KOULUTUSPORTAALI

LAHDEN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka

Opinnäytetyö

Kevät 2007

Lassi Hulkkonen

Lahden ammattikorkeakoulu
Tietotekniikan koulutusohjelma

HULKKONEN, LASSI: Koulutusportaali
Case: Flander OY:n koulutusportaali
Ohjelmistotekniikan opinnäytetyö, 52 sivua

Kevät 2007

TIIVISTELMÄ

Tämän opinnäytetyön aiheena on koulutussivuston rakentaminen www-ympäristöön. Tarkemmin keskitytään tekniikkaan nimeltä ”web-template”.

Teoriaosassa käsitellään yleisiä www-sivujen tekemiseen tarvittavia tekniikoita. Internetissä palveluita käytetään yleensä asiakas-palvelin arkkitehtuurin pohjalta. Osapuolten välinen kommunikaatio käyttää hyväkseen erilaisia standardeja, jotka liittyvät tietoliikenteeseen ja tiedon muotoon. Yleisesti käytetty muoto tehdä kattavia sivustoja on portaali, joka tarkoittaa monesta sivusta koostuvaa dynaamista kokonaisuutta. Käyttöoikeuksia on yleensä myös rajoitettu.

”Web-template” on tekniikka, joka on kehitetty helpottamaan www-sovellusten tekoa. Sitä käytetään pääosin käyttöliittymien toteutuksessa. Periaatteena on, että ulkoasun lähdekoodit kirjoitetaan erillisiin tiedostoihin, ja dynaamisiksi määritellyt osat korvataan ohjausmerkintöjen avulla. Tämän tekee järjestelmälle ominainen mallinekoneisto; sijoitettavat tiedot voidaan hankkia esimerkiksi tietokannasta muiden sovelluskerrosten kautta.

Opinnäytetyön käytännön osuus sisälsi koulutusportaalin toteutuksen IT-yritys Flander Oy:lle. Projekti käytiin läpi pienellä työntekijämäärällä, mutta se piti sisällään ohjelmistotuotannon päävaiheet. Ohjelma kirjoitettiin palvelinpään komentokielellä, ja sovelluksen toteutuksessa käytettiin hyväksi ”web-template”-tekniikkaa.

Ulkoasu on tärkeä osa www-sovellusta. Kyseisen sivuston tekemisestä kertyneiden kokemusten perusteella voidaan sanoa, että ”web-templatet” auttavat käyttöliittymän erotuksessa muista sovelluksen kerroksista. Lisäksi hyvin käytettynä ne voivat tuoda koodiin uudelleenkäytettävyyttä, mutta menetelmän tehokas käyttö vaatii kuitenkin suunnitelmallisuutta.

Avainsanat: Internet, web, portaali, komentokieli, template(mallinne), käyttöliittymä, ohjelmistokerros, koulutus.

Lahti University of Applied Sciences
Faculty of Technology

HULKKONEN, LASSI: Training portal
Case: Flander OY training portal

Bachelor's thesis in Software Engineering, 52 pages

Spring 2007

ABSTRACT

The topic of this thesis was building a training application in the WWW environment. The main focus is on a technique called "web template".

The theory section deals with the common technologies that are needed to create web pages. The services in the internet are usually based on the client-server architecture. The interaction between the parties uses many standards connected with the communication details and the form of data. A widely used form to create a vast web site is a web portal. This means an entity with several dynamic pages and at least partially restricted usage.

"Web template" is a technique that has been developed to help to create WWW applications. It is mainly used in user interface implementation. The main principle is that the source code of the layout is written in separate files. The parts that are marked as dynamic are replaced according to the control logic. This operation is carried out by a distinctive template engine. The information to be located can be gathered for example from a database through other software layers.

The case of this research constituted of creating a training portal for IT company Flander OY. The project was done by a small number of people but it contained all the main stages of a software engineering process. The program was written with a server side command language. The application was implemented using the "web template" technique.

The layout is an important part of a WWW application. Based on the experiences gathered from making this particular web site it can be said that web templates help to separate the user interface from other layers in an application. In addition, with skillful use they can create reusability to the code. However, a systematic approach is required to use the method efficiently.

Key words: internet, web, portal, command language, template, user interface, software layer, training.

SISÄLLYS

1 JOHDANTO

Viimeisen vuosikymmenen aikana on luotu paljon internetin välityksellä toimivia ohjelmistosovelluksia. Tähän on ollut suurena vaikuttajana tietoliikenneyhteyksien nopea kehitys. Samalla myös erilaiset menetelmät verkkosovellusten toteuttamiseen ovat parantuneet, ja niiden käyttö on tullut helpommaksi. Hyvin yleinen käyttöliittymä internetin välityksellä toimivissa ohjelmistoratkaisuissa on www-selain. Selaimia käytetään pääosin html-pohjaisten www-sivujen ja niihin upotettujen tietokokonaisuuksien tarkasteluun. Tiedonsiirto noudattaa yleisesti http-protokollaa. Sen yleisluontoisuus yhdessä muiden www-standardien kanssa mahdollistaa suuren määrän erilaisia palvelinratkaisuja asiakassovelluksen pysyessä käytännössä muuttumattomana. Helpottamaan palvelimen ohjelmointia edelleen on kehitetty erilaisia komentokieliä, joita suoritetaan tulkkausmoduulien avulla. Itse palvelinta ei tällöin tarvitse rakentaa uudestaan yksinkertaisia sovelluksia varten, vaan riittää, että kirjoitetaan vaaditut toiminnot suorittava skripti ja ajetaan se, kun asiakas kutsuu palvelua. Palvelimen moduuli osaa hoitaa komentojen tulkitsemisen ja suorituksen.

Lopputyön pohjaksi tehdyssä projektissa on rakennettu edellä mainittuja näkökohtia hyödyntävä koulutusjärjestelmä keskikokoisen informaatioteknologian alan yrityksen tarpeisiin. Päällimmäisinä vaatimuksina olivat koulutuspakettien lisäyksen ja suorittamisen mahdollistaminen internetin välityksellä. Näihin liittyi erottamattomana osana myös huomattava määrä erilaisia ylläpito-, tilastointi- ja rekisteröintitoimintoja. Käyttöliittymänä oli tarkoitus käyttää www-selainta ja muutenkin sovelluksen toteutuksessa ajateltiin käyttää asiakas-palvelinarkkitehtuurin tarjoamia ratkaisumahdollisuuksia. Toteutusmuoto noudattaa ”www-portaalina” yleisesti tunnettua mallia, joka perustuu www-sivujen ja asiakaskutsujen dynaamiseen käsittelyyn rajoitetussa ympäristössä. Käytännössä tämä tarkoittaa sitä, että sivusto sisältää tietyn määrän palveluja ja materiaaleja, joita haetaan www-sivulle määriteltyjen skriptien ja asiakkaan antamien parametrien perusteella. Sivuston käyttö on rajattu ainoastaan tunnistetuille käyttäjille, ja ohjelmointikielenä tällaisissa sovelluksissa käytetään yleensä jotain valitun http-palvelinsovelluksen kanssa yhteensopivaa

komentokieltä. Tässä tapauksessa alustaksi valittiin Linux-pohjainen ratkaisu ja kieleksi PHP, koska ne olivat ilmaisia ja entuudestaan melko tuttuja kokonaisuuksia.

PHP on paljon käytetty palvelinpään ohjelmointikieli. Varsinkin vähemmän kriittisissä palvelinsovelluksissa sen ominaisuudet ovat riittävät. PHP muistuttaa perussyntaksiltaan yksinkertaistettua c-kieltä ja tarjoaa nykyisin laajennettavuutensa ansiosta monipuolista toiminnallisuutta. Koska dynaamisten www-sivustojen toimintamekanismi on usein samankaltainen ja PHP on hyvin yleinen kieli, on sille kehitetty monia menetelmiä helpottamaan sivunmuodostuksen toteuttamista. Niistä on yleensä apua koodin uudelleenkäytettävyyden ja sovelluksen eri kerrosten erottelemisen kannalta. Yksi menetelmä ovat ”web template:t” yhdessä niitä prosessoivien mallinekoneistojen kanssa. Perusperiaatteena on, että koneistoon syötetään sivulle haluttuja dynaamisia tietoja, jotka se yhdistää valittuihin mallinetiedostoihin määritellyllä tavalla. Toteutustavat vaihtelevat eri ratkaisujen mukaan, mutta yleensä mallinetiedostot voivat sisältää yksinkertaista ohjauskoodia, jonka avulla tiedot saadaan tulostettua halutulla tavalla staattisen html-kuvauksen joukkoon. Tuloksena syntyy www-sivu, joka voidaan sellaisenaan lähettää asiakkaalle. ”Template”-ratkaisuja käytettiin myös koulutusportaalin toteutuksessa, ja niiden ajateltiin tuovan selkeyttä ja uudelleenkäytettävyyttä sovelluksen rakenteeseen.

2 INTERNET JA WWW

2.1 Yleistä

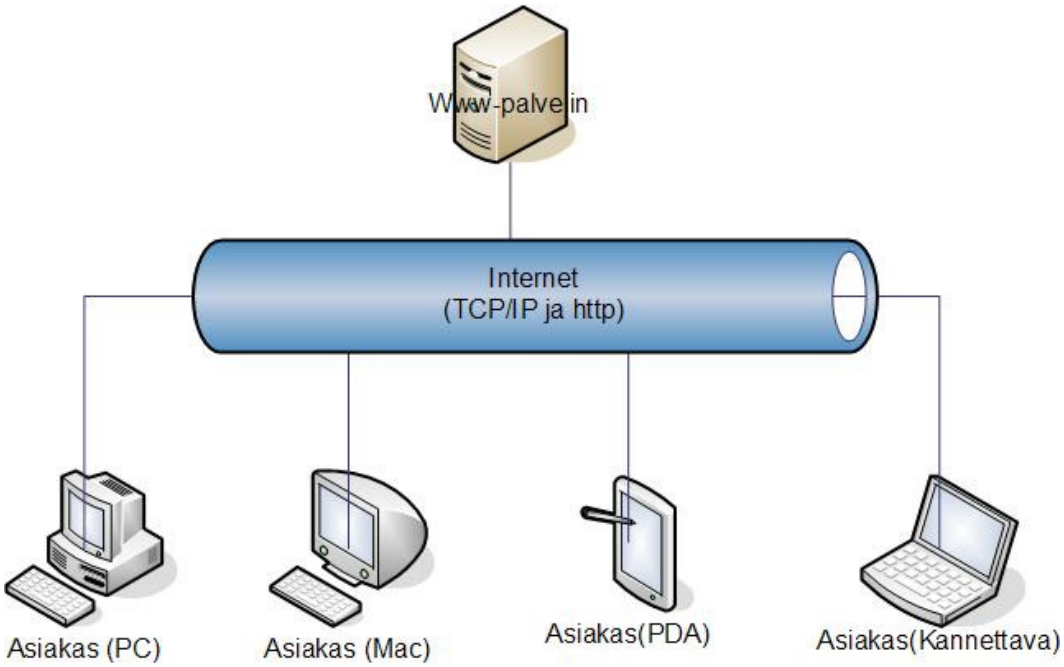
Internet on maailmanlaajuinen tietoverkko, joka koostuu lukemattomasti yhdistetyistä aliverkoista. Jokaisella verkkoon liittyvällä tietokoneella on periaatteessa mahdollisuus olla yhteydessä kaikkiin muihin verkon jäseniin internet-protokollan avulla. Käytännössä tätä mahdollisuutta kuitenkin rajoitetaan huomattavasti tietoturvasyistä. Internetin välityksellä voidaan tarjota monenlaisia palveluja, jotka sijaitsevat fyysisesti kaukana käyttäjästä. Sovellusten välisessä viestinnässä käytetään erilaisia sovellusprotokollia, jotka muodostetaan fyysiseen tiedonsiirtoon tarvittavien määritekerrosten päälle (Parker & Sportack 2000, 15).

World Wide Web on määrättömän suuri kokoelma internetin välityksellä yhdistettyjä dokumentteja ja muita resursseja. Käsitteeksi muodostuneiden ”www-sivujen” välitys on internetin käytetyin sovellusmuoto. Muita yleisiä sovelluksia ovat muun muassa sähköposti, tiedostonsiirto ja erilaiset pikaviestintämenetelmät. Www-sivut rakentuvat html-elementeistä, jotka voivat sisältää erilaisia tietokokonaisuuksia (Refsnes Data, 2007). Sivuja välitetään http-protokollan avulla, ja niitä voidaan havainnollisen tietosisältönsä lisäksi hyödyntää käyttöliittymänä monenlaisten dynaamisten verkkopalvelujen toteutuksessa.

2.2 Asiakas-palvelinarkkitehtuuri

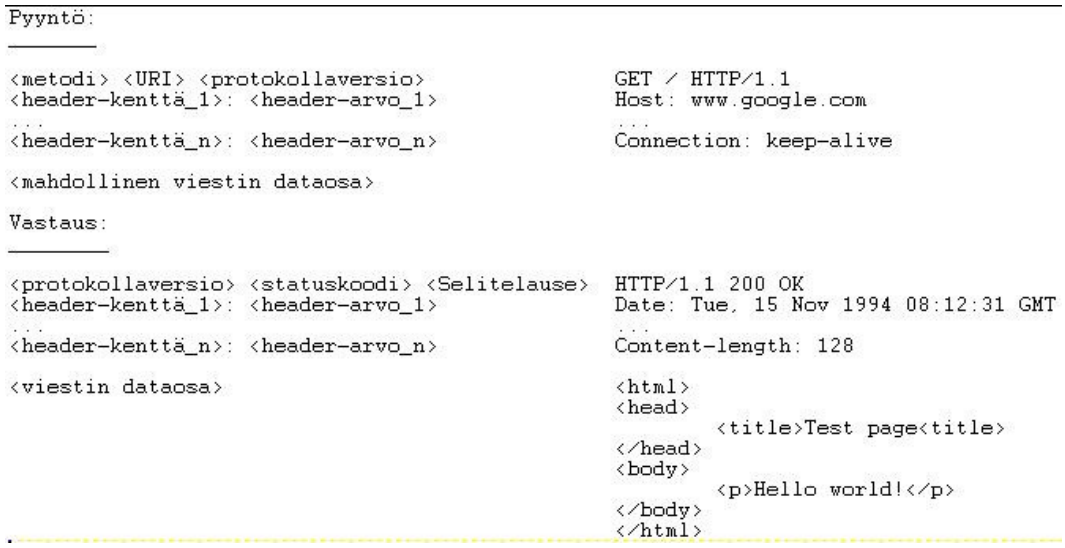
Jotta tietoliikenneyhteyksien välityksellä voidaan siirtää tietoa tehokkaasti, täytyy keskustelevien sovellusten välisen työnjaon ja tiedonsiirron vaiheiden olla selkeitä. On tärkeää, ettei tietolinjoja kuormiteta turhalla liikenteellä. Lisäksi etäällä toisistaan sijaitsevien ohjelmien täytyy pystyä ymmärtämään toistensa suorituksen vaiheita, jotta haluttu lopputulos saadaan aikaiseksi. Yksi ratkaisu näihin tarpeisiin on asiakas-palvelin arkkitehtuuri, joka rajaa sovellusten roolit tarkasti kutsuvaan ja vastaavaan osapuoleen. (Peltomäki 1998, 8.)

