Juha Kaari

KESKUSTELUOMINAISUUDEN SUUNNITTELU JA TOTEUTUS WWW-JULKAISUJÄRJESTELMÄÄN

Tietotekniikan koulutusohjelma

2008

samk
Satakunnan ammattikorkeakoulu

KESKUSTELUOMINAISUUDEN SUUNNITTELU JA TOTEUTUS WWW-JULKAISUJÄRJESTELMÄÄN

Kaari, Juha
Satakunnan ammattikorkeakoulu
Tietotekniikan koulutusohjelma
Joulukuu 2008
Aarinen, Reino
UDK: 004.42, 004.738.52
Sivumäärä: 73

Asiasanat: Microsoft .NET, ASP .NET, SQL, relaatiotietokannat, blogit

---

Nykyajan Internet on kaksisuuntaisen viestinnän väline. Useat sivustot sisältävät toimintoja, jotka mahdollistavat sivustoa käyttävien ihmisten tuottaa omaa sisältöä. Sisältö voi olla esim. yrityksen tuotteesta annettava palaute, lehtiartikkeliin liittyvä mielipide tai tietosanakirjaan lisättävä tieto. Puhutaan sosiaalisesta mediasta, Web 2.0:sta.

Tämän opinnäytetyön tarkoituksena oli suunnitella ja toteuttaa keskusteluominaisuus Hypermedia Oy:n Hyperscriptor XG 3.5 WWW-julkaisujärjestelmään. Julkaisujärjestelmää ylläpidetään web-selaimella käytettävän käyttöliittymän avulla. Ylläpitäjän ei tarvitse osata HTML-kieltä, vaan sisältö tuotetaan WYSIWYG-editorilla (what you see is what you get). Hyperscriptor on modulaarinen järjestelmä, joka sisältää perusominaisuuksien lisäksi useita lisäominaisuuksia, joita kutsutaan moduuleiksi. Tässä työssä toteutettiin järjestelmään uusi moduuli, joka mahdollistaa keskustelujen lisäämisen ylläpidettäviin sivuihin.

Internet-sovellusten toteuttamiseen on olemassa useita eri tekniikoita, kuten Java ja PHP. Hypermedia Oy:n julkaisujärjestelmä on toteutettu Microsoftin ASP.NET-tekniikalla sekä SQL Server -tietokannalla. Koska kyseessä oli järjestelmän jatkokehitys, työssä käytettyjen menetelmien valinta oli helppo tehdä. Teoreettisessa osuudessa kerrotaan .NET-arkkitehtuurista ja erityisesti siihen kuuluvasta ASP.NET-tekniikasta. Lisäksi tutustutaan relaatiotietokantoihin sekä lyhyesti Web 2.0:aan.

Opinnäytetyö aloitettiin vuoden 2008 toukokuussa. Ensimmäiseksi tehtiin yksityiskohtainen toiminnallinen määrittely, joka sisältää keskustelumoduuliin liittyvät näkymät ja ominaisuudet. Toiminnallisen määrittelyn sisältämät ominaisuudet jaettiin kolmeen osaan, joista tässä opinnäytetyössä toteutettiin ensimmäinen. Työn tekninen toteutus saatiin valmiiksi elokuun lopussa. Keskusteluominaisuus jää odottamaan ensimmäistä todellista testiään eli asiakasta, joka haluaa sen osaksi julkaisujärjestelmäänsä.

DESIGNING AND IMPLEMENTING A DISCUSSION FEATURE FOR A WWW PUBLISHING SYSTEM

Kaari, Juha
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in Information Technology
December 2008
Aarinen, Reino
UDK: 004.42, 004.738.52
Number of Pages: 73

Key Words: .NET, ASP .NET, SQL, relational databases, blogs

Nowadays the Internet is a tool for two-way communication. A considerable number of web sites include features that allow Internet users to produce their own content. The content can be, for example, feedback about a company's product, opinions concerning a written article or information for a dictionary. These characteristics are often presented as Web 2.0.

The objective of this thesis was to design and implement a discussion feature for the Hypermedia Ltd's Hyperscriptor XG 3.5 WWW publishing system. The system is administered by a graphical interface via an ordinary web browser. Skills of HTML are not required because the content is created trough a WYSIWYG (what you see is what you get) editor. Hyperscriptor is a modular system which consists of multiple modules in addition to the basic features. The result of this thesis is a new module, which enables the user to add discussions to the pages managed with the system.

There are a number of technologies, such as Java and PHP, to develop web applications. Hypermedia Ltd's publishing system is developed with the Microsoft ASP.NET web application framework and the SQL Server database. The selection between the methods was easy to make because the new module is part of an existing system and therefore there were no reasons to use any other technologies. The theoretical part of this thesis concerns .NET Framework and especially ASP.NET, which belongs to it. Relational databases and Web 2.0 are also explored.

The thesis was started in May 2008. The first part of the thesis concerned designing the new discussion module. The result of the designing was a document which includes all user interfaces and features of the new module. The features were divided in three parts and the first part was implemented in this thesis. The technical part of the thesis was completed at the end of August. The next step is to meet the real life testing situation, the first customer that wants the discussion feature as a part of its publishing system.

SISÄLLYS

TERMILUETTELO

ADO.NET	Kokoelma tietokannan käsittelyyn tarvittavia luokkia.
ASAX	ASP.NET-sovelluksen alustustiedoston tiedostopääte.
ASCX	ASP.NET-käyttäjäkontrollin tiedostopääte. Käyttäjäkontrolli on ASPX-sivuun upotettava erillinen sivu.
ASMX	ASP.NET Web Services -sovelluksen tiedostopääte.
ASP	Active Server Pages. Microsoftin kehittämä dynaamisten Internet-sivujen tekemiseen tarkoitettu tekniikka, joka yhdistää HTML-kieltä sekä skriptikoodia.
ASP.NET	Mircosoftin .NET-arkkitehtuuriin kuuluva Internet-ohjelmointitekniikka.
ASPX	ASP.NET Web Forms -sivun tiedostopääte.
C#	Microsoftin .NET-konseptia varten kehittämä ohjelmointikieli.
CLR	Common Language Runtime. .NET-arkkitehtuuriin kuuluva ohjelmien ajonaikainen ympäristö.
Code Behind	Tekniikka, jolla erotetaan ASP.NET-sovelluksen koodi käyttöliittymän toteuttavasta tiedostosta.
Eväste	Internet-käyttäjän koneelle tallennettavaa tietoa, joka mahdollistaa käyttäjäistuntojen ylläpitämisen. Web-sovellus tunnistaa käyttäjän evästeeseen sijoitettavan tunnisteen avulla.
IIS	Internet Information Services. Microsoftin WWW-palvelinohjelmisto.
JIT	Ajonaikainen käännös, jossa .NET-sovelluksen välikoodi (MSIL) käännetään lopulliseksi konekieliseksi koodiksi.
MIME-tyyppi	Määrittelee datan sisällön tyypin.

MSIL	Kehitysympäristön/kääntäjän tuottama välikoodi, jota ei voida ajaa suoraan vaan tarvitaan lisäksi täsmäkäännös (JIT).
.NET	Microsoftin kehittämä ohjelmistoteknologia, joka sisältää ajonaikaisen ympäristön sekä luokkakirjastoja.
.NET-assembly	Tarkoitetaan kokoelmaa, joka voi muodostua yhdestä tai useammasta ohjelma- sekä resurssitiedostosta. Ohjelma-tiedostot voivat olla EXE- tai DLL-tyyppisiä.
Palvelinkontrolli	Palvelinpäässä käsiteltävä komponentti esim. tekstikenttä.
PostBack	ASP.NET -sivun uudelleenlataus (lomakeohjelmat).
SQL	Relaatiotietokantojen käsittelyyn tarkoitettu kieli.
SQL Server	Microsoftin tietokantaohjelmisto.
T-SQL	SQL-kielen laajennus (Transact-SQL).
ViewState	Sivunlatausten välillä käytettävä tiedon tallennussäiliö.
Visual Basic .NET	Microsoftin .NET-perheeseen kuuluva ohjelmointikieli.
Web 2.0	Termi, jolla viitataan nykyajan Internetin sosiaalisiin ja teknisiin piirteisiin (blogit, wikit, AJAX, RSS jne.).
Web Forms	Selainpohjaisista sovelluksista, siihen kuuluvista sivuista sekä palvelinkomponenteista käytettävä nimitys.
Web Services	Nimitys puhtaasti toiminnallisista palveluista, joita toiset sovellusympäristöt käyttävät.
XML	Metakieli, jonka avulla voidaan kuvata tietoa tiedosta. Sisältää itse datan ja siihen liittyvät tiedot, kuten datan nimi.

1 JOHDANTO

”Uskomme, että bloggaaminen ei ole vain viisasta, vaan suorastaan välttämätöntä, jos yritys haluaa päästä lähemmäksi asiakkaitaan”, totesivat Robert Scoble ja Shel Israel kirjassaan Blogit ja bisnes. Interaktiivisuus on kasvavissa määrin tärkeä osa nykypäivän Internetiä, niin ihmisten jokapäiväisessä elämässä kuin yritysmaailmassakin. Yhä useampi Internet-sivusto sisältää jonkinlaisen keskustelualueen tai blogin, joka mahdollistaa omien mielipiteiden jakamisen toisten ihmisten kanssa maailmanlaajuisesti ja ennen kaikkea reaaliajassa.

Tässä opinnäytetyössä suunniteltiin ja toteutettiin Hypermedia Oy:n toimeksiannosta uusi ominaisuus yrityksen Hyperscriptor XG 3.5 WWW-julkaisujärjestelmään. Hypermedia Oy on toiminut digitaalisen viestinnän ja e-liiketoiminnan järjestelmien asiantuntijayrityksenä vuodesta 1989 lähtien. Yrityksen julkaisujärjestelmä on tehty aikaisemmin insinöörityötutkintoon kuuluvana opinnäytetyönä. Nyt järjestelmän oli aika siirtyä syvemmälle Web 2.0 -aikaan uuden keskusteluominaisuuden muodossa. Ominaisuus mahdollistaa keskustelujen lisäämisen julkaisujärjestelmän sivuihin. Keskustelu voi kohdistua esim. Internet-julkaisun artikkeliin, yrityksen tuotteeseen tai blogiin. Keskusteluja lisätään ja ylläpidetään julkaisujärjestelmän ylläpidon käyttöliittymällä, jonka käyttämiseen tarvitaan pelkästään web-selain.

Internet-ohjelmointiin on olemassa useita erilaisia tekniikoita, kuten Java, ASP.NET ja PHP. Tämän opinnäytetyön toteutukseen käytettiin ASP.NET-tekniikkaa. Tekniikan valinta oli selvä, sillä Hypermedia Oy toimii Microsoftin ympäristössä ja sen kehittämä julkaisujärjestelmä on toteutettu ASP.NET-tekniikalla sekä SQL Server -tietokannalla. Työn teoriaosassa kerrotaan Microsoftin .NET-arkkitehtuurista ja erityisesti siihen kuuluvasta ASP.NET-tekniikasta. Lisäksi tutustutaan relaatiotietokantoihin sekä nykyajan Internetiin ja sen ominaispiirteisiin, Web 2.0:aan. Varsinaisessa työosassa kerrotaan keskusteluominaisuuden suunnittelu- ja toteutusvaiheista.

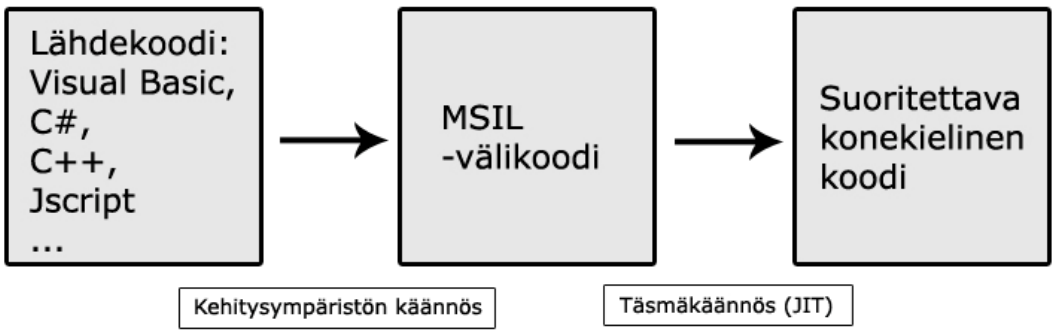
2 MICROSOFT JA INTERNET

2.1 .NET

.NET Framework on Microsoftin kehittämä ajonaikainen ympäristö, johon ohjelmoijan on helppo tehdä hyviä ja vakaita ohjelmia. Ohjelmat suoritetaan ympäristön sisällä. .NET tarjoaa hyödyllisiä ajonaikaisia ominaisuuksia kuten automaattisen muistinhallinnan ja järjestelmäpalveluiden helpon käytön. (Platt 2001, 7.)

.NET arkkitehtuuri koostuu kahdesta osasta: ohjelmien ajonaikainen ympäristö (Common Language Runtime, CLR) ja luokkakirjasto (.NET Framework Class Library). Ajonaikainen ympäristö tarjoaa palvelut ajettavien ohjelmien taustalla. Sen tehtäviin kuuluvat mm. käännetyn ohjelmakoodin ajaminen, turvallisuus ja säikeiden hallinta. Ajonaikainen ympäristö irrottaa käytettävän ohjelmointikielen suorituksesta, jonka ansiosta ohjelmointiin voidaan käyttää useita eri kieliä. Yleisimpiä kieliä ovat Visual Basic .NET ja C# (C Sharp). Arkkitehtuurin toinen osa sisältää yleiskäyttöisiä luokkia sovellusten ohjelmointiin esim. System.XML-kokoelma, josta löytyy tarvittavat ominaisuudet XML-tiedon käsittelyyn. (Inkinen 2003, 4-5.)

Kaikki CLR-yhteensopivat kehitysympäristöt kääntävät .NET-sovelluksen lähdekoodin standardoiduksi Microsoft Intermediate Language (MSIL)-välikoodiksi. Koska kaikki kehitysympäristöt tuottavat saman välikoodin, lopputulos on sama ohjelmointikielestä riippumatta. MSIL-koodia ei voida suorittaa suoraan, vaan tarvitaan täsmäkäännös (just-in-time-compilation, JIT). Täsmäkäännöksessä tuotetaan lopullinen konekielinen koodi, joka voidaan ajaa ko. alustalla. Jokaisella alustalla voi olla oma täsmäkääntäjänsä, joka tuo .NET:lle määrätyn määrän alustariippumattomuutta.



Kuva 2.1. Sovelluksen käännösvaiheet.

.NET-ympäristön automaattinen muistinhallinta perustuu roskien keruu (carbage collection) nimiseen menetelmään. Menetelmän ansiosta ohjelman ei tarvitse itse vapauttaa varaamaansa muistia, vaan ajonaikainen ympäristö vapauttaa sen automaattisesti, kun ohjelma ei enää tarvitse sitä. (Platt 2001, 20-22.)

2.2 ASP.NET

2.2.1 Taustat

Aluksi Internetiä käytettiin pelkästään staattisten sivujen jakeluun. Tällaisen toiminnon rakentaminen oli melko helppoa. Käyttäjä kirjoittaa Internet-selaimeen pyydettävän sivun osoitteen. Palvelin vastaanottaa pyynnön, etsii pyyntöä vastaavan sivun ja lähettää sen käyttäjän selaimelle. Kaikki Internetissä oleva tieto oli staattista ja se esitettiin ilman käyttäjän syötetietoja ja ohjelmalogiikkaa.

Tämä ei kuitenkaan riittänyt pitkälle. Staattisilla sivuilla ei pystytty toteuttamaan haluttuja toimintoja, kuten pankkitilin hallinta. Tarvittiin dynaamisia sivuja, joissa haetaan ja muutetaan tietoa käyttäjän syötteiden perusteella. Dynaamisen sivun luonti alkaa käyttäjän pyynnöstä, joka sisältää syötteen, kuten pankkitilin numeron. Palvelin vastaanottaa pyynnön ja hakee tilin saldon, tapahtumat ym. tiedot pankin tietokannasta. Tämän jälkeen palvelin generoi sivun, joka sisältää haetut tiedot ja lähettää sivun käyttäjän selaimelle.

Tämän kaltaisten dynaamisten toimintojen myötä on tärkeää, että palvelin tietää kuka pyytää sivua ja mitä tietoja hänelle saadaan näyttää. Käyttäjä ei myöskään ajattele,

että sivuston käyttö on yksittäisten pyyntöjen lähettämistä. Hän ajattelee sitä istuntona, joka kestää tietyn ajan. Tällöin palvelimen on muistettava mitä tietoja käyttäjä mahdollisesti syötti hetki sitten. Tähän kaikkeen tarvitaan ajonaikainen ympäristö, joka tarjoaa näihin tarpeisiin valmiit työkalut.

Microsoft julkaisi vuonna 1997 ASP (Active Server Pages) -ympäristön osana IIS (Internet Information Services) WWW-palvelinta. ASP mahdollistaa dynaamisten sivujen luomisen yhdistämällä HTML-kieltä ja skriptikoodia. Kun käyttäjä pyytää ASP-sivua, palvelin etsii sen ja käynnistää ASP-prosessin. Prosessi tulkitsee ASP-sivun skriptiosat, jotka sijaitsevat <% ja %> -merkkiyhdistelmien välissä. Lopputuloksena saadaan prosessin generoimat dynaamiset HTML-osuudet, jotka syntyvät sivun skriptiosista, sekä sivussa valmiina olevat staattiset HTML-osuudet.

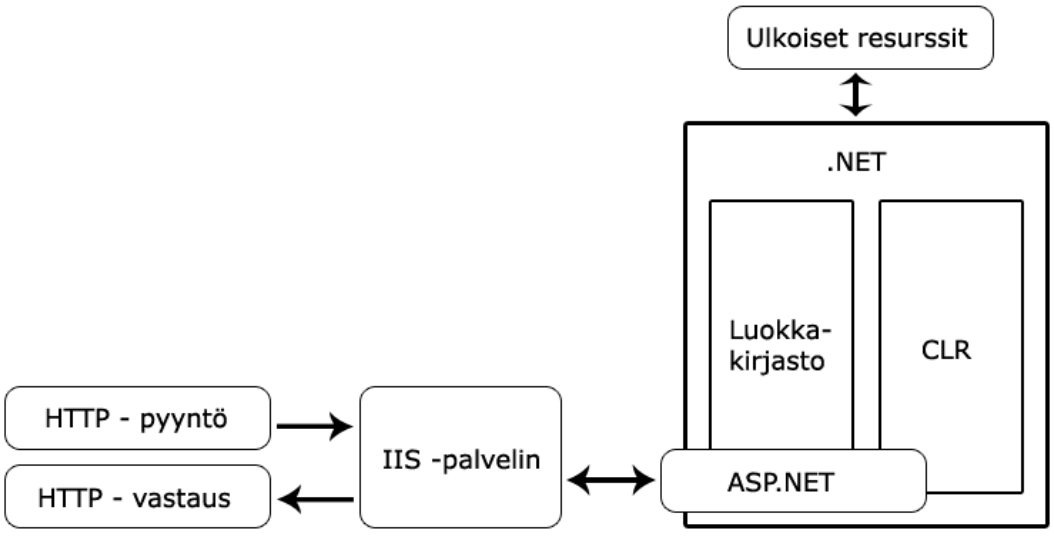
Kuva 2.2. Yksinkertainen ASP-sivu.

Kun Internet-käyttäjien vaatimukset kasvoivat, web-ohjelmoijat alkoivat tarvita lisää ominaisuuksia web-kehitysympäristöltään. ASP.NET, joka käyttää hyödyksi .NET Frameworkia, tarjosi kaivatut ominaisuudet. ASP.NET tarjoaa parempaa suorituskykyä, vakautta ja helpompaa ohjelmoitavuutta. ASP.NET erottaa HTML-kielen ja sovelluksen koodin toisistaan käyttämällä taustakoodi-tekniikkaa (Code Behind). ASP-sovelluksissa HTML ja koodi kirjoitettiin samaan tiedoston, kun taas ASP.NET-sovelluksissa koodi on omassa tiedostossaan, johon sivulla on viittaus. (Platt 2001, 93-98.)

2.2.2 Sovelluksen toiminta

Vanha ASP-tekniikka oli hidas, koska sivuilla olevat skriptikoodit tulkittiin ajon aikana. ASP.NET-ympäristössä sivut kääntyvät automaattisesti, joko asennuksen tai ensimmäisen pyynnön yhteydessä. Tämä tekee ASP.NET-ympäristöstä vanhaa ASP-ympäristöä nopeamman. Koska käytössä on .NET Frameworkin täsmäkäännös, palvelinta ei tarvitse pysäyttää, kun jokin komponentti tai sivu päivitetään. (Platt 2001, 100.)

ASP.NET-sovellukset voidaan jakaa kahteen kategoriaan: Web Forms - ja Web Services -sovellukset. Web Forms -sovellukset ovat käyttäjien kanssa vuorovaikutuksessa toimivia käyttöliittymäpohjaisia sovelluksia. Web Services -sovellukset ovat puhtaasti toiminnallisia palveluita, joita toiset sovellusympäristöt voivat käyttää. Tässä opinnäytetyössä ei käsitellä Web Services -sovelluksia, vaan keskitytään pelkästään Web Forms -sovelluksiin. ASP.NET-sovellukset ovat ennen kaikkea .NET-sovelluksia, joka mahdollistaa .NET-ympäristön ja sen resurssien täyden hyödyntämisen. WWW-palvelin tarvitaan vain HTTP-liikenteen muodostamiseen, mutta varsinainen toiminta tapahtuu .NET-ympäristön sisällä.



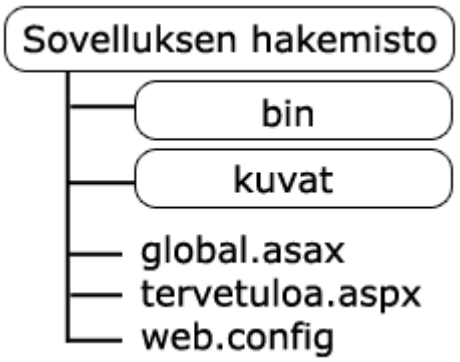
Kuva 2.3. ASP.NET-sovelluksen toiminta.

IIS-palvelin tunnistaa ASP.NET-sovellukseen kohdistuvan pyynnön tiedostomuodon perusteella. ASP.NET:n tiedostomuotoja ovat aspx, ascx, asmx, asax ja config. Asmx on Web Service -sovelluksen tiedostomuoto, joten sitä ei käsitellä enempää tässä

työssä. Config-tiedostoja ei palauteta, koska ne sisältävät tietoa järjestelmän kokoonpanosta. Lisäksi Global.asax-tiedoston suora käyttö on kielletty, koska se on sovelluksen alustustiedosto. (Inkinen 2003, 58-60.)

