden tyylien määrittelymahdollisuudet vertailukelpoisia keskenään: sisältö sijaitsee vahintään kolmen *div*-elementin sisällä. Jos *#sisalto*-elementtiä ei tehtäisi, voitaisiin joskus olla tilanteessa, että komponentin lataama raaka sisältö olisi kahden mutta kaikissa moduulialueissa kolmen *div*-elementin sisällä. Tässä tapauksessa kyse on kuitenkin ulkonäön mahdollisesta hienosäädöstä kuin olennaisesta sivuston toimintaan liittyvästä seikasta. *#box_content*-elementti on aina saman levyinen kuin *#left*-elementti, joten myös siinä käytetään aiemmin määriteltyä *$cols*-muuttujaa.

```
<div id="box_content" class="grid_<?php echo $cols; ?> clearfix alpha
omega inner">
<div id="sisalto"><jdoc:include type="component" /></div>
</div>
```

Sekä *#box_footer*, *#box_right* että *#debug*-elementtien kirjoittamisen yhteydessä yksinkertaisesti tarkistetaan, ladataanko niiden sisältämiin moduuli-

paikkoihin moduuleita ja tarvittaessa alue kirjoitetaan. Seuraavana on esimerkkinä *#box_right*-elementin kirjoitus.

```
// Tarkistetaan, ladataanko moduulipaikkaan moduuleita.
<?php if($this->countModules('box_right')) : ?>

// Jos ladataan, niin jatketaan.
<div id="right" class="grid_4 outer">
<div id="box_right" class="grid_4 clearfix alpha omega inner">

// Ladataan moduulit.
<jdoc:include type="modules" name="box_right" style="xhtml" />
</div>
</div>

// Suljetaan ehtolause.
<?php endif; ?>
```

Kuten huomaamme, noudattelee jokainen elementti yhtä CSS-määritysten ID:tä ja useaa luokkaa. *960 Grid Systemin* CSS-koodi pidetään muista tyylimäärittelyistä täysin erillisenä tiedostona. Muita tyylimäärityksiä varten tehtiin jokaiselle elementille oma ID-valitsin (ID selector). Lisäksi tehdään kaksi luokkavalitsinta (class selector): *outer* ja *inner.* Niiden avulla päästään helposti muokkaamaan juuri tiettyjen elementtien tai elementtiryhmien ominaisuuksia CSS-tiedostossa. Voimme esimerkiksi määritellä kaikkiin *#left .inner h1* -otsikkoelementteihin tietyn kokoiset kirjasimet ilman että se vaikuttaisi *#right*-elementin sisäisiin *h1*-otsikoihin. Omat määritykset tehdään *tyyli.css*-tiedostoon, eikä *960.css*- ja *reset.css*-tiedostojen tyylimäärityksiin kosketa. Määrityksiä tehdessä on muistettava, että *tyyli.css*-tiedostossa ei saa määritellä *grid*-elementtien leveyteen vaikuttavien *margin*-, *padding*- eikä *border*-ominaisuuksien arvoja, muutoin *960 Grid Systemin* mukainen asemointi rikkoutuu.

6.3 Sisältökomponentin asetteluiden ylikirjoittaminen

Joomlan oletusasennuspaketin mukana asentuu kolme tyylipohjaa. Yksi niistä, *Beez*, pyrkii täysin taulukottomaan toteutukseen [21]. *Beez*-tyylipohjassa on ylikirjoitutustiedostot kahdentoista Joomlan perusasennuksen laajennuksen asettelumalleille. Tässä luvussa analysoimme alkuperäistä sisältökomponentin *article/default.php*-asettelumallia, minkä jälkeen käymme tarkemmin läpi *Beez*-tyylipohjan vastaavaa asettelumallia. Jatkossa käytämme näistä kahdesta asettelumallista nimiä beez-malli ja oletusmalli.

Oletusmallissa otsikko kirjoitetaan yhden ja artikkelin sisältö toisen taulukon sisään. Kun artikkelin osat pirstotaan tällä tavoin useisiin taulukoihin, artikkelin semanttisuus hajoaa. Se kuitenkin säilyy, jos otsikot ovat käärittynä

XHTML-kielen mukaisten otsikkoelementtien sisään ja artikkelin sisältö seuraa niitä *p*-elementteihin käärittynä. Pääotsikoissa käytetään *H1*-elementtiä ja niitä seuraavissa alaotsikoissa luonnollisesti *H2*- ja alempitasoisia otsikkoelementtejä.

Beez-mallissa komponentin otsikko on pääotsikko. Se kääritään *H1*-elementtiin, mikä on semanttisesti aivan oikein; pääotsikko on ylimmän tason otsikko. Sitä seuraavat mahdolliset julkaisu-tiedot, artikkelin kirjoittajan nimi ja kirjoituspäivämäärä jos ne on määritelty artikkelin yhteydessä näkyväksi. Sen jälkeen kirjoitetaan artikkelin otsikko *H2*-elementtiin käärittynä, jos se on joku muu kuin komponentin otsikko. Sen jälkeen tulostetaan artikkeli kappale kappaleelta väliotsikoineen kuten artikkelin kirjoittaja on sen rakenteen tarkoittanut. Kokonaisuus kääritään yhden *div*-elementin sisään.

```
// Alussa on tuttu _JEXEC-tarkistus.
<?php // @version $Id: default.php 11917 2009-05-29 19:37:05Z ian $
defined('_JEXEC') or die('Restricted access'); ?>

// Aloitetaan sivu.
<div id="page">

// Tarkistetaan, onko käyttäjällä oikeuksia artikkelin muokkaamiseen
julkisen liittymän kautta. Jos on, niin kirjoitetaan linkit, joista
käyttäjä pääsee artikkelin muokkaustilaan.
<?php if (($this->user->authorize('com_content', 'edit', 'content',
'all')
|| $this->user->authorize('com_content', 'edit', 'content', 'own'))
&& !($this->print)) : ?>
<div class="contentpaneopen_edit
<?php echo $this->escape($this->params->get('pageclass_sfx')); ?>">
<?php echo JHTML::_('icon.edit', $this->article, $this->params,
$this->access); ?>

</div>
// Suljetaan muokkausoikeudet tarkistava ehtolause.
<?php endif; ?>

// Tarkistetaan näytetäänko valikkokohteessa määritelty sivun otsikko.
<?php if ($this->params->get('show_page_title',1)

// Tarkistetaan myös, ettei sivun otsikko ole sama kuin artikkelin
otsikko.
&& $this->params->get('page_title') != $this->article->title) : ?>

// Jos edellä olevat ehdot täyttyvät, jatketaan, ja kirjoitetaan sivun otsikko H1-elementin sisään.
<h1 class="componentheading<?php echo $this->escape($this->
params->get('pageclass_sfx')); ?>">
<?php echo $this->escape($this->params->get('page_title')); ?>
</h1>

// Lopetetaan ehtolause.
<?php endif; ?>

// Tarkistetaan, onko artikkelin otsikko määritelty näkyväksi.
<?php if ($this->params->get('show_title')) : ?>
```

```
// Jos on, niin se kirjoitetaan.
<h2 class="contentheading

/ Tarkistetaan, onko artikkelin otsikko määritelty linkiksi osoittamaan artikkeliin. Otsikko on hyvä olla linkki silloin, jos on artikkeliin on määritelty ingressiteksti ja lue lisää -erotin.
<?php echo $this->escape($this->params->get('pageclass_sfx')); ?>">
<?php if ($this->params->get('link_titles')
&& $this->article->readmore_link != '') : ?>
<a href="<?php echo $this->article->readmore_link; ?>
class="contentpagetitle<?php echo $this->escape($this->
params->get('pageclass_sfx')); ?>">
<?php echo $this->escape($this->article->title); ?></a>

// Kirjoitetaan artikkelin otsikko ja lopetetaan linkin tarkistava ehtolause.
<?php else :echo $this->escape($this->article->title); endif; ?>
</h2>

// Lopetetaan otsikon kirjoittamisen tarkistava ehtolause.
<?php endif; ?>
```

Tämän jälkeen *beez*-malli tarkistaa edellä esitettyjen ehtolauseiden avulla, onko artikkelin yhteydessä määritelty näkyväksi artikkelin luonti- ja muokkauspäivämäärät ja kirjoittajan nimi. Tiedot kääritään *p*-elementteihin, eli kappaleisiin, joista jokainen saa luokkamääreen, minkä avulla niiden tyylejä pystyy helposti muokkaamaan erillisessä CSS-tiedostossa. Ennen itse artikkelin kirjoittamista lähetetään heräte myös *afterDisplayTitle*-kuuntelijaa käyttäville liitännäisille, jotka kirjoittavat sisältönsä otsikon jälkeen. Tyylipohja tarkistaa myös, onko ylläpitoliittymässä artikkelin yhteydessä määritelty näkyväksi kolme linkkiä: artikkelin tulostus tulostimella, artikkelin muuttaminen PDF-tiedostoksi ja artikkelin lähettäminen ystävälle sähköpostitse. Ylläpitoliittymässä voidaan myös määritellä, ja *beez*-malli tarkistaa, näytetäänkö artikkelin yhteydessä linkki niihin pää- ja alaryhmiin, joihin artikkeli kuuluu.

```
// Tässä vaiheessa lähetetään heräte beforeDisplayContent-kuuntelijalle. Kaikki liitännäiset, jotka käyttävät sitä, tulostavat sisältönsä seuraavassa kohdassa.
<?php echo $this->article->event->beforeDisplayContent; ?>

// Jos artikkeli on on tehty monisivuiseksi, kirjoitetaan artikkelin sisällysluettelo.
<?php if (isset ($this->article->toc)) :
echo $this->article->toc;
endif; ?>

// Kirjoitetaan artikkelin sisältö.
<?php echo JFilterOutput::ampReplace($this->article->text); ?>

// Lopuksi lähetetään heräte afterDisplayContent-kuuntelijaa käyttäville liitännäisille, jotka kirjoittavat oman tulosteensa artikkelin perään.
<?php echo $this->article->event->afterDisplayContent; ?>

// Suljetaan sivun div-elementti.
</div>
```

Kuten edellä huomattiin, jokainen PHP-lauseke pyritään kirjoittamaan ainoastaan yhdelle riville, ja tiedostot suositellaan kirjoitettavaksi käyttäen XTHML-kieltä niin, että ne sisältävät lyhyitä PHP-pätkiä sen sijaan, että sivu kirjoitettaisiin kokonaan PHP:lla, joka muodostaisi XHTML-kielistä koodia. Asemointimallissa ei myöskään käsitellä dataa, siihen vain haetaan tarvittava sisältö. Tiedon käsittely tehdään muualla MVC-arkkitehtuuria noudattaen. [3, s. 212]

6.4 Asennuspaketti

Tyylipohjan tiedostot sijaitsevat kuvassa 2 esitellyn Joomla-asennuksen juuren *templates*-kansiossa omassa alihakemistossaan, joka on nimetty tyylipohjan nimen mukaan. Tyylipohjakansion juuressa on oltava tyylipohjan tulopiste, *index.php*, ja manifestitiedosto *templateDetails.xml*. Muut tiedostot ovat vapaaehtoisia. Tiedostojen sijoittamiseen ja nimeämiseen suositellaan sellaista tapaa, joka kertoo, mikä on kunkin kansion sisältö ja tiedoston tarkoitus. Näin tyylipohjan ylläpitäminen on helpompaa, etenkin jos tiedostoja on useita. Bloggaajan tyylipohjalle annetaan nimi *someblogger*, joten tyylipohjatiedostoille tehdään *someblog*ger-kansio. Sinne luodaan näkymien ylikirjoittaville tiedostoille *html*-kansio, CSS-tyylitiedostoille *css*-kansio ja tarvittaville kuville *images*-kansio.