Käytännössä tämä tarkoittaa sitä, että asiakas lähettää viestin palvelimelle, jossa se pyytää jotain resurssia, toimenpidettä suoritettavaksi lähettämilleen tiedoille tai molempia, minkä jälkeen asiakas jää yleensä odottamaan palvelimen vastausta. Toimiakseen oikein palvelimen tulee olla odottavassa tilassa pyynnön saapuessa. Siihen reagoidaan ja suoritetaan toimenpiteet kutsun sisältämien määritysten perusteella, ja lopuksi lähetetään vastaus, mikäli asiakas sitä odottaa. Yleensä palvelin välittää vähintään jonkinlaisen kuittauksen, vaikka se ei periaatteessa ole välttämätöntä. Hoidettuaan asiakkaan pyytämät tehtävät palvelin siirtyy kuuntelemaan uusia kutsuja. Käytännössä palvelimet joutuvat usein palvelemaan useita asiakkaita samanaikaisesti, jolloin toteutuksessa saatetaan joutua käyttämään säikeitä tai lapsiprosesseja. Kuvio yksi havainnollistaa, kuinka palvelin voi protokollien avulla ymmärtää monia erilaisia asiakkaita. Www-maailmassa tunnettu käsite on www-palvelin. Tällä viitataan sovellukseen, joka vastaa asiakkaan http-protokollaa noudattaviin pyyntöihin ja lähettää sille tarpeen mukaan erilaisia paikallisia resursseja ja www-sivuja.



KUVIO 1. Asiakas-palvelin arkkitehtuuri

Http, eli Hyper Text Transfer Protocol, on asiakas-palvelin arkkitehtuurin tarpeisiin kehitetty protokolla. Http-viestien muoto vaihtelee hieman riippuen siitä, onko lähettäjä asiakas vai palvelin. Samalla se kuitenkin myös määrittelee osapuolet. Asiakaskutsun osat ovat metodi ja halutun resurssin sijainti palvelimella, erilaiset ”header”-määritteet sekä mahdollisesti viestin mukana siirrettävä dataosa. Dataosan rakenne ja tarpeellisuus riippuu kutsun metodin asettamista vaatimuksista. Resurssin sijainti ilmaistaan URI:n avulla, joka on lyhenne sanoista ”Universal Resource Identifier”. Sitä käytetään ilmaisemaan yksittäisten resurssien sijaintia palvelimella. Palvelimen vastaus koostuu statustiedoista, ”header”-määritteistä ja dataosasta. Koska asiakas pyytää yleensä jotain resurssia http:n avulla, on palvelimen vastauksessa yleensä myös dataosa, jossa tietokokonaisuus välitetään. Statusrivillä kerrotaan tietoja palvelun onnistumisesta ja aiheeseen liittyvä viestilause (esim. 200 ”OK”). Kuviosta kaksi on nähtävissä, millaisia viestien eri osat voivat olla käytännössä. Asiakkaan ja palvelimen ei tarvitse tietää toistensa rakenteesta mitään, koska protokollan asettamat vaatimukset kuvaavat niiltä vaadittavat tehtävät tarpeeksi hyvin. (Peltomäki 1998, 16; W3C® 2007.)



KUVIO 2. Http-pyyntö ja -vastaus

Www-palvelimen ja selaimen välinen tietoliikenneyhteys hoidetaan TCP/IP-protokollan avulla, joka tarjoaa pohjan http:n käyttämiselle. IP on alimman tason yhteyskäytäntö, joka huolehtii, että tieto kulkee fyysisesti oikeaan osoitteeseen. Käytännössä tämä merkitsee päätelaitteiden osoittamista ja pakettien reitittämistä verkossa. TCP-protokolla toimii IP:n päällä. Sen keskeisiä ominaisuuksia ovat katkeamaton yhteys asiakkaan ja palvelimen välillä tiedonsiirron aikana ja lähetettävien datapakettien oikean järjestyksen varmistaminen. Tällä tavoin on mahdollista lähettää huomattavia määriä katkeamatonta tietoa ja varmistaa, että se menee perille oikeassa järjestyksessä. (Parker & Sportack 2000, 42-27.)

Ohjelmoinnin näkökulmasta katsottuna tavallisimmissa käyttöjärjestelmissä on tuki erilaisten sovellusten välisten verkkoviestintämuotojen käytölle. Tarjolla olevia ohjelmointikirjastoja käyttämällä voidaan rakentaa erilaisia verkkosovelluksia. Ohjelmallisesti määriteltävien protokollien, kuten http:n, käsittely täytyy toteuttaa sovelluspohjaisesti. Epäilemättä yleisimpiin protokolliin on olemassa valmiita ratkaisumalleja, joiden avulla viestintämekanismien toteutus helpottuu.

2.3 Html

Www-sivut koostuvat yleensä html-koodista. Html, eli Hypertext Markup Language, on standardoitu kuvauskieli, jonka avulla voidaan kuvata dokumentin rakennetta ja siihen tulostettavia erilaisia elementtejä. Koska dokumentit voivat sisältää hyperlinkkejä, sanotaan niiden olevan hypertekstiä. Www-sivuja katsellaan www-selainten avulla. Nämä sovellukset osaavat tulkita html:ää ja tulostaa dokumentin määritellyn kaltaisena tietokoneen näytölle. Ulkoasu ja tulkinta vaihtelevat hieman eri toteutusten välillä, mutta käytännössä ne kaikki yltävät standardin asettamiin vaatimuksiin. Yleensä selaimilla on graafinen käyttöliittymä, mutta myös konsolipohjaisia on olemassa. (W3C® 2007.)

Html-syntaksi koostuu erilaisista tageista ja niille ominaisista parametreista. Se muistuttaa tässä suhteessa paljon xml-kieltä. Tagit kuvaavat dokumentin osia erilaisina elementteinä ilmaistuna. Aloitus ja lopetustagin väliin sijoitetaan

varsinainen data, jonka elementin halutaan sisältävän dokumenttia tulostettaessa. Yleensä data on tekstiä, ja myös muunlaista tietoa voidaan esittää, mutta siihen tarvitaan yleensä omat erityiset elementtinsä. Esimerkiksi kuvia, javascript-koodia ja nykyisin usein videokuvaakin käytetään www-sivujen yhteydessä. Elementtien kuvaamiseen käytetään hakasulkeita. Lopetustagit sisältävät ensimmäisenä merkkinä ”/”-merkkiä ilmaisemaan eron aloitustageihin. Mahdolliset parametrit sijoitetaan aloitustagin sisään.

Html-kuvauskielisen www-sivun kautta voidaan tiedon esittämisen lisäksi myös välittää sitä palvelimen suuntaan. Tämä onnistuu parhaiten lomakkeiden avulla. Ne koostuvat tarkoitukseen soveltuvasta tagiperheestä, joka pitää sisällään muun muassa painikkeita, valintaruutuja ja erilaisia syötekenttiä. Lomake kootaan kokonaisuudessaan ”form”-elementin sisään, ja se voidaan lähettää painamalla ”submit”-tyyppistä painiketta. Lomakkeen sisältämä tieto lähetetään siinä annetun metodin määrittelemässä muodossa palvelimelle, ja lähetettyjä tietoja voidaan käsitellä palvelinpäähän rakennettujen skriptien avulla, esimerkiksi tallentaa ne tietokantaan.

2.4 PHP

PHP, nykyisin lyhenne sanoista ”PHP Hypertext Preprocessor”, on palvelinpään tulkattava komentokieli. Tämä merkitsee sitä, ettei sovelluksia käännetä, vaan niiden sisältämät ohjeet tulkitaan ajon aikana. Palvelinohjelmointikielet on kehitetty helpottamaan dynaamisten sovellusten toteuttamista www-ympäristössä. Niiden käyttö perustuu ns. CGI-tekniikkaan. Common Gateway Interface merkitsee sitä, että palvelinsovellus toimii välittäjän asiakkaan ja ulkoisen sovelluksen välillä, jonka se käynnistää kutsuttaessa (Peltomäki 1998, 478). Palvelinohjelmointikielet ovat yleisesti korkean tason ohjelmointikieliä.

PHP:tä on alusta pitäen kehitetty avoimen lähdekoodin menetelmällä ja se onkin tästä syystä hyvin edullinen käyttäjälleen. Alun perin PHP oli kokoelma erilaisia lyhyitä PERL-skriptejä, mutta on laajentunut siitä omaksi kielekseen. Nykyisin perustoimintojen lisäksi kieleen on olemassa monia laajennuksia, joiden avulla

voidaan suorittaa monipuolisia tehtäviä. Suhteessa www-palvelimeen PHP:n rooli on kuitenkin rajoittunut ainoastaan komentojen suoritukseen, joten kriittisemmät tiedonsiirto-operaatiot jäävät helposti pois sen käyttöalueesta. Toisaalta yksinkertainen toteutusarkkitehtuuri mahdollistaa PHP:n käytön monissa erilaisissa palvelinalustoissa. Toimiakseen skriptit tarvitsevat palvelinmoduulin, joka osaa paikallistaa ne, ja tulkita niiden sisältämät komennot. (The PHP Group 2007.)

PHP:n syntaksia voisi kuvata yhdistelmäksi c-kieltä ja PERL:iä. Silmukat ja ehtolauseet operaattoreineen muistuttavat kirjoitusasultaan c-kielen vastaavia. Muuttujia ei tarvitse alustaa ja niiden tyyppi määrittyy automaattisesti kulloinkin sijoitetun arvon mukaan. Tämän kaltainen suoraviivaisuus viittaa vahvasti erilaisten komentokielien suuntaan, jollainen esimerkiksi PERL on. PHP:ssä ei ole tällä hetkellä tukea nimiavaruuksien lisäämiselle. Luokkia on kuitenkin mahdollista määritellä ja käyttää rajoitetusti. Lisäksi muuttujien vaikutusalueisiin on mahdollista vaikuttaa, vaikka niitäkään ei oletuksena tarvitse määritellä.

PHP-kieli tukee palvelimelle muodostettavia istuntoja ja istuntomuuttujia. Www-maailmassa istunnot auttavat rajaamaan käyttäjäoikeuksia ja tunnistamaan käyttäjäkohtaisia asetuksia. Jotta istunnot pysyisivät yksityisinä laajassa tietoverkossa, käytetään yleensä istuntotunnisteita, jotka ovat määrämittaisia satunnaisarvoja. Tunnisteet voidaan tallentaa selaimen evästeeseen, tai niitä voidaan kuljettaa osoiterivillä. PHP tukee myös evästeiden käsittelyyn liittyviä toimintoja. (The PHP Group 2007.)

```php
<?php
    // Alustetaan taulukkoon erilaisia paikkoja
    $places = array( "earth", "moon", "sun" );

    // Tulostetaan paikat vuorotellen valmiiksi muotoillun merkkijonon kanssa
    foreach( $places as $place )
        echo "Hello, " . $place . "!";
?>
```

KUVIO 3. PHP-koodiesimerkki

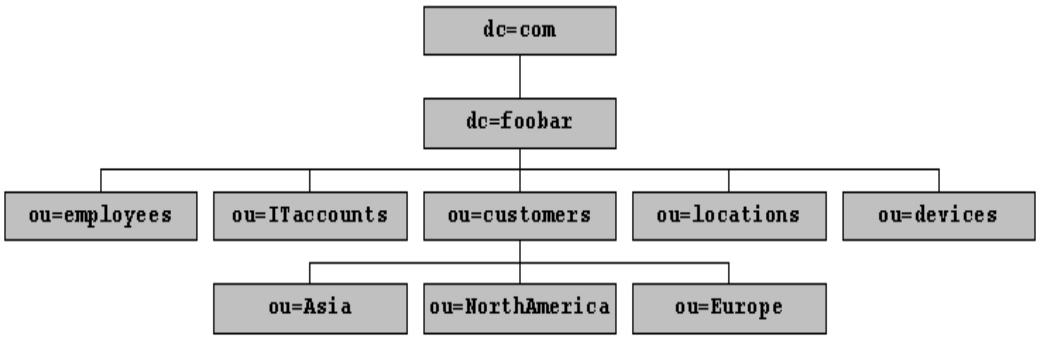
PHP-skriptit käyttävät hyväkseen palvelimen niille välittämiä URL-parametreja. URL, eli ”Universal Resource Locator”, ilmaisee tiedon absoluuttista sijaintia tietoverkossa. Käytännössä se kirjoitetaan www-selaimen osoiteriville. Periaatteessa yksi URL-rivi voi kertoa resurssin tarkan sijainnin koko internet-verkon alueella (Peltomäki 1998, 16). Parametrit sijoitetaan yleensä rivin viimeiseksi kysymysmerkin jälkeen, ja niitä tulkitsemalla voidaan generoida www-sivuihin dynaamista sisältöä, joka voi olla esimerkiksi jostain tietokannasta peräisin. Sivun sisällöksi haluttu materiaali ”tulostetaan” www-palvelimen käyttöön ”echo”- tai ”print”-komennoilla. Käytännössä tieto siirtyy jollain prosessien välisen kommunikoinnin menetelmällä. Käsitteeksi muodostunut avoimen lähdekoodin LAMP-arkkitehtuuri käyttää hyväkseen edellä mainittua periaatetta. Www-palvelin ja tietokantasovellus toimivat dynaamisen sivunmuodostuksen perustana. Sovellusalusta koostuu Linux-käyttöjärjestelmästä, Apache www-palvelimesta, MySQL-tietokantapalvelimesta ja vaihtoehtoisesta palvelinpään komentokielestä (Perl/Python/PHP). Näistä rakentuu toimiva kokonaisuus, johon voi suoraan kirjoittaa www-sovelluksia valitulla kielellä.

2.5 MySQL

MySQL on eräs tietokantapalvelinsovellus. Sen lähdekoodi on avoin. MySQL tarjoaa kattavan määrän toimintoja ja on toimintaperiaatteiltaan sekä luotettava, että joustava. Tietokannat ovat muodoltaan relaatiotietokantoja. Nämä ovat kokonaisuuksia, joissa tiedot on tallennettu riveittäin taulujen sarakkeisiin. Taulut voivat viitata toisiinsa avainten perusteella. Kyselyihin käytetään SQL-kieltä (lyhenne sanoista ”Structured Query Language”). Se on syntaksiltaan selkeä ja tehokas tapa relaatiotietokantojen käsittelyyn. PHP-kielessä on kattava rajapinta MySQL-palvelujen käyttöön. Tämä yhdistelmä onkin saavuttanut suosiota erilaisten asiakas-palvelinsovellusten yhteydessä. (MySQL AB 2007; Stephens, Plew, Morgan & Perkins 1999, 4.)

2.6 LDAP

LDAP on lyhenne sanoista ”Lightweight Directory Access Protocol”. Se on TCP/IP-kerroksen päällä toimiva sovellusprotokolla, jota käytetään pääsääntöisesti tietojen lukemiseen, johon protokolla on optimoitu. Myös kirjoittaminen on tuettu, mutta tietoja ei yleensä tarvitse muokata usein. Haettavat tiedot sijaitsevat yleensä LDAP-palvelinsovelluksen sisällä. Tietokannan muoto on hakemistopohjainen. Siinä olevat merkinnät voidaan lukea dn-merkkijonon perusteella. Se kuvaa kohteen sijaintia hakemistopuun juuresta lähtien haara kerrallaan. Kuviossa 4 näkyy esimerkki mahdollisesta puurakenteesta. Esimerkiksi kohde ”ou=Europe” sijaitsee dn-merkkijonon ”dc=com,dc=foobar,ou=customers” osoittamassa paikassa. Sovellusta käytetään usein muun muassa käyttäjätietojen tarkastamiseen ja yhteystietojen hakemiseen. (Donnelly 2000.)



KUVIO 4. LDAP-puurakenne (Donnelly 2000.)

3 WEB-TEMPLATET

3.1 Tehokas tapa tuottaa dynaamisia dokumentteja

Kun selataan jotain satunnaista www-sivustoa, voidaan helposti havaita, että sivujen ulkoasu pysyy pääosin samanlaisena sen eri sivuilla. Vaihteleva sisältö on yleensä rajoitettu tietyille ruudun alueille. Suuri osa web-portaaleista perustuu tällaiseen malliin. Dynaamista sisältöä haetaan staattisen sivupohjaan, jossa sivu jaotellaan erilaisia tehtäviä noudattaviin osiin. Sisältö rakennetaan URL:ssa olevien määritteiden perusteella. Toteutusmahdollisuuksia tällaiseen sovellukseen on karkeasti jaoteltuna kaksi: ”perinteinen” ja web-templatet.