2.2.3 Sovelluksen rakenne

Sovellukselle tulee luoda järjestelmään oma hakemisto, jonka sisään luodaan alihakemisto bin. Bin-hakemistoon tallennetaan sovelluksen taustaluokat ja .NET-assemblyt (dll-tiedostot). WWW-palvelimelle luodaan virtuaalihakemisto, joka osoittaa järjestelmään luotuun hakemistoon. Sovellukselle luodaan lisäksi Web.config-konfigurointitiedosto sekä Global.asax-alustustiedosto. Sovellus koostuu useista tiedostoista, jotka on sijoitettu ko. hakemistorakenteeseen. Sovellukseen voi kuulua em. asetustiedostojen lisäksi lähdekooditiedostoja (.vb), Web Forms -tiedostoja (.aspx), kontrollitiedostoja (.ascx) sekä HTML-tiedostoja (.html). Alla olevassa rakenneesimerkissä sovelluksessa käytettäville kuville on luotu oma alihakemistonsa.



Kuva 2.4. Esimerkki ASP.NET-sovelluksen hakemistorakenteesta.

Yksittäinen ASP.NET-sivu, jota voidaan kutsua myös Web Forms - tai aspx-sivuksi, koostuu direktiiveistä, palvelimen koodilohkosta sekä HTML-osiosta (ks. Liite 1). Direktiiveissä voidaan määritellä mm. käytettävä ohjelmointikieli. Palvelimen koodilohko sisältää tapahtumakäsittelijät ym. ohjelmoidut toiminnallisuudet, jotka voidaan sijoittaa myös erilliseen taustakooditiedostoon, johon viitataan direktiivissä. (ks. 2.2.4 Erillinen koodi ja ulkoasu). HTML-osio voi sisältää merkkauksen lisäksi ASP.NET-lohkoja vanhan ASP-tekniikan tapaan <% ja %> -merkkiyhdistelmien välissä. HTML-elementtien lisäksi voidaan käyttää Web Forms -kontrolleja (ks. 2.2.8 Web Forms -kontrollit). (Inkinen 2003, 61-66.)

Lisäksi sovellus voi sisältää käyttäjäkontrolleja (User control), joita voidaan liittää sovelluksen Web Forms -sivuihin. Käyttäjäkontrollin tiedostopääte on ascx. Kontrollit ovat sivuun upotettavia erillisiä sivuja, jotka sisältävät tietyn toiminnallisuuden, jota voidaan hyödyntää monessa sivussa. Menetelmällä vältetään saman koodin kirjoittaminen moneen sivuun. Käyttäjäkontrollin avulla voidaan toteuttaa esim. sivuston valikko, jolloin saman valikkotoiminnallisuuden tarjoama kontrolli voidaan liittää moneen sivuun. (Esposito 2005,46.)

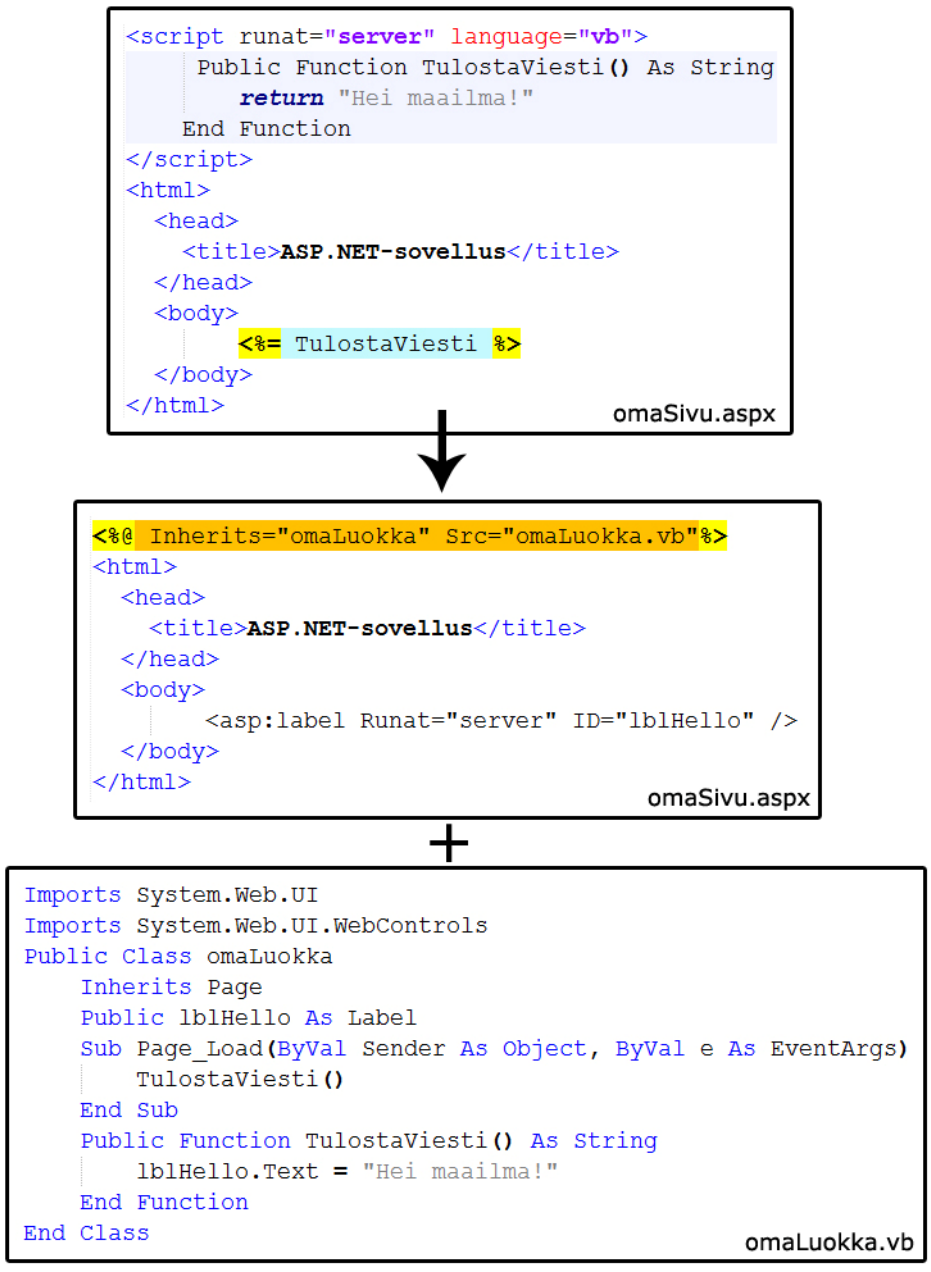
ASP.NET-sovelluksen asetustiedostot ovat XML-muodossa, mikä on nykyään varsin yleinen käytäntö erilaisten sovellusten asetustiedostoissa. ASP.NET sisältää asetustiedostoja neljällä eri tasolla: .NET-ympäristön asetukset, WWW-palvelun asetukset, sovelluksen asetukset sekä yksittäisten alihakemistojen asetukset. Kun sovelluksen yksittäiseen sivuun kohdistuu pyyntö, aloitetaan sen asetuksien määrittäminen. Ensimmäiseksi tarkistetaan .NET-ympäristön yleiset ASP.NET-asetukset, jotka sijaitsevat Machine.config-tiedostossa. Seuraavaksi tarkistetaan web-palvelutason asetukset, jotka löytyvät WWW-palvelun web.config-tiedostosta. Lopulliset asetukset määräytyvät tarkistamalla sovelluksen ja sen yksittäisten alihakemistojen asetustiedostot (web.config) aina pyydettyyn osoitteeseen saakka.

Machine.config on koko palvelinta koskeva asetustiedosto, joka määrittelee .NET-rajapinnan peruskokoonpanotiedot. Asetustiedostoja löytyy yksi kappale jokaista .NET-versiota kohden. Tiedostosta löytyy erillisiä lohkoja, joissa määritellään asetuksia palvelurajapinnoille, kuten ASP.NET. ASP.NET-ympäristöön liittyvät asetukset löytyvät system.web-lohkosta.

Web.config-asetustiedostot sisältävät yksityiskohtaisempia asetuksia ASP.NET-ympäristöön liittyen. Web.config-asetustiedostoja voi olla järjestelmässä useita. Jokaisella asetustiedostossa määriteltävällä asetuksella on oletusasetus, joka on voimassa, jos kyseistä asetusta ei määritellä erikseen. Asetuksia on paljon, mutta useimmiten asetusten määritteleminen kohdistuu vain tiettyihin asetusryhmiin. (Inkinen 2003, 13-17.)

2.2.4 Erillinen koodi ja ulkoasu

ASP.NET mahdollistaa sivun ulkoasun ja toiminnallisen koodin sijoittamisen erikseen ylläpidettäviin tiedostoihin (Code Behind -tekniikka). Tämä antaa sovelluskehittäjälle modulaarisemman tavan sovelluksen rakentamiseen aikaisemman HTML- ja ASP-koodin sekasotkun sijaan. Kooditiedosto on rakenteeltaan luokka, jolle periytetään Page-luokan ominaisuudet. Page-luokka kuuluu System.Web.UI-nimiavaruuteen, joten luokan määrittely aloitetaan Imports-lausekkeella, jolla kytkeydytään ko. nimiavaruuteen. Periytyminen määritellään Inherits-lausekkeella.



Kuva 2.5. Koodin sijoittaminen erilliseen tiedostoon.

Code Behind -luokka määritellään aspx-sivun Page-palvelindirektiiviin. Koodilohkojen käyttö aspx-sivulla voidaan välttää kokonaan käyttämällä lisäksi ASP.NET palvelinkontrolleja. Yllä olevassa esimerkissä käytetään label-kontrollia vastaavanlaisen lopputuloksen saavuttamiseksi. (Inkinen 2003,73-77.)

2.2.5 Lomakesovellukset

Sovelluksissa käytettävä vuorovaikutteisuuden mahdollistava ominaisuus on tyypillisesti HTML-lomake. Erilaisten lomakekenttien avulla sovellus voi toimia dynaamisesti käyttäjän syötteiden mukaan. Lomakkeet ovat tärkeässä roolissa myös ASP.NET-sovelluksissa. ASP.NET-lomakkeet eroavat hieman normaaleista HTML-lomakkeista. Lomakkeen juurielementti on vastaavalla tavalla form, mutta sille annetaan lisäksi runat-attribuutti. Runat-attribuutin ansiosta ASP.NET osaa käsitellä lomakkeen palvelimella. Lisäksi sivun tiedostopäätteen tulee olla aspx, jotta IIS-palvelin osaa ohjata sivun käsittelyn ASP.NET:lle. Lomakkeen elementeissä (kentissä) voidaan käyttää vastaavalla tavalla normaaleja HTML-elementtejä, jotka sisältävät runat-attribuutin. ASP.NET palvelinpäässä käsiteltävä lomake näyttää lähes samanlaiselta, kuin tavallinen HTML-lomake.

Toinen vaihtoehto on käyttää ASP-nimiavaruuteen sijoitettuja palvelinkontrolleja, joita käytetään esim. tietokannan tiedoista luotaviin tauluihin tai HTML-lomakeelementtien korvaajina laajempien ominaisuuksien kera. Tällaisia elementtejä ovat mm. <asp:textfield> ja <asp:button>.

ASP.NET sisältää postback-metodin, jonka avulla voidaan tarkistaa tapahtuiko edellisessä HTTP-pyynnössä lomakkeen lähetys. Lisäksi elementeissä voidaan käyttää automaattista takaisinpostitusta, joka tarkoittaa lomakkeen (sivun) automaattista uudelleenlataamista esim. pudotusvalikon valintaa vaihdettaessa. Ominaisuus on käytännöllinen lomakkeissa, joiden näkymät vaihtuvat käyttäjän valintojen mukaisesti. Automaattinen uudelleenlataus määritellään elementtiin attribuutilla AutoPostBack=”true”. (Inkinen 2003,66-72.)

```
<%@ Page Language="VB" %>
<script runat="server">
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
        If Page.IsPostBack Then
            If etunimi.Text <> Nothing And sukunimi.Value <> Nothing Then
                lblName.Text = "Nimesi on " & etunimi.Text & " " & sukunimi.Value
            Else
                lblName.Text = "Täytä kaikki kentät!"
            End If
        Else
            lblName.Text = "Syötä nimesi!"
        End If
    End Sub
</script>
<html>
<head runat="server">
    <title>Lomakesivu</title>
</head>
<body>
    <form id="lomake" runat="server">
        Etunimi: <asp:TextBox runat="server" ID="etunimi"/><br />
        Sukunimi: <input runat="server" type="text" id="sukunimi" /><br />
        <input type="submit" value="Lähetä" /><br /><br />
        <asp:Label runat="server" ID="lblName" />
    </form>
</body>
</html>
```

Kuva 2.6. Lomakeohjelma.

Yllä olevassa esimerkissä luodaan lomakeohjelma, joka kysyy käyttäjän etu- ja sukunimen. Koodilohko sisältää tapahtumankäsittelijän, joka suoritetaan sivun jokaisen latauksen yhteydessä. Ensimmäisessä If-lauseessa tarkistetaan tapahtuiko lomakkeen lähetys. Kun käyttäjä saapuu sivulle, sovellus antaa käyttäjälle ilmoituksen ”Syötä nimesi!”. Seuraava If-lause tarkistaa täyttikö käyttäjä molemmat kentät lähettäessään lomaketta. Jos ei täyttänyt, käyttäjää huomautetaan asiasta. Jos kentät täytettiin oikein, sovellus tulostaa tekstin ”Nimesi on Etunimi Sukunimi”. Esimerkissä on käytetty sekä HTML-, että ASP.NET-palvelinkontrolleja (ks. 2.2.8 Web Forms -kontrollit).

2.2.6 Sisäiset oliot

ASP.NET sisältää sisäisiä olioita, jotka ovat hyvin oleellinen osa web-sovellusten toiminnallisuuden rakentamisessa. Sisäisiä olioita ovat mm. Application, Request, Response ja Session.

Application-olio sisältää sovellukseen liittyvää tietoa, joka on yhteistä kaikille käyttäjille. Oliota käytetään sovellusmuuttujien ja -objektien tallentamiseen. Luokan esiintymät muodostuvat avain-arvo-pareista. Uusi avain-arvo-pari voidaan määritellä yksinkertaisesti alla olevan esimerkin mukaan. Esimerkissä kasvatetaan kävijälaskurin arvoa yhdellä. Esimerkissä käytetään lisäksi Lock- ja UnLock-metodia estämään muuttujan samanaikainen muokkaus sovelluksen toisista osista. Laskuri voidaan sijoittaa Global.asax-tiedoston istunnon aloitustapahtumakäsittelijään, jolloin laskurin arvoa lisätään yhdellä jokaisen uuden istunnon alkaessa (ks. Liite 2).

```
Application.Lock()
Application("intKavijaLaskuri") += 1
Application.UnLock()
```

Kuva 2.7. Sovellusmuuttujan käsittely.

Session-olio on vastaavanlainen, kuin Application-olio, mutta siihen tallennettavat muuttujat ovat istuntokohtaisia. Session-olion SessionID-ominaisuus sisältää sovellusistunnon tunnisteen, jonka perusteella sovellus tunnistaa käyttäjän. Sovellus antaa istunnon alussa käyttäjälle yksikäsitteisen tunnisteen, joka säilytetään istunnon loppuun saakka. Istunto päättyy, kun TimeOut-ominaisuuteen määritelty aika tulee täyteen viimeisestä HTTP-pyynnöstä laskien. Istunnon päättyessä SessionID:n vapauttamisen lisäksi vapautetaan kaikki istunnon muuttujat. Istunnon lopettamiseen voidaan käyttää tarvittaessa myös Abandon-metodia.

Request-oliota käytetään HTTP-pyyntöjen käsittelyyn. Pyynnön sisältö voidaan lukea olion ominaisuuksien avulla. ASP.NET-lomakkeella lähetettävän GET-pyynnön data käsitellään QueryString-kokoelman kautta ja POST vastaavasti Form-kokoelman kautta. Lomakkeen tietojen lisäksi olio sisältää tietoa HTTP-liikenteeseen liittyen esim. käytettävän asiakasselaimen tiedot. Request-olion avulla voidaan lisäksi mm. tallentaa lomakkeelta lähetettävä liitetiedosto sekä käsitellä evästeitä.

Alla olevassa esimerkissä on lomake, joka sisältää suku- ja etunimikentän. Lomakkeen lähetykseen käytetään GET-metodia, jolloin sen lähettämään tietoon päästään koodissa käsiksi Request-olion QueryString-kokoelmalla. Jos lomakkeen lähetykseen

käytettäisiin POST-metodia, koodissa käytettäisiin vastaavalla tavalla Form-kokoelmaa (esim. Request.Form(”etunimi”)).

```
<form id="lomake" runat="server" method="get">
    Etunimi: <input type="text" name="etunimi" /><br />
    Sukunimi: <input type="text" name="sukunimi" /><br />
    <input type="submit" value="Lähetä" /><br /><br />
    <asp:Label runat="server" ID="lblName" />
</form>
```

```
If Page.IsPostBack Then
    Dim strEtunimi As String = Request.QueryString("etunimi")
    Dim strSukunimi As String = Request.QueryString("sukunimi")
    If strEtunimi <> Nothing And strSukunimi <> Nothing Then
        lblName.Text = "Nimesi on " & strEtunimi
        lblName.Text &= " " & strSukunimi
    Else
        lblName.Text = "Täytä kaikki kentät!"
    End If
Else
    lblName.Text = "Syötä nimesi!"
End If
```

Kuva 2.8. Yksinkertainen lomakesovellus, jossa käytetään GET-metodia ja QueryString-kokoelmaa.

Response-oliota käytetään asiakkaalle lähetettävän HTTP-vastauksen käsittelyyn. Olion ominaisuuksiin kuuluvat mm. sivutulosteen puskuroinnin hallinta, tulostaminen suoraan vastaukseen sekä evästeiden hallinta. Jos vastauksessa käytetään puskurointia, näytetään vastaus asiakkaalle vasta, kun tulossivun sisältö on kokonaan prosessoitu. Alla olevassa esimerkissä asetetaan vastauksen MIME-tyyppi sekä käytettävä merkistö. Lisäksi vastaukseen kirjoitetaan ”Hei maailma!”. (Inkinen 2003, 82-102.)

```
Response.ContentType = "text/html"
Response.Charset = "utf-8"
Response.Write("Hei maailma!")
```

Kuva 2.9. Response-olion käsittely.

2.2.7 Istunnot ja tilanhallinta

Sovelluksen tilan ja istuntojen hallinta on tärkeä osa sovelluksen rakentamista. Edellisessä kohdassa esitelty sovellustaso sisältää sovelluksen kaikille käyttäjille yhteistä

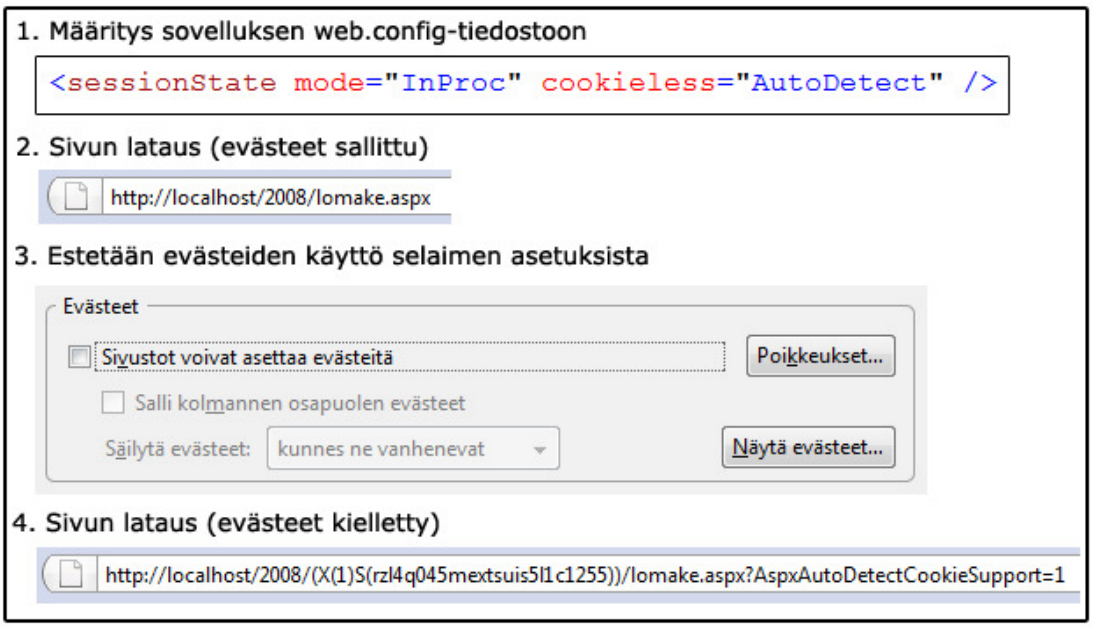
tietoa. Sovellustasoon voidaan siis tallentaa tietoa, jota sovellus tarvitsee suorituksessaan kaikkien käyttäjien kanssa. Tieto voi olla esim. tietokantayhteyden muodostamiseen tarvittava merkkijono tai sivuston kävijälaskuri. Tiedon tallennus sovellustasolle tekee sen käytöstä keskitettyä ja helposti muokattavaa. Jos tietokantayhteyteen tulee muutos, riittää että muutos päivitetään yhteen sovellustason muuttujaan.

Jokaisella sovelluksen käyttäjällä on oma henkilökohtainen istunto. Yhdessä sovelluksessa voi olla samanaikaisesti käynnissä useita istuntoja, jotka erotetaan toisistaan satunnaisesti luotavalla tunnisteella. Yksittäisen istunnon tiedot eivät ole käytettävissä koko sovelluksen laajuisesti, niin kuin em. sovellustasossa. Istuntoon voidaan tallentaa istunnossa tarvittavaa käyttäjäkohtaista tietoa. Tiedon varastona käytetään edellisessä kohdassa mainittua Session-oliota.

Sovellukseen kuuluva Global.asax-asetustiedosto on sovelluksen tilanhallinnan paras apuväline (ks. Liite 2). Asetustiedosto sisältää tapahtumankäsittelijöitä eri sovellustilojen ja istuntojen hallintaan. Sovellustason tapahtumia ovat sovelluksen käynnistys ja lopetus. Käynnistystapahtuma suoritetaan sovelluksen ensimmäisellä käyttökerralla. Lopetustapahtuma suoritetaan palvelimen tai palvelun sammuttamisen yhteydessä. Istuntoihin liittyviä tapahtumia ovat vastaavasti istunnon aloitus ja lopetus. Uusi istunto aloitetaan, kun käyttäjä lähettää ensimmäisen pyynnön sovelluksen sivuun, jolloin suoritetaan istunnon aloitustapahtumakäsittelijä. Istunto päättyy, joko keskeytysmetodin kutsumiseen tai aikakatkaisuun, jonka jälkeen suoritetaan istunnon lopetustapahtumakäsittelijä.