Tyylipohjan manifestitiedosto sisältää olennaiset tiedot tyylipohjan tiedostoista ja niiden sijainneista. Joomla-järjestelmä hyödyntää tietoja tyylipohjan asennuksessa ja tyylipohjaa käytettäessä. Manifestitiedostossa on myös tyylipohjan tekijän tiedot. Seuraavassa on osa *someblogger*-tyylipohjan manifestitiedostoa, johon on kommentoitu XML-tiedoston tagien tarkoitus.

```
// Määritellään käytettävä XML versio ja käytettävä merkistö.
<?xml version="1.0" encoding="utf-8"?>

// Aloitetaan tyylipohjan asennustiedostojen tietojen määrittely.
<install version="1.5" type="template">

// Tyylipohjan nimi, luontipäivämäärä ja tiedot tekijästä
// sekä lisenssistä.
  <name>someblogger</name>
  <creationDate>May 2010</creationDate>
    <author>Pasi Heiskanen/Angie Radtke/Robert Deutz/Eric Meyer/Dejan
    Noveski/Vasil Vangelovski/Cory Simmons</author>
    <authorEmail>pasi.heiskanen@gmail.com</authorEmail>
    <authorUrl>http://www.onnikoo.fi</authorUrl>
    <license>GNU/GPL version 2</license>

// Description-kentässä kuvaillaan tyylipohjaa.
// Tässä tapauksessa eritellään myös tyylipohjassa käytetyn avoimen
// lähdekoodin lähteet.
    <description>
```

```
    Versatile aggregator template for social media addicts.
     HTML overrides by Angie Radtke and Robert Deutz, joomla@run-
     digital.com, http://www.run-digital.com. 960.css by
     http://960ls.atomidata.com/ thanks to Dejan Noveski, Vasil Van-
     gelovski and Cory Simmons. Reset.css by Eric Meyer,
     http://meyerweb.com/eric/tools/css/reset/.
    <description>

// Esitellään asennettavat tiedostot ja tiedostojen sijainnit.
  <files>
    <filename>index.php</filename>
    <filename>templateDetails.xml</filename>
    <filename>template_thumbnail.png</filename>
    <filename>css/960.css</filename>
    <filename>css/tyyli.css</filename>
    <filename>css/reset.css</filename>
    <filename>html/index.html</filename>
    <filename>html/com_content/index.html</filename>
    <filename>html/com_content/article/default.php</filename>
    <filename>html/com_content/article/index.html</filename>
    <filename>html/com_content/article/form.php</filename>
    <filename>html/com_content/category/index.html</filename>
    <filename>html/com_content/category/blog.php</filename>
    <filename>html/com_content/category/blog_item.php</filename>
    <filename>html/com_content/category/blog_links.php</filename>
    <filename>html/com_content/category/default_items.php</filename>
    <filename>html/com_content/category/default.php</filename>
    <filename>html/com_content/frontpage/index.html</filename>
    <filename>html/com_content/frontpage/default.php</filename>
    <filename>html/com_content/frontpage/default_item.php</filename>
    <filename>html/com_content/frontpage/default_links.php</filename>
    <filename>html/com_content/section/index.html</filename>
    <filename>html/com_content/section/blog.php</filename>
    <filename>html/com_content/section/blog_item.php</filename>
    <filename>html/com_content/section/blog_links.php</filename>
    <filename>html/com_content/section/default.php</filename>

// Beez-mallissa on ylikirjoittavat tiedosto useille
// komponenteille ja moduuleille. Niiden esittely on
// tässä jätetty pois. Tässä työssä tehtävän tyylipohjan
// templateDetails.xml-tiedosto on liitteenä (liite 4).
  </files>

// Esitellään tyylipohjassa käytetyt moduulipaikat.
  <positions>
    <position>box_menu</position>
    <position>box_mini</position>
    <position>box_right</position>
    <position>box_footer</position>
    <position>debug</position>
  </positions>

// Suljetaan XML-tiedosto.
</install>
```

Kun kaikki tarvittavat tiedostot on saatu valmiiksi ja testattua kehitysympäristössä, tehdään tyylipohjakansiosta tiedostoineen ja alihakemistoineen asennuspaketti. Sen nimeämisessä ei ole tarkkaa käytäntöä, mutta yleisesti paketti on saman niminen kuin tyylipohja. Paketti asennetaan testiympäristöön, missä varmistetaan asennuksen onnistuminen ja testataan tyylipohjan toimivuus. Onnistuneen testauksen jälkeen tyylipohja on valmis asennettavaksi tuotantoympäristöön.

Asennuspaketti voi olla joko *zip-, tar.gz-* tai tar.bz2-tiedostomuotoinen [22]. Mac OS X -käyttöjärjestelmissä on huomioitava, että sen 10.3.5 ja uudemmat versiot pakkaavat *zip*-tiedostot oletusarvoisesti AppleDouble-muotoon. Ne asentuvat Joomlaan muutoin normaalisti mutta aiheuttavat *XML Parsing Error at 1:1. Error 4: Empty document* -virheilmoituksen [23]. Mac OS X -käyttöjärjestelmän tiedostoselaimen (Finder) pakkaa-toiminto käyttää aina AppleDouble-tiedostomuotoa. Päätteen (Terminal) kautta kansion voi myös Mac-ympäristössä pakata ilman AppleDouble-muotoa käyttäen *export COPYFILE_DISABLE=true* -komentoa ennen varsinaista pakkaamista [24].

Tyylipohjan asennus

Kehitysympäristössä tehtiin tyylipohja nimeltä *someblogger* ja sen tiedostot pakattiin *zip*-tiedostoksi. Ennen paketin asentamista tuotantoympäristöön, sen toiminta testattiin paikallisen koneen testiympäristössä. Paketti asennettiin laajennustenhallintatyökalulla ja otettiin käyttöön määrittelemällä se oletustyylipohjaksi tyylipohjienylläpitotyökalulla. Muut tyylipohjat poistettiin turvallisuussyistä [25].

7 TARVITTAVAT MODUULIT

Moduuli ja komponentti on helpoin erottaa toisistaan käyttäen esimerkkinä yksinkertaista äänestyslomaketta ja blogimerkintää. Komponentti tulostaa sivulle blogimerkinnän, ja moduuli näyttää sen vierellä kyselylomakkeen. On huomattava, että myös valikot ovat Joomlassa sisältöartikkeleista erillisiä moduuleita. Joomlan oletusasennuspaketissa on useita moduuleita valmiina, esimerkiksi RSS-syötteenlukija, yksinkertainen pikakysely ja tuoreimmat blogimerkinnät näyttävä moduuli. Niiden lisäksi Joomlaan on saatavilla lukuisia kolmansien osapuolten kehittämiä moduuleita.

Samoin kuten komponentteja, Joomlassa on erikseen sekä julkisen että ylläpitoliittymän moduuleita. Ne voivat olla toiminnoiltaan täysin itsenäisiä tai täydentää komponenttien toimintoja. Mutta toisin kuin komponentteja, joita näytetään aina vain yksi kerrallaan, voidaan sivulla joko näyttää useita moduuleita samaan aikaan tai olla näyttämättä yhtään moduulia. Moduuleista sanotaan, että niiden tarkoitus ei ole esittää sivun pääasiallista sisältöä, vaan täydentää komponentin toimintaa ja tulostetta [11, s. 7]. Bloggaajan verkkosivuston etusivu rakennettiin kuitenkin lähes kokonaan niiden varaan,

koska hän haluaa aloitussivun sisällön noudettavaksi useista lähteistä. Jokainen etusivulla näytettävä moduuli hakee sisältöjä eri sosiaalisesta mediasta, jolloin etusivun komponentin rooli jää hyvin pieneksi.

Tässä luvussa esitellään julkisen liittymän moduulien rakennetta käyttäen esimerkkinä blogisivustolle asennettavaa RokTwittie-nimistä moduulia. Se on itsenäinen moduuli, joka noutaa ja näyttää bloggaajan tuoreimmat *Twitter*-mikroblogipalveluun lähettämät viestit.

7.1 Moduulien arkkitehtuuri, rakenne ja toiminta

Joomlan ohjelmistokehys ei määrittele moduulisuunnitteluun niin kurinalaista arkkitehtuuria kuin komponenteille. Jokaisella moduulilla on kuitenkin oltava manifestitiedosto. Se on XML-muotoinen tiedosto, jossa on yleisiä tietoja moduulista, kuten moduulin ja moduulin kehittäjän nimi, sekä asennuksessa tarvittavat tiedot moduulipaketin sisältämistä tiedostoista [3, s. 120, s. 127]. Manifestitiedostossa esitellään myös moduulin asetukset, joita voi säätää ylläpitoliittymän kautta. Moduulit asennetaan *modules*-kansion alihakemistoon, joka nimetään moduulin mukaan niin, että siihen tulee *mod_*-etuliite. RokTwittie-moduuli asentuu siis *modules/mod_roktwittie*-hakemistoon. Myös moduulilla on tulopiste, jonka kautta Joomla lähestyy sitä. Esimerkkimoduulin tulopiste on nimeltään *mod_roktwittie.php*, ja sen ensimmäisellä rivillä on kappaleessa 5.2.1 mainittu *_JEXEC*-tarkistusrivi, kuten kuuluukin olla.

Tietojen näyttämiseen liittyvät tehtävät on eroteltu moduulin muista toiminnoista samaan tapaan kuin komponenttien käyttämässä MVC-arkkitehtuurissa [luku 5.1]. Moduulin asemointimallit sijaitsevat *tmpl*-alihakemistossa. Oletusasemointimalli on nimeltään *default.php*, ja asemointimalleja voi olla useita. Käytettävää asemointimallia kutsutaan staattisen *JModuleHelper*-luokan *getLayoutPath()*-metodia kutsumalla [3, s. 124]. *JModuleHelper*-luokka sijaitsee Joomlan kirjastossa *libraries/joomla/application/module*-hakemistossa.

helper.php

```
233    function getLayoutPath($module, $layout = 'default')
234    {
235        global $mainframe;
236
237        // Build the template and base path for the layout
238        $tPath = JPATH_BASE.DS.'templates'.DS.$mainframe->getTemplate().DS.'html'.DS.$module.DS.$layout.'.php';
239        $bPath = JPATH_BASE.DS.'modules'.DS.$module.DS.'tmpl'.DS.$layout.'.php';
240
241        // If the template has a layout override use it
242        if (file_exists($tPath)) {
243            return $tPath;
244        } else {
245            return $bPath;
246        }
247    }
```

Kuva 17. GetLayoutpath()-metodi palauttaa käytettävän asemointimallin polun.