Yksinkertainen tapa on tehdä jokaiselle sivutyypille oma skriptinsä, joka linkitetään kiinteästi muihin vastaaviin, joita sivustolla käytetään. Eri sivut voivat suorittaa erilaisia tehtäviä, mutta niiden lähdekoodin täytyy myös sisältää paljon päällekkäistä toiminnallisuutta yhtenäisen ulkoasun säilyttämiseksi. Html-koodi joudutaan yleensä sisällyttämään sovelluslogiikan joukkoon, koska kaikki sivunmuodostuksen vaatimat operaatiot ovat kiinteänä osana kirjoitettua skriptiä. Tämä saattaa aiheuttaa ongelmia, kun sivun ulkoasua halutaan muokata. Tiedostoista saattaa tulla huomattavan laajoja kokonaisuuksia, joita on vaikea hahmottaa varsinkin, jos ei hallitse kyseessä olevaa komentokieltä. Laajennettavuusmahdollisuudet eivät kuitenkaan juuri kärsi, koska uusi sivutyyppi täytyy joka tapauksessa rakentaa aina kokonaisuudessaan alusta. Vähäistä käsittelyä vaativia osia voidaan myös sisällyttää suurempiin kokonaisuuksiin, jolloin niiden kirjoitusta ei tarvitse toistaa.

Web-templatet ovat toinen menetelmä. Ne ratkaisevat joitain edellisessä kappaleessa mainitun toteutusmuodon ongelmista. Menetelmän perusteena on, että staattiset osat sijoitetaan erillisiin mallinetiedostoihin, joihin dynaaminen sisältö lisätään määriteltyihin kohtiin mallinekoneiston avulla. Ensi näkemältä tämä vaikuttaa samankaltaiselta kuin erillisiin skripteihin perustuva toteutus. Mallineet sisältävät myös usein ohjauskoodia, mutta sen tarkoituksena ei ole enää hankkia tietoa vaan lähinnä muokata sivun ulkoasua halutun näköiseksi. Näin

voidaankin sanoa, että käyttöliittymän esitysmuoto on erotettu varsinaisesta sovelluslogiikasta. Tämä on yksi olennaisimpia syitä web-templatejen käyttöön www-sovellusten kehittämisessä.

Käytännön esimerkkinä mallineiden käytöstä voidaan kuvitella yksinkertainen www-sivu, jossa tulostetaan tietokannassa sijaitsevan taulun sisältö html-taulukkoelementin sisään. Jos sivu tehdään yksinkertaisimmalla mahdollisella tavalla käyttäen PHP-kieltä, ensin pyydetään tietokannasta taulun sisältö. Sitten tulostetaan kaikki elementit tietoineen ”echo” tai ”print” komentojen avulla silmukassa, jossa rivit käydään läpi. Tapa on kieltämättä tehokas suoritusta ajatellen, mutta monimutkaisemmissa sovelluksissa koodista voi tulla huomattavan epäselvää. Web-templateja käytettäessä tietokannan tiedot sijoitettaisiin johonkin mallinejärjestelmälle välitettävään taulukkomuuttujaan. Taulukon läpikäynti kuvattaisiin tiedostossa, joka pitäisi sisällään ainoastaan kuvauksen silmukan suorittamisesta ja tarvittavat html-merkinnät. Tiedot sijoitettaisiin määriteltyihin kohtiin elementtien joukossa. Parhaiten eron huomaa, kun tarkastelee tiedostojen sisältöä. Mallineisiin perustuva järjestelmä sisältää ainoastaan käyttöliittymää kuvaavaa tietoa ja on tässä tapauksessa myös hyvin selkeän näköinen (kuvio 5).

Yksinkertainen PHP:

```
<html>
<body>
    <?php
        // Connecting, selecting database
        $link = mysql_connect('mysql_host', 'mysql_user', 'mysql_password')
           or die('Could not connect: ' . mysql_error());
        echo 'Connected successfully';
        mysql_select_db('my_database') or die('Could not select database');

        // Performing SQL query
        $query = 'SELECT * FROM my_table';
        $result = mysql_query($query) or die('Query failed: ' . mysql_error());
    ?>
    <table>
        <tr>
            <th>id</th>
            <th>data1</th>
            <th>data2</th>
        </tr>
    <?php
        // Printing results in HTML
        while ($line = mysql_fetch_array($result, MYSQL_ASSOC)) {
           echo "\t<tr>\n";
           foreach ($line as $col_value) {
              echo "\t\t<td>$col_value</td>\n";
           }
           echo "\t</tr>\n";
        }
        echo "</table>\n";

        // Free resultset
        mysql_free_result($result);

        // Closing connection
        mysql_close($link);
    ?>
    <table>
</body>
</html>
```

Template-tiedosto:

```
<html>
<body>
    <table>
        <tr>
            <th>id</th>
            <th>data1</th>
            <th>data2</th>
        </tr>
    {foreach $table as $row}
        <tr>
            <th>{$row['id']}</th>
            <th>{$row['data1']}</th>
            <th>{$row['data2']}</th>
        </tr>
    {/foreach}
    <table>
</body>
</html>
```

KUVIO 5. PHP ja Web-template

PHP on paljon käytetty ohjelmointikieli. Sille on kehitetty useita erilaisia web-templateja auttamaan www-sovelluskehitystä. On kuitenkin hyvä muistaa, että PHP-kieli itsessään on jo eräänlainen mallinejärjestelmä. Vaikka mallineet helpottavat käyttöä, saattavat ne myös osaltaan tuoda päällekkäistä toiminnallisuutta sovelluksiin. Lisäksi niiden käyttäminen voi kuormittaa palvelinta enemmän, kuin mitä ehkä olisi tarpeen. Olemassa on myös muita varteenotettavia vaihtoehtoja, joista saadaan samankaltaisia hyötyjä.

3.2 Mallinetiedostot

Suurimmalle osalle web-template-sovelluksia on yhteistä, että ne käyttävät mallinetiedostoja www-sivujen muodostamisessa. Nämä tiedostot sisältävät staattisen html-koodin, jonka sekaan voidaan yleensä tulostaa erilaisia muuttujia ja taulukoita. Tulostuksen kulkua voidaan puolestaan ohjata erilaisilla silmukoilla ja ehtolausekkeilla. Ominaisuudet vaihtelevat toteutuskohtaisesti. Ohjausrakenteet noudattavat aina kulloinkin kyseessä olevalle web-template-järjestelmälle ominaista syntaksia, jota sen koneisto osaa tulkita. Tiedoston dynaamisesti määrittyvät osat erotetaan kiinteästä datasta jonkinlaisilla erikoismerkeillä. Yleisiä tapoja ovat haka- ja aaltosulkeet. Web-template-sovelluksen käyttämän ohjauskoodin syntaksi on usein avainasemassa käyttömukavuuden kannalta, sillä mallinetiedostojen tehokas rakentaminen vaatii tämän kattavaa osaamista.

Tiedoston perusrakenne on www-sivu, joka on koottu ”html”-elementin sisään. Sivu sisältää yleensä myös muut oletusmääritykset. Tärkeintä on, että se on selaimen ymmärtämää muotoa. Monissa järjestelmissä on myös mahdollista sisällyttää mallinetiedostoon tietyillä komennoilla muita mallineita. Näitä voidaan kutsua ”alimallineiksi”, jotka saattavat olla erikoistuneita jonkin rajatun tietokokonaisuuden tulostukseen. Alimalline voi esimerkiksi tuottaa lomakkeen tai tietynlaisen taulukon. Sivujen ulkoasun muokkaus selkeytyy, koska se on jaettu pienempiin kokonaisuuksiin. Myös koodin uudelleenkäyttävyys paranee,

sillä pienemmille erikoistuneille kokonaisuuksille löytyy usein käyttöä muissakin kohteissa. Käytännössä on mahdollista koota tällä tavoin kokonaisia mallinekirjastoja jonkin tietyn sivuston käyttöön. Kuviosta 6 on nähtävissä miten edellä mainittuja seikkoja voidaan toteuttaa Smarty:n avulla. Se on yksi melko laajalle levinnyt web-template ratkaisu. Alimallineet sisällytetään ”include”-komennon avulla. ”Output”-osa kuvaa, miltä valmis html-koodi näyttää mallinekoneiston prosessoinnin jälkeen.

header.tpl

```
<html>
<head>
<title>{$title|default:"no title"}</title>
</head>
<body>
```

footer.tpl

```
</body>
</html>
```

index.tpl

```
{include file="header.tpl" title="User Info"}

User Information:<p>

Name: {$name|capitalize}<br>
Address: {$address|escape}<br>

{include file="footer.tpl"}
```

output

```
<html>
<head>
<title>User Info</title>
</head>
<body>

User Information:<p>

Name: George Smith<br>
Address: 45th &amp;amp; Harris<br>

</body>
</html>
```

KUVIO 6. Smarty-mallinetiedoston syntaksi (New Digital Group, inc. 2007)

Mallinetiedostojen sisältöä ei ole välttämätöntä rajoittaa pelkän html-koodin muodostamiseen. Niihin voi sisällyttää myös muita www-ohjelmoinnin piirin kuuluvia menetelmiä, kuten javascript-ohjelmia ja css-tyylitiedostoja. Dynaaminen sisältö voi puolestaan olla mitä tahansa data, joka voidaan muuntaa merkkijonoiksi.

3.3 Mallinekoneisto

Jotta mallinetiedostot saadaan koottua esitettävään muotoon, täytyy ne ajaa mallinekoneiston läpi. Tämä tulkitsee tiedostoon sijoitetut ohjauskoodit ja suorittaa tarvittavat toimenpiteet niiden perusteella. Lopputuloksena saadaan täysimuotoinen www-sivu, joka voidaan lähettää palvelimen kautta asiakkaalle. Mallinekoneisto on tunnettu termi myös muualla kuin www-ohjelmointiympäristössä. Periaate on kuitenkin aina samankaltainen: määritellyt osat korvataan dynaamisesti uudella sisällöllä.

Mallinekoneistot on yleensä kirjoitettu samalla ohjelmointikielellä, jonka yhteydessä niitä käytetään. Suorittava ohjelma hyödyntää kuitenkin ainoastaan koneen rajapintoja. Mallinekoneistot ovat kokonaisuuksia, jotka toteuttavat kaikki mahdolliset mallinetiedostojen sisältämien ohjauskoodien vaatimat toiminnot. Yleinen periaate on, että mallinetiedostoon merkityt ohjauskoodit korvataan suoritettavaksi määritellyissä operaatioissa muodostetuilla arvoilla. Koneistot tarvitsevat myös toimintoja, joiden avulla niiden käyttäytymistä voidaan kontrolloida suoritettavasta ohjelmasta käsin. Toiminnoista tärkeimpiä ovat metodit tai funktiot, jotka määrittelevät mallinetiedostoissa esiintyvien muuttujien arvoja. Kuviossa 7 on esitetty, miten muuttujien arvot asetetaan Smarty-järjestelmässä. Siinä kuvatut operaatiot alustavat ja prosessoivat edellisessä kappaleessa kuvatut mallinetiedostot. ”Assign” on yleisesti käytetty termi mallinekoneiston muuttujien antamisen yhteydessä, ja lisäksi on tavallista, että täytyy asettaa erilaisia initialisointiparametreja, jotta koneisto toimisi oikein.

index.php

```
include('Smarty.class.php');

// create object
$smarty = new Smarty;

// assign some content. This would typically come from
// a database or other source, but we'll use static
// values for the purpose of this example.
$smarty->assign('name', 'george smith');
$smarty->assign('address', '45th & Harris');

// display it
$smarty->display('index.tpl');
```

KUVIO 7. Smarty:n käyttö PHP-skriptissä (New Digital Group, inc. 2007)

Keskeinen osa mallinekoneiston toimintaa on tulkita lähdekoodi oikein, ja menetelmät tähän vaihtelevat toteutustavan mukaan. Perusperiaatteena kuitenkin on, että tiedostojen sisällöt luetaan muistiin ja käydään ne läpi kohta kohdalta. Ohjauslogiikka tunnistetaan erikoismerkkien ja avainsanojen perusteella. Mallinetiedostojen käyttäminen on eräänlaista skriptausta, mutta www-ympäristössä myös tulkitseva kieli on usein komentopohjainen, kuten PHP.

Yleensä web-template-järjestelmissä on mahdollista kääntää lähdetiedostot nopeammin suoritettavissa olevaksi kokonaisuudeksi mallinekoneiston avulla. Käytännössä tämä tarkoittaa sitä, että katsotaan, minkälaisia operaatioita tiedoston suorittaminen vaatii ja kirjoitetaan ne ylös väliaikaistiedostoon käytössä olevalla komentokielellä ilmaistuna. Näiden tiedostojen muoto voi vaihdella riippuen toteutustavasta. Suorittamalla syntynyt koodi saadaan sama lopputulos kuin aiemmin tulkitsemalla mallinetiedostot kokonaisuudessaan. Jotta tällainen järjestely toimisi, täytyy mallinekoneiston kuitenkin pysyä ajan tasalla siitä, onko lähdetiedostoihin tehty muutoksia. Myös tätä hienostuneemmat väliaikaistiedostot ovat mahdollisia, jolloin palvelimen suorituskyky paranee. Esimerkiksi osa dynaamisesti muodostettavasta tiedosta voi olla jo valmiiksi tiedostossa, jos se pysyy usein samana.

3.4 Kääntäminen ja välitiedostot

Monissa mallinekoneistoissa on ominaisuus, jolla mallinetiedostot voidaan kääntää helpommin tulkittavissa olevaan muotoon käyttöä varten. Tämä tarkoittaa sitä, että samalla kun koneisto tulkitsee merkinnät ensimmäistä kertaa, se kirjoittaa uuden tiedoston. Siihen sijoitetaan ainoastaan suoritettaviksi tarkoitetut operaatiot. Tiedosto ei enää sisällä mallinekoneiston omaa koodia, vaan on kirjoitettu käytössä olevalla ohjelmointikielellä, esimerkiksi PHP:llä. Tämä ei tarkoita sitä, ettei mallinekoneiston omia operaatioita voitaisi käyttää. Tiedosto voidaan ajaa suoraan, kun kutsutaan pohjana toiminutta mallinetta; näin koneiston oman ohjauskielen tulkitsemisesta aiheutuvat suorituskustannukset saadaan poistettua lähes kokonaan.

Joissain tapauksissa voidaan mennä vielä lähdemallineiden kääntämistä pidemmälle, ja sivuista on mahdollista tehdä staattisia väliversioita. Käytännössä tämä tarkoittaa usein sitä, että mallinetiedoston perusteella tehdään html-tiedosto, joka sisältää jossain tietyssä tilanteessa haetut dynaamiset tiedot. Välitiedostoa kutsutaan, kun tilanteen vaatimat ehdot täyttyvät. Kääntäminen tapahtuu monesti oletuksena, koska se ei juuri vaikuta sivujen toimintaan. Välitiedostojen muodostaminen voi puolestaan vaikuttaa toimintanopeuteen dramaattisesti, mutta samalla sivujen dynaamisuus voi kärsiä. Ohjelmoijan täytyy huolehtia siitä, että sivuja päivitetään tarpeen vaatiessa.

3.5 Erilaisia web-template-järjestelmiä

3.5.1 Smarty

Jo aiemmin mainittu Smarty on eräs PHP:llä kirjoitettu web-template-järjestelmä. Siinä käytetään mallinetiedostoja, joissa ohjauskoodi on oletuksena merkitty kaarisulkujen sisään. Merkintätavan voi määritellä uudestaan tarvittaessa. Smarty tukee huomattavaa määrää erilaisia toimintoja ja on melko helppokäyttöinen. Sen suunnitteluperiaatteena on esitysmuodon erottaminen varsinaisesta sovelluksesta. Tämä idea sallii melko kattavan loogisten ominaisuuksien käytön mallineissa,

kunhan ne eivät tee muita kuin esitykseen liittyviä tehtäviä. Tällainen lähestymistapa voi muuan muassa selkeyttää www-ohjelmistoprojektissa toimivien henkilöiden työnjakoa. Smarty tukee myös näkyvyysalueeltaan rajoitettujen parametrien välitystä alimallinetiedostoihin. Mallinetiedoissa suoritettaviin operaatioihin on mahdollisuus tehdä omia laajennuksia. Yksi huomionarvoinen piirre on, että Smarty:ssa on mahdollista määritellä rajoitetusti näkyviä muuttujia alimallineisiin. Tämä lisää ohjelmointiin selkeyttä. (New Digital Group, inc. 2007.)