Sovelluksen asetustiedosto (Web.config) sisältää käyttäjäistuntoihin liittyviä asetuksia. Asetuksiin voidaan määritellä mm. evästeiden käyttö, istunnon umpeutumisaika sekä istuntojen käsittelytapa. Jos evästeitä ei käytetä, käyttäjän istuntotunniste kuljetetaan osoitteen mukana. Tämä mahdollistaa istuntojen ylläpitämisen myös selaimissa, jotka eivät tue evästeiden käyttöä tai ne on asetettu pois päältä. Istuntojen käsittelytavalla määritellään käytetäänkö istuntoja ja jos käytetään, niin millä periaatteella. Vaihtoehtoisia istunnon käsittelytapoja ovat tallennus WWW-palvelun yhteyteen, tallennus tietokantaan sekä tallennus ulkopuoliseen palvelinprosessiin. Alla olevassa esimerkissä istuntojen käsittelytavaksi on määritelty tallennus WWW-palvelun yhteyteen (”InProc”). Evästeiden käyttö on asetettu automaattiseksi. Evästeitä käyte-

tään jos asiakasselain sallii sen. Muussa tapauksessa istunnon tunniste kuljetetaan automaattisesti osoitteen mukana.



Kuva 2.10. Istunnon käsittelytapa ja evästeiden käyttö.

Tietokantatallennuksen etuna on istuntotietojen säilyvyys esim. palvelimen alasajon yhteydessä. Istuntojen tallennukseen käytettävä tietokanta voidaan luoda ASP.NET:n mukana tulevalla SQL-proseduurilla. Ulkopuolisen tilapalvelimen etuna on istuntojen riippumattomuus ASP.NET- ja IIS-prosesseista. Tällöin esim. WWW-palvelun uudelleenkäynnistäminen ei vaikuta istuntojen tilaan. Tilapalvelimen käyttö edellyttää palvelimen osoitteen ja portin määrittelyn sovelluksen asetustiedostoon. Tilapalvelin sisältyy .NET-Frameworkiin ja se voi sijaita myös kokonaan erillisellä palvelimella.

ASP.NET-sovelluksen yksittäisen sivun tilaa voidaan hallita ViewState-ominaisuuden avulla. Ominaisuus mahdollistaa sivuun sijoitettujen komponenttien automaattisen arvojen säilytyksen sivun PostBack-tapahtumien välillä. Arvot tallennetaan tilasäiliöön ja ne ilmenevät lopputuloksessa, eli HTML-merkkauksessa, ASP.NET:n generoimina hidden-tyyppisinä lomakekenttinä. ViewState-ominaisuus on sivukohtainen, joten toiselle sivulle siirryttäessä tiedot häviävät. Ominaisuuden saa pois päältä sivu- ja komponenttikohtaisesti, jolloin voidaan välttää turha resurssien tuhlaus esim. tilanteessa, jossa sivun kenttiin haetaan tiedot tietokannasta jokaisen

sivunlatauksen yhteydessä. ViewState voidaan tarvittaessa kytkeä pois koko sovelluksen laajuisesti sovelluksen asetustiedostosta.

Tilasäiliötä, jota voidaan ajatella myös sivukohtaisena väliaikaistiedon varastona, voidaan käyttää myös sovelluksen tietojen tallennukseen PostBack-tapahtumien välillä (ei pelkästään lomakekenttien). Tilasäiliön tieto on tallennettu avain-arvo-pareina. Alla olevassa esimerkissä on demonstroitu tilasäiliön käyttö. Kun käyttäjä saapuu sivulle, asetetaan tilasäiliöön uusi avain-arvo-pari. Tässä vaiheessa tulostus ei palauta mitään käyttäjälle, koska tulostus suoritetaan ennen avain-arvo-parin asettamista. Sivu sisältää lisäksi painikkeen, jonka painaminen aiheuttaa PostBack-tapahtuman. Painikkeen painamisen jälkeen sivu ladataan uudelleen ja asetettu arvo tulostetaan käyttäjälle. (Inkinen 2003, 102-113.)

```
Response.Write(ViewState("nimi"))
If Not Page.IsPostBack Then
    ViewState("nimi") = "arvo"
End If
```

Kuva 2.11. Tilasäiliön käyttö.

2.2.8 Web Forms -kontrollit

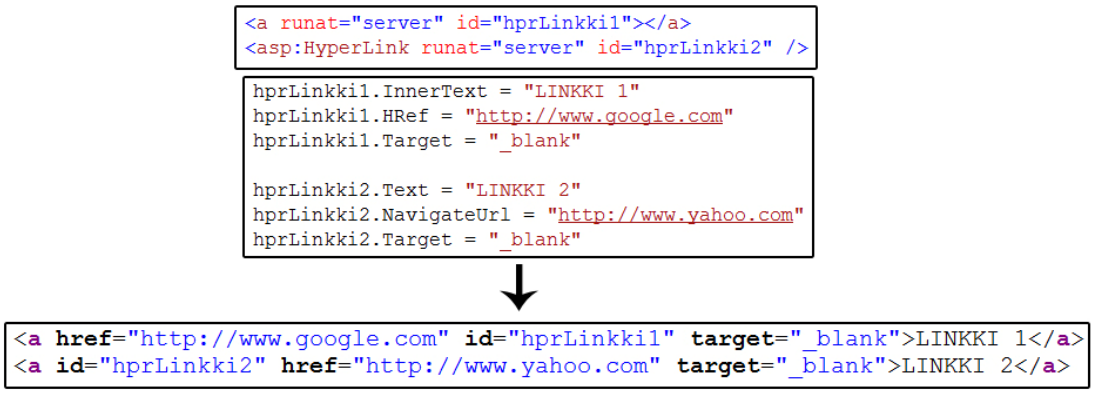
Web Forms -termillä voidaan tarkoittaa yleisesti ASP.NET-selainsovelluksia tai niissä käytettävää komponenttikokoelmaa. Suurin osa ASP.NET-sivun toiminnallisuudesta saadaan aikaan Web Forms -kontrolleilla, jotka ovat laaja kokoelma palvelinkomponetteja. Palvelinkomponentit ovat ASP.NET-luokkia, joita voidaan käsitellä ohjelmallisesti. ASP.NET generoi komponentit näyttövaiheessa HTML-merkkauskieleksi. Tämän ansiosta sovelluskehittäjä voi keskittyä täysillä toiminnallisuuksien toteuttamiseen ilman, että tarvitsee huolehtia selainriippuvaisuuksista.

Palvelinkomponentteja on kahdenlaisia: HTML- sekä ASP.NET-kontrollit. HTML-kontrollit ovat tavallisia HTML-elementtejä, jotka on varustettu palvelintoiminnallisuudella. Tämä mahdollistaa normaalin HTML-elementin käsittelyn sovelluksen koodista käsin. Minkä tahansa tavallisen HTML-elementin saa palvelinpohjaiseksi lisäämällä elementtiin runat- ja id-attribuutit. Runat-attribuutin arvo ”server” kertoo, että elementti suoritetaan palvelinpohjaisesti ja id-attribuutti vaaditaan, jotta element-

tiä voidaan käsitellä koodissa. HTML-kontrollit sijaitsevat System.Web.UI.HtmlControls-nimiavaruudessa.

ASP.NET-kontrollit ovat perusajatukseltaan vastaavanlaisia kuin HTML-kontrollit, mutta ne sisältävät enemmän ominaisuuksia ja ne on luotu helpottamaan sovelluskehityksessä usein vastaantulevia tehtäviä. Kontrolleissa käytetään omaa etuliitettä (”asp”) esim. painike-kontrolli määritellään sivuun seuraavasti: <asp:button .../>. ASP.NET-kontrollit sijaitsevat System.Web.UI.WebControls-nimiavaruudessa.

Seuraavaksi käydään läpi muutamia kontrolleja. Palvelinpohjainen linkin käsittely voidaan suorittaa molemmilla kontrollityypeillä. HTML-linkin palvelinvastineita ovat HTML-palvelinkontrolli ”HtmlAnchor” ja vastaavasti ASP.NET-palvelionkontrolli ”Hyperlink”. Alla olevassa esimerkissä käsitellään linkkiä molemmilla kontrollityypeillä. ASP.NET:n generoima HTML on vastaavanlainen molempia tapoja käytettäessä.



Kuva 2.12. Linkin luonti Web Forms -kontrolleilla.

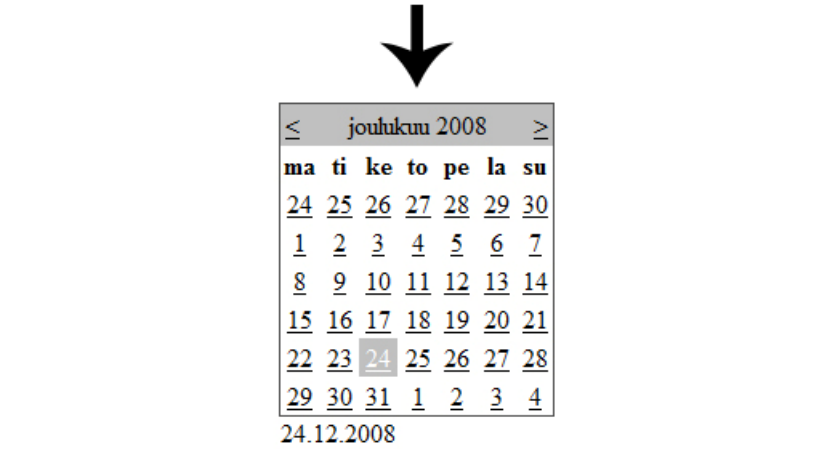
ASP.NET-sivun takaisinpostitukseen vaaditaan form-elementti. Jos sivu sisältää lomake-elementtejä, niin form-elementti on pakollinen osa lomakkeen toimivuuden mahdollistamiseksi. Elementti otetaan käyttöön kirjoittamalla normaalin HTML-form-elementin sisään attribuutti runat=”server”.

Niin kuin edellä mainittiin, ASP.NET-palvelinkontrollit ovat ominaisuuksiltaan runsaampia ja tarjoavat ratkaisuja usein vastaantuleviin tehtäviin. Yksi tällainen komponentti on kalenteri (Calendar). Kalenterikomponentin avulla voidaan toteuttaa helposti kalenteriliittymiä. Komponentti sisältää paljon muokkausmahdollisuuksia, jon-

ka avulla kalenterista saadaan halutun näköinen ja muuhun ulkoasuun soveltuva. Peruskäytössä kalenterin luonti on varsin yksinkertaista. Alla olevassa esimerkissä luodaan kalenteri, jossa valittu päivämäärä tulostetaan kalenterin alapuolelle. Kalenterin osalta esimerkkisovelluksen tekeminen vaatii vain tunnisteen määrittelyn sekä tapahtumakäsittelijän luonnin.

```
<asp:Calendar ID="kalenteri" runat="server" />
<asp:Label ID="lblPaiva" runat="server" />
```

```
Protected Sub kalenteri_SelectionChanged(ByVal
    lblPaiva.Text = kalenteri.SelectedDate
End Sub
```



Kuva 2.13. Kalenterikontrollin käyttö.

Toinen esimerkki usein tarvittavan toiminnon toteuttavasta komponentista on DataGrid-komponentti. DataGrid mahdollistaa tiedon taulukkomuotoisen esittämisen ASP.NET-sivulla. Tiedon näyttämisen lisäksi, sen ominaisuuksin kuuluvat tiedon lajittelu ja sivutus sekä tiedon valinta ja muokkaaminen. Tietolähteenä voidaan käyttää tietokannan lisäksi myös XML-tiedostoa. Esimerkissä tulostetaan XML-tiedoston sisältö taulukkomuotoisena (ks. Liite 3).

Placeholder on erittäin hyödyllinen kontrolli, kun samalle sivulle halutaan luoda eri näkymäryhmiä, joita halutaan näyttää eri aikaan. Esim. lomake, jossa tehtävien valintojen perusteella halutaan näyttää tiettyjä lisävalintoja, voidaan toteuttaa ko. kontrollilla. Kontrollin sisään sijoitettujen komponenttien näkyvyyttä säädetään Visible-ominaisuuden avulla. (Inkinen 2003, 132-183.)

2.2.9 Tietokannan käyttö

Tietokantojen käyttö on hyvin merkittävä osa ASP.NET-sovelluksia, sillä suurin osa dynaamisista web-sovelluksista on tietokantapohjaisia. Yksinkertaisimmillaan tietokanta voi olla tekstitiedosto, mutta vaativimmissa tilanteissa tarvitaan edistyneempää tietokantaa ja sen tarjoamia ominaisuuksia. Yhteistä tietokantapohjaisissa sovelluksissa on tietokantaliittymän tarve (liittymärajapinta), joka mahdollistaa tietokannan käsittelyn. .NET-ympäristön tietokantaliittymien rakennusmallina toimii ADO.NET, joka on kokoelma tietokannan käsittelyyn liittyviä luokkia. Luokat on sijoitettu System.Data-nimiavaruuteen.

Luokkien toiminnot voidaan jakaa viiteen pääosaan, joista jokainen sisältää kaksi erillistä luokkaa. Syy kahteen erilliseen luokkaan on erikseen Microsoftin SQL Server -tietokannalle optimoidut luokat, joiden avulla SQL Serverin käyttö on nopeampaa. Toinen yleisempikäyttöinen luokka on OLEDB, jota voidaan käyttää myös muihin tietokantoihin. Perusluokkia ovat Connection, Command, DataSet, DataAdapter sekä DataReader.

Tietokantayhteydet hoitaa Connection-luokka. Yhteyden avaamiseen käytetään luokan Open-metodia ja sulkemiseen vastaavasti Close-metodia. Muut luokan metodit tarjoavat tietoa mm. yhteyden tilasta. Command-luokan tehtävänä on suorittaa tietokantakomentoja, kuten tiedon hakua, päivityksiä sekä lisäyksiä.

DataReader-luokkaa käytetään nopeiden tietokantahakujen suoritukseen. DataReader-oliolla käsiteltävä tietokantahaun tulosjoukko luetaan yksisuuntaisesti alusta loppuun. Tulosjoukon selaaminen tapahtuu käyttämällä Read-metodia, jonka kutsuminen siirtää lukutapahtuman seuraavaan tietueeseen eli riviin. Tietueen kenttien arvoja voidaan tarkastella Item-kokoelman avulla, josta kentän sisältämän tiedon voi hakea kentän nimen perusteella.

Alla olevassa esimerkissä haetaan yrityksen tietokannasta kaikkien työntekijöiden sukunimet. Ensimmäiseksi määritellään yhteyden muodostamiseen tarvittava lauseke, joka sisältää tietokantapalvelimen, käyttäjätunnuksen, salasanan sekä käytettävän

tietokannan nimen. Käyttäjätunnus on etukäteen määritelty tietokantapalvelimelle (Microsoft SQL Server 2005).

```
Dim strConn As String
strConn = "Data Source=.\SQLExpress;User ID=WebsiteUser;"
strConn &= "Password=salasana;Initial Catalog=yritys_demo"
Dim sqlConn As SqlConnection = New SqlConnection(strConn)
Dim sqlComm As SqlCommand = New SqlCommand
sqlComm.Connection = sqlConn
sqlConn.Open()
sqlComm.CommandText = "SELECT Sukunimi FROM Tyontekija"
Dim rdrTyontekijat As SqlDataReader = sqlComm.ExecuteReader
If rdrTyontekijat.HasRows Then
    Do While rdrTyontekijat.Read()
        Response.Write(rdrTyontekijat("Sukunimi") & "<br />")
    Loop
End If
rdrTyontekijat.Close()
sqlConn.Close()
```

Kuva 2.14. DataReader-luokan käyttö tietokantahaussa.

DataSet on tietokannasta saatavan datan käsittelyyn tarkoitettu työkalu (luokka), jolla käsitellään tietokannasta haettua tietoa. Luokka on tietolähdetyypistä riippumaton, joten siihen voidaan hakea tietoa myös esim. XML-tiedostosta. DataSet-olion käsittely ei vaikuta suoraan alkuperäiseen tietokantaan, mutta siihen tehdyt muutokset voidaan päivittää varsinaiseen tietokantaan DataAdapter-olion avulla. DataSet-olion sisältämä tieto on DataTable-rakenteissa, joka vastaa relaatiotaulun rivi-sarake-rakennetta. Yksittäinen DataSet-olio voi sisältää useita DataTable-rakenteita, joten niiden ylläpitämiseen käytetään Tables-kokoelmaa, joka sisältää rakenteiden nimet.

DataAdapter-luokan tehtävä on toimia tietolähteen sekä DataSet-olion välisenä linkkinä. Luokan ilmentymälle annetaan muodostuksen yhteydessä suoritettava tietokantakomento sekä tietokantayhteys (Connection-olio). Hakutuloksen palauttamat tiedot siirretään DataSet-olioon DataAdapter-luokan Fill-metodilla, jonka parametreiksi annetaan DataSet-olion nimi sekä tallennettavan tulosjoukon nimi. Nimen avulla tulosjoukkoa voidaan käsitellä DataSet-oliossa. Alla olevassa esimerkissä haetaan yrityksen työntekijöiden tiedot, jotka palautetaan asiakkaan selaimelle XML-muodossa. GetXml-metodi generoi dataset-olion sisällön XML-muotoon. Jos käytettäisiin kommentoidun osuuden koodia XML-tulostuksen sijaan, saataisiin vastaava lopputulos, kuin edellisessä esimerkissä.

```
Dim strSQLSelect As String = "SELECT * FROM Tyontekija"
Dim sqlConn As New SqlConnection(strConn)
Dim sqlAdapt As New SqlDataAdapter(strSQLSelect, sqlConn)
Dim dsYritys As New DataSet
sqlAdapt.Fill(dsYritys, "Tyontekija")
'For Each rivi In dsYritys.Tables("Tyontekija").Rows
'    Response.Write(rivi("sukunimi") & "<br />")
'Next
Response.ContentType = "text/xml"
Response.Write(dsYritys.GetXml)
Response.End()
```

Kuva 2.15. DataAdapter- ja DataSet-luokkien käyttö tietokantahaussa.

Rivien lisäys, muokkaus sekä poisto ovat melko yksinkertaisia toimenpiteitä. Ensimmäiseksi avataan tietokantayhteys vastaavalla tavalla, kuin aiemmissa esimerkeissä. Seuraavaksi määritellään suoritettava SQL-lause, joka suoritetaan käyttämällä ExecuteNonQuery-metodia. Alla olevassa esimerkissä annetaan kaikille työntekijöille viiden prosentin palkankorotus. Samalla periaatteella voidaan suorittaa myös tiedon lisäys- ja poisto-toimenpiteitä. SQL-kieltä käydään läpi seuraavassa luvussa. (Inkinen 2003, 203-226.)

```
sqlConn.Open()
sqlComm.CommandText = "UPDATE Tyontekija SET Palkka = Palkka * 1.05"
sqlComm.ExecuteNonQuery()
sqlConn.Close()
```

Kuva 2.16. Tietokantataulun päivitys.

3 RELAATIOTIETOKANNAT

3.1 Perusteet

Relaatiotietokannoissa tieto esitetään tauluissa, joita kutsutaan myös relaatioiksi. Taulussa oleva tieto sijaitsee riveillä, joita kutsutaan tietueiksi. Taulun jokainen rivi sisältää saman määrän kenttiä (sarakkeita). Relaatiotietokannasta voidaan hakea tietoa taulun nimen, kenttien ja niiden arvojen perusteella. Alla olevassa kuvassa on yksinkertainen tietokantataulu, joka sisältää yrityksen työntekijöiden tietoja. Taulus-

sa on 3 kenttää: työntekijän id, etunimi ja sukunimi. Yrityksessä on yhteensä 5 työntekijää, joten taulussa on vastaava määrä rivejä.

Tyontekija		
TyontekijaID	Etunimi	Sukunimi
1001	Matti	Meikäläinen
1002	Olli	Ostaja
1003	Tarmo	Tuotanto
1004	Ville	Virtanen
1005	Maija	Meikäläinen

Kuva 3.1. Relaatiotietokannan taulu.

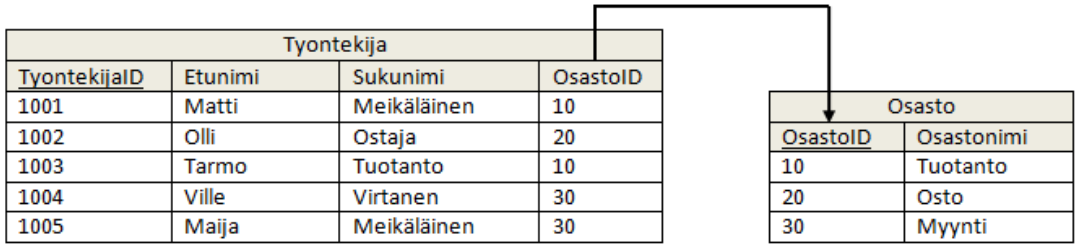
Tietokantataulun kenttää luotaessa tulee määritellä kentän nimi, tietotyyppi ja tiedon pakollisuus. Lisäksi voidaan määritellä mm. maksimipituus, oletusarvo ja oikeellisuustarkistus. Kentän tietotyyppi voi olla esim. numeerinen, teksti, päivämäärä ja/tai kellonaika, looginen (tosi/epätosi) tai bittijono.

Jos sarake määritellään pakolliseksi, uutta riviä syötettäessä sille tulee antaa jokin arvo. Jos kenttää ei ole määritelty pakolliseksi ja sille ei anneta arvoa, kentän arvoksi asetetaan NULL. Kenttään määritetyn oletusarvon avulla voidaan tarvittaessa välttää NULL-arvot. NULL on merkintä puuttuvalle arvolle, joka ei tarkoita nollaa, vaan tuntematonta arvoa.

Kenttään määriteltävän oikeellisuustarkistuksen avulla voidaan asettaa numeeriselle tietotyypille sallittu arvoväli. Jos esim. halutaan määritellä, että kentän arvon tulee sisältää neljä numeroa (1000 - 9999), kuten edellä olleen työntekijätaulun TyontekijaID-kenttä. Tällöin kentän ehdoksi voidaan määritellä, että arvon tulee olla suurempi kuin 999 ja pienempi kuin 10000.

Jokaiseen tauluun tulee määritellä perusavain, joka voi muodostua taulun yhdestä tai useammasta kentästä. Perusavaimen tehtävä on yksilöidä taulun rivit. Jokaisella rivillä tulee olla arvo tai arvojen yhdistelmä, joka erottaa sen taulun kaikista muista riveistä. Perusavaimen arvo ei saa olla NULL. Useammasta kentästä muodostettua perusavainta kutsutaan yhdistetyksi avaimeksi. Tietokannan taulut pidetään perusavaimen mukaisessa järjestyksessä, joka tekee tiedon hakemisesta tehokkaampaa.