RokTwittie-moduulissa on kaksi asemointimallia. Oletuksena on *default.php*- ja Twitter-yhteyksien virheilmoitusten näyttämisessä käytetään *error.php*-asemointimallia. Jos oletusasemointimalli haluttaisiin ylikirjoittaa, sijoitettaisiin oma *default.php*-tiedosto tyylipohjakansion *html/mod_roktwittie*-alihakemistoon. Näin ei kuitenkaan tässä opinnäytetyössä tehdä, koska RokTwittien oletusasemointimalli käärii sisällön *div*-elementtien sisään, ja sen ulkonäkö on helposti muokattavissa ainoastaan CSS-tyylimäärittelyitä käyttäen.

7.2 RokTwittie-moduulin lisenssi

Joomla perustuu yleisen lisenssin (GNU GPL) mukaiseen avoimeen lähdekoodiin, eli Joomlaa saa muokata ja levittää vapaasti. Lisenssiehtojen mukaisesti myös siihen kehitetyt lisäosat ovat saman tai vapaamman lisenssin mukaisia. Lisäosia jaettaessa tulee antaa saataville myös lisäosan lukukelpoinen lähdekoodi sekä oikeudet lähdekoodin muokkaamiseen ja muokatun teoksen jakeluun samoin lisenssiehdoin.

Täysin ilmaisia lisäosia on vapaasti ladattavissa lisäosahakemistossa, *Joomla Extensions Directoryssa, http://www.joomla.org*-sivustolla. Siellä on paljon erinomaisia mutta myös joitakin tuotantoympäristöön asennuskelvottomia lisäosia. Sivuston yhteisö on vilkas ja tarjoaa varsin hyvää tukea laajennusten virittelyyn. Keskusteluja seuraamalla ja niihin osallistumalla saa parhaan käsityksen, mikä lisäosa soveltuu parhaiten mihinkin tarkoitukseen.

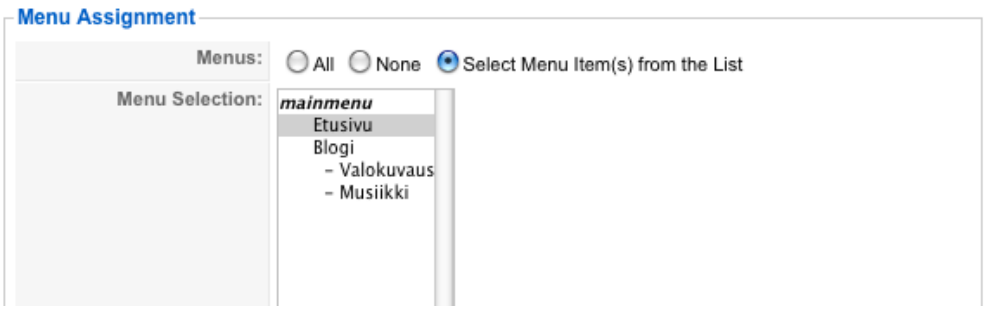
RokTwittie on ladattavissa vain maksullisen *RocketTheme*-klubin jäsenyyden myötä *http://www.rockettheme.com*-sivustolta. Kaupalliset ohjelmistokehittäjät tarjoavatkin laajennusten jakelun yhteydessä maksullista asennus- ja ylläpitotukea rahoittaakseen lisäosien kehittämistyötä. Jakelumalli, jonka mukaan tukipalvelu on ostettava ennen oikeutta lisäosan lataamiseen, on

hieman kyseenalainen. Joka tapauksessa lisenssin periytyminen koskee myös RokTwittietä.

7.3 RokTwittien asennus ja asetukset

RokTwittie-moduuli asennetaan Joomlan ylläpitoliittymässä laajennustenhallintatyökalulla. Asennuksen jälkeen moduulin asetuksia määritellään moduulienhallintatyökalulla.

Asennuksen jälkeen moduulilla on yksi ilmentymä moduulienhallintatyökalun moduulilistauksessa. Ilmentymä voidaan kopioida ja jokaisen ilmentymän ominaisuuksia määritellä erikseen. RokTwittiestä tehtiin esimerkkisivustolle kaksi erilaista ilmentymää, joista toinen on näkyvissä etusivulla *box_menu*-moduulipaikassa ja toinen kaikilla alisivuilla *box_right*-moduulipaikassa.

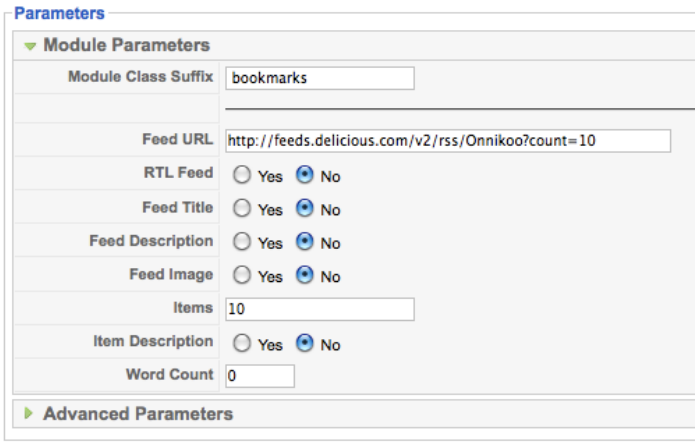


Kuva 18. RokTwittie-moduulin toinen ilmentymä määriteltiin näkyväksi etusivulla.

7.4 RSS-syötettä lukevien moduulien tekeminen

Bloggaaja haluaa näyttää sivuillaan sisältöjä myös *Delicio.us*-kirjanmerkkipalvelun kanavaltaan sekä *Last.fm*-musiikkipalvelun profiilistaan. Molemmat palvelut tarjoavat henkilökohtaista RSS-syötettä (Really Simple Syndication). Joomlan perusasennuksessa on syötteen näyttämistä varten lukijamoduuli valmiina, joten sitä ei tarvitse erikseen asentaa.

Molemmissa tapauksissa tehtiin moduulienylläpitotyökalulla uusi moduuli ja määriteltiin sen tyypiksi *Feed Display*. Tämän jälkeen moduulin asetuksiin tallennettiin syötteen osoite ja näytettävien syötekohteiden (Items) lukumäärä sekä määriteltiin muut parametrit [kuva 5]. *Delicio.us*-kirjanmerkkisyötemoduulille annettiin nimi *Jaetut kirjanmerkkini* ja *Last.fm*-syötemoduulille *Tätä kuuntelen nyt*. Molemmat määriteltiin näkymään *box_right*-moduulipaikassa kaikkien valikkokohteiden osoittamilla sivuilla.



Kuva 19. RSS-syötteenlukijamoduulin asetuksia.

8 TARVITTAVAN LIITÄNNÄISEN ASENTAMINEN

Liitännäisten (plugin) avulla voidaan muokata Joomlan toiminnallisuuksia kajoamatta järjestelmän sisäiseen koodiin [3, s. 133]. Liitännäiset ovat tapahtumaohjattuja. Joomlassa on useita tapahtumia valmiina, joita liitännäiset voivat hyödyntää, ja liitännäisillä voi myös olla omia tapahtumia [26]. Joomlassa on sisäänrakennettuna mekaniikka tapahtumien kuuntelemiseen ja niiden ohjaamiseen.

Blogisivustolle asennettiin Paul Thompsonin suunnittelema Flickr Album -niminen liitännäinen [27]. Sen avulla artikkelin yhteyteen voidaan liittää kuvia bloggaajan *Flickr*-kuvapalvelun tililtä. Liitännäisten toimintaa esitellään käyttäen sitä esimerkkinä.

8.1 Liitännäisten rakenne ja toiminta

Julkisen liittymän komponentit asentuvat Joomla-sivuston juurihakemistoon *components*-kansioon ja moduulit *modules*-kansioon. Tästä poiketen liitännäiset asentuvat *plugins*-hakemiston alihakemistoihin sen mukaan, minkä tyyppinen liitännäinen on. Joomlassa on kahdeksan eri tarkoituksiin soveltuvaa liitännäisryhmää. Englanninkielinen termi on kunkin ryhmän alihakemiston nimi.

- Todentamisliitännäisten (authentication) avulla Joomlaan voidaan lisätä käyttäjän tunnistautumismenetelmiä. Joomlan perusasennukseen kuuluu LDAP-, OpenID- ja Gmail-tunnistautumisten lisäksi Joomlan oma tunnistautumismenetelmä.

- Sisältöliitännäisillä (content) voidaan muokata tietokannasta haettua sisältöä ennen sen näyttämistä käyttäjälle. Flickr Album kuuluu tähän ryhmään.
- Editoriliitännäisillä (editor) käytetään sisällön kirjoittamiseen ja tallentamiseen.
- Editorinappulaliitännäiset (editor-button) laajentavat editorin toimintoja esimerkiksi kuvan lisäämiseksi artikkelin yhteyteen.
- Hakuliitännäisiä (search) käytetään sivuston hakutoimintojen toteuttamiseen.
- Järjestelmäliitännäisiä (system) käytetään Joomlan ydintoimintoihin, kuten artikkeleiden julkaisuun sekä järjestelmän ja lisäosien asennukseen.
- Käyttäjäliitännäisten (user) tarjoamia palveluita hyödynnetään eri järjestelmien käyttäjäoikeuksien hallinnassa ja niiden välisissä toiminnoissa.
- XML-RPC-ryhmän liitännäiset ovat vastuussa *XML Remote Procedure Call* -protokollan mukaisista etäyhteyksistä.

Kaikkien liitännäisten tavoin Flickr Album perii Joomlan kirjaston *JPlugin*-luokan [3, s. 137]. *JPlugin*-luokan kaikki asennetut ja liitännäistenylläpitotyökalulla aktivoidut alaluokat rekisteröidään automaattisesti globaaliin *JDispatcher*-luokkaan.