Smarty:n mallinekoneisto koostuu käytännössä yhdestä luokasta, joka sisältää kaikki suorituksen kannalta olennaisimmat toiminnot. Erilaisille mukana tuleville laajennuksille ja asetusmekanismeille on omat lähdetiedostonsa. Resurssihakemistot, kuten mallinetiedostojen sijainnit, määritellään ensimmäisenä, kun luokasta on luotu instanssi. Tämän jälkeen on mahdollista käyttää olion eri metodeja sivunmuodostusoperaation määrittelyä varten. Sivu tulostetaan lopuksi ”display”-metodilla. Smarty tukee monipuolisesti välitiedostoja. Tiedostoista muodostetaan käännetty versio ensimmäisen ajon yhteydessä. (New Digital Group, inc. 2007.)

3.5.2 FastTemplate

FastTemplate on myös PHP-kielen yhteydessä käytettävä web-template-sovellus. Sen erikoispiirteenä on huomattavan yksinkertainen toteutusmalli. FastTemplate käyttää mallinetiedostoja, joissa muuttuja merkitään kaarisulkeiden sisään. Smarty:sta poiketen tämä sovellus ei kuitenkaan tue monimutkaisempia loogisia merkintöjä. Toimintaperiaatteena on, että ainoastaan tiedostoihin kirjoitetut muuttujat korvataan niille asetetuilla arvoilla. Tämä luonnollisesti asettaa joitain rajoituksia esitysasun muodostamisen suhteen ja pakottaa joitain muotoiluun liittyviä toimintoja sovelluslogiikan puolelle. Oikein toteutettuna FastTemplate:ssa on menetelmä sisäkkäin sijoitettuja lähdetiedostojen käsittelemiseen. (CDI & Moore 1999.)

Käytön eri vaiheet muistuttavat huomattavasti edellä mainitun Smarty:n vastaavia. Niitä on kuvattu alla olevassa koodiesimerkissä (kuvio 8). Ensin alustetaan pääluokka ja määritellään käytettävät mallinetiedostot. Mallinetiedostot muunnetaan vuorotellen ”parse”-metodilla puhtaaseen html-muotoon, ja lopuksi suoritetaan kokonaisuuden tulostus, johon käytetään ”FastPrint”-metodia. Koodiesimerkki muodostaa www-sivun, jossa on kolmirivinen taulukko. Esimerkistä on helppo huomata, kuinka taulukon rivien tulostaminen hoidetaan silmukalla isäntäsovelluksesta käsin. FastTemplate ei tue välitiedostoja tai kääntämistä, joten sen yksinkertainen rakenne saattaa jopa hidastaa palvelimen toimintaa jossain tapauksissa. (CDI & Moore 1999.)

```
<?
include("class.FastTemplate.php3");
$tpl = new FastTemplate("/path/to/templates");
$tpl->define( array( main   => "main.tpl",
                     table  => "table.tpl",
                     row    => "row.tpl"    ));

$tpl->assign(TITLE,"FastTemplate Test");

for ($n=1; $n <= 3; $n++)
{
    $Number = $n;
    $BigNum = $n*10;
    $tpl->assign( array(  NUMBER      =>  $Number,
                          BIG_NUMBER  =>  $BigNum ));

    $tpl->parse(ROWS,".row");
}
$tpl->parse(MAIN, array("table","main"));
Header("Content-type: text/plain");
$tpl->FastPrint();
exit;
?>
```

KUVIO 8. FastTemplate:n toimintaa (CDI & Moore 1999)

3.5.3 SmartTemplate

PHP-kehittäjät pitävät itseään selvästi älykkäänä joukkona, kun toinenkin alustalle suunniteltu web-template-järjestelmä on nimetty näin osuvasti. Ominaisuuksiltaan SmartTemplate:n voisi sanoa olevan kahden edellä mainitun

sovelluksen välimaastosta. Se sisältää hieman tulostuksessa käytettävää logiikkaa ja tukee mallinetiedostojen kääntämistä sekä varastoimista. Erilaisten toimintojen määrä ei kuitenkaan yllä Smarty:n tasolle.

Käytettävissä mallinetiedostoissa on kahdenlaisia merkintöjä ohjauskoodin kuvaukseen. Muuttujat merkitään kaarisulkeiden sisään. Ehtolauseet ja silmukat sen sijaan kirjoitetaan samalla tavoin kuin normaalit html-kommentit. Tämä voi aiheuttaa sekaannusta varsinaisten kommenttien kanssa. Tiedostoja voi asettaa sisäkkäin ”include”-komennon avulla. SmartTemplate:en voi kirjoittaa omia laajennuksia. Ne merkitään kaarisulkeiden sisään, ja niille on mahdollista antaa muuttujia parametreina. Kuviossa 9 on esimerkki mallinetiedostosta, jossa on itse määritelty laajennus. (Endelwar 2007.)

```
<html>
<body>
<!-- IF world -->
    <h1>Hello, {myExtension:world}!</h1>
<!-- ELSE -->
    <p style="color: red;">ERROR!</p>
<!-- ENDIF world -->
</body>
</html>
```

KUVIO 9. SmartTemplate mallinetiedosto

SmartTemplate perustoiminnallisuus on hyvin samankaltaista Smarty:n kanssa. Ensin luodaan mallinnusluokka ja määritellään sen asetukset. Sitten asetetaan muuttujien arvot. Lopuksi kokonaisuus parsitaan koneistossa lopulliseen muotoon ja tulostetaan.

3.5.4 Xml ja xslt www-ympäristössä

Xml ja xslt eivät ole suoranaisesti www-ohjelmoinnin tarpeisiin kehitettyjä menetelmiä, mutta niiden ominaisuudet sopivat mainiosti myös tähän

tarkoitukseen. Xml, EXtensible Markup Language, on kuvauskieli, jolla voidaan kuvata tietoa erilaisissa muodoissa. Sen perussyntaksi koostuu tageista html:n tapaan. Olennaisin ero on, että xml ei ole rajoitettu tiettyihin nimikkeisiin, vaan elementtien nimet voidaan vapaasti keksiä. Koodia ei myöskään koskaan varsinaisesti suoriteta. Sitä käytetään tietovarastona. Kun elementtejä sijoitetaan sisäkkäin, on mahdollista muodostaa puumaisia rakenteita, joissa ulommat tagit kokoavat sisemmät yhtenäiseksi kokonaisuudeksi. Aloitus- ja lopetusmerkkien väliin sijoitetaan haluttu informaatiosisältö. Elementeille voidaan myös asettaa erilaisia attribuutteja. Kuviosta 10 voi nähdä xml-tiedoston perusrakenteen.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="cdcatalog.xsl"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
.
.
.
</catalog>
```

KUVIO 10. Xml-tiedosto, jolle on määritelty xsl-tyylitiedosto (Refsnes Data 2007)

Xslt on lyhenne sanoista ”EXtensible Stylesheet Language Transformation”. Sen avulla voidaan toteuttaa erilaisia xml-koodin tulostustapoja. Kuvaukset kirjoitettaan xsl-kielellä. Kuten xml-tiedostoissa myös xsl:ää käytettäessä tarvitaan tarkat tyyppimääritykset, jotta sitä osattaisiin tulkita oikein kaikissa sovelluksissa. Jos xsl-tiedostolla halutaan muodostaa html-pohjainen www-sivu, täytyy lähdekoodiin sisällyttää sille ominaiset merkinnät. Koodin sekaan sijoitetaan xsl-elementit, jotka hoitavat xml-tiedon esityksen. Ne voivat sisältävää myös muita hyödyllisiä toimintoja web-templatejen ohjauskielien tapaan. Xml-tiedoston alkuun määritellään käytettävä xsl-tyylitiedosto. Kuviossa 11 on xsl-tiedosto, josta muodostetaan www-sivu. (Refsnes Data 2007.)

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
    <tr bgcolor="#9acd32">
      <th align="left">Title</th>
      <th align="left">Artist</th>
    </tr>
    <xsl:for-each select="catalog/cd">
    <tr>
      <td><xsl:value-of select="title"/></td>
      <td><xsl:value-of select="artist"/></td>
    </tr>
    </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>

</xsl:stylesheet>
```

KUVIO 11. Yksinkertainen xsl-tiedosto (Refsnes Data 2007)

Toimiakseen xml-xsl-yhdistelmä vaatii jonkinlaisen muuntimen, joka käsittelee lähdekoodit ja muodostaa niistä standardeja noudattavan tulosteen. Useissa ohjelmointikielissä on olemassa kirjastot xml-tekniikoiden käsittelyyn. Nämä suorittavat samanlaisia tehtäviä, kuin mallinekoneistot varsinaisissa web-template-järjestelmissä. Yleisimmissä selaimissa on sisäänrakennettu tulkki xml-pohjaisille kielille. Kuitenkin jos halutaan generoida valmiita www-sivuja dynaamisesti xsm/xslt-yhdistelmän avulla, muunnokset täytyy pääsääntöisesti suorittaa palvelinpäässä. Esimerkiksi PHP:ssä on olemassa tuki tälle menetelmälle. Kuviossa 12 olevassa koodiesimerkissä käytetään Sablotron-moduulia. On myös huomionarvoista, että www-standardien kehittämisestä vastaava W3consertium suosittelee xml/xslt-yhdistelmän käyttöä. Xml-kielen käyttö tiedon kuvaamiseen on standardi, jota monet erilaiset sovellukset ymmärtävät. Toisaalta, jos tieto ei ole valmiiksi tässä muodossa, voi xslt-menetelmän käyttö osoittautua työlääksi. (Makogon 2003; Refsnes Data 2007.)

```
<?php
include("class.xslt.php");

$t = new XSLTransformer('test.xml', 'test.xsl');

print $t->getOutput();

$t->destructXSLTansform();
?>
```

KUVIO 12. Xsl-muunnos PHP-ympäristössä (Makogon 2003)

3.5.5 Java Server Pages

Java on täysipainoinen ohjelmointikieli ja tarjoaa näin laajat mahdollisuudet erilaisten sovellusten kehittämiselle. Pelkästään www-ohjelmointia varten on olemassa joukko erilaisia tekniikoita, kuten esimerkiksi servlet- ja applet-sovellukset. ”Java Server Pages” kuuluu myös tähän joukkoon ja perustuu mallinetiedostojen käyttöön. Sen käyttökynnys riippuu pitkälti siitä, miten hyvin Java-maailma on entuudestaan tuttu. Periaatteessa JSP on kuitenkin selkeä ja vaivaton tapa muodostaa dynaamisia www-sivuja. Siitä tavoiteltavat hyödyt käyvät hyvin yksiin muiden web-template-järjestelmien kanssa. Esityksen erottaminen sovelluslogiikasta on päämääränä.

Mallinetiedostojen päätteeksi asetetaan ”.jsp”. Sisältö noudattaa pääpiirteissään samankaltaista muotoa kuin aiemmin mainituissa mallinejärjestelmissä. Html-koodin sekaan upotetaan erikoistageja, joiden avulla kerrotaan näissä kohdissa suoritettavat toimenpiteet tai määritellään muodostettavalle sivulle ominaisia attribuutteja. Oletusmerkintätapana näissä suoritusta ohjaavissa skripteissä käytetään hakasulkeita, joiden sisäpuolella on prosenttimerkit. Näiden merkintöjen sisään kirjoitettava koodi voi olla melko vapaamuotoista. Lausekkeet voivat jopa katketa ja jatkua html-koodin jälkeen seuraavan erikoistagin sisässä. Pitkää skriptillä toteutettua ohjelmanosaa kutsutaan ”scriptlet”:ksi. Tämän lisäksi JSP-ympäristö tunnistaa joukon erityisiä jsp-tageja, joilla monet sivujen tärkeimmistä toiminnoista kuvataan. Kuviosta 13 on nähtävissä edellä mainittuja JSP-tiedostolle tyypillisiä asioita. Jos muotoa verrataan esimerkiksi PHP:llä

kirjoitetuihin skripitiedostoihin, havaitaan tiettyä samankaltaisuutta tyylissä, jolla dynaamiset osat liitetään staattisten joukkoon. Voidaankin päätellä, että JSP-mekanismi on rakenteeltaan varsin erilainen kuin PHP:llä toteutetut mallinejärjestelmät, koska se muistuttaa enemmän itse kielen toimintaa. Käytännössä JSP:tä käyttävä Java-järjestelmä voi toteuttaa kaikki mahdolliset palvelinpäässä suoritettavat komennot mallinetiedostossa asetettujen määritysten pohjalta. Esimerkiksi PHP-pohjaisissa sovelluksissa mallinekoneita kutsutaan suorituksessa olevista komentotiedostoista käsin, jolloin suorituksen kulku määräytyy näiden koodien mukaisesti. (Sun Microsystems, inc. 1999.)

```
<%@ page import="hello.NameHandler" %>

<jsp:useBean id="mybean" scope="page"

          class="hello.NameHandler" />
<jsp:setProperty name="mybean" property="*" />

<html>
<head><title>Hello, User</title></head>
<body bgcolor="#ffffff" background="background.gif">

<%@ include file="dukebanner.html" %>

<table border="0" width="700">
<tr>
<td width="150">   </td>
<td width="550">
<h1>My name is Duke. What's yours?</h1>
</td>
</tr>
<tr>
<td width="150"   </td>
<td width="550">
<form method="get">
<input type="text" name="username" size="25">
<br>
<input type="submit" value="Submit">
<input type="reset" value="Reset">
</td>
</tr>
</form>
</table>

<%
    if ( request.getParameter("username") != null ) {
%>

<%@ include file="response.jsp" %>

<%
    }
%>

</body>
</html>
```

KUVIO 13. Eräs JSP-tiedosto (Sun Microsystems, inc. 1999)

JSP-sivuihin voidaan upottaa ulkoisia Java-ohjelmia, jotka suoritetaan sivuja kutsuttaessa. Ohjelmille voidaan asettaa parametreja, ja niistä saadut paluuarvot on mahdollista tulostaa sivulle. Tämä tuo html-koodiin uudelleenkäytettävyyttä ja laajennettavuutta, ja erottaa käyttöliittymän selkeästi muista kerroksista. Tapa,

jolla dynaaminen tieto hankitaan, ei vaikuta www-sivun rakenteeseen. Tiedot saatetaan esimerkiksi hakea tietokannasta tai parsia xml-tiedostosta. Lomakkeista saatua tietoa voidaan käsitellä ja tallentaa palvelimelle. JSP-sivut tulkitaan kahdessa vaiheessa. Nämä vaiheet tukevat hyvin html-lomakkeen käsittelyä. Kun sivu ladataan ensimmäisen kerran, JSP-koneisto tulkitsee tiedoston sisällöstä ne osat, jotka se tässä vaiheessa pystyy. Tuloksena muodostetaan Java-luokka, yleensä ”servlet”. Kun käyttäjä esimerkiksi lähettää sivulla olevan lomakkeen, käsitellään muodostettu luokka saatujen lomakeparametrien perusteella. (Sun Microsystems, inc. 1999.)

Kun Javaa käytetään asiakas-palvelin-arkkitehtuurin yhteydessä, asettaa se palvelinsovellukselle joitain erikoisvaatimuksia. Java on itsessään tulkattava kieli. Palvelimen välityksellä suoritettavat ohjelmat täytyy ajaa virtuaalikoneen avulla, mikä voi vaikuttaa palvelimen suorituskykyyn joissain tapauksissa. Java-palvelinympäristöiksi on kehitetty ratkaisuja, jotka tarjoavat monipuoliset ja tehokkaat käyttömahdollisuudet, kunhan niiden toimintaan perehtyy tarpeeksi hyvin. JSP-tiedostojen tulkinta suoritetaan yleensä Java-palvelimen yhteydessä olevalla moduulilla.