Tauluihin voidaan määritellä myös viiteavainkenttiä. Viiteavain viittaa jonkin toisen taulun perusavaimeen. Viiteavainkentän tulee sisältää jokin arvo, joka löytyy viittauksen kohteena olevan taulun pääavainkentästä. Jos viiteavainkenttään yritetään syöttää arvo, jota ei löydy kohdetaulusta, syöttöä ei hyväksytä. Tällaista taulujen välistä yhteyttä kutsutaan viite-eheydeksi.



Tyontekija			
TyontekijaID	Etunimi	Sukunimi	OsastoID
1001	Matti	Meikäläinen	10
1002	Olli	Ostaja	20
1003	Tarmo	Tuotanto	10
1004	Ville	Virtanen	30
1005	Maija	Meikäläinen	30

Osasto	
OsastoID	Osastonimi
10	Tuotanto
20	Osto
30	Myynti

Kuva 3.2. Taulujen välille on muodostettu viite-eheys.

Yllä olevassa kuvassa on kaksi taulua, joiden välille on luotu viite-eheys. Työntekijätaulu sisältää viiteavainkentän (OsastoID), joka viittaa osastotaulun vastaavannimiseen pääavainkenttään. Työntekijätaulun OsastoID-kenttään voidaan syöttää vain sellaisia arvoja, jotka löytyvät osastotaulun pääavainkentästä. Osastotaulun OsastoID-kentän arvoja voidaan yrittää muuttaa tai poistaa, mutta jos se aiheuttaa viite-eheyden rikkomisen, suoritetaan etukäteen valittu toimenpide. Viite-eheyttä rikotaan, jos muokataan sellaista kentän arvoa, johon löytyy viittaus toisesta taulusta. Vaihtoehtoisia toimenpiteitä ovat viiteavaimen nollaus, oletusarvon asettaminen, johdannaismuutos sekä rajoitettu muutos.

Eri toimenpiteillä tapahtuvat muutokset, kun edellä olleen esimerkin osastotaulun myyntiosaston id 30 vaihdetaan arvoksi 40:

- Viiteavaimen nollaus: kaikkien myyntiosastoon kuuluvien työntekijöiden osaston arvoksi asetetaan NULL.
- Oletusarvon asettaminen: kaikkien myyntiosastoon kuuluvien työntekijöiden osaston arvoksi asetetaan työntekijätaulun osastokenttään määritelty oletusarvo.
- Johdannaismuutos: kaikkien myyntiosastoon kuuluvien työntekijöiden osaston arvoksi asetetaan 40.

- Rajoitettu muutos: toimenpidettä ei voida suorittaa, koska vain sellaisia osastotaulun perusavaimia voidaan muuttaa tai poistaa, joihin ei ole viittauksia työntekijätaulusta.

(Lahtonen 2002, 4-13.)

3.2 SQL-kieli

SQL (Structured Query Language) on relaatiotietokantojen käsittelyyn käytettävä kieli. SQL-kielellä voidaan määritellä tietokannan rakenne ja käsitellä sen sisältöä. Kyselyt ovat SQL-kielen yleisin käyttökohde, sillä tietokannan käsittelystä suurin osa on erilaisten raporttien ja koosteiden luomista.

3.2.1 Taulujen määrittely

SQL ei ole nimestään huolimatta pelkkä kyselykieli, vaan sitä voidaan käyttää myös taulujen määrittelyyn ja luontiin. Tietokannan tauluja luotaessa tulee määritellä taulun ja sen kenttien nimet sekä tietotyypit. Jotta osaisi valita kullekin kentälle sopivan tietotyypin, tulee niiden merkitys ymmärtää. Seuraavaksi käydään läpi SQL92-standardin mukaiset tietotyypit.

Merkkijonojen tallentamiseen on olemassa kaksi eri tietotyyppiä: CHAR (CHARACTER) ja VARCHAR (CHARACTER VARYING). CHAR on kiinteämittainen merkkijono, joka on aina määrätyn pituuden mittainen. Jos CHAR-kenttään syötetään vähemmän merkkejä kuin siihen määritelty pituus on, tyhjäksi jäävät merkit täytetään välilyönneillä. CHAR-tietotyyppiä kannattaa käyttää silloin kun tiedetään, että merkkijono on aina tietyn mittainen. Esim. henkilötunnus sisältää aina 11 merkkiä. VARCHAR on puolestaan vaihtuvamittainen merkkijono, joka on maksimissaan annetun pituuden mittainen. VARCHAR sopii sellaisten merkkijonokenttien tietotyypiksi, joiden pituus vaihtelee esim. henkilön etu- ja sukunimi.

Numeerisiin tietotyyppeihin kuuluvat: NUMERIC, DECIMAL, INTEGER, SMALLINT ja FLOAT. NUMERIC on tarkka desimaaliluku, joka sisältää täsmälleen määrätyn maksimimäärän numeroita, joista tietty osa on varattu desimaaleille.

DECIMAL on tarkka desimaaliluku, joka sisältää vähintään määrätyn maksimimäärän numeroita, joista tietty määrä on desimaaleja. INTEGER ja SMALLINT ovat kokonaislukutietotyyppejä, joista SMALLINT-tietotyypin lukualue on pienempi. FLOAT on liukulukutietotyyppi, jolle voidaan asettaa haluttu tarkkuus.

Binäärimuotoisen datan tallentamiseen tarkoitetut tietotyypit ovat BIT ja BIT VARYING. BIT on kiinteämittainen eli täsmälleen määritetyn pituuden mittainen. BIT VARYING on vaihtuvamittainen bittijono, joka on vastaavalla tavalla maksimissaan annetun pituuden mittainen kuin VARCHAR-merkkijonotietotyyppi.

Aikaa määritteleviä tietotyyppejä ovat INTERVAL, DATE, TIME sekä TIMESTAMP. INTERVAL on vuosista ja kuukausista tai vaihtoehtoisesti päivistä, tunneista, minuuteista ja sekunneista koostuva ajanjakso. DATE sisältää vuoden (0001-9999), kuukauden (01-12) ja päivän (01-31). TIME-tietotyyppi sisältää tunnit, minuutit ja sekunnit tuhannesosan tarkkuudella. TIMESTAMP sisältää vuoden, kuukauden, päivän, tunnit, minuutit ja sekunnit. TIMESTAMP-tietotyypin rajoitukset ovat vastaavat, kuin DATE- ja TIME-tietotyypeissä. Lisäksi TIMESTAMP voi sisältää tiedon aikavyöhykkeestä.

Uuden taulun luonti aloitetaan CREATE TABLE -komennolla, jolle kerrotaan taulun nimi. Nimen jälkeen luetellaan tauluun luotavien kenttien nimet ja niiden tietotyypit sekä tietotyyppien mahdolliset pituudet. Jokaiseen kenttään voidaan määritellä tiedon pakollisuus. Jos halutaan, että kenttään on pakko syöttää jokin arvo, kirjoitetaan kentän tyyppimäärittelyn jälkeen NOT NULL. Alla olevassa esimerkissä luodaan taulu työntekijöiden tietojen tallennusta varten. Perusavain määritellään kirjoittamalla tietotyypin jälkeen PRIMARY KEY. Perusavainkenttä on myös automaattisesti NOT NULL -tyyppinen kenttä, koska perusavainkentän arvo ei saa olla NULL.

```
CREATE TABLE Tyontekija (
TyontekijaID     SMALLINT          PRIMARY KEY,
Etunimi          VARCHAR(32)       NOT NULL,
Sukunimi         VARCHAR(64)       NOT NULL,
OsastoID         INTEGER           NOT NULL,
Hetu             CHAR(11)          NOT NULL UNIQUE,
Palkka           FLOAT             NOT NULL,
Aloituspvm       TIMESTAMP         DEFAULT CURRENT_TIMESTAMP,
Sahkoposti       VARCHAR(128),

CONSTRAINT Tyontekija_FK_Osasto
  FOREIGN KEY (OsastoID)
  REFERENCES Osasto (OsastoID)
    ON UPDATE CASCADE
    ON DELETE NO ACTION,

CONSTRAINT Tyontekija_Check_ID
  CHECK(TyontekijaID > 999 AND TyontekijaID < 10000)
)
```

Kuva 3.3. Relaatiotietokantaulun luonti.

Taulun kenttien oletusarvo voidaan määritellä DEFAULT-määreellä. Ilman määritystä kentän oletusarvo on NULL. Yllä luotavassa työntekijätaulussa aloituspäivämäärä-kentän oletusarvoksi määritellään sen hetkinen ajankohta käyttämällä funktiota CURRENT_TIMESTAMP. Kentän arvoksi tulee siis automaattisesti uuden työntekijän lisäyksen ajankohta.

CREATE TABLE -lauseen yhteyteen voidaan määritellä kenttien sisältöä rajoittavia ehtoja käyttämällä CONSTRAINT-määritettä. Rajoitemäärityksiä ovat PRIMARY KEY (perusavain), UNIQUE (kaksoisarvojen poisto), FOREIGN KEY (viiteavain) sekä CHECK (tarkistukset).

UNIQUE-määritteen avulla voidaan estää kaksoisarvot tietyssä kentässä tai kenttäyhdistelmässä. Yllä olevan työntekijätaulun henkilötunnus-kenttä määritellään yksikäsitteiseksi. Määritteen ansiosta ei ole mahdollista lisätä kahta työntekijää samalla henkilötunnuksella. Jos halutaan estää samat etu- ja sukunimiyhdistelmät eli täyskaimat, voidaan lisätä alla olevan esimerkin mukainen rajoitus. Rajoitus lisätään käyttäen taulun muokkauskomentoa, josta kerrotaan myöhemmin lisää.

```
ALTER TABLE Tyontekija
ADD CONSTRAINT Tyontekija_kaimarajoitus
UNIQUE(Etunimi,Sukunimi)
```

Kuva 3.4. Saman etu- ja sukunimi yhdistelmän esto.

Viiteavaimen määrittelyyn käytetään FOREIGN KEY- ja REFERENCES- määritteitä. Ennen viiteavaimen luontia viitattavan taulu on oltava olemassa. Edellä olleen työntekijätaulun luontiesimerkissä osastotaulu tulee luoda siis ensin. FOREIGN KEY -määritteen jälkeen kirjoitetaan kenttä, johon viite-eheys luodaan. Tämän jälkeen tulee REFERENCES-määrite, jolle annetaan viitattavan taulun nimi sekä taulun kenttä, johon viittaus kohdistetaan. Lisäksi voidaan valita toimenpide, joka määrittelee miten toimitaan, jos eheyttä rikotaan poistamalla tai päivittämällä äititaulun kenttiä. Toimenpiteitä ovat NO ACTION, SET NULL (viiteavaimen nollaus), SET DEFAULT (oletusarvon asettaminen), CASCADE (johdannaismuutos) sekä RESTRICT (rajoitettu muutos).

NO ACTION -määritys estää viite-eheyden rikkovan päivityksen tai poiston tapahtumisen. Ts. määritys ei salli muutoksia äititaulun arvoihin, joihin viitataan lapsitaulussa. SET NULL -määrite ei estä äititauluun tehtävää muutosta, vaan asettaa niihin lapsitaulun kenttiin NULL, jotka viittaavat muutoksen kohteena olevaan äititaulun kenttään. Määritys edellyttää, että lapsitaulun kenttään saa syöttää NULL-arvoja. SET DEFAULT toimii vastaavalla tavalla kuin SET NULL, mutta lapsitaulun kentän arvoiksi asetetaan kenttään määritetty oletusarvo. CASCADE-määritys tekee lapsitaulun kentille vastaavat toimenpiteet kuin äititauluun. Jos äititaulusta poistetaan rivi, niin ko. riviin viittaavat lapsitaulun rivit poistetaan myös. Samoin äititauluun tehtävät päivitykset astuvat voimaan myös lapsitaulun vastaaviin arvoihin. RESTRICT-määritys toimii vastaavalla tavalla kuin NO ACTION, mutta tietokannan hallintajärjestelmä huomaa viite-eheyden rikkoutumisen ennen varsinaista muutosta, jolloin sitä ei pääse tapahtumaan. NO ACTION -määritys huomaa rikkoutumisen vasta muutoksen jälkeen ja peruuttaa tehdyn muutoksen.

Työntekijä-taulun luontiesimerkissä (Kuva 3.3) luotiin viite-eheys työntekijätaulun OsastoID-kentästä osastotaulun OsastoID-kenttään. Viite-eheyteen määriteltiin kaksi eri sääntöä. Ensimmäinen sääntö estää osastotaulun rivin poistamisen, jos työntekijä-

taulussa on viittauksia ko. riviin (ON DELETE NO ACTION). Toinen sääntö aiheuttaa OsastoID-kenttään tehtävien muutoksien päivittymisen myös työntekijätaulun vastaavaan kenttään (ON UPDATE CASCADE).

CHECK-lausetta käytetään kenttien tarkistussääntöjen luontiin. CHECK-lause voi sisältää monimutkaisia ehtoja, jotka on rakennettu SELECT-lauseilla, mutta pelkän arvoalueen määrittely on melko yksinkertaista. SELECT-lauseet käsitellään myöhemmässä vaiheessa. Työntekijätaulun luonnissa työntekijälle määriteltiin henkilökohtainen tunnistekenttä, joka toimii myös taulun pääavainkenttänä. Tunnistekenttään määritettiin CHECK-lauseella arvoalue, joka määrittää että kenttään syötettävän arvon on oltava nelinumeroinen (TyontekijaID > 999 AND TyontekijaID < 10000).

Taulun rakennetta voidaan jälkeenpäin muuttaa ALTER TABLE -lauseella. Alla olevassa esimerkissä suoritetaan eri muokkaustoimenpiteitä. Ensimmäiseksi lisätään työntekijätauluun uusi puhelinnumerokenttä. Jos muutoksia tehdään olemassa olevaan kenttään, käytetään ALTER COLUMN -komentoa. Toisessa kohdassa asetetaan palkkakentän oletusarvoksi 3000. Kolmannessa kohdassa poistetaan työntekijätaulun sähköpostikenttä. Viimeisessä kohdassa poistetaan työntekijätaulu kokonaisuudessaan. (Lahtonen 2002, 37-53.)

```
1. ALTER TABLE Tyontekija
   ADD COLUMN Puhelinnumero VARCHAR(20)

2. ALTER TABLE Tyontekija
   ALTER COLUMN Palkka SET DEFAULT 3000

3. ALTER TABLE Tyontkeija
   DROP COLUMN Sahkoposti

4. DROP TABLE Tyontekija
```

Kuva 3.5. Tietokannan muokkauskomentoja.

3.2.2 Sisällön käsittely

Tietokannan sisällön haku, muokkaus ja lisääminen ovat SQL-kielen tärkein osa-alue. Näitä toimintoja vastaavat SQL-kielen komennot ovat SELECT (kysely), INSERT (lisäys), UPDATE (päivitys) ja DELETE (poisto). SQL-kielen käytetyin toiminto on kyselylause SELECT, joka sisältää haettavan taulun ja sarakkeiden nimet sekä mahdolliset kyselyehdot. Kyselylauseen perussyntaksi on seuraavanlainen.

```
SELECT Etunimi, Sukunimi
FROM Tyontekija
```

Kuva 3.6.

Kysely palauttaa kaikkien työntekijöiden etunimen ja sukunimen. Haettavat kentät tulevat SELECT-käskyn jälkeen pilkulla erotettuina. Jos halutaan hakea taulun kaikkien kenttien kaikki tiedot, korvataan kenttien nimet yhdellä *-merkillä. Lisäksi kyselylauseeseen voidaan sijoittaa ehtoja, joka määrittelevät mitkä rivit haetaan. Jos esim. halutaan tulostaa vain tietyn osaston työntekijöiden tiedot, kysely on seuraavanlainen.

```
SELECT *
FROM Tyontekija
WHERE OsastoID = 10
```

Kuva 3.7.

Ehdoissa voidaan käyttää vertailuoperaattoreja yhtä suuri (=), pienempi kuin (<), suurempi kuin (>), pienempi tai yhtä suuri kuin (<=), suurempi tai yhtä suuri kuin (>=) sekä erisuuri kuin (<>). Ehto saa kohdistua myös sellaiseen kenttään, joka ei tule lopputulokseen. Seuraavassa haussa haetaan kaikkien työntekijöiden tiedot, joiden sukunimi tulee aakkosissa sukunimen Virtanen jälkeen.

```
SELECT *
FROM Tyontekija
WHERE Sukunimi > 'Virtanen'
```

Kuva 3.8.

Päivämääräkenttään kohdistuvassa hakuehdossa tulee ottaa huomioon päivämäärän esitysmuoto. Päivämäärien tulkinnassa on suuria maa-, ympäristö ja ohjelmistokohtaisia eroja. Kannattavinta on käyttää ISO-standardin mukaista muotoa vvvv-kk-pv. Tässä muodossa suomalaisittain ilmoitettu päivämäärä 24.12.2008 on muodossa 2008-12-24. Seuraavassa haussa haetaan kaikkien sellaisten työntekijöiden etu- ja sukunimi, jotka ovat tulleet töihin ennen 1.6.2008.

```
SELECT Etunimi, Sukunimi
FROM Tyontekija
WHERE Aloituspvm < '2008-6-1'
```

Kuva 3.9.

Ehtolauseen merkitys voidaan kääntää päinvastaiseksi käyttämällä NOT-operaatiota ehtolauseen edessä. Seuraavassa kyselyssä haetaan kaikki työntekijät, jotka ovat aloittaneet työnsä jonain muuna vuonna, kuin 2008. Haussa käytetään lisäksi EXTRACT-funktiota, jolla saadaan erotettua vuosi ajankohtaa ilmaisevasta kentästä.

```
SELECT *
FROM Tyontekija
WHERE NOT EXTRACT(YEAR FROM Aloituspvm) = 2008
```

Kuva 3.10.

Merkkijonoista voidaan tehdä kyselyjä myös osittain täsmäävillä arvoilla. Tähän toimintoon käytetään LIKE-operaattoria sekä prosentti- ja alaviivamerkkiä. %-merkki tarkoittaa mitä tahansa merkkijonoa ja _-merkki mitä tahansa yksittäistä merkkiä. Alaviivaa voidaan käyttää esim. kyselyssä, jossa halutaan hakea tietyn mittaiset nimet. Tällöin haettavaan merkkijonoon sijoitetaan vastaava määrä alaviivoja. Seuraava kysely tulostaa kaikkien työntekijöiden tiedot, joiden sukunimi päättyy ”nen”.

```
SELECT *
FROM Tyontekija
WHERE Sukunimi LIKE '%nen'
```

Kuva 3.11.

Ehtolauseiden muodostamisessa voidaan käyttää AND- ja OR-operaattoreita, joiden avulla voidaan yhdistää useampia hakuehtoja. AND-operaattorin avulla voidaan yhdistää hakuehdot, joiden on molempien toteuduttava, jotta käsiteltävä rivi palautetaan. OR-operaattorilla yhdistettävistä hakuehdoista vain toisen on toteuduttava. Operaattoreita voi käyttää myös sekaisin, jolloin niiden suoritusjärjestys kannattaa varmistaa sulkujen avulla. Alla olevassa esimerkissä haetaan kaikki 2000-luvulla aloittaneet työntekijät, jotka eivät kuulu osastoon numero 10. Lisäksi haetaan kaikki työntekijät, jotka kuuluvat osastoon 20.

```
SELECT *
FROM Tyontekija
WHERE (EXTRACT(YEAR FROM Aloituspvm) >= 2000
AND OsastoID <> 10)OR osastoID = 20
```

Kuva 3.12.

Kyselyn tuottamat hakutulokset voidaan järjestää haluttuun järjestykseen tietyn kentän mukaan. Lisäksi voidaan määritellä onko järjestys laskeva (DESC) vai nouseva (ASC). Alla oleva kysely palauttaa kaikkien työntekijöiden tiedot sukunimen mukaisessa aakkosjärjestyksessä.

```
SELECT *
FROM Tyontekija
ORDER BY Sukunimi ASC
```

Kuva 3.13.

Jos halutaan hakea rivejä siten, että tietty arvo toistuu vain kerran, voidaan käyttää DISTINCT-määrettä. Seuraava kysely palauttaa työntekijöiden eri sukunimet. Jokainen sukunimi esiintyy vain kerran, vaikka yrityksessä olisi useampi työntekijä, jolla on sama sukunimi.

```
SELECT DISTINCT Sukunimi
FROM Tyontekija
```

Kuva 3.14.

SQL tarjoaa koostefunktioita, joita voidaan käyttää SELECT-lauseen kenttäluettelossa tai HAVING-lauseessa. Funktioita ovat AVG (keskiarvo), COUNT (arvojen lukumäärä), MAX (suurin arvo), MIN (pienin arvo) sekä SUM (summa). COUNT(*) laskee kaikkien palautettavien tulosrivien lukumäärän, myös NULL-arvon sisältävät. Jos COUNT-funktiolle annetaan jonkin kentän nimi, lasketaan vain niiden rivien lukumäärä, joiden arvo ei ole NULL. Kaikki muut funktiot jättävät NULL-arvot huomioimatta. Seuraava esimerkki tulostaa työntekijöiden lukumäärän, sekä lukumäärän monellako heistä on sähköpostiosoite. Lisäksi kyselyn palauttamille kentille asetetaan otsikot AS-sidesanaa käyttämällä.

```
SELECT COUNT(*) AS Työntekijöitä,
COUNT(Sahkoposti) AS Sähköposteja
FROM Tyontekija
```

Kuva 3.15.

Jos kyselyssä käytetään tavallisten kenttien lisäksi koostefunktioita, tulee kaikki tavalliset kentät luetella GROUP BY -lauseessa. Seuraavassa esimerkissä haetaan työntekijöiden palkkojen keskiarvo osastoittain. Toisessa esimerkissä lisätään HAVING-lause, jolla voidaan määritellä mitkä ryhmittelyn tulosriveistä haetaan. Esimerkissä tulostetaan sellaisten osastojen keskipalkka, joissa on enemmän, kuin 1 työntekijä.

```
SELECT OsastoID, AVG(Palkka) AS Palkka_KA
FROM Tyontekija
GROUP BY OsastoID
```

```
SELECT OsastoID, AVG(Palkka) FROM Tyontekija
GROUP BY OsastoID
HAVING COUNT(TyontekijaID) > 1
```

Kuva 3.16.