Flick Album käyttää *onPrepareContent*-nimistä kuuntelijaa. Saman niminen kuuntelija on useissa muissakin sisältöliitännäisissä. Kaikkien aktiivisten sisältöliitännäisten *onPrepareContent*-metodit aktivoituvat näytettävän sisällön luontivaiheessa kun sisältökomponentti luo artikkelin näkymää. Esimerkiksi luvussa 5.2.4 esitellyssä *view.html.php*-tiedostossa sijaitsevan *ContentViewArticle*-luokan *display()*-metodin alussa luodaan viite tapahtumankäsittelijäolioon, joka sisältää listan kaikista aktiivisista liitännäisistä [28]:

```
$dispatcher =& JDispatcher::getInstance();
```

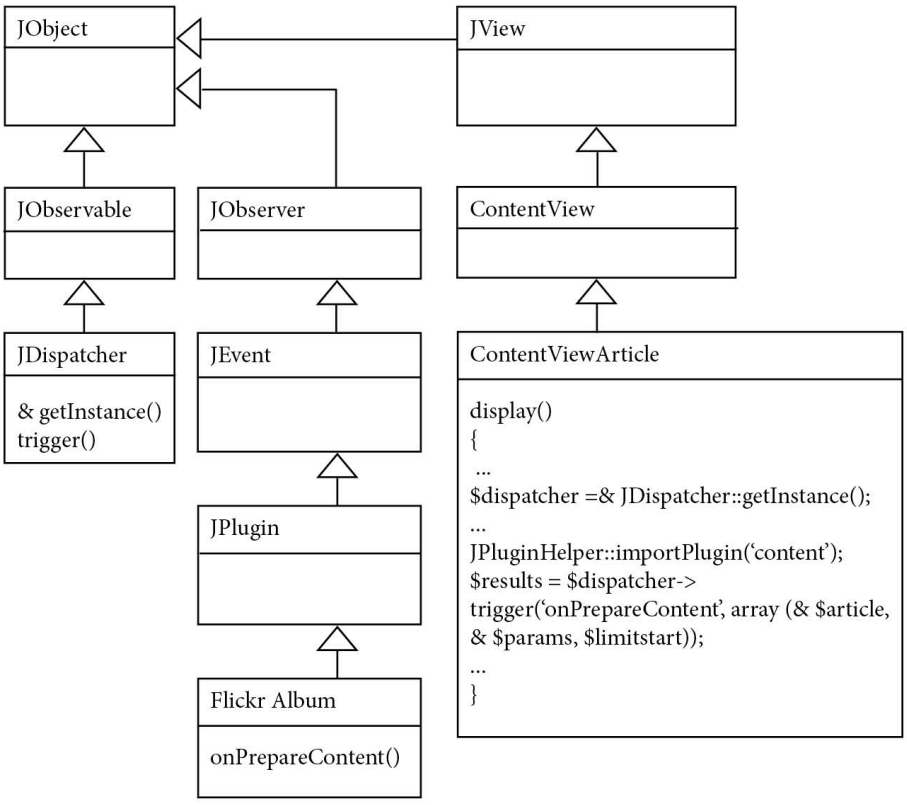
Ennen kuin *view.html.php*-tiedostossa halutaan aktivoida kuuntelija, joka kuuluu sisältöliitännäisiin, on ladattava sisältöliitännäisryhmän kaikki aktiiviset liitännäiset. Se tapahtuu käyttäen *JPluginHelper*-apuluokkaa:

```
JPluginHelper::importPlugin( 'content' );
```

ImportPlugin()-metodi siis lataa sisältöliitännäiskansiosta kaikki ne liitännäiset, jotka ovat ylläpitoliittymän liitännäistenylläpitotyökalulla aktivoitu. Seuraavaksi lähetetään *onPrepareContent*-tapahtumalle heräte:

```
$results = $dispatcher->trigger( 'onPrepareContent', array(
&$artist, &$title ) );
```

Alla olevassa kuvassa [kuva 20] havainnollistetaan liitännäisten tapahtumiin ja tapahtumien käsittelyyn liittyvien luokkien keskinäiset perimissuhteet.



Kuva 20. Liitännäisten kuuntelemiseen liittyvä luokkakaavio.

Kaikille *onPrepareContent*-tapahtumille välitetään parametrina viite sisältöolioon, jonka sisältöä on tarkoitus muokata. Sisältöolio sisältää tietokannasta haetun artikkelin seuraaviin osioihin jaoteltuna [3, s. 144]:

- *created* - sisältöartikkelin luontipäivämäärä
- *modified* - artikkelin muokkauspäivämäärä
- *text* - artikkelin sisältö
- *title* - artikkelin otsikko
- *toc* - monisivuisen artikkelin sisällysluettelo.

Flickr Albumin *onPrepareContent*-tapahtuma hakee viitatun sisältöolion *text*-osiosta *{flickr-album}* ja *{/flickr-album}* merkinnöillä erotettua tekstiä. Merkintöjen avulla välitetään parametreina tietoja siitä, mitä kuvia *Flickristä* halutaan näyttää ja miten ne halutaan esittää. Parametreina voidaan kertoa esimerkiksi, että haluamme näyttää kuvasetin (Flickr Photoset), minkä lisäksi

välitetään kyseisen albumin tunnistetieto, ID-numero: ”*{flickr-album} Type=Photoset, Photoset=72157622435118468 {/flickr-album}”*. Niiden avulla Flickr Album -liitännäinen muodostaa oman XHTML-muotoisen koodin, jolla se korvaa sisältöolion *text*-osiossa olevan merkinnän. Tämän jälkeen muokattu sisältöolio lähetetään eteenpäin selaimelle tulostettavaksi. Vastaavalla tavalla Joomlan sisäinen *Email Cloaking* -liitännäinen muuntaa sisällössä olevat sähköpostiosoitteet automaattisesti JavaScript-muotoon ja piilottaa ne siten roskapostirobottien ulottumattomiin.

Toisin kuin komponenteilla ja moduuleilla, liitännäisillä ei ole vastaavanlaista mekanismia niiden muodostaman koodin ylikirjoittamiseen tyylipohjien avulla. Flickr Album muodostaa tarvittavan XTHML-koodin *flickrAlbum.php*-tiedostossa sijaisevilla funktioilla ja niiden kutsumilla JavaScript-koodeilla. Niihin ei tässä opinnäytetyössä kajota, mutta sen sijaan muutamme liitännäisen omaa *flickrAlbum.css.php*-tiedostoa, joka generoi liitännäisen käyttämän CSS-tyylin. *FlickrAlbum.css.php*-tiedosto sijoittuu liitännäisen asentamisen yhteydessä *plugins/content/flickrAlbum/css*-kansioon.

8.2 Flickr Album -liitännäisen asennus

Flickr Album -liitännäinen asennettiin laajennustenylläpitotyökalulla. Onnistuneen asennuksen jälkeen liitännäistenylläpitotyökalulla Flickr Albumin asetuksiin syötettiin *Flickr*-palvelussa tehty ns. API-avain sekä bloggaajan *Flickr*-tilin käyttäjätunniste [29], [30].

Asetusten määrittelyn jälkeen voidaan jokaisen artikkelin yhteyteen kirjoittaa lyhyt merkintä siihen kohtaan, johon halutaan muodostaa kuvagalleria tietyn Flickr-tilin kuva-albumin kuvista. Esimerkiksi kirjoittamalla *{flickr-album}Type=Photoset, Photoset=72157623795671649{/flickr-album}* noudetaan asetuksissa määritellyn käyttäjän tililtä tunnistetta 72157623795671649 vastaava kuva-albumi.

Kuva 21. Flickr Album hakee kuvia Flickr-tililtä ja muodostaa niistä gallerian.

9 YHTEENVETO

Joomla on erittäin monipuolinen ja mukautuva julkaisualusta. Sen ohjelmistokehys tarjoaa joustavia mahdollisuuksia järjestelmän laajentamiseen tarpeisiin sopivaksi. Siksi Joomla-alusta sopii erinomaisesti myös vaativiin verkkosivuprojekteihin. Sosiaalisen median rajapintojen hyödyntämiseen on Joomlaan tarjolla tukuttain valmiita avoimen lähdekoodin laajennuksia. Bloggaajan käyttämien sosiaalisten medioiden sisällöt saadaan niiden avulla koottua luontevaksi osaksi blogisivustoa. Tyylipohjien avulla sivustosta voidaan suunnitella visuaalisesti täysin yksilöllinen.

Joomla-sivuston asennus ja etenkin laajennusten suunnittelu vaativat hyviä tietoteknisiä taitoja ja ohjelmoinnin osaamista. Sivuston ylläpito sen sijaan onnistuu keneltä tahansa, joka hallitsee Internetin ja tietokoneen käytön perusteet.

Tämän opinnäytetyön aikana asennettiin Joomla-laajennusten kehitys-, testi- ja tuotantoympäristöt. Tuotantoympäristö on julkinen ja se sijaitsee osoitteessa http://someblogger.onnidesign.net.

Työn aikana tehtiin Joomla-tyylipohja, joka soveltuu blogin kirjoittamiseen ja useista lähteistä olevien sisältöjen kokoamiseen. Tyylipohjan sommittelussa hyödynnettiin *960 Grid System* -ruudokkopohjaa. Se toteutettiin Joomlan tyylipohjien suunnitteluun tarkoitetuilla PHP-ehtolauseilla. Tyylipohjan avulla taulukkotaittoon perustuvat Joomlan oletusasemointimallit korvattiin sellaisil-

la asemointimalleilla, joissa on semanttinen XTHML-rakenne. Tyylipohja-asennuspaketti, *someblogger.zip*, siirrettiin ladattavaksi *Joomlaportal.fi*-sivustolle GNU GPL 2.0 -lisenssin alaisena.

VIITELUETTELO

[1] Framework [verkkodokumentti]. 25.2.2010 [viitattu 1.5.2010]. Saatavissa: http://docs.joomla.org/Framework.

[2] Joomla! A Users Guide Building A Successful Joomla! Powered Website, Barrie M. North, Prentice Hall, 2008.

[3] Mastering Joomla! 1.5 Extension And Framework Development, James Kennard, Packt Publishing, 2007.

[4] Professional Joomla!, Dan Rahmel, Wiley Publishing Inc. 2007.

[5] Setting up your workstation for Joomla! Development [verkkodokumentti]. 29.4.2010 [viitattu 2.5.2010]. Saatavissa: http://docs.joomla.org/Setting_up_your_workstation_for_Joomla!_development.

[6] XAMPP [verkkodokumentti]. 15.12.2009 [viitattu 2.5.2010]. Saatavilla: http://www.apachefriends.org.

[7] Setting up your workstation for Joomla! Developmet -- Part 2 [verkkodokumentti]. 2.1.2010 [viitattu 2.5.2010]. Saatavissa: http://docs.joomla.org/Setting_up_your_workstation_for_Joomla!_development_--_Part_2.

[8] Download Joomla! [verkkodokumentti]. [viitattu 23.5.2010]. Saatavissa: http://www.joomla.org/download.html.

[9] Joomla! 1.5 Installation Manual [verkkodokumentti]. 30.10.2007 [viitattu 23.5.2010]. Saatavissa: http://downloads.joomlacode.org/docmanfileversion/1/7/4/17471/1.5_Installation_Manual_version_0.5.pdf.

[10] Article Manager [verkkodokumentti]. 22.11.2009 [viitattu 30.5.2010]. Saatavissa: http://help.joomla.org/content/view/1456/196/.

[11] Learning Joomla! 1.5 Extension Development, Joseph LeBlanc, Packt Publishing, 2007.

[12] Developing a Model View Controller Component - Part 1 [verkkodokumentti]. 15.4.2010 [viitattu 2.5.2010]. Saatavissa: http://docs.joomla.org/Developing_a_Model-View-Controller_Component_-_Part_1#Creating_the_Entry_Point.