3.6 Hyvät puolet

Web-template-järjestelmien käytössä on monia etuja, vaikka ne eivät olekaan aina täysin yksiselitteisiä. Mallinetiedostojen avulla saadaan käyttöliittymän ja ulkoasun rakentamiseen ylimääräinen kehitysrajapinta. Tätä kautta voidaan kirjoittaa kaikki selainpäässä käytettävät osat, mikä käytännössä tarkoittaa html-koodia ja Javascript-ohjelmia. Selaimessa suoritettavien skriptien ja css-tyylitiedostojen dynaaminen valinta helpottuu, ja esimerkiksi www-sivun ulkoasun graafisia ominaisuuksia suunniteltaessa ei tarvitse tuntea kuin html-kuvauskieli ja mallinekoneistolle ominainen syntaksi ohjausrakenteiden tekemiseen. On myös mahdollista, että dynaamiset osat kirjoitetaan valmiiksi ja luodaan perusmalli sivuasettelulle, jolloin html ja tyylitiedostot jäävät ainoiksi osa-alueiksi, jotka tarvitsee tuntea. Tämä voi auttaa jakamaan työtehtäviä ohjelmointiin ja graafiseen suunnitteluun perehtyneiden henkilöiden kesken.

Jos käyttöliittymään sisällytettävän dynaamisen tiedon muoto määritellään riittävän selkeästi, on sovelluslogiikka helppo erottaa omaksi kerroksekseen. Tällöin voidaan ajatella, että määritysten puitteissa koko käyttöliittymä voidaan esimerkiksi tehdä kokonaan uusiksi ilman, että sillä on vaikutusta muihin sovelluksen osiin. Käänteisesti katsottuna käyttöliittymä voidaan rakentaa sovelluksen mallinekoneistoon syöttämien tietojen määritysten pohjalta. Muoto täytyy vain olla jokaisessa tapauksessa hyvin tiedossa. Jotain tiettyä käyttöliittymää ei yleensä pysty ainakaan kokonaisuudessaan käyttämään usean eri sovellusten kanssa. Kuitenkin, toteutuksen dynaamisuuden asteesta riippuen, osia käyttöliittymästä voidaan käyttää uudelleen erilaisissa kohteissa, jos mallinetiedostojen kirjoittamisessa on hyödynnetty tarpeeksi modulaarisuutta.

3.7 Huonot puolet

Web-templatejen huonoja puolia on vaikea mennä suoralta kädeltä määrittelemään. Yleisellä tasolla tarkasteltuna mallinekoneistot voivat ratkoa monia ongelmia. Olemassa olevat ratkaisut pääsevät pitkälti tavoitteisiin, jotka niille on asetettu. On kuitenkin tärkeä muistaa, että tarve täytyy olla olemassa, ennen kuin siihen voidaan etsiä ratkaisua. PHP-kielen yhteydessä asiaa voidaan pohtia siinä mielessä hieman kriittisemmin, että se on itsessään jo tulkattava kieli. Näin mallineiden kanssa joudutaan tekemään kaksi eri tulkitsemisoperaatiota: PHP ja mallineet. Lisäksi mallinekoneistojen kieli poikkeaa yleensä hieman PHP:stä, jolloin sen syntaksin opetteleminen on tarpeellista, joskaan ei yleensä ohjelmointiin perehtyneelle kovin työlästä. PHP-kielellä ohjelmoitaessa on myös muita tapoja erottaa käyttöliittymä omaksi kerroksekseen. Lisäksi mallinekoneiston ohjauskielenä voitaisiin käyttää myös perusmuotoista PHP:tä, jos sovellus tehtäisiin sopivalla tavalla. (Lozier 2003.)

Liittyen mallinetiedostojen tulkitsemiseen ja muuttujien asettamiseen web-template-järjestelmillä on vaikutusta myös palvelimen suorituskykyyn. Valmiiden ratkaisujen monimutkaisuuteen on hankala vaikuttaa. Nopeuden ratkaisee loppujen lopuksi tekniikka, jolla tulkinnat ja muunnokset suoritetaan, sekä

mahdollisuus ohjauskoodien kääntämiseen ja välitiedostoihin. Yleisesti ottaen mallineiden muuntaminen on selkeästi hitaampaa kuin pelkän PHP-kielen tulkinta, jos kaikki suorituksen vaiheet käydään läpi. Täytyy kuitenkin muistaa, että samojen ominaisuuksien aikaansaaminen vaatisi toteutuksessa käytettävistä työkaluista riippumatta monimutkaisia sovellusrakenteita. Rakenteellinen selkeys, laajennettavuus ja käyttöönoton helppous tulevat usein ohjelmistotuotannossa suorituskyvyn kustannuksella.

4 IT-YRITYKSEN KOULUTUSPORTAALI

4.1 Alkutilanne

Projekti lähti liikkeelle, kun Flander Oy otti sähköpostiviestillä yhteyttä Lahden Ammattikorkeakoulun ohjelmistotekniikan opettajiin. Yhteydenoton aiheena oli löytää tekijä yrityksen koulutuskäyttöön kaavaillulle www-sovellukselle. Samalla tämä työ voisi toimia opinnäytetyön aiheena jollekin oppilaalle. Opettajat ehdottivat minua työhön, koska olin suorittanut suuren osan opinto-ohjelmaan valinnaisena kuuluvan www-ohjelmointimoduulin kursseista.

Aluksi pidimme muutamia tapaamisia yrityksen paikallisen yksikön edustajien kanssa, joissa päätimme toteuttaa hankkeen. Projekti noudatti alusta lähtien ohjelmistokehityksen perusprosessia. Ensimmäiseksi määrittelimme aiotun sovelluksen päävaatimuksia ja toteutuksen pääpiirteitä, kuten ohjelmointikielen ja -ympäristö. Alustavasti ideana oli, että ohjelma olisi rakennettu yrityksen Java-pohjaisen ”Daisy”-dokumentinhallintajärjestelmän yhteyteen (Outerthought bvba 2007). Tästä kuitenkin luovuttiin, koska toteutus tämän sovelluksen avulla osoittautui liian monimutkaiseksi, vaikka ehkä olisikin ollut mahdollinen. Päädyimme perinteisempään www-sovellusratkaisuun, jossa koko sivusto rakennetaan alusta lähtien omaksi kokonaisuudekseen. Melko pian aloin kirjoitella ensimmäisiä versioita määrittelydokumenteista omalla ajallani.

Dokumenteista tehtiin muutamia versioita yhteistyössä Flanderin vastuuhenkilön kanssa ja vaatimusluettelo alkoi täydentyä. Ajankohta läheni kevätlukukauden loppua. Sovelluksen varsinainen suunnitteluvaihe, jossa toteutukseen lopullinen rakenne päätettäisiin, saattoi alkaa. Kun kouluvuosi loppui, pääsin tekemään työtä yrityksen tiloihin ja samalla sain suoritettua puuttuvia työharjoittelukuukausia. Uusia vaatimusmäärityksiä tuli vielä toteutusvaiheen aikana jonkin verran. Noin puolen vuoden kuluttua tärkeimmät toiminnot oli koodattu ja sovellusta päästiin testaamaan. Löydetyt virheet ja puutteet korjattiin. Lopuksi hoidettiin dokumentointi kuntoon niiltä osin, kuin se oli puutteellinen.

4.2 Tärkeimmät vaatimukset

Koulutussovelluksen tuli olla laajasti yrityksen työntekijöiden ja tarvittaessa myös asiakkaiden käytössä. Pääajatuksena oli, että käyttäjät voisivat suorittaa erilaisia koulutuspaketteja sovelluksen avulla, kunhan heillä vain olisi www-selain ja yhteys internetiin. Myös järjestelmän ylläpidon ajateltiin tapahtuvan www-pohjaisen käyttöliittymän kautta. Koulutusten suorituksen tuli onnistua lukemalla aihetta varten kootut materiaalit ja vastaamalla html-lomakkeisiin tulostettaviin kysymyksiin. Tulokset otettaisiin talteen, ja niitä voitaisiin myös myöhemmin tarkastella, jos siihen olisi tarvetta.

Koulutusmateriaalit oli tarkoitus saada muodostettua yhdistelemällä dokumentteja, joita oli varastoitu yrityksen dokumentinhallintajärjestelmään. Tähän tarkoitukseen käytettiin ilmaista, Java-pohjaista sovellusta nimeltä ”Daisy”. Erilaisten vaihtoehtojen tutkimisen jälkeen päädyttiin eriyttämään koulutussivusto omaksi kokonaisuudekseen. Sivustolle ominaista määrittelyä noudattavat dokumentit voitaisiin tuoda siihen käsin, mikä onnistuisi lataamalla ne palvelimelle html-lomakkeen kautta. Järjestelmään syötettävien tiedostojen muodoksi määriteltiin lopulta www-dokumentti, koska sen elementit olisi helppo näyttää sivustossa. Suorituksen tehostamiseksi dokumentti ajateltiin pilkkoa pienempiin osiin, jotka voitaisiin vuorotellen ladata ruudulle. Materiaalin uudelleenmuodostukseen tarvittiin ”parseri”-tyyppinen moduuli. Se tarkoittaa ohjelmanosaa, joka käy läpi syötetyn lähdemateriaalin ja muokkaa sen uuteen muotoon löytämiensä määritemerkintöjen perusteella. Esitysten sisältöä tuli myös pystyä päivittämään. Samalla versionumeroista täytyi pitää kirjaa.

Koulutussivusto haluttiin rajata yrityksen työntekijöihin ja mahdollisesti joidenkin asiakkaiden käyttöön. Tämä vaati käyttäjätunnusten muodostamista ja tarkistusta kirjauduttaessa järjestelmään. Flander Oy:n työntekijöiden käyttäjätiedot oli valmiiksi kerätty yrityksen LDAP-tekniikkaa käyttävälle palvelimelle. Sitä tuli hyödyntää myös tekeillä olleen sivuston käyttäjäntunnistuksessa. Varsinaisia käyttäjäoikeuksia ei kuitenkaan ollut mahdollista hakea LDAP:sta, koska niitä ei ollut sinne asetettu. Käyttäjien oikeudet täytyi tallentaa sivuston paikalliseen tietokantaan. Sinne kerättiin myös asiakaskäyttäjien kaikki tiedot, koska niitä ei

olisi ollut muualta saatavissa. LDAP-käyttäjälle täytyi luoda paikalliset tunnukset automaattisesti ensimmäisen kirjautumisen yhteydessä, ja samalla häneltä piti kerätä työhön liittyvää informaatiota tilastointia varten. Työntekijöiden sähköpostiosoitteet haettiin LDAP-palvelimelta. Käyttäjäntunnistuksen piti siis osata hakea käyttäjätietoja kahdesta paikasta.

Myös sivuston sisäisiä toimintoja haluttiin rajoittaa erilaisia tehtäviä hoitaville henkilöille. Käyttäjätasoja tehtiin kolme: Alin taso käsitti normaalit käyttäjät, jotka saattoivat suorittaa koulutuspaketteja ja muokata rajallisesti omia käyttäjätietojaan. Koulutuksista oli mahdollista suorittaa ainoastaan ne, joihin käyttäjä oli ilmoitettu. Seuraava taso tarkoitettiin niiden henkilöiden käyttöön, jotka lisäisivät omia koulutuspakettejaan sivustolle. Näillä käyttäjillä olisi oikeus lisätä kysymyksiä ja esityksiä sekä muokata niitä, joissa heidät on merkitty omistajiksi. Lisäksi tämä käyttäjäryhmä saisi tarkastella tilastoja ja muokata muiden käyttäjien ilmoittautumisia luomilleen kursseille. Korkein käyttäjätaso oli tarkoitettu järjestelmän ylläpitäjille. Heillä olisi oikeus muokata kaikkia tietoja, joita www-käyttöliittymän kautta oli mahdollista. Ainoastaan ylläpitokäyttäjillä olisi oikeus lisätä, muokata tai poistaa käyttäjätietoja.

Kuten jo aiemmin mainittiin, sovelluksen käytön tuli tapahtua yleisen www-selaimen avulla. Käyttöliittymän vaatimukset oli jaettavissa kahteen osaan. Tärkeintä oli, että käyttäjät voisivat kirjautua sisään sivustolle ja suorittaa kursseja, joihin heidät on ilmoitettu. Toisaalta sivuston toiminnan mahdollistaminen vaati erilaisia ylläpitotyökaluja, joihin olisi rajoitettu pääsy. Kurssien suoritusosaan suunniteltiin malli, jossa esitysmateriaalia selataan sivu kerrallaan eteenpäin ja tarvittaessa taaksepäin. Pääkappaleiden välissä näytetään määritelty määrä välikysymyksiä, jotka arvotaan tietokannasta. Kun näihin on vastattu, näytetään selite. Se kertoo, miksi oikea vastaus oli juuri kyseinen. Esityksen lopuksi käyttäjälle asetetaan lopputentti, mikäli hänen ilmoittautumisensa on vielä voimassa. Kokeen kysymyksistä täytyy saada tietty prosenttimäärä oikein, jotta kurssi läpäistään onnistuneesti. Kysymykset voivat olla samoja kuin aiemmin, mutta ne arvotaan kaikkien kappaleiden joukosta. Viimeisellä sivulla suunniteltiin näytettävän yhteenveto käyttäjän suoritteista kyseessä olevassa esityksessä. Selaamisen mahdollistavien hyperlinkkien lisäksi

käyttöliittymän haluttiin navigaatiopalkki, jonka avulla olisi mahdollista siirtyä nopeasti esityksen osasta toiseen. Se kertoisi myös visuaalisin merkein, missä kohtaa esitystä ollaan menossa. Ylläpitotyökaluille ei ollut alun perin kovin tarkkoja määritteitä, ja ne muotoutuivat lopulliseen muotoonsa sovelluksen kehityksen lomassa. Niissä oli kuitenkin tarkoitus käyttää html-lomakkeita tiedon lähettämiseen palvelimelle. Ylläpitotyökalut olivat jaettavissa kolmeen eri osa-alueeseen: rekisteröinti, tilastot sekä tietojen haku ja muokkaus.

Sivuston oli myös tarkoitus lähettää käyttökohteen mukaan vaihtelevia sähköpostiviestejä asiakkaille. Näin he saisivat tietää esimerkiksi, onko heidät kirjattu uudelle kurssille tai ovatko heidän ilmoittamisensa vanhenemassa. Myöhemmässä vaiheessa vaatimuksiin lisättiin myös salasanan uusiminen sivustoa käyttävien asiakashenkilöiden osalta. Tämä tarkoitti niitä, joiden tietoja ei tarkasteta LDAP-palvelimen avulla. Uusi salasana lähetettäisiin käyttäjälle automaattisen sähköpostiviestin mukana.

4.3 Ohjelmointiympäristö

Sovelluksen alustaksi suunniteltiin alusta alkaen jotain maksutonta ohjelmointiympäristöä. Sisäisen palvelun toteutukseen ei haluttu varata turhia resursseja. Kun vaatimukset alkoivat selvitä, päädyttiin varsin nopeasti tulokseen, että LAMP-arkkitehtuuri olisi paras ratkaisu tähän tarkoitukseen. Se pitää sisällään tunnettuja, vapaalla lisenssillä kehitettyjä, tuotteita: Linux-käyttöjärjestelmä, Apache-www-palvelin ja MySQL-tietokantapalvelin. Www-palvelimeen voidaan liittää jokin yhteensopiva palvelinohjelmointikieli, kuten PHP, Perl tai Python. Alusta tarjoaa vakaan ja monipuolisen pohjan vaativammillekin verkkosovelluksille. Se on myös helppo saada käsiinsä ja melko vaivaton pystyttää.