Kun halutaan hakea tietoa useammasta taulusta kerralla, luetellaan FROM-lauseessa ko. taulut pilkuilla erotettuna. Hakulauseessa voidaan viitata kaikkiin haettujen taulujen kenttiin. Jos taulut sisältävät samannimisiä kenttiä, niihin tulee viitata kyselyssä lisäämällä kentän nimen eteen taulun nimi sekä piste. Se, miten taulut liittyvät toisiinsa, kerrotaan WHERE-lauseessa. Liitos luodaan kahden sarakkeen välille, joista

pitää löytyä toisiaan vastaavia arvoja. Yleensä yhdistäminen tehdään äititaulun perusavainkentän ja lapsitaulun viiteavainkentän välille. Seuraavassa esimerkissä haetaan työntekijöiden etu- ja sukunimi sekä osaston nimi.

```
SELECT Etunimi, Sukunimi, Nimi
FROM Tyontekija, Osasto
WHERE Tyontekija.OsastoID = Osasto.OsastoID
```

Kuva 3.17.

Ilman liitosehtoa yllä oleva kysely palauttaisi kaikki mahdolliset etunimi-sukunimi plus osasto -kombinaatiot. Ts. kaikki työntekijät kuuluisivat tulosteen mukaan jokaiseen osastoon.

Yksi SQL:n hyödyllinen ominaisuus on alikysely. Alikyselyiden avulla voidaan rajata varsinaisen kyselyn palauttamaa arvojoukkoa sijoittamalla pääkyselyn sisään toinen hakulause. Myös alikysely voi sisältää oman alikyselyn, joten sisäkkäisiä alikyselyjä voi olla useita. Jos alikysely palauttaa vain yhden rivin, voidaan käyttää normaaleja vertailuoperaattoreita. Alla olevassa kyselyssä haetaan kaikki sellaiset työntekijät, jotka ovat aloittaneet yrityksessä aikaisemmin, kuin työntekijöiden aloituspäivämäärän keskiarvo on.

```
SELECT *
FROM Tyontekija
WHERE Aloituspvm < (
  SELECT AVG(Aloituspvm) FROM Tyontekija
)
```

Kuva 3.18.

Jos alikysely palauttaa useamman rivin, tarvitaan operaattoreita. IN-operaattorilla voidaan tarkistaa löytyykö alikyselyn palauttamasta arvojoukosta vastaava arvo, kuin pääkyselyssä. Seuraavassa kyselyssä haetaan kaikki Porin osastoissa työskentelevät työntekijät.

```
SELECT *
FROM Tyontekija
WHERE OsastoID IN (
  SELECT OsastoID
  FROM Osasto
  WHERE sijainti LIKE 'PORI'
)
```

Kuva 3.19.

Tietojen hakemisen lisäksi oleellista on tiedon lisääminen, muuttaminen sekä päivittäminen. Uuden rivin lisääminen tietokantaan suoritetaan INSERT INTO -komennolla. Alla olevassa esimerkissä lisätään työntekijätauluun uusi työntekijä. INSERT INTO -komennon jälkeen kerrotaan taulun nimi, jonka jälkeen tulee VALUES-määre. VALUES-määreen jälkeen luetellaan taulun kenttiin syötettävät arvot.

```
INSERT INTO Tyontekija
VALUES (1000,'Joni','Johtaja',50,
'010476-N394',8000, '2008-11-02 15:44:41',
'joni.johtaja@yritys.com')
```

Kuva 3.20.

Jos kaikkiin kenttiin ei haluta syöttää arvoa tai syötettävät arvot halutaan luetella eri järjestyksessä, kuin tauluun on määritelty, luetellaan kenttien nimet erikseen. Alla olevassa esimerkissä suoritetaan vastaavanlainen rivin syöttö kuin edellisessä esimerkissä, mutta tiedot syötetään eri järjestyksessä ja kaikkia kenttiä ei täytetä. Sähköpostikenttää ei ole määritelty pakolliseksi, joten sen voi jättää kokonaan syöttämättä. Aloituspvm-kenttään on määritelty oletusarvoksi CURRENT_TIMESTAMP -funktio, joten ajankohdan syöttäminen erikseen on turhaa.

```
INSERT INTO Tyontekija(TyontekijaID,Etunimi,
Sukunimi,OsastoID,Hetu,Palkka)
VALUES (1000,'Joni','Johtaja',50,
'010476-N394',8000)
```

Kuva 3.21.

Tietojen muuttaminen tapahtuu UPDATE-komennolla. Alla olevassa esimerkissä muutetaan työntekijän kuukausipalkkaa. UPDATE-komennon jälkeen kirjoitetaan taulun nimi. Taulun nimen jälkeen tulee SET-määre, jolle luetellaan kentät ja niihin päivitettävät arvot pilkuilla erotettuna. Lisäksi UPDATE-komennossa voidaan rajoittaa päivitettäviä kenttiä käyttämällä WHERE-ehtoa. Ilman WHERE-ehtoa tiedon päivitys suoritetaan kaikkiin riveihin. Alla olevassa esimerkissä WHERE-ehtoon sijoitetaan työntekijän henkilökohtainen tunniste.

```
UPDATE Tyontekija
SET palkka = 2600
WHERE TyontekijaID = 1003
```

Kuva 3.22.

UPDATE-lauseessa voidaan käyttää myös laskutoimituksia. Seuraavassa esimerkissä annetaan yrityksen kaikille työntekijöille yleinen 10 prosentin palkankorotus.

```
UPDATE Tyontekija
SET palkka = palkka * 1.1
```

Kuva 3.23.

Tiedon poistaminen suoritetaan DELETE-komennolla. Koko taulun tyhjentäminen onnistuu kirjoittamalla DELETE FROM sekä taulun nimi. Tämän vuoksi on tärkeä muistaa käyttää WHERE-ehtoa, jotta vältetään turha tiedon poistaminen. Seuraavassa esimerkissä poistetaan yrityksen työntekijä. Ilman WHERE-ehtoa poistettaisiin kaikkien työntekijöiden tiedot. (Lahtonen 2002, 60-115.)

```
DELETE FROM Tyontekija
WHERE TyontekijaID = 1007
```

Kuva 3.24.

3.3 Microsoft SQL Server

SQL Server on Microsoftin kehittämä relaatiotietokannan hallintajärjestelmä (RDBMS). SQL Serverin kyselykielenä käytetään Microsoftin ja Sybasen kehittämää T-SQL-kieltä (Transact-SQL), joka on SQL-kielen laajennus. (Wikipedia 2008a.)

T-SQL laajentaa SQL:ää ominaisuuksilla, kuten kontrolli- ja silmukkarakenteet sekä paikalliset muuttujat. Ko. ominaisuuksilla voidaan toteuttaa ehtolauseita (if-else-rakenne) sekä silmukoita (while-rakenne), jotka sisältävät suoritettavia tietokantakomentoja. Paikallisen, eli vain suoritettavalle komennolle näkyvän muuttujan avulla voidaan määritellä mm. silmukan suorituskerrat. Lisäksi T-SQL sisältää useita funktioita, joiden avulla voidaan käsitellä mm. tekstiä ja päivämääriä sekä suorittaa erilaisia laskutoimituksia. (Wikipedia 2008b.)

SQL Server tukee eri tietotyyppejä, kuten INTEGER, FLOAT, VARCHAR, BINARY ja TEXT. Lisäksi siinä voidaan määritellä ja käyttää omia yhdistelmätyyppejä, jotka koostuvat eri tietotyypeistä. SQL Server käyttää transaktioita varmistamaan, että jokainen operaatio suoritetaan kokonaan tai ei lainkaan. Ts. operaatio perutaan jos aiheutuu jokin virhetilanne. Ajettavista tietokantakomennoista voidaan muodostaa ryhmiä, jolloin varmistetaan, että kaikki komennot suoritetaan onnistuneesti. Transaktioiden ansiosta tietokantaa ei koskaan jätetä minkäänlaiseen välitilaan.

SQL Serveriin on sisältynyt versiosta 2005 lähtien SQL Server Management Studio, jonka avulla tietokantaa voidaan hallita graafisen käyttöliittymän välityksellä. Työkalun avulla tietokantaa voidaan hallita sekä komentoeditorilla, että graafisilla työkaluilla. Työkalun toimintoihin kuuluvat SQL Serverin hallinta, kaikkien siihen kuuluvien komponenttien ylläpito sekä itse tietokantojen luonti, muokkaus sekä sisällön haku. (Wikipedia 2008a.)

4 WEB 2.0

4.1 Taustat

O'Reilly Media ja Medialive International järjestivät lokakuussa 2004 konferenssin, jossa termi Web 2.0 lanseerattiin maailmalle. Web 2.0 on termi, jolla viitataan World Wide Webin toiseen vaiheeseen. Sillä ei tarkoiteta minkäänlaista ohjelmistopäivitys-

tä, vaan verkon nykyisiä sosiaalisia ja teknisiä piirteitä. Web 2.0 koostuu kahdesta päälinjauksesta: siirtyminen WWW-pohjaisiin sovelluksiin sekä sosiaalisempi lähestymistapa sisällön tuottamiseen ja jakamiseen. Perustana toimii vanhan staattisen verkon muuttuminen dynaamiseksi ja joustavaksi kokonaisuudeksi. Aikaisemmin verkko toimi yksisuuntaisen kommunikoinnin välineenä, jossa sisällöntuottajat tekivät sivuja, joita verkon käyttäjät lukivat. Tiedon kulku oli yksisuuntaista. Nykyään käyttäjät ovat vuorovaikutuksessa toisiinsa erilaisten blogien ja keskustelupalstojen välityksellä.

Muutos vuorovaikutteisempaan Internetiin näkyy myös siinä, että omien kotisivujen tekemisen sijaan ihmiset kerääntyvät erilaisten yhteisöjen käyttäjiksi. Perinteiset kotisivutkaan eivät silti ole kadonneet, vaan niiden luonne on muuttunut. Nykyään kotisivut rakennetaan usein blogin ympärille. Blogi on sivusto, johon sen ylläpitäjä tuottaa ajankohtaista sisältöä. Blogit mielletään usein verkkopäiväkirjoiksi. Bloggaaminen on päiväkirjan tyyliin päivämäärällä merkatun tekstin tuottamista, mutta se on aihealueeltaan usein perinteistä päiväkirjaa laajempi. Blogit ovat usein vapaasti kommentoitavia, yhteisöllisiä ja nopeasti päivittyviä tietolähteitä, joiden sisältö voi koostua esim. maailman tapahtumista, uusista trendeistä tai henkilökohtaisesta elämästä. Blogien perustaminen on helppoa erilaisten bloggauspalvelujen avulla. Blogit edustavat hyvin Web 2.0:n ominaispiirteitä ja ovat oikeastaan sitä parhaimmillaan sekä teknisesti että ideologisesti.

Verkkopohjaisiin sovelluksiin siirtyminen merkitsee vallankumousta ohjelmistoajattelussa. Esim. Microsoftin Office-paketit ovat olleet sidoksissa käytettävään käyttöjärjestelmään ja päivityksiä on saatu vain uuden version julkistuksen myötä. Siirtyminen WWW-pohjaisiin sovelluksiin tarkoittaa järjestelmä- ja versioriippuvuuden poistumista. Ohjelmat toimivat verkossa ja erillisen ohjelmistoasennuksen sijaan käyttäjät voivat hoitaa esim. taulukkolaskennan pelkän Internet-selaimen avulla. Käyttäjällä on käytettävissään aina uusin versio ohjelmasta, sillä yksittäisten suurpäivitysten sijaan ohjelmistoja päivitetään sitä mukaa, kun uusia ominaisuuksia saadaan valmiiksi. Web 2.0 -ilmiö voidaan jakaa kolmeen osaan näiden perusominaisuuksien perusteella: entistä sosiaalisempi verkko, verkon muuttuminen riippumattomaksi alustaksi sekä yritysten asema ilmiön keskellä. (Tirronen 2008, 9-24.)

”Uskomme, että bloggaaminen ei ole vain viisasta, vaan suorastaan välttämätöntä, jos yritys haluaa päästä lähemmäksi asiakkaitaan. Voimme nähdä jo silmissämme sen aivan lähitulevaisuuteen sijoittuvan hetken, jolloin bloggaamisen sikseen jättäneisiin yrityksiin suhtaudutaan hienoisella varauksella. Ihmiset miettivät, mahtaako sellaisella yrityksellä olla jotain salattavaa. Ehkä omistajat ovat huolissaan siitä, mitä heidän alaisillaan olisi sanottavanaan?”. (Scoble & Shel 2006, 13.)

Ennen bloggaamista pidettiin yritysmaailmassa vain ohimenevänä ilmiönä. Nykyään yrityksissä nähdään kuitenkin bloggamisen tarjoamat mahdollisuudet. Scoble ja Shel uskovat, että ”se pienten tapahtumien ketju, joka johti blogien syntyyn, tulee vaikuttamaan vallankumouksellisella tavalla liiketoimintaan, markkinointiin, asiakastukeen, sisäiseen viestintään, yrityksen ja sijoittajien välisiin suhteisiin ja jopa tutkimukseen ja tuotekehittelyyn”.

Osallistumalla keskusteluun yritykset synnyttävät luottamusta ja inhimillistävät itseään. Blogien välityksellä tulevat asiakkaat voivat tutustua yrityksen työntekijöihin jo ennen varsinaista kaupankäyntiä. Tämä helpottaa kaupankäynnin syntymistä, sillä mitä enemmän ihmiset keskustelevat keskenään, sitä paremmin he oppivat ymmärtämään toisiaan. Liiketoimintaan puolestaan ryhdytään mieluiten sellaisten ihmisten kanssa, jotka tunnetaan ja joihin luotetaan. Blogi on edullinen ja tehokas viestintäkanava, sillä sen avulla voidaan tavoittaa miljoonia ihmisiä pienellä vaivalla ja erittäin edullisesti. (Scoble & Shel 2006, 41-44.)

4.2 Tekniikka

Web 2.0 -toimintojen toteuttamiseen tarvitaan erilaisia tekniikoita ja toimintamalleja. Seuraavaksi kerrotaan mitä on AJAX, RSS-syöte sekä mashup.

AJAX on ns. hybriditekniikka, joka yhdistää eri tekniikoita, kuten JavaScript, dynaaminen HTML, CSS sekä XML. Tekniikoissa ei ole sinänsä mitään uutta, vaan niitä käytetään uudella tavalla. AJAX-tekniikan ajatuksena on välttää web-sivun uudelleenlataaminen jokaisen käyttäjätoiminnon jälkeen. Sen sijaan, että ladataan koko sivu uudelleen, päivitetään vain tarvittavat osat. AJAX-sovellus on alustariippumaton

eikä sen käyttöönotto vaadi käyttäjälle minkäänlaista päivitysoperaatiota. (Hintikka 2007, 38-39.)

Normaalisti tiedon vaihtaminen palvelimen kanssa tapahtuu HTML-lomakkeella. Lomakkeen tiedot lähettääkseen, käyttäjän on painettava lomakkeen lähetysnappia ja odotettava sivun uudelleenlatausta. Jatkuva lomakkeen lähettäminen tekee sovelluksen käytöstä hidasta. AJAX-tekniikassa tiedon vaihtaminen selaimen ja palvelimen välillä tapahtuu taustalla JavaScript XMLHttpRequest-objektin välityksellä. Tämän avulla sovellus voi tehdä pyyntöjä palvelimelle ilman sivun uudelleenlatausta. Yksi hyvä sovellusesimerkki on Googlen hakutoiminto (Google Suggest), jossa käyttäjän hakukenttään syöttämät kirjaimet lähetetään palvelimelle, joka palauttaa listan hakusanaehdotuksia. (W3Schools 2008.)

RSS- eli otsikkosyötteen avulla voidaan kertoa WWW-palvelussa tapahtuvista muutoksista. Otsikkosyötettä voidaan käyttää mm. uutisten välittämiseen. Käyttäjät voivat lukea kätevästi uusimmat ja suosituimmat uutiset eri aihealueisiin liittyen yhdestä paikasta. Syötteet ovat hyödyllisiä sekä sisällön tuottajille, että sen lukijoille. Sisällön tuottaja pystyy tarjoamaan lukijoille reaaliaikaisen tavan tiedonsaantiin ilman, että käyttäjän tarvitsee jatkuvasti käydä sisällöntuottajan web-sivuilla. Ts. uutiset tulevat käyttäjän luo eikä toisinpäin. Otsikkosyötettä voidaan hyödyntää uutisotsikoiden lisäksi tarjoamaan tietoa mm. pörssikursseista sekä tuoteuutuuksista, jolloin kohderyhminä toimivat kuluttajat sekä sijoittajat. Syötteitä voidaan vastaanottaa monella eri tavalla mm. tallentamalla syöte omaan selaimeen, käyttämällä yksittäistä Internet-sivustoa, jonne on koottu syötteitä useasta eri lähteestä tai käyttämällä erillistä RSS-keräinohjelmaa. (Hintikka 2007, 25-26.)

Mashup on uudenlainen tapa tuottaa web-sovelluksia. Mahsup yhdistää eri käyttöliittymiä ja datavarantoja yhteen muodostaen yhden monipuolisen kokonaisuuden. Yksi suosituimmista käyttöliittymistä on Google Maps, johon voidaan yhdistää paikkatietoja eri aiheisiin liittyen, kuten tietoa lähialueella olevista kaupoista ja ravintoloista. Mashup perustuu käyttäjän selaimella suoritettavaan JavaScript-kieleen. Lisäksi käytetään usein jotain palvelinpäässä suoritettavaa kieltä, joka hakee sovellukseen tarvittavat tiedot. Tieto haetaan erilaisista ulkoisista ja sisäisistä lähteistä (palvelimista),

joiden yhteen kokoaminen muodostaa itse mashup-sovelluksen. (Hintikka 2007, 40-43.)

5 KESKUSTELUOMINAISUUS

5.1 Lähtökohdat

Työn lähtökohtana oli Hypermedia Oy:n Hyperscriptor XG WWW-julkaisujärjestelmä ja sen jatkokehitys. Julkaisujärjestelmä on aikaisemmin kehitetty insinöörityötkinnon opinnäytetyönä (Tietokantapohjainen www-julkaisujärjestelmä ASP.Net-kielellä / Niko Erkintalo). Tuolloin yrityksen vanhasta ASP-pohjaisesta julkaisujärjestelmästä tehtiin kokonaan uusi versio, koska siirryttiin käyttämään uudempaa ASP.NET-tekniikkaa. Lisäksi samaan julkaisujärjestelmään liittyen on tehty ammattikorkeakoulun ylempään insinöörityötkintoon kuuluva opinnäytetyö, joka käsittelee verkkokauppasovelluksen kehittämistä (Tuotetiedon ja sähköisen kaupankäynnin www-julkaisuympäristön kehittäminen / Mika Peuralahti). Verkkokauppasovellus sisältää julkaisujärjestelmän lisäksi verkkokauppatoiminnallisuudet. Julkaisujärjestelmän tärkeimpiä ja erityisesti tämän opinnäytetyön kannalta oleellisia ominaisuuksia käydään läpi seuraavassa alaluvussa.

Kuten edellisessä luvussa kävi ilmi, interaktiivisuus on hyvin tärkeä osa nykypäivän Internetiä. Monet sivustot sisältävät keskustelufoorumin tai blogin. Hypermedia Oy:n WWW-julkaisujärjestelmän oli aika siirtyä syvemmälle Web 2.0 -aikaan. Järjestelmä ei ennestään sisältänyt ominaisuutta, joka mahdollistaa käyttäjien välisen keskustelun. Hypermedia Oy:n toimeksiannosta tässä opinnäytetyössä toteutettiin julkaisujärjestelmään keskustelu-moduuli.

Lähtökohtana oli uuden moduulin kokonaisvaltainen suunnittelu ja toteutus. Uusi ominaisuus mahdollistaa kommenttien lisäämisen julkaisujärjestelmän sivuihin julkisesta Internet-käyttöliittymästä käsin. Kommentin lisääminen voi kohdistua esim. verkkokaupassa olevaan tuotteeseen, verkkolehden uutisartikkeliin, blogiin tai vie-

raskirjaan. Keskusteluja ylläpidetään julkaisujärjestelmän ylläpitokäyttöliittymästä käsin, jonka käyttämiseen tarvitaan pelkästään Internet-selain.

Hyperscriptor on toteutettu Microsoftin ASP.NET-tekniikalla (ks. 2 Microsoft ja Internet) ja SQL Server -tietokannalla (ks. 3.3 SQL Server). Ohjelmointiin käytetään Microsoftin Visual Studio .NET -kehitysympäristöä ja Visual Basic .NET -ohjelmointikieltä. Minulla ei ollut aikaisempaa kokemusta em. tekniikoista, joten yhtenä lähtökohtana oli niihin tutustuminen.

Ennen suunnittelun aloittamista oli syvennyttävä julkaisujärjestelmään, sen ominaisuuksiin, käyttölogiikkaan ja rakenteeseen. Hyperscriptor on modulaarinen järjestelmä, joka sisältää useita erillisiä ominaisuuksia, joita kutsutaan moduuleiksi. Kukin julkaisujärjestelmän ostava asiakas voi tarpeensa mukaan valita mitä moduuleja tarvitsee perusominaisuuksien lisäksi. Keskustelutyökalun suunnitteluvaiheessa oli otettava huomioon järjestelmän perusominaisuuksien lisäksi myös yhteensopivuus muiden moduulien kanssa. Lisäksi keskustelutyökalun ylläpidon tulisi noudattaa samaa ulkoasua ja logiikkaa kuin muu järjestelmä.

Työn tekemiseen varattiin aikaa 1.5. - 31.8.2008 välisenä aikana 12,5 tuntia viikossa. Työn kirjallinen osuus oli tarkoitus saada valmiiksi vuoden 2008 loppuun mennessä.

5.2 Hyperscriptor XG 3.5 WWW-julkaisujärjestelmä

Hyperscriptor XG 3.5 on Hypermedia Oy:n kehittämä ohjelmistotuote Internet-julkaisuun. Järjestelmää käyttävien ylläpitäjien ei tarvitse osata HTML-kieltä, vaan sivujen ylläpito onnistuu käyttöliittymällä, joka toimii Internet-selaimella. Päivittäminen onnistuu siis mistä tahansa ja milloin tahansa, kunhan käytössä on Internet-yhteys ja -selain. Seuraavaksi käydään läpi julkaisujärjestelmän rakenne ja perusominaisuudet sekä erityisesti tämän opinnäytetyön kannalta oleelliset asiat.

Järjestelmällä luotavien Internet-sivujen sisältö tuotetaan wysiwyg-editorilla (what you see is what you get). Sisältöä luodaan käyttämällä tekstinkäsittelyohjelmista tuttuja tekstin muotoilupainikkeita. Editori sisältää myös HTML-näkymän, joka mah-

dollistaa sisällön muokkaamisen HTML-kielellä. Jokainen sivu voi sisältää rajattoman määrän kappaleita, joista jokaista muokataan erikseen editorilla.