[13] _JEXEC explained [verkkodokumentti]. 20.1.2009 [viitattu 3.5.2010]. Saatavilla: http://www.evontech.com/login/topic/542.html.

[14] Understanding Output Overrides [verkkodokumentti]. 27.4.2010 [viitattu 2.5.2010].
Saatavissa: http://docs.joomla.org/Understanding_Output_Overrides.

[15] Miksi taulukkotaitto on typerää [verkkodokumentti]. [viitattu 2.5.2010] Saatavissa: http://www.aimedia.fi/stupid_tables/everything.html.

[16] HTML doctype declaration [verkkodokumentti]. [viitattu 3.5.2010]. Saatavissa: http://w3schools.com/tags/tag_doctype.asp.

[17] 960 Grid System [verkkodokumentti]. 15.5.2010 [viitattu 15.5.2010]. Saatavissa: http://960.gs/.

[18] CSS Tools: Reset CSS [verkkodokumentti]. [viitattu 16.5.2010]. Saatavissa: http://meyerweb.com/eric/tools/css/reset/ .

[19] Joomla! 1.5 Templates - Conditional Statements [verkkodokumentti]. 16.8.2008 [viitattu 16.5.2010]. Saatavilla: http://portal.hrpr.com/joomla-15-templates-conditional-statements.

[20] CSS Clearfix in CSS [verkkodokumentti]. [viitattu 16.5.2010]. Saatavissa: http://www.webtoolkit.info/css-clearfix.html.

[21] Joomla Beez, what is it? [verkkodokumentti]. [viitattu 17.5.2010]. Saatavissa: http://joomla-beez.com/index.php?option=com_content&view=article&id=49.

[22] Unpacking a package file [verkkodokumentti]. 19.1.2008 [viitattu 23.5.2010]. Saatavissa: http://docs.joomla.org/Unpacking_a_package_file.

[23] AppleDouble [verkkodokumentti]. 27.4.2010 [viitattu 23.5.2010]. Saatavissa: http://docs.joomla.org/AppleDouble.

[24] Creating a basic Joomla! Template - Note to Mac OS X users [verkkodokumentti]. 4.5.2009 [viitattu 23.5.2010]. Saatavissa: http://docs.joomla.org/Tutorial:Creating_a_basic_Joomla!_template#Note_to_Mac_OS_X_users.

[25] Security Checklist 4 - Joomla Setup [verkkodokumentti]. 7.4.2010 [viitattu 1.6.2010]. Saatavissa: http://docs.joomla.org/Security_Checklist_4_-_Joomla_Setup.

[26] Tutorial:Plugins [verkkodokumentti]. 30.11.2009 [viitattu 17.9.2010]. Saatavissa: http://docs.joomla.org/Tutorial:Plugins.

[27] Captbunzo’s Flickr Album [verkkodokumentti]. 19.2.2009 [viitattu 3.6.2010]. Saatavissa: http://joomla.paulthompson.net/.

[28] Supporting Plugins In Your Component [verkkodokumentti]. 9.7.2008 [viitattu 18.10.2010]. Saatavissa: http://docs.joomla.org/Supporting_plugins_in_your_component.

[29] The App Garden, API Keys [verkkodokumentti]. [viitattu 23.5.2010]. Saatavissa: http://www.flickr.com/services/api/misc.api_keys.html.

[30] Idgettr [verkkodokumentti], [viitattu 8.9.2010]. Saatavissa: http://idgettr.com.

SOMEBLOGGER-TYYLIPOHJAN INDEX.PHP-KOODI

```
<?php defined( '_JEXEC' ) or die( 'Restricted access' ); ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="<?php echo
$this->language; ?>" lang="<?php echo $this->language; ?>" >
<head>
<jdoc:include type="head" />
<link rel="stylesheet" href="templates/system/css/system.css" ty-
pe="text/css" />
<link rel="stylesheet" href="templates/system/css/general.css" ty-
pe="text/css" />
<link href="templates/<?php echo $this->template?>/css/reset.css"
rel="stylesheet" type="text/css" />
<link href="templates/<?php echo $this->template?>/css/960.css"
rel="stylesheet" type="text/css" />
<link href="templates/<?php echo $this->template?>/css/tyyli.css"
rel="stylesheet" type="text/css" />
<?php
// #left-elementin leveys
$cols=8;
if(!$this->countModules('box_right'))
     { $cols = 12; };
// #box_menu- ja #box_mini-elementtien leveydet
$menucols=5;
$minicols=3;
// Tarkistetaan, onko  box_right-, box_mini- ja
// box_menu-elementeissä sisältöä ja muutetaan
// arvoja tilanteen mukaan sopivaksi.
if($this->countModules('box_right')
|| $this->countModules('box_mini'))
     { $menucols = 8; };
if($this->countModules('box_right')
&& !$this->countModules('box_menu'))
     { $minicols = 8; };
if(!$this->countModules('box_right')
&& $this->countModules('box_menu')
&& $this->countModules('box_mini'))
     { $menucols = 8; $minicols = 4; };
if(!$this->countModules('box_right')
&& !$this->countModules('box_mini'))
     { $menucols = 12; };
if(!$this->countModules('box_right')
&& !$this->countModules('box_menu'))
     { $minicols = 12; };
?>
</head>
<body>
<div id="wrapper" class="container_12">
<div id="left" class="grid_<?php echo $cols; ?> outer">
<?php if( $this->countModules('box_menu')) : ?>
<div id="box_menu" class="grid_<?php echo $menucols; ?>
clearfix alpha <?php if( !$this->countModules('box_mini')) : ?>ome-
ga<?php endif; ?> inner">
<jdoc:include type="modules" name="box_menu" style="xhtml" />
</div>
<?php endif; ?>
<?php if( $this->countModules('box_mini')) : ?>
<div id="box_mini" class="grid_<?php echo $minicols; ?>
clearfix <?php if( !$this->countModules('box_menu')) : ?>alpha
<?php endif; ?> omega inner">
<jdoc:include type="modules" name="box_mini" style="xhtml" />
</div>
<?php endif; ?>
```

```
<div id="box_content" class="grid_<?php echo $cols; ?> clearfix alpha
omega inner">
<div id="sisalto"><jdoc:include type="component" /></div>
</div>
<?php if($this->countModules('box_footer')) : ?>
<div id="box_footer" class="grid_<?php echo $cols; ?> clearfix alpha
omega inner">
<jdoc:include type="modules" name="box_footer" style="xhtml" />
</div>
<?php endif; ?>
</div>
<?php if($this->countModules('box_right')) : ?>
<div id="right" class="grid_4 outer">
<div id="box_right" class="grid_4 clearfix alpha omega inner">
<jdoc:include type="modules" name="box_right" style="xhtml" />
 </div>
</div>
<?php endif; ?>
<?php if($this->countModules('debug')) : ?>
<div id="debug" class="grid_12 clearfix outer">
<jdoc:include type="modules" name="debug" style="xhtml" />
</div>
<?php endif; ?>
</div>
</body>
</html>
```