Ohjelmointikieleksi valittiin PHP, koska sen ominaisuudet olivat riittävät vaadittujen toimintojen toteuttamiseen. Siinä on rajapinnat kommunikoinnille MySQL- ja LDAP-palvelimien kanssa. Lisäksi PHP oli tekijälle entuudestaan kaikkein tutuin mahdollisista palvelinpään ohjelmointimenetelmistä. Siihen on

myös saatavilla kattavat ohjeistukset internetissä. Suureen osaan yleisimmistä ongelmista löytyy tietoa.

Sovelluksen kehitystä varten pystytettiin erillinen fyysinen palvelin, johon sijoitettiin www-palvelinsovellus ja tietokanta. Toimintaa oli mahdollista kokeilla koodin kirjoituksen yhteydessä ilman, että se häiritsi yrityksen muita verkkopalveluja. LDAP-palvelin sijaitsi yrityksen sisäisessä verkossa omassa fyysisessä sijainnissaan. Kaikki ohjelmakoodit sijoitettiin valmistellulle kehityspalvelimelle. Myöhemmin sen rinnalla alettiin käyttää paikallisessa verkossa ollutta ”Subversion”-versionhallintajärjestelmää. Sen tarkoitus oli tässä tapauksessa lähinnä säilyttää varmuuskopioita. Versionhallintaan ei tarvinnut kirjoitusvaiheessa kiinnittää juuri huomiota, sillä projektissa oli vain yksi ohjelmoija. Käytössä olevaan koodiin ei siten tullut muita poikkeamia versiointijärjestelmässä olevaan verrattuna, kuin ne joita ohjelmoija teki.

4.4 Web-template-arkkitehtuuri

PHP on ominaisuuksiltaan laaja ohjelmointikieli, ja se tukee näin monenlaisia ohjelmistoarkkitehtuureja. Toisaalta sen muoto palvelinpäässä suoritettavana komentokielenä asettaa joitain rajoituksia käyttömahdollisuuksille. Lähtökohtaisesti oli tarkoitus tehdä sovelluksesta mahdollisimman hyvin laajennettava. Lisäksi koodin uudelleenkäytettävyyteen haluttiin kiinnittää jonkin verran huomiota. Yleinen tapa lisätä ohjelman ylläpidon tehokkuutta on erottaa käyttöliittymä sovelluslogiikasta. Tätä tarkoitusta varten on PHP:n yhteyteen kehitetty erilaisia web-template-ratkaisuja. Erään tällaisen valintaan päädyttiin, koska se vaikutti helppokäyttöiseltä ja toiminnoiltaan riittävän monipuoliselta.

Yksinkertaisen mallinejärjestelmän toteuttaminen itsekään ei olisi kovin monimutkaista. Valitulle ohjelmointikielelle on kuitenkin olemassa niin monia valmiitakin ratkaisuja, että oman toteutuksen tekeminen pienimuotoisempaan sovellukseen ei yleensä kannata. Tässä tapauksessa tarkoitukseen valittiin sovellus nimeltä ”Smarty”, joka on yksi suosituimmista ja monipuolisimmista tarjolla olevista ratkaisuista. Smarty:n käyttöön on myös saatavissa runsaasti

ohjeita. Ohjelma on ”Lesser GNU General Public Lisence”-lisenssin alainen. Tämä merkitsee sitä, että sovellusta voi käyttää ilmaiseksi, jos noudattaa tiettyjä lähdekoodin julkaisemiseen liittyviä sääntöjä. Smarty:n liitettävät, itse kirjoitetut ohjelmanosat täytyy julkaista, jos tehtyä tuotetta on aikomus myydä tai jaella ulkopuolisille. Tässä tapauksessa kirjoitetut koodit tulivat kuitenkin ainoastaan yrityksen sisäiseen käyttöön. (Free Software Foundation, inc. 2007.)

Mallinejärjestelmä perustuu ajatukseen, että osa tiedoston sisällöstä korvataan dynaamisella datalla. Loput tiedostossa olevat merkinnät pysyvät siinä muodossa, kun ne on kirjoitettu. Muuttumattomiksi määritellyt osat kuvaavat käyttöliittymän perusrakennetta. Korvattavat osat puolestaan sisältävät yleensä sivun dynaamisen sisällön ja mahdollisesti joitain navigaatioelementtejä. Hieman monimutkaisemman käyttöliittymän tulostuksessa tarvitaan yleensä erilaisia ehtolauseita ja silmukoita. Ohjauskoodia ei tästä huolimatta pitäisi sotkea varsinaisen sovelluslogiikan kanssa. Muuten käyttöliittymän erottamisen periaate kärsii. Samalla ohjelman rakenteellinen selkeys vähenee. Tiedon haku ja muodostus tulee suorittaa sovelluslogiikan puolella. Näytettävät osat tulostetaan näiden määritysten perusteella.

Käyttöliittymälle asetetut vaatimukset ohjasivat mallinetoteutuksen rakenteen muodostamista. Yrityksellä oli sivuasettelu, jota käytettiin muutamassa muussakin www-sovelluksessa. Sitä aiottiin noudattaa myös tässä tapauksessa. Asettelussa sivun yläosaan tulostettiin ”pääosa”. Se piti sisällään pari kuvaa ja linkkejä kohteisiin, joista sai tietoa yrityksestä. Sivun vasemmassa reunassa oli navigaatiopalkki, josta pääsi hyperlinkkien avulla käsiksi kyseessä olevan sivuston eri toimintoihin. Oikeaan reunaan puolestaan tulostettiin palkki, joka saattoi muun muassa näyttää uutisia ja muuta yleistä tietoa. Keskelle sivua oli varattu tila varsinaiselle sisällölle. Koulutussivustossa päätettiin käyttää tätä samaa mallia. Sivun keskiosaan tulostettaisiin esitykset ja työkalut, kulloisenkin sivutyypin määritysten mukaan. Ylä- ja sivupalkit muodostettaisiin osittain dynaamisten määritteiden pohjalta, mutta suureksi osaksi ne olisi kirjoitettu staattisesti omiin tiedostoihinsa. Oikealla olevaan palkkiin päädyttiin loppujen lopuksi sijoittamaan ohjeita kulloisenkin toiminnon käyttämiseen.

Alimallineet olivat hyvä työkalu käyttöliittymän suunnittelussa ja toteutuksessa. Koska sivun keskiosaan tulostettavat toiminto-osat saattoivat poiketa toisistaan huomattavasti, täytyi niille myös rakentaa erilaiset alimallinetiedostot. Näytettävät alimallineet valittiin päämallinetiedostossa välillisesti URL:ssa kuljetettavan ”type”-parametrin mukaan. Päämallinetiedosto, ”index.tpl”, sisälsi sivun perusrakenteen. Sivun asettelu ja ositus hoidettiin html-kuvauskielen taulukkoelementin avulla. Sen sisältämiin soluihin ladattiin pääsääntöisesti omat alimallineensa Smartyn ”include”-komennolla. Näin eri osien kuvaukset saatiin asetettua omiin tiedostoihinsa. Tämä helpotti hahmottamaan niitä erillisinä kokonaisuuksina ja selkeytti päätiedoston rakennetta huomattavasti. Kuviosta 14 on nähtävissä portaalin pääsivu tulostettuna selaimeen. Asettelun eri osat ovat erotettavissa.

KUVIO 14. Portaalin pääsivu

Sivutyypin mukaan määrittyviä mallinetiedostoja tehtiin kymmenen erilaista:

- login.tpl.
- logout.tpl.
- forgot.tpl.

- main.tpl.
- show.tpl.
- questions.tpl.
- search.tpl.
- edit.tpl.
- register.tpl.
- stats.tpl.

Käytännössä jokainen sisälsi käyttöliittymän omalle sivutyypilleen ominaisen toimintojen suorittamiseen. Toiminnot muodostettiin yleisten vaatimusten pohjalta. Niitä olivat kirjautumiseen liittyviin operaatiot, pääsivun muodostus, resurssien etsiminen, lisäys, poisto ja muokkaus, esityksen selaus ja kysymysten suorittaminen sekä rekisteröinti ja tilastojen katselu. Salasanan uusiminen sai myös oman sivutyyppinsä, vaikka se onkin melko pieni toiminto. Sivutyypit tunnistivat erilaisia parametreja, jotka vaihtelivat toimintokohtaisesti. Näiden avulla pystyttiin hakemaan saatavilla olevista tietolähteistä erilaisia sisältöjä mallinetiedoston käyttöön. Tarvittaessa määriteltiin myös sivun ulkoasuun liittyviä ominaisuuksia. Toiminnallisuuteen keskittyneiden alimallineiden lisäksi tehtiin tiedostoja, jotka muodostivat erilaisia pienempiä kokonaisuuksia sivuston tarpeisiin. Muun muassa tietojen lisäämiseen ja muokkaukseen käytettävän lomakkeen tulostus on toteutettu tiedostossa nimeltä ”form_fields.tpl”. Siinä käsitellään taulukkoa, joka pitää sisällään kenttien arvot ja määritteitä siitä, miten ne tulisi näyttää sivulla.

Mallinetiedostorakennelman yhteyteen asetettiin kaksi erityistä toimintoa helpottamaan sovelluksen sisällön laajennettavuutta. Vaikka alun perin oli ajatuksena käyttää mahdollisimman vähän Javascript selainohjelmointia, muutamat toiminnot oli käytännöllisintä toteuttaa tällä menetelmällä. Javascript-koodit täytyy liittää html:n sekaan tai viitata ulkoiseen tiedostoon, jossa ne sijaitsevat. Näin selain onnistuu löytämään ne. Päämallinetiedostoon tehtiin ehtolause, jonka perusteella osataan liittää kulloinkin kyseessä olevalle sivulle siihen kuuluvat Javascript ohjelmat. Järjestelmän generoimat viestit puolestaan sijoitettiin kaikki samaan mallinetiedostoon. Niitä näytetään tilanteissa, joissa käyttäjä odottaa tietoa esimerkiksi toiminnon suorituksen onnistumisesta. Eri

viesteille asetettiin liput, joiden perusteella niitä näytetään. Liput muodostetaan sovelluslogiikan puolella ja asetetaan Smartyn käyttöön. Viestitekstien muokkaus helpottui, koska kaikki ovat löydettävissä samasta sijainnista.

Perustarpeet täyttävän ohjauslogiikan lisäksi Smartyn ominaisuuksiin kuului joukko erikoistuneita funktioita. Näitä oli mahdollista käyttää muun muassa tulosteiden ulkoasun lopullisen muodon määrittelemiseen ja muiden käyttöliittymään kiinteästi liittyvien yksityiskohtien tuottamiseen. Funktioille oli mahdollista asettaa myös parametreja, jotta tuloksesta saataisiin halutun kaltainen. Näitä pienoismoduuleja, eli ”ekstensioita”, oli myös mahdollista kirjoittaa itse haluamiinsa tarkoituksiin. Tätä Smartyn ominaisuutta ei kuitenkaan osattu rakennetussa sovelluksessa juuri hyödyntää. Suurin osa käyttöliittymän vaatimista määrityksistä asetettiin valmiiksi jo ennen Smarty:n kutsumista. Sitten oikeat vaihtoehdot valittiin tulostettaviksi mallinetiedostossa olevilla ehtolauseilla. Tämä vähensi ehkä kirjoitettavan koodin määrä, mutta lisäsi sen monimutkaisuutta jonkin verran. (New Digital Group, inc. 2007.)

4.5 Syötteiden käsittely

Olennainen osa dynaamista www-sovellusta ovat erilaiset parametrit, joita välitetään http-pyynnön mukana asiakkaalta palvelimelle. Näiden avulla pystytään päättelemään, mitä resursseja ja toimintoja halutaan käytettäviksi. On olemassa kaksi yleistä tapaa, joilla parametreja välitetään. Jos käytetään ”get”-metodia, ne määritellään URL-rivillä. Tämä tarkoittaa usein sitä, että sivuston sivujen sisältämät hyperlinkit luodaan dynaamisesti käsittämään erilaisia muunnelmia tarjolla olevasta parametrivalikoimasta. Linkkien osoittamiin kohteisiin rakennetaan sivut linkissä olleiden määritteiden perusteella. Toinen tapa välittää tietoa on ”post”-metodi, jota käytetään lähinnä html-lomakkeiden yhteydessä. Se lähettää lomakkeen sisältämät parametrien arvot palvelimelle. Erikoispiirteenä on se, että mukana siirrettävää tietomäärää ei ole rajoitettu eivätkä parametrit näy käyttäjälle lähetyksen jälkeen. Tämä tapa on hyvä palvelimen tietokantojen muokkaamiseen. Lomake lähetetään ja tiedot tarkastetaan, ja jos mukana on toimintoja määrittäviä parametreja, suoritetaan niiden pyytämät operaatiot.

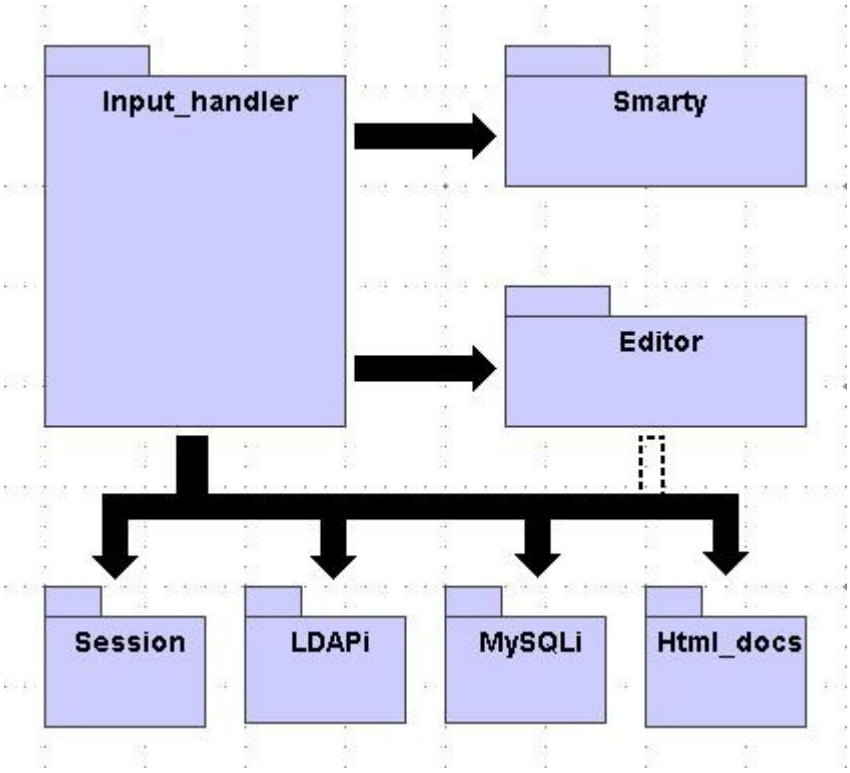
Tehdyn www-sovelluksen ydin rakentui syötteiden tulkitsemisen ympärille. Sen suoritus alkoi luomalla instanssi ”Input_handler”-luokasta. PHP-moduuli tallensi post- ja get-parametrit vastaaviin globaalisti määriteltyihin ”$_POST” ja ”$_GET” -taulukoihin. Toteutuksessa näiden käyttö oli kuitenkin rajattu ainoastaan Input_handler-luokalle. Tällaisen järjestelyn ajateltiin tuovan ohjelman rakenteeseen selkeyttä. Ensimmäiseksi tulkittiin ”type”-parametri, joka määritteli kohteena olevan sivutyypin. Arvon perusteella osattiin valita suoritettavaksi Input_handler-luokasta metodi, joka piti sisällään sivulle ominaisen ohjelmakoodin. Sivun luontiin tarkoitettujen metodien sisällöt olivat melko vapaamuotoisia. Jokaisella sivutyypillä oli omansa, mutta niiden rakenteessa oli kuitenkin myös tiettyjä samankaltaisuuksia. Ensin tulkittiin mahdolliset toimintakoodit ja suoritettiin vaaditut toimenpiteet, jonka jälkeen koottiin sivun sisältö määritellyistä lähteistä. Lopuksi välitettiin Smarty:lle tulostukseen tarvittavat muuttujat ja tietosisällöt. Metodit palauttivat totuusarvon sen mukaan, oliko tarpeellista renderoida sivu mallinekoneiston avulla. Jos tulos oli epätosi, kyseessä oli yleensä selaimen uudelleenohjaus jonkin toiminnon seurauksena. Tällöin sivun vartalo-osaa ei tarvinnut muodostaa ollenkaan.