Julkaisujärjestelmässä voi olla sivuja seitsemässä eri tasossa (tasot 0 - 6). Päätaso, eli 0-taso, tarkoittaa sivuhierarkian ylintä tasoa. Päätason sivut näkyvät Internet-sivun navigoinnissa ylimpänä. Jokainen päätason sivu muodostaa oman sivuhaaran, jonka alapuolella voi olla sivuja tasoilla 1-6. Päätason sivun alaisuuteen luotava sivu sijoittuu siis tasolle 1. Tällä periaatteella sivuja voidaan luoda puumaiseen tyyliin aina tasolle 6 saakka.

Hyperscriptor julkaisujärjestelmässä on kahdenlaisia ylläpitäjiä: pääkäyttäjät ja ylläpitäjät. Pääkäyttäjillä on automaattisesti täydet oikeudet järjestelmän kaikkiin sivuihin ja asetuksiin. Pääkäyttäjän tehtäviin kuuluvat uusien ylläpitäjien ja pääkäyttäjien luonti, WWW-sivuston päätasojen luonti ja muokkaus sekä ylläpito-oikeuksien määrittely sivuhaaroihin.

Ylläpitäjät pystyvät muokkaamaan pelkästään sellaisia sivuhaaroja, joihin heille on myönnetty ylläpito-oikeus. Uuden sivun voi luoda vain sellaisen sivuhaaran alaisuuteen, johon on ylläpito-oikeus. Ylläpito-oikeudet periytyvät alemmille tasoille. Tämä tarkoittaa sitä, että päätason sivulle myönnetty ylläpito-oikeus pätee myös päätason alapuolella oleviin sivuihin. Ylläpito-oikeuden voi määrittää alkamaan myös alemmalta tasolta, jolloin ylläpitäjällä ei ole oikeutta muokata ko. sivun yläpuolella olevia sivuja.

Julkaisujärjestelmään on saatavilla myös useita lisäominaisuuksia, joita kutsutaan moduuleiksi. Yksi esimerkki moduulista on extranet. Extranet mahdollistaa WWW-sivustolle sivuosioiden luonnin, joihin vain tietyillä käyttäjillä on pääsyoikeus. Järjestelmän pääkäyttäjä luo käyttäjät ja määrittelee mihin extranet-ryhmään kukin käyttäjä kuuluu. Tämän jälkeen ryhmille annetaan erikseen oikeudet halutuille sivuille. Käyttäjät pääsevät lukemaan extranet-sivuja kirjautumalla sisään WWW-sivuston julkisesta käyttöliittymästä.

5.3 Suunnittelu

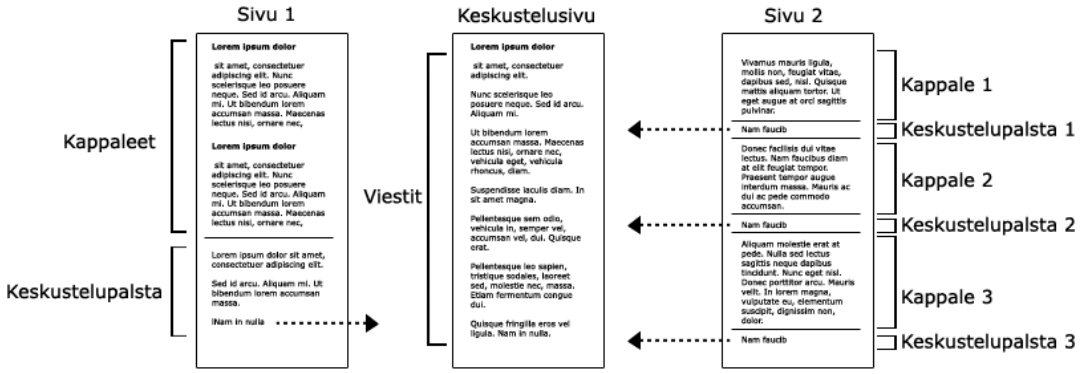
Keskustelutyökalun suunnittelu toteutettiin yksityiskohtaisella toiminnallisella määrittelyllä. Toiminnallinen määrittely kuului työn ensimmäiseen vaiheeseen ja sen tekemiseen varattiin puolet työn tekemiseen varatusta kokonaisajasta. Määrittelyyn liittyi keskustelutyökalun ominaisuuksien, käyttöliittymän, käyttöoikeuksien ja käyttölogiikan suunnittelu sekä integroiminen julkaisujärjestelmän ominaisuuksiin ja moduuleihin.

Jo suunnittelun alkumetreillä selvisi, ettei opinnäytetyön toteutus tulisi sisältämään kaikkia määrittelyn sisältämiä ominaisuuksia niiden runsaasta määrästä johtuen. Toteutus jaettiin kolmeen vaiheeseen, joista tässä työssä toteutettiin ensimmäinen. Tässä alaluvussa käsitellään kaikkia toiminnallisen määrittelyn sisältämiä ominaisuuksia, mutta toteutusta käsittelevässä alaluvussa käsitellään vain ensimmäisen vaiheen mukaisia ominaisuuksia, koska vain sen mukaiset toiminnot toteutettiin.

5.3.1 Keskustelun rakenne

Lähtökohtaisena ajatuksena oli, että julkaisujärjestelmän sivuun voidaan liittää kommentointi-ominaisuus, jonka avulla esim. yrityksen tuotesivulla asiakkaat voivat kertoa omia käyttökokemuksiaan ja parannusehdotuksia tuotteesta. Tästä ajatuksesta rakenne muokkaantui hiljalleen laajemmaksi kokonaisuudeksi.

Julkaisujärjestelmän yksi asiasivu voi koostua useista kappaleista. Sekä sivu, että sen jokainen kappale voi sisältää keskustelun. Keskustelu ilmenee kappaleen tai sivun alareunassa olevassa keskustelupalstassa. Kappaleeseen liitettävää keskustelua voidaan hyödyntää mm. blogissa tai ylipäänsä sivussa, jonka halutaan sisältävän monta keskustelua. Alla olevassa kuvassa on sivuun liitetty keskustelu (Sivu 1) sekä kappaleisiin liitettyjä keskusteluja (Sivu 2).



Kuva 5.1. Sivuun ja kappaleisiin liitetyt keskustelut.

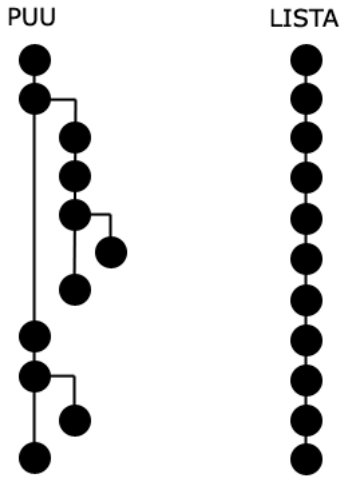
Keskustelupalstaan suunniteltiin kolme erilaista näkymää, joista ylläpitäjät voivat valita yhden keskustelua luodessaan. Yksinkertaisin näkymä sisältää keskusteluun kirjoitettujen viestien lukumäärän, viimeisimmän viestin kirjoitusajankohdan sekä linkin, josta siirrytään erilliselle keskustelusivulle lukemaan ja kirjoittamaan viestejä. Linkkitekstinä voi olla jokin keskustelun aiheeseen liittyvä houkutteleva teksti. Yllä olevassa kuvassa viitataan nuolilla keskustelusivulle siirtymistä.

Lisäksi suunniteltiin näkymä, jossa näytetään keskustelun muutama ensimmäinen tai viimeinen viesti. Tyyli on hyvin perinteinen verkkojulkaisujen parissa ja sillä voidaan helpommin herättää sivua selaavan henkilön mielenkiinto osallistua keskusteluun. Myös tämä näkymä sisältää linkin erilliselle keskustelusivulle. Kolmas vaihtoehto on näyttää keskustelupalstassa keskustelun kaikki viestit, joka vastaa keskustelusivun näkymää. Tällöin erillistä keskustelusivua ei käytetä. Viestit voidaan näyttää useammalla sivulla sivutuksen avulla. Ylläpito voi valita yhdellä sivulla näytettävien viestien lukumäärän keskustelukohtaisesti.

Erillinen keskustelusivu voi sisältää keskustelun otsikon ja viestien lisäksi aloituskappaleen. Aloituskappale on hyödyllinen lisä, jos keskusteluun ei siirrytä asiasivun keskustelupalstan kautta, vaan suoralla linkillä esim. selaimen kirjanmerkistä. Aloituskappale voi sisältää esim. jonkin artikkelin referaatin, jolloin keskusteluun osallistuvalle selviää heti mistä keskustellaan. Aloituskappale ei ole pakollinen.

Keskustelun viestit voidaan esittää kahdella eri rakenteella: lista ja puu. Käytettävä rakenne voidaan valita keskustelukohtaisesti. Listarakenteessa keskustelun viestit

näytetään perinteiseen tapaan allekkain listattuna. Puurakenteella tarkoitetaan sitä, että käyttäjät voivat kohdistaa kirjoittamansa viestin haluamaansa viestiin. Tällöin kirjoitettava viesti sijoittuu ko. ”isäntäviestin” alaisuuteen muodostamalla uuden haaran. Yhteen viestiin voi kohdistua useita vastauksia. Myös vastauksiin voi kohdistua vastauksia jne. Ylläpito voi rajoittaa keskustelukohtaisesti kuinka moneen tasoon viestejä voidaan maksimissaan kirjoittaa. Asetuksella voidaan estää turhan sekavan keskusteluraken teen muodostuminen. Kun käytetään puurakennetta, jokainen viesti sisältää erillisen ”Vastaa viestiin”-linkin.



Kuva 5.2. Keskustelun rakenteet (pallo kuvaa yhtä viestiä).

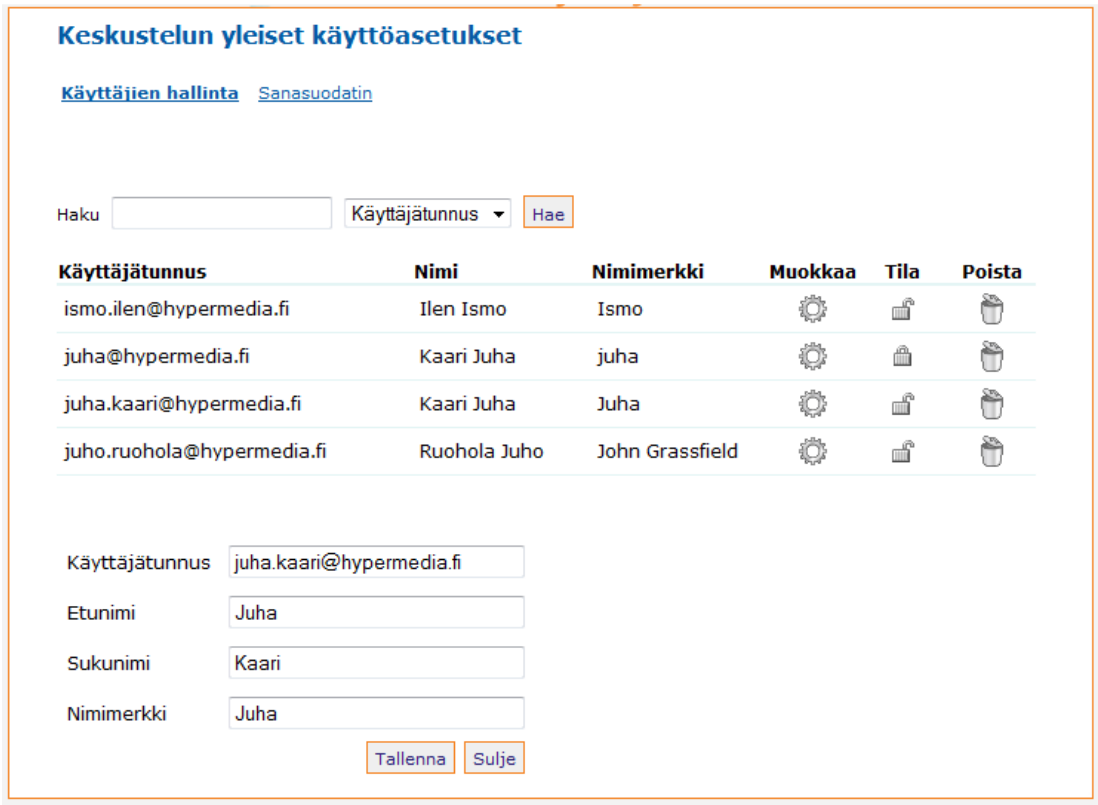
Lisäksi julkinäkymään voidaan luoda keskustelufoorumi-tyyppinen sivu, jota kutsutaan keskusteluaihesivuksi. Tämän ominaisuuden avulla sivustoon voidaan luoda sivuja, jotka sisältävät useita keskusteluja samaan aihealueeseen liittyen. Keskusteluaihesivun tärkein ominaisuus on se, että sivuston käyttäjät voivat luoda siihen omia keskustelujaan. Tällöin keskustelujen ei myöskään tarvitse liittyä minkään sivun sisältöön, vaan käyttäjät voivat aloittaa uuden keskustelun haluamastaan aiheesta.

5.3.2 Keskustelujen ylläpito

Ylläpidon suunnittelussa pyrittiin käyttämään samaa logiikkaa ja tyyliä kuin järjestelmän muissa ominaisuuksissa. Yhtenäisyys muiden ominaisuuksien kanssa helpottaa järjestelmän käyttöä yleisesti.

Niin kuin alaluvussa 5.2 kerrottiin, ylläpitäjiä voi olla kahdenlaisia: pääkäyttäjiä ja ylläpitäjiä. Pääkäyttäjillä on pääsy kaikkiin mahdollisiin julkaisujärjestelmän asetuksiin, mutta ylläpitäjille on asetettu joitain rajoituksia. Järjestelmän ylläpidossa oma erillinen näkymä, joka sisältää vain pääkäyttäjille sallittuja asetuksia. Pääkäyttäjänäkymän valikkoon suunniteltiin yksi lisäkohta keskustelutyökalun asetuksia varten. Kohta sisältää keskusteluun rekisteröityneiden käyttäjien hallintasivun sekä keskusteluista suodatettavien sanojen määrittelyn. Nämä ovat sellaisia ominaisuuksia, joita ei haluttu tuoda kaikkien ylläpitäjien käsille, vaan asetuksista vastaavat vain pääkäyttäjähenkilöt.

Käyttäjien hallintasivu sisältää kaikki järjestelmään rekisteröityneet keskustelijat. Käyttäjän tietoja ovat käyttäjätunnus, nimi ja nimimerkki. Pääkäyttäjät pystyvät tarvittaessa muokkaamaan käyttäjien tietoja. Käyttäjiä pystyy hakemaan hakusanan avulla, joka voidaan kohdistaa eri sarakkeisiin. Hakutoiminto on lähes välttämätön, sillä ilman sitä suuresta määrästä käyttäjiä yksittäisen käyttäjän hakeminen olisi melko työlästä. Lisäksi rivejä pystyy järjestämään sarakkeittain aakkosten mukaan nousevaan tai laskevaan järjestykseen sarakkeen otsikkoa klikkaamalla. Pääkäyttäjähenkilö voi asettaa yksittäisen käyttäjän tilan passiiviseksi, jolloin ko. käyttäjä ei pysty kirjautumaan järjestelmään. Toimintoa voidaan tarvita tilanteessa, jossa käyttäjän toiminta keskusteluissa todetaan häiritseväksi ja häneltä halutaan estää keskusteluun osallistuminen. Sanasuodatus käsitellään kohdassa 5.3.3 Suojaukset.

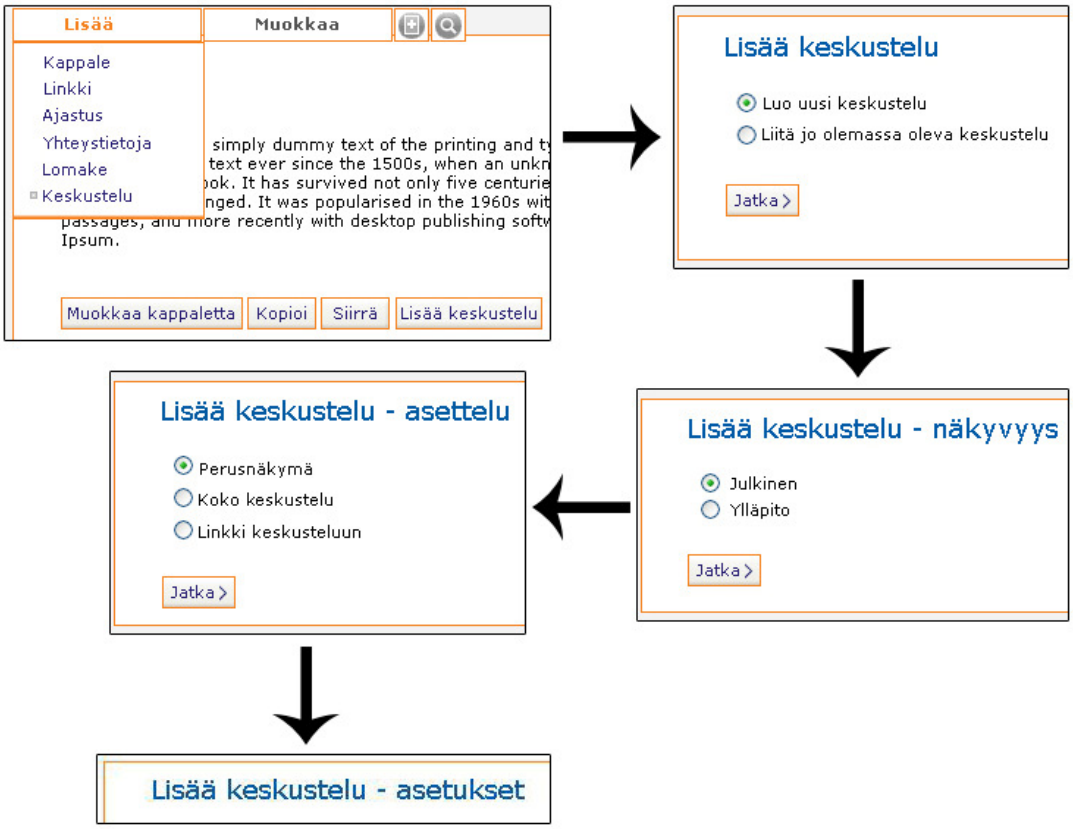


Kuva 5.3. Keskustelijoiden hallintanäkymä.

Järjestelmän ylläpitonäkymään, johon on pääsyoikeus pääkäyttäjien lisäksi myös ylläpitäjillä, suunniteltiin erillinen keskustelujen hallintasivu. Hallintasivun ajatuksena on keskustelujen keskitetty hallinta, joka tekee niiden ylläpidosta huomattavasti helpompaa. Muuten eri sivuille liitetyt keskustelut näkyisivät vain sivun omassa muokkausnäkymässä, jolloin ylläpitohenkilöillä ei olisi selvää käsitystä sivustolla olevista keskusteluista ja niiden tilasta. Hallintasivun ominaisuuksiin kuuluvat uuden keskustelun lisäys sekä olemassa olevien keskustelujen muokkaus ja ylläpito.

Kuva 5.4. Keskustelujen hallintanäkymä.

Keskustelun lisäys sivuun tehdään sivun muokkausnäkymästä käsin. Sivuun voidaan lisätä, joko ennestään olemassa oleva keskustelu tai täysin uusi keskustelu. Jos sivuun lisätään olemassa oleva keskustelu, se valitaan järjestelmän tulostamasta listasta (ks. Kuva 5.6). Toinen vaihtoehto on luoda täysin uusi keskustelu, joka liitetään automaattisesti ko. sivuun.



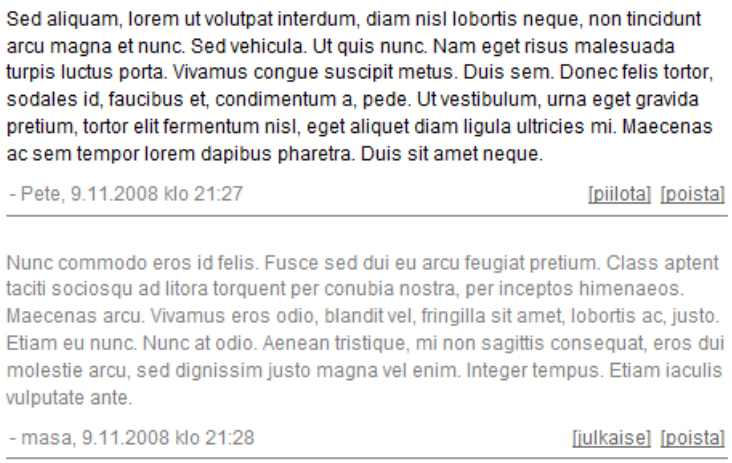
Kuva 5.5. Uuden keskustelun lisäys.

Kuvassa 5.3. käydään läpi uuden keskustelun lisäys kohta kohdalta. Ensimmäisessä kuvassa näkyy sivun muokkausnäkymä. Jos keskustelu halutaan lisätä sivuun, valitaan ”Lisää”-valikosta kohta ”Keskustelu”. Jos keskustelu halutaan lisätä sivun yksittäiseen kappaleeseen, valitaan vastaavasti kappalekohtaisesta painikkeesta ”Lisää keskustelu”. Kuvassa näkyy sivun ensimmäinen kappale. Kappaleita voi olla sivussa useita, jolloin jokaisen kappaleen kohdalla on oma keskustelun lisäyspainike. Lopuksi siirrytään keskustelun asetuksiin, joka sisältää mm. otsikon sekä aloituskappaleen määrittelyn. Kaikkia asetuksia voi jälkeenpäin muokata keskustelujen hallintasivulta.

Kuva 5.6. Olemassa olevan keskustelun liittäminen sivuun.

Keskustelun liitoksen voi tehdä myös hallintasivun liitostyökalulla, jolla voidaan hallita yhden keskustelun liitoksia keskitetysti. Ominaisuus on hyödyllinen, koska yksi keskustelu voi kuulua moneen sivuun. Saman keskustelun liittäminen moneen sivuun voi olla tarpeellista samaa aihealuetta käsittelevillä sivuilla. Tällöin sivuilla näkyy samat viestit eli kyseessä on täsmälleen sama keskustelu, joka on vain liitetty moneen sivuun.

Keskusteluun määritetyt ylläpitäjät sekä järjestelmän pääkäyttäjät voivat ylläpitää keskustelua julkaisemalla, piilottamalla sekä poistamalla viestejä. Ylläpitoon käytetään järjestelmän julkista keskustelusivua, joka sisältää normaalin keskustelunäkymän lisäksi painikkeet jokaisen viestin kohdalla em. tehtävien suorittamiseen (ks. Kuva 5.7). Vaihtoehtoisena ylläpitonäkymänä suunnitteluvaiheessa oli erillisen keskustelunäkymän luonti ylläpidon käyttöliittymään, joka olisi noudattanut ylläpidon ulkoasua. Tämä todettiin kuitenkin tarpeettomaksi ja toimintoon hyödynnettiin julkista keskustelusivua, joka olisi toteutettu joka tapauksessa. Ylläpitoon siirrytään keskustelujen hallintasivulta.