COM_CONTENT-KOMPONENTIN VIEW.HTML.PHP-KOODI

```
<?php

// Check to ensure this file is included in Joomla!
defined('_JEXEC') or die( 'Restricted access' );


// Tuodaan view.php.
require_once (JPATH_COMPONENT.DS.'view.php');

/**
 * HTML Article View class for the Content component
 *
 * @package             Joomla
 * @subpackage          Content
 * @since 1.5
 */


// Luodaan ContentViewArticle luokka, joka perii ContentView-luokan
ominaisuudet.
class ContentViewArticle extends ContentView
{
  function display($tpl = null)
  {
    global $mainframe;

    $user   =& JFactory::getUser();
    $document           =& JFactory::getDocument();
    $dispatcher         =& JDispatcher::getInstance();
    $pathway            =& $mainframe->getPathway();
    $params             =& $mainframe->getParams('com_content');

    // Initialize variables
    $article            =& $this->get('Article');
    $aparams                        =& $article->parameters;
    $params->merge($aparams);

    if($this->getLayout() == 'pagebreak') {
      $this->_displayPagebreak($tpl);
      return;
    }

    if($this->getLayout() == 'form') {
      $this->_displayForm($tpl);
      return;
    }

    if (($article->id == 0))
    {
      $id = JRequest::getVar( 'id', '', 'default', 'int' );
      return JError::raiseError( 404, JText::sprintf( 'Article # not
found', $id ) );
    }

    $limitstart         = JRequest::getVar('limitstart', 0, '',
'int');

    if (!$params->get('intro_only') && ($this->getLayout() == 'de-
fault') && ($limitstart == 0))
    {
      $model =& $this->getModel();
      $model->hit();
    }

    // Create a user access object for the current user
    $access = new stdClass();
```

```
    $access->canEdit    = $user->authorize('com_content', 'edit',
'content', 'all');
    $access->canEditOwn = $user->authorize('com_content', 'edit',
'content', 'own');
    $access->canPublish = $user->authorize('com_content', 'publish',
'content', 'all');

    // Check to see if the user has access to view the full article
    $aid= $user->get('aid');

    if ($article->access <= $aid) {
      $article->readmore_link = JRou-
te::_(ContentHelperRoute::getArticleRoute($article->slug, $article-
>catslug, $article->sectionid));;
    } else {
      if ( ! $aid )
      {
        // Redirect to login
        $uri                        = JFactory::getURI();
        $return                     = $uri->toString();

        $url  = 'index.php?option=com_user&view=login';
        $url .= '&return='.base64_encode($return);;

        //$url          = JRoute::_($url, false);
        $mainframe->redirect($url, JText::_('You must login first')
);
      }
      else{
        JError::raiseWarning( 403, JText::_('ALERTNOTAUTH') );
        return;
      }
    }

    /*
     * Process the prepare content plugins
     */
    JPluginHelper::importPlugin('content');
    $results = $dispatcher->trigger('onPrepareContent', array (& $ar-
ticle, & $params, $limitstart));

    /*
     * Handle the metadata
     */
    // because the application sets a default page title, we need to
get it
    // right from the menu item itself
    // Get the menu item object
    $menus = &JSite::getMenu();
    $menu  = $menus->getActive();

    if (is_object( $menu ) && isset($menu->query['view']) && $menu-
>query['view'] == 'article' && isset($menu->query['id']) && $menu-
>query['id'] == $article->id) {
      $menu_params = new JParameter( $menu->params );
      if (!$menu_params->get( 'page_title')) {
        $params->set('page_title',  $article->title);
      }
    } else {
      $params->set('page_title',    $article->title);
    }
    $document->setTitle( $params->get( 'page_title' ) );

    if ($article->metadesc) {
      $document->setDescription( $article->metadesc );
    }
    if ($article->metakey) {
      $document->setMetadata('keywords', $article->metakey);
    }
```

```
    if ($mainframe->getCfg('MetaTitle') == '1') {
      $mainframe->addMetaTag('title', $article->title);
    }
    if ($mainframe->getCfg('MetaAuthor') == '1') {
      $mainframe->addMetaTag('author', $article->author);
    }

    $mdata = new JParameter($article->metadata);
    $mdata = $mdata->toArray();
    foreach ($mdata as $k => $v)
    {
      if ($v) {
        $document->setMetadata($k, $v);
      }
    }

    // If there is a pagebreak heading or title, add it to the page
title
    if (!empty($article->page_title))
    {
      $article->title = $article->title .' - '. $article-
>page_title;
      $document->setTitle($article->page_title.' -
'.JText::sprintf('Page %s', $limitstart + 1));
    }

    /*
     * Handle the breadcrumbs
     */
    if($menu && $menu->query['view'] != 'article')
    {
      switch ($menu->query['view'])
      {
        case 'section':
          $pathway->addItem($article->category, 'in-
dex.php?view=category&id='.$article->catslug);
          $pathway->addItem($article->title, '');
          break;
        case 'category':
          $pathway->addItem($article->title, '');
          break;
      }
    }

    /*
     * Handle display events
     */
    $article->event = new stdClass();
    $results = $dispatcher->trigger('onAfterDisplayTitle', array
(&$article, &$params, $limitstart));
    $article->event->afterDisplayTitle = trim(implode("\n", $re-
sults));

    $results = $dispatcher->trigger('onBeforeDisplayContent', array
(&$article, &$params, $limitstart));
    $article->event->beforeDisplayContent = trim(implode("\n", $re-
sults));

    $results = $dispatcher->trigger('onAfterDisplayContent', array
(&$article, &$params, $limitstart));
    $article->event->afterDisplayContent = trim(implode("\n", $re-
sults));

    $print = JRequest::getBool('print');
    if ($print) {
      $document->setMetaData('robots', 'noindex, nofollow');
    }

    $this->assignRef('article', $article);
    $this->assignRef('params' , $params);
```

```
    $this->assignRef('user'   , $user);
    $this->assignRef('access' , $access);
    $this->assignRef('print', $print);

    parent::display($tpl);
  }

  function _displayForm($tpl)
  {
    global $mainframe;

    // Initialize variables
    $document           =& JFactory::getDocument();
    $user   =& JFactory::getUser();
    $uri  =& JFactory::getURI();
    $params             =& $mainframe->getParams('com_content');

    // Make sure you are logged in and have the necessary access
rights
    if ($user->get('gid') < 19) {
      JResponse::setHeader('HTTP/1.0 403',true);
        JError::raiseWarning( 403, JText::_('ALERTNOTAUTH') );
      return;
    }

    // Initialize variables
    $article            =& $this->get('Article');
    $aparams            =& $article->parameters;
    $isNew  = ($article->id < 1);

    $params->merge($aparams);

    // At some point in the future this will come from a request ob-
ject
    $limitstart         = JRequest::getVar('limitstart', 0, '',
'int');

    // Add the Calendar includes to the document <head> section
    JHTML::_('behavior.calendar');

    if ($isNew)
    {
      // TODO: Do we allow non-sectioned articles from the fron-
tend??
      $article->sectionid = JRequest::getVar('sectionid', 0, '',
'int');
      $db = JFactory::getDBO();
      $db->setQuery('SELECT title FROM #__sections WHERE id =
'.(int) $article->sectionid);
      $article->section = $db->loadResult();
    }

    // Get the lists
    $lists = $this->_buildEditLists();

    // Load the JEditor object
    $editor =& JFactory::getEditor();

    // Build the page title string
    $title = $article->id ? JText::_('Edit') : JText::_('New');

    // Set page title
    // because the application sets a default page title, we need to
get it
    // right from the menu item itself
    // Get the menu item object
    $menus = &JSite::getMenu();
    $menu  = $menus->getActive();
    $params->set( 'page_title', $params->get( 'page_title' ) );
    if (is_object( $menu )) {
```

```
      $menu_params = new JParameter( $menu->params );
      if (!$menu_params->get( 'page_title')) {
        $params->set('page_title',  JText::_( 'Submit an Article' ));
      }
    } else {
      $params->set('page_title', JText::_( 'Submit an Article' ));
    }
    $document->setTitle( $params->get( 'page_title' ) );

    // get pathway
    $pathway =& $mainframe->getPathWay();
    $pathway->addItem($title, '');

    // Unify the introtext and fulltext fields and separated the
fields by the {readmore} tag
    if (JString::strlen($article->fulltext) > 1) {
      $article->text = $article->introtext."<hr id=\"system-
readmore\" />".$article->fulltext;
    } else {
      $article->text = $article->introtext;
    }

    $this->assign('action',         $uri->toString());

    $this->assignRef('article',     $article);
    $this->assignRef('params',      $params);
    $this->assignRef('lists',       $lists);
    $this->assignRef('editor',      $editor);
    $this->assignRef('user',        $user);


    parent::display($tpl);
  }

  function _buildEditLists()
  {
    // Get the article and database connector from the model
    $article = & $this->get('Article');
    $db  = & JFactory::getDBO();

    $javascript = "onchange=\"changeDynaList( 'catid', sectioncatego-
ries, docu-
ment.adminForm.sectionid.options[document.adminForm.sectionid.selecte
dIndex].value, 0, 0);\"";

    $query = 'SELECT s.id, s.title' .
        ' FROM #__sections AS s' .
        ' ORDER BY s.ordering';
    $db->setQuery($query);

    $sections[] = JHTML::_('select.option', '-1', '-
'.JText::_('Select Section').' -', 'id', 'title');
    $sections[] = JHTML::_('select.option', '0',
JText::_('Uncategorized'), 'id', 'title');
    $sections = array_merge($sections, $db->loadObjectList());
    $lists['sectionid'] = JHTML::_('select.genericlist',  $sections,
'sectionid', 'class="inputbox" size="1" '.$javascript, 'id', 'title',
intval($article->sectionid));

    foreach ($sections as $section)
    {
      $section_list[] = (int) $section->id;
      // get the type name - which is a special category
      if ($article->sectionid) {
        if ($section->id == $article->sectionid) {
          $contentSection = $section->title;
        }
      } else {
        if ($section->id == $article->sectionid) {
          $contentSection = $section->title;
```

```
        }
      }
    }

    $sectioncategories = array ();
    $sectioncategories[-1] = array ();
    $sectioncategories[-1][] = JHTML::_('select.option', '-1',
JText::_( 'Select Category' ), 'id', 'title');
    $section_list = implode('\', \'', $section_list);

    $query = 'SELECT id, title, section' .
        ' FROM #__categories' .
        ' WHERE section IN ( \''.$section_list.'\' )' .
        ' ORDER BY ordering';
    $db->setQuery($query);
    $cat_list = $db->loadObjectList();

    // Uncategorized category mapped to uncategorized section
    $uncat = new stdClass();
    $uncat->id = 0;
    $uncat->title = JText::_('Uncategorized');
    $uncat->section = 0;
    $cat_list[] = $uncat;
    foreach ($sections as $section)
    {
      $sectioncategories[$section->id] = array ();
      $rows2 = array ();
      foreach ($cat_list as $cat)
      {
        if ($cat->section == $section->id) {
          $rows2[] = $cat;
        }
      }
      foreach ($rows2 as $row2) {
        $sectioncategories[$section->id][] =
JHTML::_('select.option', $row2->id, $row2->title, 'id', 'title');
      }
    }

    $categories = array();
    foreach ($cat_list as $cat) {
      if($cat->section == $article->sectionid)
        $categories[] = $cat;
    }

    $categories[] = JHTML::_('select.option', '-1', JText::_( 'Select
Category' ), 'id', 'title');
    $lists['sectioncategories'] = $sectioncategories;
    $lists['catid'] = JHTML::_('select.genericlist',  $categories,
'catid', 'class="inputbox" size="1"', 'id', 'title', intval($article-
>catid));

    // Select List: Category Ordering
    $query = 'SELECT ordering AS value, title AS text FROM #__content
WHERE catid = '.(int) $article->catid.' AND state > ' .(int) "-1" . '
ORDER BY ordering';
    $lists['ordering'] = JHTML::_('list.specificordering', $article,
$article->id, $query, 1);

    // Radio Buttons: Should the article be published
    $lists['state'] = JHTML::_('select.booleanlist', 'state', '',
$article->state);

    // Radio Buttons: Should the article be added to the frontpage
    if($article->id) {
      $query = 'SELECT content_id FROM #__content_frontpage WHERE
content_id = '. (int) $article->id;
      $db->setQuery($query);
      $article->frontpage = $db->loadResult();
    } else {
```

```
      $article->frontpage = 0;
    }

    $lists['frontpage'] = JHTML::_('select.booleanlist', 'frontpage',
'', (boolean) $article->frontpage);

    // Select List: Group Access
    $lists['access'] = JHTML::_('list.accesslevel', $article);

    return $lists;
  }

  function _displayPagebreak($tpl)
  {
    $document =& JFactory::getDocument();
    $document->setTitle(JText::_('PGB ARTICLE PAGEBRK'));

    parent::display($tpl);
  }
}
```

COM_CONTENT-KOMPONENTIN VIEW.PHP-KOODI

```
<?php
defined('_JEXEC') or die('Restricted access');

// Tuodaan Joomlan kirjastosta view.php.
jimport( 'joomla.application.component.view');

// Luodaan luokka ContentView, joka perii edellä tuodussa view.php-
// tiedostossa olevan JView-luokan.
class ContentView extends JView
{
  function __construct($config = array())
  {
    parent::__construct($config);

    //Add the helper path to the JHTML library
    JHTML::addIncludePath(JPATH_COMPONENT.DS.'helpers');
  }
}
```