Sivunmuodostuksessa käytettiin tarjolla olevia työkaluja. Nämä oli suurimmaksi osaksi tehty itse ja ne pitivät sisällään erilaisia menetelmiä, joiden avulla saatettiin päästä käsiksi ulkoisiin tietolähteisiin. Input_handler-luokasta oli yhteys kaikkiin muihin sovelluksen luokkiin. Näitä olivat:

- Smarty.
- Session.
- MySQLi.
- Html_docs.
- LDAPi.
- Editor.

Niiden avulla saatiin tietoja, joita käytettiin sivuston sisällön tuottamiseen ja istunnon hallintaan. ”Editor”-luokka poikkesi muista mainituista siinä mielessä, että se tehtiin suorittamaan erilaisia tiedon muokkausoperaatioita ja lisätoimintoja. Esimerkiksi suurin osa palvelimelle tallennettavasta tiedosta kulki sen läpi.

Sähköpostiviestien lähetys ja salasanojen generointi suoritettiin myös Editor-luokan avulla. Kuvio 15 näyttää miten sovelluksen eri luokat jakautuvat palvelinpäässä kerroksiin ja miten ne kutsuvat toisiaan.



KUVIO 15. Palvelinpään ohjelmistokerrokset

Sivun muodostukseen vaikuttivat omalta osaltaan myös istunnon voimassaolo ja käyttöoikeudet. Näihin ei URL-parametreilla ollut suoraa vaikutusta. Istunnon tiedot tallennettiin istuntomuuttujiin ja evästeisiin, ja käyttäjäoikeudet oli osittain rajattu sivutyyppikohtaisesti. Sallitut tyypit ja resurssit ladattiin määrätyissä sivuston sijainneissa istunnon tietoihin, josta ne saatettiin tarkastaa. Jos istunto ei ollut voimassa, käyttäjä ohjattiin automaattisesti kirjautumissivulle. Jos hän taas pyysi resurssia tai toimintoa, johon hänellä ei ollut oikeutta, tulostettiin pääsivu ja siihen virheestä ilmoittava viesti.

4.6 Liitynnät ulkoisiin tietolähteisiin

4.6.1 LDAP-liityntä

Sovellus hakee Flanderin työntekijöiden käyttäjätietoja LDAP-palvelimelta. Sitä käytetään melko yleisesti käyttäjäntunnistukseen yrityksen sisäisessä tietoverkossa. LDAP-yhteyksiä varten rakennettiin oma luokkansa nimeltä ”LDAPi”. Sitä ei tarvita jokaisella sivulla. Luokan toiminnot on toteutettu PHP:n LDAP-rajapinnan avulla. Tämä vaatii jonkin yhteensopivan LDAP-asiakasmoduulin asennusta PHP:n yhteyteen. LDAP-funktiot on kapseloitu tämän luokan sisään, eikä niitä käytetä muissa sovelluksen osissa. Kun funktioiden toimintoja tarvitaan, kutsutaan LDAPi:a. Luokan muodostimessa otetaan yhteys LDAP-palvelimeen. Sen sisältämillä metodeilla voidaan autentikoida käyttäjiä ja noutaa haluttuja tietokonaisuuksia. Käyttäjien yhteystietoja, kuten sähköpostiosoite, on mahdollista hakea anonyymisti, koska LDAP-järjestelmää käytetään usein juuri osoitetietojen hakuun. Autentikointi sen sijaan tapahtuu siten, että käyttäjän tunnuksien oikeellisuus tarkastetaan ”ldap_bind”-komennolla. Se ei varsinaisesti hae tietoja vaan sitoo yhteyden LDAP-hakemistoon. Kirjautumisen onnistuminen päätellään funktion paluuarvon perusteella. Näin käyttäjien salasanoja ei tarvitse ollenkaan käsitellä LDAP-palvelimen ulkopuolella. Paikallisessa tietokannassa säilytettävät salasanat on myös suojattu. Tähän käytettiin md5-hajautusmenetelmää.

4.6.2 MySQL-liityntä

Sivujen muodostuksessa ja tiedon varastoimisessa käytetään runsaasti MySQL-tietokantapalvelinta. Myös sen käyttöä varten on tehty oma rajapintaluokkansa ”MySQLi”. Kuten LDAPi:n kohdalla, myös tässä luokassa on toteutettu periaatetta, että PHP:n erikoisfunktiot kapseloidaan niitä käyttävän luokan sisään. MySQL-kutsut ovat olleet niin tärkeä osa PHP-kielisiä sovelluksia, että tuki niille on yleensä oletuksena mukana. Yhteys palvelimeen otetaan MySQLi-luokan muodostimessa. Muut metodit rakentuvat pääosin SQL-kielen ”insert”-, ”delete”-,

”update”- ja ”select”-komentojen ympärille. Näiden avainsanojen perään kootaan dynaamisesti erilaisia syntaksia noudattavia hakuehtoja ja parametreja. Nämä puolestaan saadaan metodeille annettavista parametreista, jotka on asetettu kutsuvassa luokassa. Muodostettujen merkkijonojen avulla suoritetaan kyselyt MySQL-tietokantaan käyttäen ”mysql_query”-komentoa. Jos kyseessä on tiedon haku, palautetaan saatu kokonaisuus metodin paluuarvossa. Nämä ovat yleensä taulukkomuotoa. Muuten käytetään arvoja tosi ja epätosi kuvaamaan operaation onnistumisastetta.

Tietokantoja käsitellessä täytyi myös ottaa huomioon muutamia yleisiä tietoturvariskejä. Hakulauseet täytyi suojata ”SQL injection”-haavoittuvuudelta. Tässä haittamenetelmässä SQL-kyselyyn yritetään lisätä ylimääräisiä ehtoja tai komentoja. Ehkäisyyn käytettiin ”msql_real_escape_string”-funktiota, joka muuttaa haitallisten erikoismerkkien muotoa lisäämällä niihin ”escape”-määreen ”\”. Toinen samankaltainen tietoturvariski on ”Cross Site Scripting”, jossa puolestaan on päämääränä tallentaa haitallisia www-selaimen tulkitsemia koodeja tietokantaan lomakkeiden kautta. Jos niitä käytetään sellaisenaan www-sivulla, voi se aiheuttaa tietoturva-aukkoja ynnä muuta haitallista toimintaa. Tämän ongelman ehkäisemiseksi on olemassa ”htmlspecialchars”-PHP-funktio, joka muuttaa html-kuvauskielen erikoismerkit koodattuun muotoon. Kaikki tietokannasta haettava tieto ajettiin sovelluksessa tämän lauseen läpi.

4.6.3 Esitysten lataus ja muokkaus

Omalaatuisin osa sovelluksen arkkitehtuuria on tapa, jolla koulutusmateriaalista koostuvat esitykset tallennetaan sivustolle. Hoitamaan tähän tarkoitukseen vaadittuja toimintoja tehtiin ”Html_docs”-luokka. Se toimii rajapintana esityksen lähdetiedostojen ja sovelluksen ydintoimintojen välillä. Luokka täytyy aina initialisoida ennen käyttöä asettamalla vaaditut lähdepolut sen jäsenmuuttujiin. Niiden avulla osataan etsiä tiedostoja oikeasta sijainnista. Polut on tallennettu tietokannan esitystauluun. Lisäksi suurin osa mahdollisista toimenpiteistä vaatii, että esityshakemiston sisällöstä muodostetaan lista sovelluksen muistiin.

Esitykset koostuvat käytännössä html-tiedostoista, kuvista, termistöstä ja kysymysten määrittelytiedostosta. Html-tiedostot sisältävät jokainen yhden esityksen kappaleen ja ne on nimetty sen otsikon mukaan. Tiedostot saadaan muodostettua parsimalla palvelimelle lähetettävä yhtenäinen www-dokumentti osiin sen sisältämien otsikkonumeroiden perusteella. Tämä lähdetiedosto voi sisältää erilaisia html-elementtejä. Jos kuvia on mukana, täytyy ne pakata päätiedoston kanssa samaan zip-pakettiin, jotta useamman tiedoston lähetys onnistuu samalla kertaa. Termit puolestaan sijoitetaan lähdedokumentin loppuun. Ne merkitään otsikolle, jonka sisällöksi tulee avainsana ”_TERMS_”. Termit ja niiden selitykset asetetaan kaksisarakkeiseen taulukkoon, josta ne osataan ottaa talteen.

Palvelimella esitykset tallennetaan kaikki omiin kansioihinsa. Tämä koskee myös saman esityksen eri versioita. Kansioiden nimet määritellään esityksen nimen ja versionumeron mukaan ja niihin asetetaan kaikki asiaankuuluvat tiedostot kuvia lukuun ottamatta. Esityskansiot sijaitsevat yhtenäisessä sijainnissa, joka määritellään sivuston asetuksissa. Myös kuvakansioiden polut määritellään näin. Kuvat sijoitetaan siten, että ne ovat esitysversion mukaan nimetyssä hakemistossa, josta ne voidaan ladata http-palvelimen kautta. Kun esityksen sisältö päivitetään, asetetaan sen kohdalle tietokantaan uusi versionumero ja hakemisto. Näin materiaaleja osataan etsiä uudesta sijainnista.

Edellä mainitun kaltaiseen toteutukseen lähdemateriaalin käsittelemiseksi päädyttiin, koska sivuston toiminnan ajateltiin nopeutuvan, kun tiedostot ovat valmiiksi melko pieniä. Esitysten sisältö ladataan www-sivulle suoraan näistä tiedostoista. Järjestely sisältää itsessään tiedostojen esitysjärjestyksen, jota ei tarvitse erikseen tallentaa tai muokata. Toisaalta sen selvittämiseen menee näin enemmän prosessoriaikaa, kuin joissain muissa toteutusvaihtoehdoissa. Esityksen lopullinen muoto näkyy kuviossa 16.

KUVIO 16. Esityksen selaus

4.6.4 Session-luokka ja istunto

Yleensä PHP-kieltä käytettäessä istuntomuuttujille on varattu oma taulukkonsa ”$_SESSION”. Jokainen taulukon solu voi pitää sisällään avain-arvoparin, joka säilyy tarvittaessa niin kauan, kuin istunto on voimassa. Tehdyssä sivustossa istuntotaulukon käyttö haluttiin kapseloida oman luokkansa sisään. Muutenkin istunnonhallinta oli hyvin tärkeässä osassa sovelluksen toimintaa ajatellen, koska tilojen hallinta on olennainen osa portaalityyppisen sivuston toimintaa. Näin syntyi ”Session”-luokka. Sen metodien avulla voitiin käynnistää tai lopettaa istunto, asettaa istuntomuuttujien arvoja sekä tarkastaa, onko istunto voimassa. Metodeja kutsuttiin yleensä Input_handler-luokasta käsin. Istunnon tarkastus suoritettiin aina sivuja kutsuttaessa. Jos se ei ollut voimassa, käyttäjä ohjattiin kirjautumissivulle. Muutamille sivuille tosin pääsi myös ilman kirjautumista.

Suurin osa istuntomuuttujista tallennettiin siinä vaiheessa, kun istunto luotiin. Tämä tapahtui välittömästi, kun käyttäjän syöttämät kirjautumistiedot oli

varmistettu oikeellisiksi. Käyttäjätunnus, asiakkaan ip-osoite, käyttäjätaso ja lista sallituista sivutyypeistä olivat tärkeimmät istuntoon asetettavat tiedot. Sallittujen esitysten lista otettiin myös ylös. Sitä päivitettiin aina käyttäjän saapuessa portaalin pääsivulle. Istunnon satunnainen tunnus tallennettiin asiakasselaimen evästeeseen. Ilman tämän sallimista sivuille ei ollut mahdollista kirjautua. Jokaisen sivun yhteydessä evästettä päivitettiin siten, että sen voimassaoloa kasvatettiin asetetun minuuttimäärän verran. Jos eväste ehti vanheta, täytyi käyttäjän kirjautua uudestaan sisään. Istunnon saattoi myös päättää valinnaisesti siihen tarkoitetusta linkistä. Tällöin kaikki istuntomuuttujat ja evästeet tuhottiin.

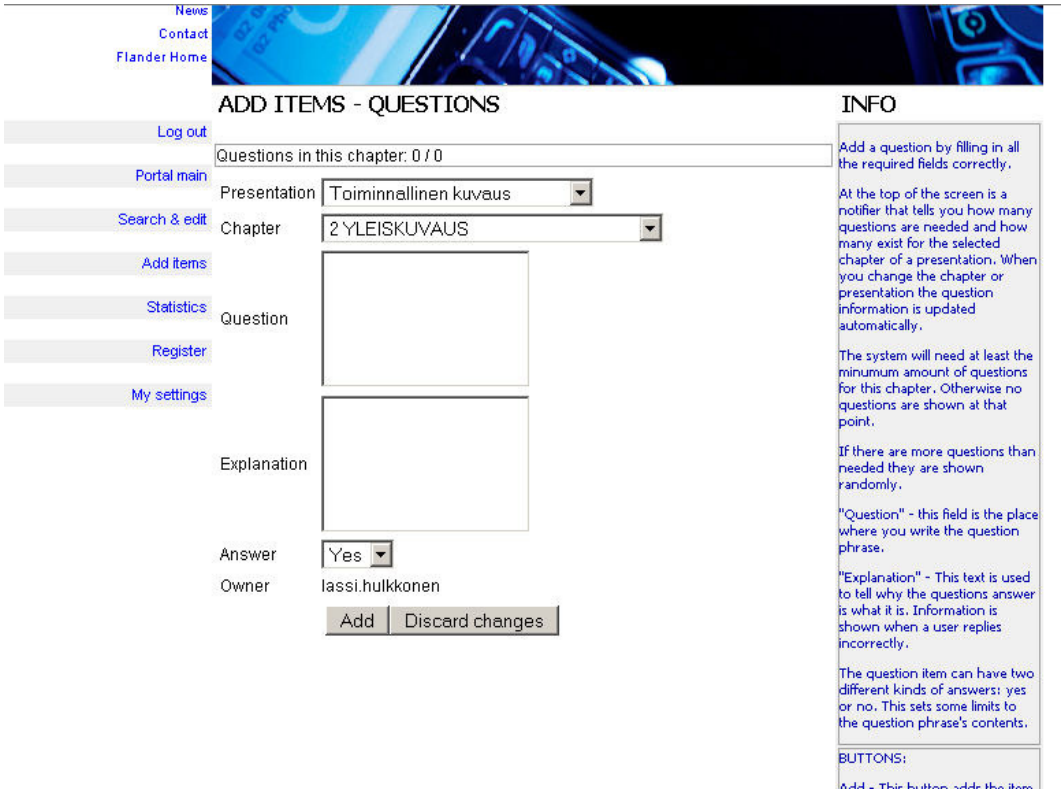
4.7 Lopullinen ulkoasu ja käyttö

Sivuston ulkoasun toteutunut muoto noudatti pitkälti sille asetettuja vaatimuksia. Sivuasettelu oli järjestelty näyttämään yrityksen mallin mukaiselta. Kirjautumissivulla, josta toiminnallisuus yleensä alkaa, oli kaksi kenttää joihin saattoi syöttää käyttäjätunnuksen ja salasanan. Lisäksi täällä oli linkki, joka johti salasanan uusimissivulle. Sivu koostui pääosin yhdestä kentästä, johon piti syöttää olemassa oleva käyttäjätunnus (mieluiten käyttäjän oma). Tunnuksen perusteella generoitiin käyttäjälle uusi salasana ja lähetettiin hänen sähköpostiosoitteeseensa. Hyväksytyn kirjautumisen jälkeen käynnistettiin istunto ja käyttäjä ohjattiin pääsivulle. Tässä näkymässä tulostettiin lista kursseista, joihin käyttäjä oli ilmoitettu, sekä tietoa niiden statuksesta, muun muassa paras suoritus ja umpeutumispäivämäärä. Kurssiotsikot sisälsivät linkit, joista pääsi katselemaan niiden sisältöjä.