Kuva 5.7. Keskustelun ylläpito.

Keskustelutyökaluun suunniteltiin myös ominaisuus, joka mahdollistaa ylläpidon sisäisten keskustelujen luonnin. Ylläpidon keskusteluja voidaan hyödyntää mm. yrityksen WWW-sivuston kehityskeskustelussa ylläpitäjien kesken. Jokaiseen järjestelmän sivuun voidaan luoda yksi ylläpidon keskustelu. Ylläpidon keskustelun pohjana käytetään julkisen näkymän keskustelusivua.

5.3.3 Suojaukset

Kun on kyseessä Internet ja sen valtava määrä käyttäjiä, on erittäin tärkeää ottaa huomioon mahdollinen tahallinen ilkivalta. Tähän varauduttiin suunnittelemalla keskustelutyökaluun erilaisia suodattimia ja turvaominaisuuksia. Ominaisuuksia ovat sanojen suodatus, saman peräkkäisen viestin kirjoitusesto, käyttäjän tunnistus, viestin julkaisu ylläpidon kautta, ihmistodennus, sähköpostivarmennus sekä kulmasulkujen suodatus.

Sanojen suodatus on hyödyllinen ominaisuus jos keskusteluissa halutaan pitää yllä tiettyä perussensuuria. Sanojen suodatus voidaan asettaa erikseen päälle tai pois keskustelukohtaisesti. Suodatettavien sanojen lista on kaikissa keskusteluissa kuitenkin sama. Sanalistan ylläpito kuuluu järjestelmän pääkäyttäjän tehtäviin. Suodatettavan sanan voi katkaista *-merkillä. Jos sanasuodattimeen määritelty sana katkaistaan sen molemmista päistä, suodatin suodattaa ko. sanan myös muiden merkkien keskeltä esim. yhdyssanassa. Isoja kirjaimia ei oteta erikseen huomioon, vaan ne suodatetaan vastaavien pienten kirjainten mukaisesti. Kaikki suodatettavat sanat voidaan siis määritellä pienillä kirjaimilla. Jos keskusteluun kirjoitetaan kielletty sana, joka sisältää sekä pieniä että isoja kirjaimia, suodatin osaa silti suodattaa ko. sanan.

Sanasuodatin ei kuitenkaan ole vedenpitävä, sillä sanan kirjoittamiseen löytyy aina jokin keino, kuten täytemerkkien käyttö kirjainten välissä. Tämä tekee täydellisen sanasuodatuksen toteutuksesta erittäin vaikeaa. Jos julkaisujärjestelmällä ylläpidettävä sivusto on monikielinen (kielimoduuli), niin sanasuodatin sisältää sanalistan jokaiselle kielelle erikseen. Sanasuodatus tapahtuu keskustelua ladattaessa, joten tietokantaan tallennetaan aina alkuperäinen viesti. Tämä mahdollistaa sanasuodatuksen

poistamisen keskustelusta, jolloin kaikki viestit näkyvät alkuperäisessä muodossaan. Tämän ansiosta myös uusien kiellettyjen sanojen lisääminen astuu heti voimaan.

Saman viestin uudelleenlähetys tapahtuu helposti, joko vahingossa tai tahallaan, painamalla selaimen päivitys-nappia. Tällöin selain lähettää viestin kirjoituslomakkeen tiedot uudelleen. Kun järjestelmä huomaa, että käyttäjä yrittää lähettää samaa viestiä uudelleen, se estää viestin tallennuksen ja antaa käyttäjälle virheilmoituksen.

Keskusteluissa voidaan käyttää tarvittaessa käyttäjätunnistusta, jolloin keskusteluun kirjoittaminen vaatii rekisteröitymisen ja kirjautumisen. Tällöin jokaisella keskusteluun kirjoittavalla käyttäjällä on oma nimimerkki. Tämä on varsin yleinen tapa Internetissä käytävissä keskusteluissa ja se tuo mukaan jonkinasteista luottamusta keskustelijan vakavuuteen. Vaikka tällä menetelmällä ei saadakaan estettyä käyttäjiä antamasta vääriä tietoja tai kirjoittamasta ylimääräistä roskaa, se karsii kuitenkin turhia ja etenkin äkillisenä päähän pistoksena kirjoitettavia viestejä. Rekisteröitymisen yhteydessä käyttäjän tulee antaa oma toimiva sähköpostiosoitteensa, sillä tunnukset astuvat voimaan vasta sähköpostiin lähetettävän viestin vahvistamisen jälkeen. Jos käyttäjä vaihtaa myöhemmin sähköpostiosoitettaan, on tunnus aktivoitava uudelleen.

Kirjoitusoikeuden lisäksi myös keskustelun lukemista voidaan rajoittaa. Keskustelutyökalu suunniteltiin toimimaan yhdessä extranet-moduulin kanssa siten, että extranet-sivuun liitettyyn keskusteluun pääsee osallistumaan vain sivuun määritetyt käyttäjäryhmät. Ominaisuus ei ole keskustelutyökalun sisäinen, vaan se vaatii myös extranet-moduulin toimiakseen. Jos keskustelu on liitetty extranet-sivun lisäksi myös normaaliin julkiseen sivuun, siihen voivat osallistua myös muut käyttäjät normaaliin tapaan. Oikeuksista kerrotaan tarkemmin myöhemmin.

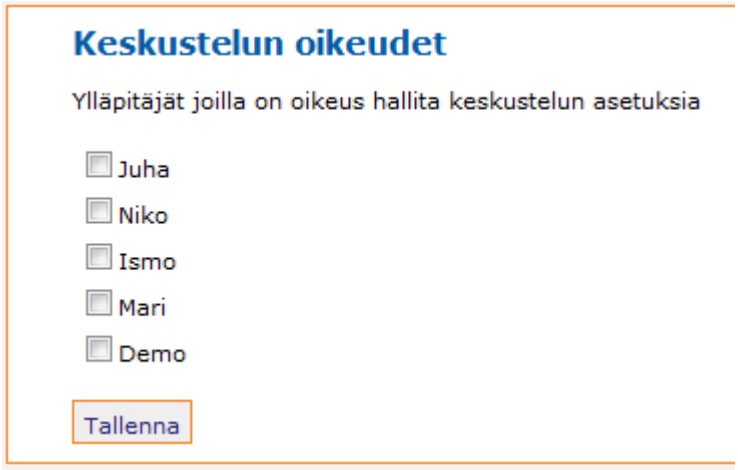
Keskusteluun kirjoitettavien viestien julkaisutapaa voidaan muokata keskustelukohtaisesti. Viestit voidaan julkaista joko suoraan tai ylläpidon vahvistuksen kautta. Lisäksi molemmissa menettelytavoissa voidaan käyttää ihmistodennusta sekä käyttäjän sähköpostiin lähetettävää vahvistuslinkkiä. Ylläpidon kautta julkaistavien viestien avulla saadaan tehokkaimmin karsittua pois kaikki epäsoveliaat viestit. Ominaisuuden haittapuolena on se, että se vaatii aktiivista ylläpitoa, jotta keskustelijoiden viestit julkaistaisiin edes melkein reaaliajassa. Jos tällaiseen ylläpitoon ei ole resursseja,

on järkevämpää käyttää ihmistodennusta ja/tai sähköpostiin lähetettävää vahvistusviestiä. Ihmistodennuksessa käyttäjän on kirjoitettava tekstikenttään kuvaan satunnaisesti generoitu merkkijono ennen viestin lähetystä. Ihmistodennuksen avulla karsitaan mm. haittaohjelmien kirjoittamat mainosviestit. Sähköpostivahvistuksessa käyttäjän kirjoittama viesti julkaistaan vasta sähköpostiin lähetettävän viestin vahvistamisen jälkeen (vahvistuslinkki). Suoraa julkaisua ja varmistustoimintoja käytettäessä ylläpitäjät voivat jälkeenpäin poistaa epäsoveliaat viestit.

Kulmasulkujen suodatus on sisäinen ominaisuus, jota ei voi asettaa pois päältä. Toiminto suodattaa kaikista viesteistä kulmasulut eli suurempi - ja pienempi kuin merkit. Tällä estetään HTML-merkkauskielen ja Java Script -koodin kirjoittaminen viesteihin. Jos viesteissä sallittaisiin HTML:n kirjoitus, tarvittaisiin erillinen lisätoiminto merkkauksen oikeellisuuden tarkistamiseen. Muuten sivun ulkoasu saattaisi helposti rikkoutua virheellisesti kirjoitettujen HTML-elementtien johdosta.

5.3.4 Käyttöoikeudet

Niin kuin aikaisemmissa kohdissa tuli esille, pääkäyttäjillä on täydet oikeudet lisätä, muokata, poistaa sekä ylläpitää keskusteluja. Ylläpitäjille asia on toisenlainen, sillä heille tulee antaa erikseen keskustelun ylläpito-oikeudet em. toimintojen tekemiseen. Keskustelun ylläpito-oikeuden voi myöntää, joko järjestelmän pääkäyttäjä tai keskustelun ylläpito-oikeuden ennestään omaava ylläpitäjä. Kun ylläpitäjä luo uuden keskustelun, hän saa automaattisesti ylläpito-oikeuden siihen. Ylläpito-oikeudet määritellään keskustelujen hallintanäkymän erillisellä työkalulla, joka listaa kaikki järjestelmän ylläpitäjät (ks. Kuva 5.8).



Kuva 5.8. Ylläpito-oikeuksien määrittely.

Niin kuin alaluvussa 5.2 kerrottiin, myös järjestelmän sivuihin liittyy ylläpito-oikeudet. Tämän vuoksi ylläpitäjällä on oikeus lisätä keskusteluja vain sellaisiin sivuihin, joihin hänellä on ylläpito-oikeus. Lisäksi ylläpitäjä saa liittää ylläpitämilleen sivuille vain sellaisia keskusteluja, joihin hänellä on ylläpito-oikeus. Keskustelujen hallintasivuille pääsee kaikki ylläpitäjät oikeuksista riippumatta. Tämän takia ylläpitäjille näytetään vain sellaisten keskustelujen hallintapainikkeet, joihin heillä on ylläpito-oikeus.

	Sivun ylläpito-oikeudet	Keskustelun ylläpito-oikeudet	Oikeudet
1	EI	EI	R/W/-
2	EI	KYLLÄ	R/W/M
3	KYLLÄ	EI	R/W/-
4	KYLLÄ	KYLLÄ	R/W/M

Kuva 5.9. Ylläpitäjän oikeudet julkiseen keskusteluun (R = luku, W = kirjoitus, M = muokkaus).

Kuvassa 5.9 on taulukko, joka sisältää tietyn sivun ja julkisen keskustelun eri oikeuskombinaatiot. Alla selitetään eri tilanteet.

1. Ylläpitäjällä ei ole oikeutta liittää keskustelua sivuun.
2. Ylläpitäjällä on oikeus muokata keskustelua, mutta ei liittää sitä sivulle.
3. Ylläpitäjällä on oikeus muokata sivua, mutta ei liittää keskustelua siihen.
4. Ylläpitäjällä on oikeus muokata keskustelua ja liittää se sivulle.

Kaikilla ylläpitäjillä on oikeus osallistua, eli lukea ja kirjoittaa viestejä kaikkiin julkisiin keskusteluihin. Ylläpidon sisäiseen keskusteluun on osallistumisoikeus vain jos

on ylläpito-oikeus ko. keskusteluun tai on määritelty ylläpitäjäksi sellaiseen sivuun, johon ko. keskustelu on liitettynä.

	Sivun ylläpito-oikeudet	Keskustelun ylläpito-oikeudet	Oikeudet
1	EI	EI	-/-/-
2	EI	KYLLÄ	R/W/M
3	KYLLÄ	EI	R/W/-
4	KYLLÄ	KYLLÄ	R/W/M

Kuva 5.10. Ylläpitäjän oikeudet ylläpidon sisäiseen keskusteluun (R = luku, W = kirjoitus, M = muokkaus).

Kuvassa 5.10 on taulukko, joka sisältää tietyn sivun ja ylläpidon sisäisen keskustelun eri oikeuskombinaatiot. Alla selitetään eri tilanteet.

1. Ylläpitäjällä ei ole oikeutta osallistua keskusteluun.
2. Ylläpitäjällä on täydet oikeudet osallistua ja ylläpitää keskustelua.
3. Ylläpitäjällä on oikeus osallistua keskusteluun, mutta ei ylläpitää sitä.
4. Ylläpitäjällä on täydet oikeudet osallistua ja ylläpitää keskustelua.

Julkisessa näkymässä voidaan lisäksi rajoittaa käyttäjien (ei ylläpitäjien) osallistumisoikeutta keskusteluun. Käyttäjätunnistus, joka voidaan asettaa keskustelukohtaisesti, rajoittaa keskusteluun kirjoittamista. Tällöin kirjoittaminen vaatii rekisteröitymisen ja kirjautumisen järjestelmään. Lukuoikeutta voidaan rajoittaa yhdessä extranet-moduulin kanssa. Alla olevassa taulukossa on havainnollistettu keskustelun lukuja kirjoitusoikeudet eri tilanteissa. Taulukko sisältää eri kombinaatiot saman keskustelun liittämisestä normaaliin sivuun sekä extranet-sivuun. Esimerkissä olevan extranet-käyttäjän ryhmälle on annettu lukuoikeudet extranet-sivuun. Extranet-käyttäjällä, jolle ei ole annettu oikeuksia, on vastaavat oikeudet kuin rekisteröityneellä keskustelijalla.

	Tapaus 1	Tapaus 2	Tapaus 3	Tapaus 4	Tapaus 5
Extranet-sivu	X			x	x
Normaali sivu		X	x	x	x
Käyttäjätunnistus päällä	(x)		x		x
Extranet-käyttäjä	R/W	R/W	R/W	R/W	R/W
Rekisteröitynyt keskustelija	*/*	R/W	R/W	R/W	R/W
Kaikki	*/*	R/W	R/*	R/W	R/*

Kuva 5.11. Keskustelun luku- ja kirjoitusoikeudet julkisen näkymän käyttäjille (R = luku, W = kirjoitus).

Normaaliin julkiseen sivuun tehty liitos on etusijalla extranet-liitokseen verrattuna ja tekee keskustelusta julkisen. Tämän vuoksi yllä olevan kuvan tapauksissa on toisiaan vastaavia lopputuloksia (oikeuksia). Selitykset eri tapauksiin:

1. Keskustelu on liitetty pelkästään extranet-sivuun. Vain kyseiseen sivuun määritetyillä extranet-ryhmillä on oikeus osallistua keskusteluun. Käyttäjätunnistus ei vaikuta tässä tapauksessa kirjoitusoikeuteen, koska kaikilla extranet-käyttäjillä on kirjoitusoikeus oletuksena.
2. Keskustelu on liitetty normaaliin sivuun, jolloin kaikilla Internet-käyttäjillä on oikeus osallistua siihen.
3. Keskustelu on liitetty normaaliin sivuun, mutta siinä käytetään käyttäjätunnistusta. Rekisteröitymättömiltä käyttäjiltä on tällöin estetty keskusteluun kirjoittaminen.
4. Keskustelu on liitetty extranet-sivun lisäksi normaaliin sivuun. Tällöin kaikilla Internet-käyttäjillä on täysi oikeus osallistua keskusteluun, koska se on liitetty myös normaaliin julkiseen sivuun.
5. Keskustelu on liitetty extranet-sivun lisäksi normaaliin sivuun. Lisäksi käytetään käyttäjätunnistusta. Rekisteröitymättömiltä käyttäjiltä on tällöin estetty keskusteluun kirjoittaminen.

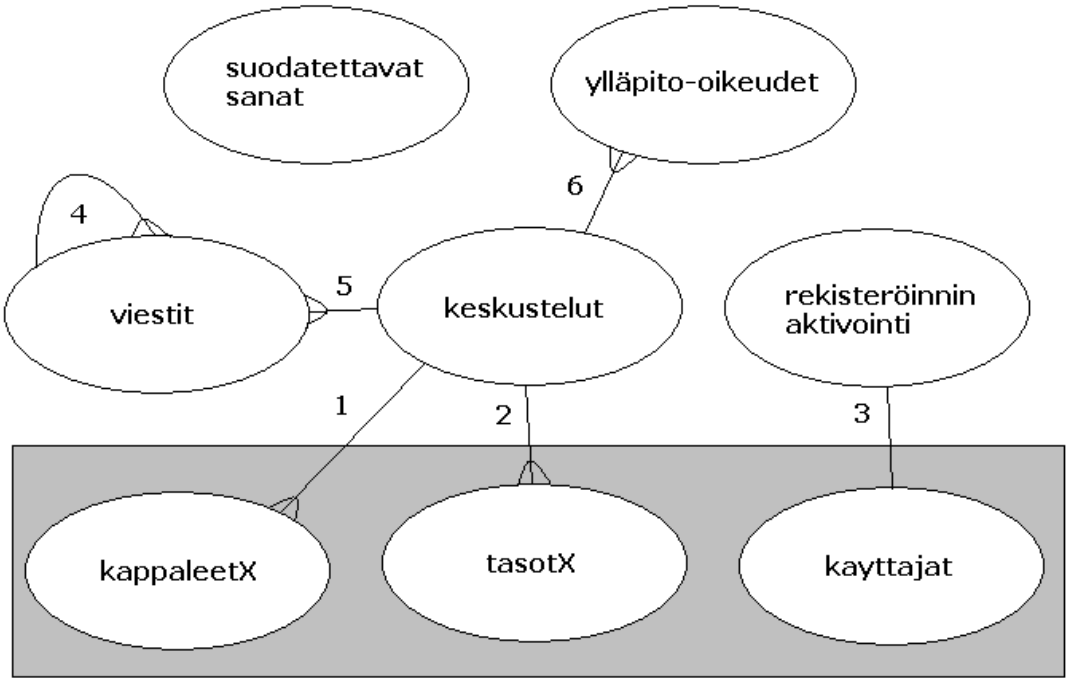
5.3.5 Tietokanta

Keskustelutyökalun myötä julkaisujärjestelmän tietokantaan suunniteltiin uusia tauluja sekä kenttiä jo olemassa oleviin tauluihin. Uusia tauluja määriteltiin yhteensä viisi kappaletta: keskustelut, viestit, suodatettavat sanat, ylläpito-oikeudet sekä rekisteröinnin aktivointi.

Keskustelut-taulua käytetään julkaisujärjestelmän kaikkien keskustelujen tallennuspaikkana. Taulun yksi rivi sisältää yhden keskustelun tiedot ja asetukset, kuten otsikko ja aloituskappale. Viestit-taulua käytetään kaikkien keskustelujen viestien tallennuspaikkana. Taulun yksi rivi sisältää yhden viestin ja siihen liittyvät tiedot, kuten viestin kirjoittaja ja kirjoitusajankohta. Ylläpito-oikeudet-taulun yksi rivi määrittää tietylle ylläpitäjälle oikeuden tiettyyn keskusteluun. Taulun yhdelle riville tallennetaan keskustelun sekä ylläpitäjän yksikäsitteiset tunnistenumerot.

Sanasuodattimen sanat tallennetaan suodatettavat sanat –tauluun. Taulun yksi rivi sisältää yhden sanan sekä sen kieliversion. Rekisteröinnin aktivointi -tauluun sijoitetaan käyttäjän rekisteröitymisen yhteydessä generoitava tunniste, jonka perusteella käyttäjä aktivoi tunnuksensa sähköpostiin lähetettävän linkin avulla. Aktivoimatta jääneet tunnukset poistetaan tietokannasta automaattisesti vuorokauden välein, jolloin myös aktivointitunniste poistetaan.

Alla olevassa kuvassa harmaalla taustalla erotetut taulut kuuluvat ennestään julkaisujärjestelmään. Vanhoihin tauluihin suunniteltiin muutamia uusia kenttiä. Keskustelun liitoksia varten kappale- sekä sivutauluihin luotiin uusi kenttä, joka sisältää keskustelun tunnisteen. Lisäksi käyttäjätauluun suunniteltiin kolme uutta saraketta: keskustelijan nimimerkki, rekisteröitymispäivämäärä sekä tunnuksen aktiivisuus. Tunnuksen aktiivisuudella viitataan siihen, että pystyykö käyttäjä kirjautumaan keskusteluun, jossa käytetään käyttäjätunnistusta.



Kuva 5.12. Tietokantataulut.

Yllä olevien tietokantataulujen väliset yhteydet:

1. Yksi keskustelu voi kuulua moneen kappaleeseen. Yhteen kappaleeseen voi kuulua vain yksi keskustelu.
2. Yksi keskustelu voi kuulua moneen sivuun (jokainen rivi tasot-taulussa vastaa yhtä sivua). Yhteen sivuun voi kuulua vain yksi keskustelu.
3. Yhdelle käyttäjälle voi kuulua vain yksi rekisteröinnin aktivointi. Yksi aktivointi voi kuulua vain yhdelle käyttäjälle.
4. Yhteen viestiin voi kuulua monta lapsiviestiä. Yksi viesti voi kuulua vain yhteen isäntäviestiin (puumalli).
5. Yhteen keskusteluun voi kuulua monta viestiä. Yksi viesti kuuluu vain yhteen keskusteluun.
6. Yhteen keskusteluun voi kuulua monta ylläpito-oikeutta. Yksi ylläpito-oikeus kuuluu vain yhteen keskusteluun.

5.3.6 Valmis suunnitelma

Suunnittelun aikana käytiin lukuisia keskusteluja Hypermedian Oy:n opinnäytetyön vastaavan kanssa sekä palavereita isommalla ryhmällä keskustelutyökalun ominai-

suuksiin liittyen. Lopputuloksena syntyi 25 sivuinen toiminnallinen määrittely. Toteutus päästiin aloittamaan suunnitellun aikataulun mukaisesti. Jo tässä vaiheessa oli selvää, että toteutuksen edetessä toiminnallinen määrittely tulisi vielä muokkaantumaan useita kertoja eri tilanteissa. Lopullinen versio määrittelystä tulisi olemaan valmis vasta toteutuksen valmistuttua. Työn valmistumisen jälkeen määrittely auttaa mahdollisessa jatkokehityksessä sekä ominaisuuden esittelyssä asiakkaille.

5.4 Toteutus

Keskustelutyökalun toiminnallinen määrittelyn sisältämät ominaisuudet jaettiin kolmeen eri vaiheeseen, joista tässä opinnäytetyössä toteutettiin ensimmäinen. Toteutus oli suurimmaksi osaksi koodaamista toiminnalliseen määrittelyyn perustuen, mutta muutamaan otteeseen oli palattava myös suunnittelupöydän ääreen. Seuraavassa kohdassa käydään läpi ensimmäiseen vaiheeseen kuuluvat ominaisuudet. Toteutus itsessään sisälsi kolme eri osaa: tietokannan päivitys, julkisen näkymän sekä ylläpidon toteutus.