JOOMLAN KIRJASTON VIEW.PHP-KOODI:

```
<?php
/**
 * @version             $Id: view.php 14401 2010-01-26 14:10:00Z lo-
uis $
 * @package             Joomla.Framework
 * @subpackage          Application
 * @copyright Copyright Copyright (C) 2005 - 2010 Open Source Mat-
ters. All rights reserved.
 * @license             GNU/GPL, see LICENSE.php
 * Joomla! is free software. This version may have been modified pur-
suant
 * to the GNU General Public License, and as distributed it includes
or
 * is derivative of works licensed under the GNU General Public Li-
cense or
 * other free or open source software licenses.
 * See COPYRIGHT.php for copyright notices and details.
 */

// Check to ensure this file is within the rest of the framework
defined('JPATH_BASE') or die();

/**
 * Base class for a Joomla View
```

```
 *
 * Class holding methods for displaying presentation data.
 *
 * @abstract
 * @package             Joomla.Framework
 * @subpackage          Application
 * @since   1.5
 */
class JView extends JObject
{
  /**
   * The name of the view
   *
   * @var   array
   * @access protected
   */
  var $_name = null;

  /**
   * Registered models
   *
   * @var   array
   * @access protected
   */
  var $_models = array();

  /**
   * The base path of the view
   *
   * @var   string
   * @access            protected
   */
  var $_basePath = null;

  /**
   * The default model
   *
   * @var string
   * @access protected
   */
  var $_defaultModel = null;

  /**
   * Layout name
   *
   * @var   string
   * @access            protected
   */
  var $_layout = 'default';

  /**
   * Layout extension
   *
   * @var   string
   * @access            protected
   */
  var $_layoutExt = 'php';

  /**
  * The set of search directories for resources (templates)
  *
  * @var array
  * @access protected
  */
  var $_path = array(
    'template' => array(),
    'helper' => array()
  );

  /**
```

```
  * The name of the default template source file.
  *
  * @var string
  * @access private
  */
  var $_template = null;

  /**
  * The output of the template script.
  *
  * @var string
  * @access private
  */
  var $_output = null;

  /**
     * Callback for escaping.
     *
     * @var string
     * @access private
     */
    var $_escape = 'htmlspecialchars';

   /**
     * Charset to use in escaping mechanisms; defaults to urf8 (UTF-
8)
     *
     * @var string
     * @access private
     */
    var $_charset = 'UTF-8';

  /**
   * Constructor
   *
   * @access            protected
   */
  function __construct($config = array())
  {
    //set the view name
    if (empty( $this->_name ))
    {
      if (array_key_exists('name', $config))  {
        $this->_name = $config['name'];
      } else {
        $this->_name = $this->getName();
      }
    }

     // set the charset (used by the variable escaping functions)
        if (array_key_exists('charset', $config)) {
            $this->_charset = $config['charset'];
        }

     // user-defined escaping callback
        if (array_key_exists('escape', $config)) {
            $this->setEscape($config['escape']);
        }

    // Set a base path for use by the view
    if (array_key_exists('base_path', $config)) {
      $this->_basePath  = $config['base_path'];
    } else {
      $this->_basePath  = JPATH_COMPONENT;
    }

    // set the default template search path
    if (array_key_exists('template_path', $config)) {
      // user-defined dirs
      $this->_setPath('template', $config['template_path']);
```

```
    } else {
      $this->_setPath('template', $this-
>_basePath.DS.'views'.DS.$this->getName().DS.'tmpl');
    }

    // set the default helper search path
    if (array_key_exists('helper_path', $config)) {
      // user-defined dirs
      $this->_setPath('helper', $config['helper_path']);
    } else {
      $this->_setPath('helper', $this->_basePath.DS.'helpers');
    }

    // set the layout
    if (array_key_exists('layout', $config)) {
      $this->setLayout($config['layout']);
    } else {
      $this->setLayout('default');
    }

    $this->baseurl = JURI::base(true);
  }

  /**
  * Execute and display a template script.
  *
  * @param string $tpl The name of the template file to parse;
  * automatically searches through the template paths.
  *
  * @throws object An JError object.
  * @see fetch()
  */
  /* JView-luokan display-luokan display-metodi, jonka ContentViewAr-
ticle-luokka ylikirjoittaa, mutta jota sen lopussa kutsutaan uudel-
leen. */
  function display($tpl = null)
  {
    $result = $this->loadTemplate($tpl);
    if (JError::isError($result)) {
      return $result;
    }

    echo $result;
  }

  /**
  * Assigns variables to the view script via differing strategies.
  *
  * This method is overloaded; you can assign all the properties of
  * an object, an associative array, or a single value by name.
  *
  * You are not allowed to set variables that begin with an undersco-
re;
  * these are either private properties for JView or private varia-
bles
  * within the template script itself.
  *
  * <code>
  * $view = new JView();
  *
  * // assign directly
  * $view->var1 = 'something';
  * $view->var2 = 'else';
  *
  * // assign by name and value
  * $view->assign('var1', 'something');
  * $view->assign('var2', 'else');
  *
  * // assign by assoc-array
  * $ary = array('var1' => 'something', 'var2' => 'else');
```

```
  * $view->assign($obj);
  *
  * // assign by object
  * $obj = new stdClass;
  * $obj->var1 = 'something';
  * $obj->var2 = 'else';
  * $view->assign($obj);
  *
  * </code>
  *
  * @access public
  * @return bool True on success, false on failure.
  */
  function assign()
  {
    // get the arguments; there may be 1 or 2.
    $arg0 = @func_get_arg(0);
    $arg1 = @func_get_arg(1);

    // assign by object
    if (is_object($arg0))
    {
      // assign public properties
      foreach (get_object_vars($arg0) as $key => $val)
      {
        if (substr($key, 0, 1) != '_') {
          $this->$key = $val;
        }
      }
      return true;
    }

    // assign by associative array
    if (is_array($arg0))
    {
      foreach ($arg0 as $key => $val)
      {
        if (substr($key, 0, 1) != '_') {
          $this->$key = $val;
        }
      }
      return true;
    }

    // assign by string name and mixed value.

    // we use array_key_exists() instead of isset() becuase isset()
    // fails if the value is set to null.
    if (is_string($arg0) && substr($arg0, 0, 1) != '_' &&
func_num_args() > 1)
    {
      $this->$arg0 = $arg1;
      return true;
    }

    // $arg0 was not object, array, or string.
    return false;
  }


  /**
  * Assign variable for the view (by reference).
  *
  * You are not allowed to set variables that begin with an undersco-
re;
  * these are either private properties for JView or private varia-
bles
  * within the template script itself.
  *
  * <code>
```

```
  * $view = new JView();
  *
  * // assign by name and value
  * $view->assignRef('var1', $ref);
  *
  * // assign directly
  * $view->ref =& $var1;
  * </code>
  *
  * @access public
  *
  * @param string $key The name for the reference in the view.
  * @param mixed &$val The referenced variable.
  *
  * @return bool True on success, false on failure.
  */

  function assignRef($key, &$val)
  {
    if (is_string($key) && substr($key, 0, 1) != '_')
    {
      $this->$key =& $val;
      return true;
    }

    return false;
  }

  /**
     * Escapes a value for output in a view script.
     *
     * If escaping mechanism is one of htmlspecialchars or htmlenti-
ties, uses
     * {@link $_encoding} setting.
     *
     * @param  mixed $var The output to escape.
     * @return mixed The escaped value.
     */
    function escape($var)
    {
        if (in_array($this->_escape, array('htmlspecialchars',
'htmlentities'))) {
            return call_user_func($this->_escape, $var, ENT_COMPAT,
$this->_charset);
        }

        return call_user_func($this->_escape, $var);
    }

  /**
   * Method to get data from a registered model or a property of the
view
   *
   * @access            public
   * @param string      The name of the method to call on the model,
or the property to get
   * @param string      The name of the model to reference, or the
default value [optional]
   * @return mixed      The return value of the method
   */
  function &get( $property, $default = null )
  {

    // If $model is null we use the default model
    if (is_null($default)) {
      $model = $this->_defaultModel;
    } else {
      $model = strtolower( $default );
    }
```

```
    // First check to make sure the model requested exists
    if (isset( $this->_models[$model] ))
    {
      // Model exists, lets build the method name
      $method = 'get'.ucfirst($property);

      // Does the method exist?
      if (method_exists($this->_models[$model], $method))
      {
        // The method exists, lets call it and return what we get
                $result = $this->_models[$model]->$method();
                return $result;
      }

    }

    // degrade to JObject::get
    $result = parent::get( $property, $default );
    return $result;

  }

  /**
   * Method to get the model object
   *
   * @access            public
   * @param string      $name       The name of the model (optional)
   * @return            mixed                                   JModel
object
   */
  function &getModel( $name = null )
  {
    if ($name === null) {
      $name = $this->_defaultModel;
    }
    return $this->_models[strtolower( $name )];
  }

  /**
  * Get the layout.
  *
  * @access public
  * @return string The layout name
  */

  function getLayout()
  {
    return $this->_layout;
  }

  /**
   * Method to get the view name
   *
   * The model name by default parsed using the classname, or it can
be set
   * by passing a $config['name'] in the class constructor
   *
   * @access            public
   * @return            string The name of the model
   * @since 1.5
   */
  function getName()
  {
    $name = $this->_name;

    if (empty( $name ))
    {
      $r = null;
      if (!preg_match('/View((view)*(.*(view)?.*))$/i',
get_class($this), $r)) {
```

```
        JError::raiseError (500, "JView::getName() : Cannot get or
parse class name.");
      }
      if (strpos($r[3], "view"))
      {
        JError::raiseWarning('SOME_ERROR_CODE',"JView::getName() :
Your classname contains the substring 'view'. ".

  "This causes problems when extracting the classname from the name
of your objects view. " .

  "Avoid Object names with the substring 'view'.");
      }
      $name = strtolower( $r[3] );
    }

    return $name;
  }

  /**
   * Method to add a model to the view.  We support a multiple model
single
   * view system by which models are referenced by classname.  A ca-
veat to the
   * classname referencing is that any classname prepended by JModel
will be
   * referenced by the name without JModel, eg. JModelCategory is
just
   * Category.
   *
   * @access            public
   * @param object      $model                  The model to add to
the view.
   * @param boolean     $default    Is this the default model?
   * @return            object
  The added model
   */
  function &setModel( &$model, $default = false )
  {
    $name = strtolower($model->getName());
    $this->_models[$name] = &$model;

    if ($default) {
      $this->_defaultModel = $name;
    }
    return $model;
  }

  /**
  * Sets the layout name to use
  *
  * @access public
  * @param string $template The template name.
  * @return string Previous value
  * @since 1.5
  */

  function setLayout($layout)
  {
    $previous                       = $this->_layout;
    $this->_layout = $layout;
    return $previous;
  }

  /**
   * Allows a different extension for the layout files to be used
   *
   * @access            public
   * @param string      The extension
   * @return            string      Previous value
```

```
   * @since 1.5
   */
  function setLayoutExt( $value )
  {
    $previous           = $this->_layoutExt;
    if ($value = preg_replace( '#[^A-Za-z0-9]#', '', trim( $value )
)) {
      $this->_layoutExt = $value;
    }
    return $previous;
  }

   /**
    * Sets the _escape() callback.
    *
    * @param mixed $spec The callback for _escape() to use.
    */
    function setEscape($spec)
    {
        $this->_escape = $spec;
    }

  /**
   * Adds to the stack of view script paths in LIFO order.
   *
   * @param string|array The directory (-ies) to add.
   * @return void
   */
  function addTemplatePath($path)
  {
    $this->_addPath('template', $path);
  }

  /**
   * Adds to the stack of helper script paths in LIFO order.
   *
   * @param string|array The directory (-ies) to add.
   * @return void
   */
  function addHelperPath($path)
  {
    $this->_addPath('helper', $path);
  }

  /**
   * Load a template file -- first look in the templates folder for
an override
   *
   * @access            public
   * @param string $tpl The name of the template source file ...
   * automatically searches the template paths and compiles as nee-
ded.
   * @return string The output of the the template script.
   */

// loadTemplate-funktiolla haetaan käytettävä asettelumalli.
  function loadTemplate( $tpl = null)
  {
    global $mainframe, $option;

    // clear prior output
    $this->_output = null;

    //create the template file name based on the layout
    $file = isset($tpl) ? $this->_layout.'_'.$tpl : $this->_layout;
    // clean the file name
    $file = preg_replace('/[^A-Z0-9_\.-]/i', '', $file);
    $tpl  = preg_replace('/[^A-Z0-9_\.-]/i', '', $tpl);

    // load the template script
```

```
    jimport('joomla.filesystem.path');
    $filetofind         = $this->_createFileName('template', ar-
ray('name' => $file));
    $this->_template = JPath::find($this->_path['template'], $fileto-
find);

    if ($this->_template != false)
    {
      // unset so as not to introduce into template scope
      unset($tpl);
      unset($file);

      // never allow a 'this' property
      if (isset($this->this)) {
        unset($this->this);
      }

      // start capturing output into a buffer
      ob_start();
      // include the requested template filename in the local scope
      // (this will execute the view logic).
      include $this->_template;

      // done with the requested template; get the buffer and
      // clear it.
      $this->_output = ob_get_contents();
      ob_end_clean();

      return $this->_output;
    }
    else {
      return JError::raiseError( 500, 'Layout "' . $file . '" not
found' );
    }
  }

  /**
   * Load a helper file
   *
   * @access            public
   * @param string $tpl The name of the helper source file ...
   * automatically searches the helper paths and compiles as needed.
   * @return boolean Returns true if the file was loaded
   */
  function loadHelper( $hlp = null)
  {
    // clean the file name
    $file = preg_replace('/[^A-Z0-9_\.-]/i', '', $hlp);

    // load the template script
    jimport('joomla.filesystem.path');
    $helper = JPath::find($this->_path['helper'], $this-
>_createFileName('helper', array('name' => $file)));

    if ($helper != false)
    {
      // include the requested template filename in the local scope
      include_once $helper;
    }
  }

  /**
  * Sets an entire array of search paths for templates or resources.
  *
  * @access protected
  * @param string $type The type of path to set, typically 'templa-
te'.
  * @param string|array $path The new set of search paths.  If null
or
  * false, resets to the current directory only.
```

```
  */
  function _setPath($type, $path)
  {
    global $mainframe, $option;

    // clear out the prior search dirs
    $this->_path[$type] = array();

    // actually add the user-specified directories
    $this->_addPath($type, $path);

    // always add the fallback directories as last resort
    switch (strtolower($type))
    {
      case 'template':
      {
        // set the alternative template search dir
        if (isset($mainframe))
        {
          $option = preg_replace('/[^A-Z0-9_\.-]/i', '', $option);
          $fallback = JPATH_BASE.DS.'templates'.DS.$mainframe-
>getTemplate().DS.'html'.DS.$option.DS.$this->getName();
          $this->_addPath('template', $fallback);
        }
      } break;
    }
  }

  /**
  * Adds to the search path for templates and resources.
  *
  * @access protected
  * @param string|array $path The directory or stream to search.
  */
  function _addPath($type, $path)
  {
    // just force to array
    settype($path, 'array');

    // loop through the path directories
    foreach ($path as $dir)
    {
      // no surrounding spaces allowed!
      $dir = trim($dir);

      // add trailing separators as needed
      if (substr($dir, -1) != DIRECTORY_SEPARATOR) {
        // directory
        $dir .= DIRECTORY_SEPARATOR;
      }

      // add to the top of the search dirs
      array_unshift($this->_path[$type], $dir);
    }
  }

  /**
   * Create the filename for a resource
   *
   * @access private
   * @param string      $type  The resource type to create the fi-
lename for
   * @param array       $parts An associative array of filename in-
formation
   * @return string The filename
   * @since 1.5
   */
  function _createFileName($type, $parts = array())
  {
    $filename = '';
```

```
    switch($type)
    {
      case 'template' :
        $filename = strtolower($parts['name']).'.'.$this-
>_layoutExt;
        break;

      default :
        $filename = strtolower($parts['name']).'.php';
        break;
    }
    return $filename;
  }
}
```