Kun käyttäjä aloitti esityksen suorituksen, ensimmäisenä hän näki ruudun, jossa selostettiin, kuinka kurssin suoritus tapahtuu käytännössä. Tämän jälkeen alkoi varsinainen esitys. Sitä saattoi selata eteen ja taakse tähän tarkoitukseen muodostetuista linkeistä. Kaikkien esityssivujen yläosaan tulostettiin palkki, jossa kerrottiin senhetkinen kohta. Aluksi näytettiin opetusmateriaaleja kappale kerrallaan, kuitenkin siten, että toiseksi ylimmän kappaletason kanssa samaan näkymään sisällytettiin kaikki sen aliluvut. Materiaalisivuilla näytettiin yläpalkissa tietoa esityksen sisältämistä kappalenumeroista. Välikysymykset

tulivat pääkappaleiden lopussa, jos niitä oli kappaleeseen asetettu. Käyttäjän tehtäväksi jäi valita oikea vaihtoehto kahdesta jokaiseen kysymykseen. Vastaamisen jälkeen näytettiin, menivätkö vastaukset oikein ja selitteet niihin. Viimeisen kappaleen jälkeen tuli lopputenttisivu. Se oli samanlainen kuin muutkin kysymyssivut, ja ainoa poikkeus oli varmistusruutu, joka piti asettaa, ennen vastausten lähetystä. Lopputentin jälkeen näytettiin saadut pisteet ja kerrottiin sanallisesti, oliko suorite onnistunut. Jos se ei ollut, tulostettiin linkki, josta pääsi koettamaan lopputenttiä uudelleen. Kysymysosuuksia ei näytetty, jos kurssin suoritusaika oli vanhentunut tai se oli jo suoritettu.

Sivun vasempaan reunaan tulostettiin linkit ylläpitotyökaluihin, jos käyttäjän oikeudet olivat riittävät. Myös hakutoiminto laskettiin näihin tässä tapauksessa. Hakusivulle tulostettiin kenttä, johon oli mahdollista syöttää hakusanoja. Lisäksi piti valita, mitä kohdetyyppiä haetaan. Tyypit muodostettiin suoraan tietokannan taulujen perusteella. Jos kenttä jätettiin tyhjäksi, haettiin kaikki valitun tyypin kohteet. Tulokset asetettiin listaan, jossa kerrottiin niiden tietoja. Vasemmassa reunassa oli valintalaatikot kullekin tulosriville, joista saattoi valita kohteita muokattavaksi. Valintamahdollisuutta rajoitettiin omistusoikeuden mukaan. Muokkaussivulla tulostettiin lomake, johon sijoitettiin käsittelyssä olevan kohteen arvot. Lomakkeiden muodot vaihtelivat kyseessä olevan tyypin mukaan. Kuviossa 17 on kysymysten lisäykseen tarkoitettu lomake. Tietojen oikeellisuus tarkistettiin tallennettaessa muutoksia. Mahdollisista virheistä ilmoitettiin käyttäjälle seikkaperäisesti. Vasemmasta navigointipalkista pääsi myös tietokohteiden lisäykseen. Siinä valittiin ensin tyyppi, sitten tulostettiin samanlainen lomake kuin tietojen muokkauksessa. Muutenkin toiminta oli samankaltaista. Jos kyseessä oli esitysten muokkaus tai lisäys, näytettiin kuvaus sen kappalerakenteesta, mikäli lähdemateriaali oli onnistuneesti lähetetty palvelimelle.



KUVIO 17. Kysymyslomake

”Manager”- ja siitä korkeamman luokan käyttäjille oli pääsy myös tilastojen katseluun ja rekisteröintityökaluun. Tilastoissa valittiin ensin, halutaanko asioita tarkastella käyttäjien vai kurssien näkökulmasta. Tämän jälkeen tulostettiin lista kaikista valintaan liittyvistä mahdollisuuksista. Kun kohde oli valittu, näytettiin siitä yksityiskohtaisia tilastotietoja, muun muassa pistekeskiarvoja ja onnistuneiden suoritteiden suhdetta rekisteröintien määrään. Rekisteröintityökalussa oli suodatin, josta oli mahdollista rajoittaa sivulle haettavia käyttäjiä erilaisten määritteiden perusteella. Listattuja käyttäjiä oli mahdollista rekisteröidä valittavissa olevalle kurssille. Sivulla olevaan alasvetovalikkoon koottiin kaikki kurssit, jotka senhetkinen käyttäjä oli luonut. Myös tietylle kurssille rekisteröidyt käyttäjät oli mahdollista hakea listaan. Tämän avulla oli mahdollista uusia käyttäjien rekisteröintejä, jos ne olivat vanhentuneet, tai poistaa heitä kurssilta.

5 JOHTOPÄÄTÖKSET

Koulutusportaalin tekeminen oli monessa suhteessa haastava projekti. Ennen kaikkea ohjelmistosuunnittelun eri vaiheiden merkitykset onnistuneen lopputuloksen kannalta korostuivat. Tämä ei tarkoita sitä, että kaikki olisi sujunut täysin suunnitelmien mukaan. Määrittelyn ja suunnittelun tärkeys tuli esiin enemmänkin työn aikana nousseiden ongelmien kautta. Toisaalta sovelluksen kehityksessä noudatetut periaatteet laajennettavuudesta ja koodin uudelleenkäytettävyydestä toivat helpotusta, kun vaatimuksia jouduttiin muokkaamaan tai lisäämään. Sovelluksen arkkitehtuuri oli riittävän joustava uusien toimintojen lisäämiseen. Tämä kuitenkin aiheutti monesti sen, että koodi ei ollut enää niin selkeää, kuin olisi toivonut. Sovelluksen arkkitehtuuri oli ihan tyydyttävä, jos ottaa huomioon suunnittelijan vähäisen kokemuksen tässä suhteessa. Web-template-järjestelmän käyttö osoittautui yhdeksi toimivaksi ratkaisuksi toteutetun kaltaisen www-sovelluksen tarpeisiin.

Www-ympäristö asettaa omat rajoituksensa sovelluksen toiminnalle. Luonteenomaisin piirre on käyttöliittymän toteutus internetin välityksellä. Ulkoasu rakennetaan html-kuvauskielellä kirjoitettujen määritysten perusteella ja toimintojen suorittamineen tapahtuu sivun päivitysten mukaan. Vaikka menetelmä alun perin tarkoitettiin dokumenttien lukemiseen, on se osoittautunut yllättävän hyväksi myös interaktiivisten sovellusten toteutukseen. Myös html-elementtien rajoitetusta määrästä huolimatta on mahdollista tehdä monipuolisia käyttöliittymiä, jos niiden eteen on valmis näkemään hieman vaivaa. Usein www-sivuista halutaan ulkoisesti vetävän näköisiä, koska ne palvelevat yleensä ihmisten välittömiä informaatiotarpeita. Monesti voitaisiin jopa sanoa, että sivujen koristeellisuus ja hyödyllinen tietosisältö menevät toistensa kanssa ristiin. Tasapainoinen yhdistelmä toiminnallisuutta ja visuaalista näyttävyyttä olisi varmasti ihanteellinen tavoite.

Loppukäyttäjälle selvästi näkyvin osa on sivun ulkoasu. Riippumatta siitä, miten kattavia ja hienostuneita operaatioita palvelimella suoritettaisiin, käyttäjä näkee ainoastaan rajapintana toimivan www-dokumentin, eikä yleensä ole kovin tietoinen sovelluksen muusta suorituksesta. Käyttöliittymän vasteaika täytyy

tietysti pyrkiä pitämään siedettävänä. Jos palvelu sijoitetaan internetiin, sen halutaan yleensä tavoittavan suuren käyttäjäkunnan. Tämä asettaa sovelluksen käytettävyydelle tiettyjä vaatimuksia. Mainittujen seikkojen valossa voidaan sanoa, että www-sovelluksen käyttöliittymän suunnittelu vaatii erityistä huomiota. Visuaalisten elementtien ja varsinaisen sovelluslogiikan toteutus voi vaatia varsin erilaisia osaamisalueita. Tästä syystä nämä tehtävät monesti jaetaankin aloihin erikoistuneiden henkilöiden kesken. Tällöin olisi hyvä, että html-dokumenttien suunnittelu ei vaatisi ohjelmoinnin osaamista, sillä esimerkiksi graafikot eivät usein ole myös tämän alan asiantuntijoita. Työnjaon tärkeys riippuu usein ennen kaikkea olemassa olevasta käytännöstä ohjelmointiprojektien suhteen ja rakennettavan sovelluksen laajuudesta. Koulutusportaalia tehdessä työnjako ei ollut tarpeellista, koska se oli ajateltu lähinnä yhden henkilön tehtäväksi. Toisaalta käyttöliittymä kehittyi vielä toteutusvaiheen aikana ja vaatimusten muodostamiseen liittyy useita henkilöitä. Voidaan ajatella, että käytetty kehitysmalli toi sovellukseen joustavuutta, joka salli käyttöliittymän muutokset vielä toteutusvaiheen aikana.

Käytännön ratkaisuna käyttöliittymän toteutukseen web-template-järjestelmä toimi riittävän hyvin. Kokonaisuus oli selkeä työstää ja lähdekoodit kirjoitettiin oman tyyppisiin tiedostoihinsa erilleen muusta koodista. Rajapinta sovelluslogiikan ja mallinejärjestelmän välillä oli sopivan yksinkertainen, mutta samalla sen toiminnot riittivät hyvin, eikä kaikkia järjestelmän hienouksia ylipäätään tullut käytetyksi. Näiltä osin luottamus tällaiseen ratkaisuun säilyi hyvin vielä työn loputtuakin. Valitun järjestelmän yhteydessä myös ohjauskoodien syntaksi oli varsin johdonmukaista, ja se oli nopea oppia. Muutenkin mallineet oli helppo integroida osaksi kokonaisuutta. Tämä johtui varmasti osittain siitä, että ohjelmointikieli oli yhteinen muiden sovelluksen osien kanssa, jolloin ohjelmointiympäristön tarjoama tuki oli melko aukoton.

Arkkitehtuurin osalta monipuolisen ja tehokkaan sovelluksen rakentaminen mallinejärjestelmän yhteyteen vaatii selvästi paneutumista. Jos ratkaisua käytettäisiin esimerkiksi suoraan lineaarisesti suoritettavien komentosarjojen kanssa, hyödyt jäisivät minimiin. Hienostuneempi, sovelluksen vaatimusten pohjalta suunniteltu, oliopohjainen ratkaisu olisi suositeltava. On hyvä huomioida,

että esimerkiksi PHP-kielen olio-ohjelmointiominaisuuksia on parannettu uusimpien versioiden myötä. Ei ole kovin hyödyllistä erottaa käyttöliittymäkerrosta, jos sovelluksessa ei ole muita selviä kerroksia. Myös palvelimen suorituskyky saattaa laskea, koska mallinejärjestelmät vaativat enemmän prosessoriaikaa suorittaessaan muunnoksia. Tätä pystytään kuitenkin pitkälti rajoittamaan, jos mallinetiedostot voidaan purkaa väliaikaismuotoon, jolloin ne on supistettu sisältämään ainoastaan välittömästi tarvittavat komennot.

Sivuja ei ehditty käyttää varsinaisessa tarkoituksessaan, eli etätyönä suoritettavaan koulutukseen. Testikäytössä ne tuntuivat vasteajaltaan melko nopeilta, jos ottaa huomioon, että suoritettavia komentoja oli monissa osissa melko paljon. Myös testaus oli sen verran kattavaa, että pahimmat virheet varmasti saatiin poistettua. Jos www-selaimen kautta tapahtuvaa koulutusta ajatellaan laajemmalta kannalta, se soveltuu hyvin yksiselitteisten asioiden opettamiseen, jotka eivät vaadi käytännön opastusta tai mielipiteiden vaihtoa. Portaalin kursseja voisi myös käyttää tiedotteiksi tai dokumenteiksi, jotka jonkin ryhmän kaikkien jäsenten halutaan varmasti lukevan tai etätyönä suoritettavana alustuksena johonkin laajempaan koulutuskokonaisuuteen. Selkeys ja hyvät käyttöohjeet auttavat itse opetusjärjestelmän käyttöönotossa, jos näihin tavoitteisiin on onnistuttu pääsemään.

LÄHTEET

CDI & Moore, J. 1999. FastTemplate 1.1.0 – PHP extension for managing templates, and performing variable interpolation [verkkojulkaisu]. The WebMaster's Net [viitattu 31.3.2007]. Saatavissa: http://www.thewebmasters.net/php/FastTemplate.phtml

Donnelly, M. 2000. An Introduction to LDAP [verkkojulkaisu]. Michael Donnelly – The Ldapman [viitattu 31.3.2007]. Saatavissa: http://www.ldapman.org/articles/intro_to_ldap.html

Endelwar 2007. SmartTemplate: a fast and simple template engine [online]. SourceForge.net [viitattu 31.3.2007]. Saatavissa: http://smarttemplate.sourceforge.net/

Free Software Foundation, inc. 2007. GNU Lesser General Public License [verkkojulkaisu]. Free Software Foundation, inc. [viitattu 31.3.2007]. Saatavissa: http://www.gnu.org/licenses/lgpl.html

Lozier, B. 2003. Template Engines [verkkojulkaisu]. Lozier, B. [viitattu 15.4.2007]. Saatavissa: http://www.massassi.com/php/articles/template_engines/

Makogon, S. 2003. Smarty vs. XML/XSLT [verkkojulkaisu]. iNet Interactive [viitattu 31.3.2007]. Saatavissa: http://www.devpapers.com/article/18

MySQL AB. 2007. MySQL – The world's most popular open source database [online]. MySQL AB [viitattu 31.3.2007]. Saatavissa: http://www.mysql.com

New Digital Group, inc. 2007. Smarty Template Engine [online]. The PHP Group [viitattu 31.3.2007]. Saatavissa: http://smarty.php.net

Orfali, R., Harkey, D. & Edwards, J. 1999. Client/Server Survival Guide. Uudistettu kolmas painos. Yhdysvallat: John Wiley & Sons, inc.

Outerthought bvba. 2007. Daisy the Open Source CMS [online]. Outerthought bvba [viitattu 31.3.2007]. Saatavissa: http://cocoondev.org/daisy/index.html

Parker, T. & Sportack, M. 2000. TCP/IP Unleashed. Toinen painos. Yhdysvallat: Sams Publishing.

Peltomäki, J. 1998. WWW-ohjelmointi. Jyväskylä: Teknolit Oy.

Refsnes Data. 2007. W3Schools [online]. Refsnes Data [viitattu 31.3.2007]. Saatavissa: http://www.w3schools.com

Stephens, R., Plew, R., Morgan, B. & Perkins, J. 1999. SQL-tietokantaohjelmointi – Trainer. Helsinki: IT Press.

Sun Microsystems, inc. 1999. JSP by Example [verkkojulkaisu]. Sun Microsystems, inc. [viitattu 31.3.2007]. Saatavissa: http://java.sun.com/products/jsp/html/jspbasics.fm.html

The PHP Group. 2007. PHP: Hypertext Preprocessor [online]. The PHP Group [viitattu 31.3.2007]. Saatavissa: http://www.php.net

W3C®. 2007. World Wide Web Consortium [verkkojulkaisu]. W3C® [viitattu 31.3.2007]. Saatavissa: http://www.w3c.org