TYYLI.CSS

```
@CHARSET "UTF-8";

body {
  font: 1em/1.5em Georgia,"Times New Roman",Times,serif;
  line-height: 140%;
  background: url(../images/tausta_ala.gif) #ccc;
}

a:link { color: #036; }
a:visited { color: #352B66; }
a:hover { color: #990; text-decoration: none; }
a:active { color: #900; }

.twitter {
  font-size: 0.9em;
  }

a img { border: 0; }

div.outer { margin-top: 20px; }

div#left .moduletable, div#left div#sisalto {
  margin-bottom: 20px;
  border: 1px dashed #b11;
  background-color: #FFF;
  padding: 20px;
}

div#box_right .moduletable {
  margin-bottom: 20px;
  padding: 20px;
  border: 1px dashed #b11;
  background-color: #fff;
  }

h2.contentheading {
  font-size: 35px;
  font-weight: normal;
  font-family:Tahoma, Geneva, sans-serif;
  color: #900;
  padding-bottom: 20px;
  }

h2.contentheading a:link, h2.contentheading a:visited {
  color: #800;
  }

h2.contentheading a:hover {
  color: #c00;
  text-decoration: none;
  }

p.articleinfo, p.iteminfo, p.pageinfo {
  color: #666;
  font-size: 0.8em;
  }

p.articleinfo, p.pageinfo {
  margin-bottom: 10px;
  }

p.articleinfo a:link, p.articleinfo a:visited, p.pageinfo a:link,
p.pageinfo a:visited {
```

```
  color: #666;
  }

p.articleinfo a:hover, p.pageinfo a:hover {
  color: #900;
  text-decoration: none;
  }

#roktwittie .bio {
  padding-top: 10px;
  padding-bottom: 10px;
  display: block !important;
  }
```

TEMPLATE_DETAILS.XML

```
<?xml version="1.0" encoding="utf-8"?>
<install version="1.5" type="template">
  <name>someblogger</name>
   <creationDate>May 2010</creationDate>
   <author>Pasi Heiskanen</author>
   <authorEmail>pasi.heiskanen@gmail.com</authorEmail>
   <authorUrl>http://www.onnikoo.fi</authorUrl>
   <license>GNU/GPL version 2</license>
   <description>HTML overrides by Angie Radtke/Robert Deutz, joom-
    la@run-digital.com, http://www.run-digital.com</description>
   <files>
      <filename>index.php</filename>
      <filename>templateDetails.xml</filename>
      <filename>template_thumbnail.png</filename>
       <filename>css/960.css</filename>
       <filename>css/tyyli.css</filename>
       <filename>css/reset.css</filename>
       <filename>html/index.html</filename>
       <filename>html/com_content/index.html</filename>
       <filename>html/com_content/article/default.php</filename>
       <filename>html/com_content/article/index.html</filename>
       <filename>html/com_content/article/form.php</filename>
       <filename>html/com_content/category/index.html</filename>
       <filename>html/com_content/category/blog.php</filename>
       <filename>html/com_content/category/blog_item.php</filename>
       <filename>html/com_content/category/blog_links.php</filename>
       <filename>html/com_content/category/default_items.php
         </filename>
       <filename>html/com_content/category/default.php</filename>
       <filename>html/com_content/frontpage/index.html</filename>
       <filename>html/com_content/frontpage/default.php</filename>
       <filename>html/com_content/frontpage/default_item.php
         </filename>
       <filename>html/com_content/frontpage/default_links.php
         </filename>
       <filename>html/com_content/section/index.html</filename>
       <filename>html/com_content/section/blog.php</filename>
       <filename>html/com_content/section/blog_item.php</filename>
       <filename>html/com_content/section/blog_links.php</filename>
       <filename>html/com_content/section/default.php</filename>
       <filename>html/com_search/search/default.php</filename>
       <filename>html/com_search/search/default_error.php</filename>
       <filename>html/com_search/search/default_form.php</filename>
       <filename>html/com_search/search/default_results.php
         </filename>
       <filename>html/com_search/search/index.html</filename>
       <filename>html/com_search/index.html</filename>
       <filename>html/editor_content.css</filename>
       <filename>html/mod_latestnews/default.php</filename>
       <filename>html/mod_latestnews/index.html</filename>
       <filename>html/mod_login/default.php</filename>
       <filename>html/mod_login/index.html</filename>
       <filename>html/mod_newsflash/_item.php</filename>
       <filename>html/mod_newsflash/default.php</filename>
       <filename>html/mod_newsflash/horiz.php</filename>
       <filename>html/mod_newsflash/vert.php</filename>
       <filename>html/mod_newsflash/index.html</filename>
       <filename>html/mod_poll/default.php</filename>
       <filename>html/mod_poll/index.html</filename>
       <filename>html/mod_feed/default.php</filename>
       <filename>html/mod_search/default.php</filename>
       <filename>html/mod_search/index.html</filename>
       <filename>html/modules.php</filename>
       <filename>html/pagination.php</filename>
       <filename>html/com_poll/poll/default.php</filename>
       <filename>html/com_poll/poll/default_graph.php</filename>
```

```
      <filename>html/com_poll/poll/index.html</filename>
      <filename>html/com_poll/index.html</filename>
      <filename>html/com_newsfeeds/categories/default.php</filename>
      <filename>html/com_newsfeeds/categories/index.html</filename>
      <filename>html/com_newsfeeds/category/default.php</filename>
      <filename>html/com_newsfeeds/category/default_items.php
        </filename>
      <filename>html/com_newsfeeds/category/index.html</filename>
      <filename>html/com_newsfeeds/newsfeed/default.php</filename>
      <filename>html/com_newsfeeds/newsfeed/index.html</filename>
      <filename>html/com_newsfeeds/index.html</filename>
      <filename>html/com_weblinks/categories/default.php</filename>
      <filename>html/com_weblinks/categories/index.html</filename>
      <filename>html/com_weblinks/category/default.php</filename>
      <filename>html/com_weblinks/category/default_items.php
        </filename>
      <filename>html/com_weblinks/category/index.html</filename>
      <filename>html/com_weblinks/weblink/form.php</filename>
      <filename>html/com_weblinks/weblink/index.html</filename>
      <filename>html/com_weblinks/index.html</filename>
      <filename>html/com_user/user/index.html</filename>
      <filename>html/com_user/user/default.php</filename>
      <filename>html/com_user/user/form.php</filename>
      <filename>html/com_user/login/index.html</filename>
      <filename>html/com_user/login/default_login.php</filename>
      <filename>html/com_user/login/default.php</filename>
      <filename>html/com_user/login/default_logout.php</filename>
      <filename>html/com_user/register/default.php</filename>
      <filename>html/com_user/register/index.html</filename>
      <filename>html/com_user/register/default_message.php
        </filename>
      <filename>html/com_user/index.html</filename>
      <filename>html/com_user/remind/index.html</filename>
      <filename>html/com_user/remind/default.php</filename>
      <filename>html/com_user/remind/default_message.php</filename>
      <filename>html/com_user/reset/index.html</filename>
      <filename>html/com_user/reset/default.php</filename>
      <filename>html/com_user/reset/confirm.php</filename>
      <filename>html/com_user/reset/complete.php</filename>
    </files>
    <positions>
      <position>box_menu</position>
      <position>box_mini</position>
      <position>box_right</position>
      <position>box_footer</position>
      <position>debug</position>
    </positions>
</install>
```