5.4.1 Ominaisuudet

Julkisen WWW-näkymän keskustelupalstaan toteutettiin ensimmäisessä vaiheessa yksi asettelumalli. Asettelumalli sisältää linkin erilliselle keskustelusivulle, viimeisen viestin kirjoitusajankohdan sekä keskusteluun kirjoitettujen viestien lukumäärän. Järjestelmään voidaan luoda julkisia keskusteluja, jotka voidaan liittää yhteen tai useampaan sivuun. Erillinen keskustelusivu sisältää elementit: otsikko, aloituskappale, viestit (lista-rakenne), sivutusnavigointi, uuden viestin kirjoituslomake sekä kirjautumislomake.

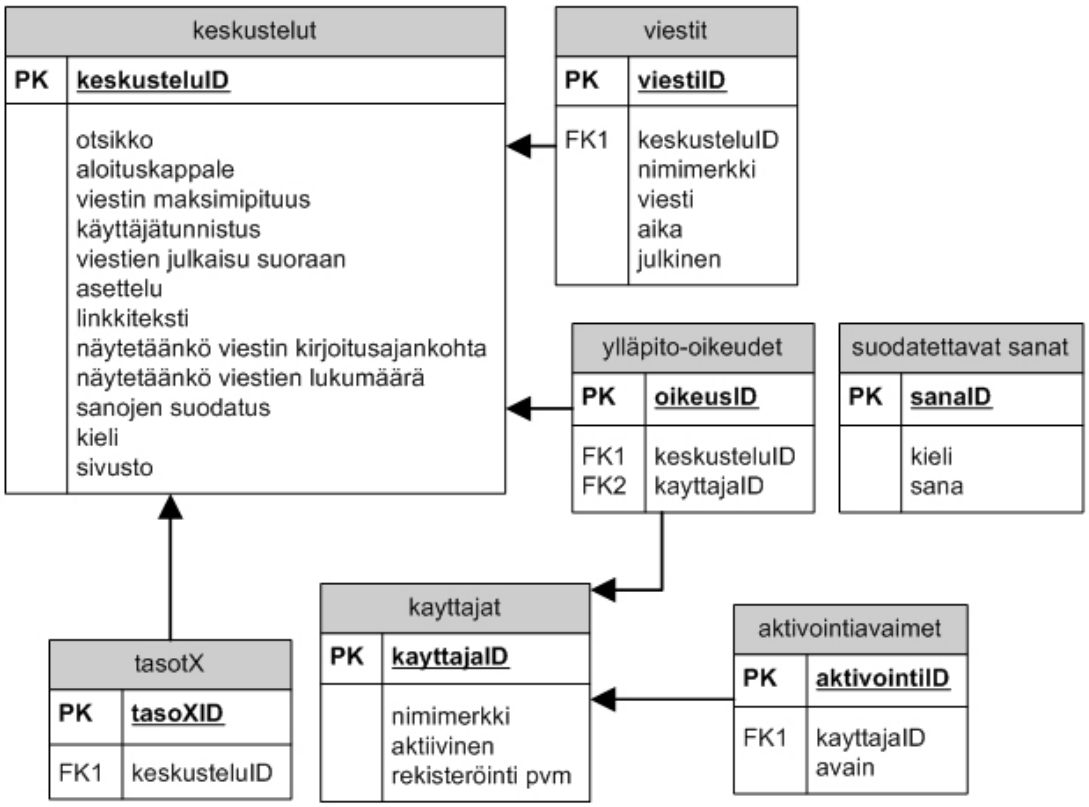
Lisäksi julkinen näkymä sisältää rekisteröitymis-toiminnon sekä rekisteröityneen keskustelijan asetusten muokkausnäkymän. Kohdassa 5.3.3 läpi käydyistä suojausominaisuuksista toteutettiin kulmasulkujen suodatus, kirosanasuodatin, käyttäjätunnistus, ylläpidon kautta julkaistavat viestit sekä saman peräkkäisen viestin kirjoitusesto. Lisäksi ylläpitäjät voivat rajoittaa keskusteluun kirjoitettavien viestien maksimipituutta.

Ylläpidon näkymään toteutettu keskustelujen hallintasivu sisältää seuraavat keskusteluihin liittyvät ominaisuudet: ylläpito, asetusten muokkaus, oikeuksien määrittely, poistaminen ja lisääminen. Yksittäisen asiasivun muokkausnäkymästä voidaan lisätä ja irrottaa keskustelu. Lisäys voi kohdistua olemassa olevaan tai uuteen keskusteluun. Pääkäyttäjänäkymään toteutettiin suodatettavien sanojen määrittely sekä rekisteröityneiden käyttäjien hallintasivu.

5.4.2 Tietokanta

Työn toteutus aloitettiin tietokannan päivittämisellä. Tietokantaan lisättiin suunnitelman mukaiset taulut ja kentät. Luotuihin tauluihin syötettiin käsin keskusteluja, niiden asetuksia ja viestejä, jotta ylläpidon ja julkisen näkymän toteutus olisi helpompaa. Julkaisujärjestelmän tietokantana käytetään Microsoftin SQL Server 2005 tietokantaa, joten uudet taulut määriteltiin Microsoftin SQL Server Management Studio -tietokannanhallintatyökalulla. Graafisen työkalun ansiosta tietotyyppien asetus ym. hoituvat helposti ilman SQL-kielen kirjoittamista.

Koska keskustelutyökalusta toteutettiin vain ensimmäinen vaihe, tietokantaan ei määritelty kenttiä kaikille määrittelyn mukaisille ominaisuuksille. Alla olevassa kuvassa on ensimmäisessä vaiheessa tietokantaan määritetyt keskustelutyökaluun liittyvät taulut ja niiden kentät. Kenttien tarkoituksiin viitataan myöhemmin.



Kuva 5.13. Keskustelumoduuliin liittyvät tietokantataulut.

Taulujen ja kenttien lisäksi tietokantaan määriteltiin säännöllisesti suoritettava SQL-komento. Komento suoritetaan vuorokauden välein, jolloin poistetaan kaikki sellaiset käyttäjät, jotka eivät ole aktivoineet tunnustaan ja joiden rekisteröitymisestä on kulunut yli vuorokausi.

```
DELETE
FROM kayttajat
WHERE viimeisin_kirjautumisajankohta IS NULL
AND aktiivinen = 0
AND (rekisterointi_pvm + 1 < GETDATE())
```

Kuva 5.14. Aktivoimatta jääneiden käyttäjien poisto.

5.4.3 Julkinen näkymä

Ensimmäisen vaiheen mukainen keskustelutyökalu ilmenee WWW-julkaisujärjestelmän julkisessa käyttöliittymässä kahdessa paikassa: asiasivun alareunaan sijoittuvassa keskustelupalstassa sekä erillisessä keskustelusivussa. Keskustelunäkymät toteutettiin käyttäen samaa käyttäjäkontrollia (ascx-tiedosto), joka liitet-

tiin em. sivuihin. Tämä todettiin järkeväksi tavaksi etenkin tulevaisuutta ajatellen, sillä seuraavissa vaiheissa toteutettava keskustelupalstan näkymävaihtoehto, jossa näytetään keskustelun kaikki viestit, vastaa täysin erillisen keskustelusivun näkymää. Tällöin on järkevä käyttää samaa käyttäjäkontrollia myös keskustelupalstassa, jolloin vältetään saman koodin kirjoittaminen kahteen sivuun.

Käyttäjäkontrolli tunnistaa pyyntökohtaisesti kummassa sivussa sitä käytetään. Kontrolli liitetään sivuun alla olevan esimerkin mukaisesti. Kontrollille annetaan ID- sekä Runat-attribuutti. ID saadaan selville käyttäjäkontrollin taustakoodissa ID-metodilla. Ts. riittää että keskustelupalstaan sekä erilliseen keskustelusivuun liitettävässä käyttäjäkontrollissa käytetään eri ID:tä, jolloin ne voidaan erottaa toisistaan. Käyttäjäkontrollin kohdesivuun ei tarvitse koodata erikseen näkyvyyttä säätävää logiikkaa, vaan se voidaan sijoittaa itse käyttäjäkontrolliin. Käyttäjäkontrollin näkymää muokataan sen mukaan, käytetäänkö sitä asiasivun keskustelupalstassa vai erillisessä keskustelusivussa. Näkymää säädellään piilottamalla kontrolliin sijoitettuja elementtejä, kuten otsikko, aloituskappale ja viestit.

```
<%@ Page Language="VB" %>
<%@ Register TagPrefix="omat" TagName="keskustelu" Src="keskustelu.ascx" %>
<html>
 <head><title>Keskustelu</title></head>
 <body>
  <omat:keskustelu runat="server" ID="keskustelusivu" /><br />
  <omat:keskustelu runat="server" ID="keskustelupalsta" />
 </body>
</html>
```

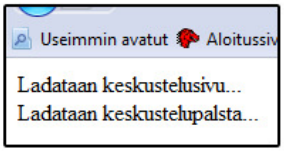
aspx-sivu

```
<%@ Control Language="VB" CodeFile="keskustelu.ascx.vb" Inherits="keskustelu
<asp:Label runat="server" ID="lblKeskustelu" />
```

keskustelu.ascx

```
If Me.ID = "keskustelusivu" Then
    lblKeskustelu.Text = "Ladataan keskustelusivu..."
Else
    lblKeskustelu.Text = "Ladataan keskustelupalsta..."
End If
```

keskustelu.ascx.vb



Kuva 5.15. Käyttäjäkontrollin liittäminen sivuun.

Lisäksi keskustelusivu sisältää kirjautumislomakkeen. Lomake näytetään uuden viestin kirjoituslomakkeen tilalla sellaisten keskustelujen lopussa, jotka käyttävät käyttäjätunnistusta. Lisäksi lomake sisältää linkit rekisteröitymiseen sekä unohtuneen salasanan tilaukseen. Rekisteröitymislomakkeessa kysytään käyttäjän etu- ja sukunimi, nimimerkki sekä käyttäjätunnus, jonka tulee olla toimiva sähköpostiosoite. Kun käyttäjä lähettää rekisteröitymislomakkeen, järjestelmä käy läpi seuraavat tarkistukset.

1. Tarkistetaan onko kaikki kentät täytettyinä. Jos jokin kenttä jää tyhjäksi, järjestelmä antaa virheilmoituksen.
2. Tarkistetaan onko käyttäjätunnus ja nimimerkki vapaa. Jos käyttäjätunnus (sähköpostiosoite) tai nimimerkki on käytössä, järjestelmä antaa virheilmoituksen.
3. Tarkistetaan onko salasana ohjeistetun mittainen (5-20 merkkiä). Jos salasana on liian pitkä tai lyhyt, järjestelmä antaa virheilmoituksen.
4. Salasana kysytään kahteen kertaan kirjoitusvirheiden välttämiseksi. Jos salasanat eivät täsmää toisiaan, järjestelmä antaa virheilmoituksen.

Jos yllä mainitut tarkistukset läpäistään, järjestelmä tallentaa käyttäjän tiedot tietokannan käyttäjätauluun. Käyttäjän tila asetetaan passiiviseksi asettamalla käyttäjän aktiivisuuden ilmaisevan kentän arvoksi epätosi. Seuraavaksi järjestelmä generoi aktivointiavaimen (merkkijono), joka tallennetaan aktivointiavaintauluun yhdessä käyttäjän tunnisteen kanssa. Lopuksi käyttäjän antaman käyttäjätunnuksen mukaiseen sähköpostiosoitteeseen lähetetään viesti, joka sisältää linkin rekisteröitymisen aktivointiin. Linkki sisältää generoidun aktivointiavaimen. Linkkiä klikkaamalla, käyttäjän tila aktivoidaan asettamalla aktiivisuutta ilmaisevan kentän arvoksi tosi. Ennen aktivointia, käyttäjä ei pääse kirjautumaan keskusteluun. Jos käyttäjä myöhemmässä vaiheessa vaihtaa tunnustaan, on aktivointi suoritettava uudelleen. Alla olevassa esimerkissä demonstroidaan avaimen generointi sekä sähköpostin lähetys.

```
Dim strAvain = NewGuid.ToString() ' Generoidaan aktivointiavain
Dim mailMessage As New MailMessage
mailMessage.From = "asiakaspalvelu@kotisivu.com" ' Viestin lähettäjä
mailMessage.To = "matti.meikalainen@osoite.com" ' Viestin vastaanottaja
mailMessage.Subject = "Rekisteröityminen" ' Viestin aihe
' Viestin sisältö
mailMessage.Body = "http://kotisivu.com/reg.aspx?avain=" & strAvain
SmtpMail.SmtpServer = "smtp.kotisivu.com" ' Sähköpostipalvelin
SmtpMail.Send(mailMessage) ' Viestin lähetys
```

Kuva 5.16. Avaimen generointi ja sähköpostin lähetys.

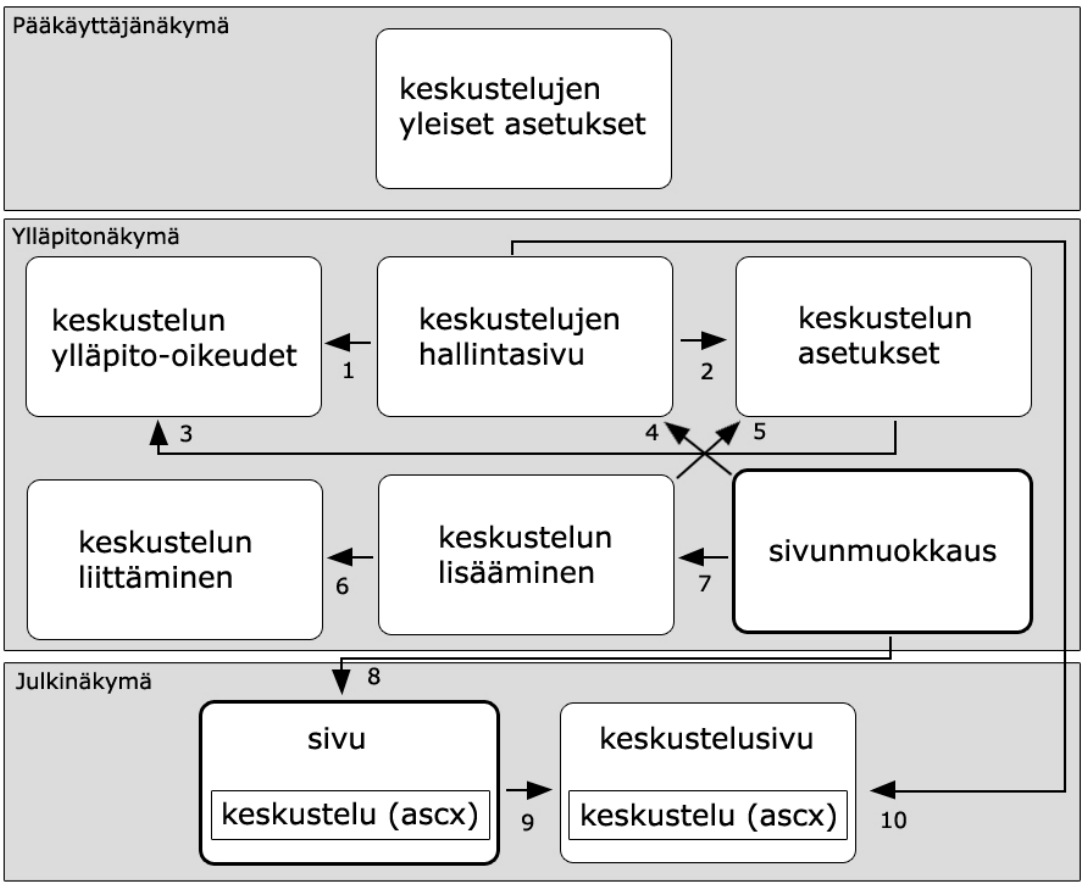
Keskustelusivun rakennuksessa yksi tärkeä osa oli käyttöoikeuksien tarkistus. Käyttöoikeudet käytiin läpi kohdassa 5.3.4. Käyttöoikeuksien toteutus aloitettiin tekemällä kaavio suoritettavista tarkistuksista, jotka käydään läpi kun käyttäjä saapuu kes-

kustelusivulle (ks. Liite 5). Tarkistuksen yhteydessä määritellään luku-, kirjoitus- sekä ylläpito-oikeudet, joista muodostuu neljä eri oikeustasoa. Oikeustason mukaan tiedetään mitä toimintoja sivu saa tarjota kullekin käyttäjälle tiettyyn keskusteluun. Tarkistuksessa otettiin huomioon myös seuraavissa vaiheissa toteutettavat ominaisuudet, kuten ylläpidon sisäiset keskustelut. Kaavion perusteella oli helppo toteuttaa käyttöoikeustarkistuksen tekevä funktio, joka palauttaa käyttäjän oikeustason.

- Taso 0: ei oikeuksia
- Taso 1: lukuoikeudet
- Taso 2: luku- ja kirjoitusoikeudet
- Taso 3: luku-, kirjoitus- ja ylläpito-oikeudet

5.4.4 Ylläpidon näkymä

Toiminnallisen määrittelyn mukaiset suunnitelmat vaativat uusien sivujen toteuttamista sekä muutaman olemassa olevan muokkaamista. Ylläpidon näkymään toteutettiin viisi uutta keskustelujen ylläpitoon liittyvää sivua. Lisäksi pääkäyttäjänäkymään toteutettiin sivu, joka sisältää käyttäjien sekä suodatettavien sanojen hallinnan. Olemassa olevista sivuista jouduttiin muokkaamaan valikoita sekä sivun muokkausnäkymää, joka sisältää keskustelun lisäys- sekä irrotus-toiminnon.



Kuva 5.17. Keskustelumoduuliin liittyvät sivut ja niiden välinen navigointi.

Yllä olevassa kuvassa esitetään kaikki keskustelumoduuliin liittyvät sivut sekä niiden välinen navigointi. Vahvalla reunuksella esitetyt sivut ovat ennestään järjestelmään kuuluneita sivuja. Pääkäyttäjänäkymän keskustelujen yleiset asetukset sisältää aiemmin esitellyt käyttäjien hallinnan sekä suodatettavien sanojen määrittelyn. Ko. sivulta ei siirrytä muille sivuille, vaan asetukset hoidetaan sivun sisäisesti.

Keskustelujen hallintasivulta on pääsy keskustelukohtaisiin asetuksiin (2), ylläpito-oikeuksien määrittelyyn (1) sekä ylläpitoon (10). Keskustelun asetussivua käytetään sekä uuden, että olemassa olevan keskustelun määrittämiseen. Sivunmuokkauksesta voidaan siirtyä suoraan muokkaamaan siihen liitetyn keskustelun asetuksia (4). Asetusnäkymänä käytetään keskustelujen hallintasivua. Tällöin muokattava keskustelu korostetaan erillisellä taustavärillä. Uusi keskustelu voidaan luoda keskustelujen hallintasivulta käsin, jolloin sitä ei liitetä sivuun (2). Toinen vaihtoehto on luoda uusi keskustelu suoraan sivunmuokkauksesta, jolloin keskustelu liitetään automaattisesti ko. sivuun (7, 5 ja 3). Olemassa oleva keskustelu voidaan liittää sivuun suoraan sivunmuokkauksesta käsin (7 ja 6). Kohta 8 tarkoittaa sivun esikatseluun siirtymistä,

joka kuuluu julkaisujärjestelmän perusominaisuuksiin. Kohdassa 9 kuvataan julkinäkymän keskustelupalstasta erilliselle keskustelusivulle siirtymistä.

Ylläpidon käyttöoikeuksien tarkistusta tarvitaan kaikissa keskustelun ylläpitonäkymän sivuissa. Tämän vuoksi oikeuksien tarkistusta varten toteutettiin erillinen luokka, joka sisältää metodit keskustelun sekä sivun ylläpito-oikeuksien tarkistamiseen. Tarkistuksia tarvitaan, koska eri ylläpitosivuille siirryttäessä tietoja kuljetetaan osoiteparametreina, jolloin niitä pystyy muokkaamaan myös käsin.

Keskustelujen ylläpito-oikeudet määritellään käyttämällä kuvan 5.8. mukaista näkymää. Näkymä sisältää kaikki järjestelmään kuuluvat ylläpitäjät sekä valintaruudun jokaisen ylläpitäjän kohdalla. Yhden käyttäjän ylläpito-oikeus yhteen keskusteluun vastaa tietokannan ylläpito-oikeustaulun yhtä riviä, joka sisältää keskustelun sekä käyttäjän yksikäsitteiset tunnisteet (ks. Kuva 5.13).

Seuraavaksi käydään läpi yhden keskeisen ylläpitosivun toteutus. Keskustelujen hallintasivu toteutettiin ”toistimella” eli Repeater-palvelinkontrollilla. Kontrollin avulla voidaan tulostaa tietolähteestä saatava tulosjoukko haluttuun muotoon. Kontrollin sisään voidaan sijoittaa muita palvelinkontrolleja, jonka avulla voidaan rakentaa yhden rivin esitysmuotoon käytettävä näkymä. Repeater-kontrolliin sidotaan datalähde vastaavalla tavalla kuin aiemmin esille tulleessa DataGrid-kontrollissa (ks. Liite 3). Tällä kertaa datalähteenä käytetään DataReader-oliota, joka sisältää suoritetun SQL-kyselyn tulosjoukon. Tulosjoukon sisältämät keskustelut ja niiden tiedot käsitellään yksitellen Repeater-kontrollin ItemDataBound-tapahtumakäsittelijässä. Keskustelut käsitellään otsikon mukaisessa aakkosjärjestyksessä. Järjestyksen määrää SELECT-lauseen ORDER BY -määre.

Keskustelujen hallintasivulla näytetään kaikille ylläpitäjille vähintään keskustelun otsikko. Hallintapainikkeet näytetään vain keskustelun ylläpito-oikeudet omaaville henkilöille. Painikkeet on sijoitettu Placeholder-kontrollin sisään, jolloin ne voidaan kätevästi piilottaa rivikohtaisesti. Jokaisen keskustelun kohdalla tarkistetaan käyttäjän ylläpito-oikeudet ko. keskusteluun, jonka mukaan painikkeiden näkyvyyttä säädellään. Lisäksi jokaiseen hallintapainikkeeseen sijoitetaan linkki, joka sisältää keskustelun tunnisteen osoiteparametrina. Liitteen 4. esimerkissä demonstroidaan yksin-

kertaistettuna keskustelujen hallintasivun Repeater-kontrollin toiminta. Lopputulokseksi saadaan kuvan 5.4 mukainen näkymä.

6 YHTEENVETO

Opinnäytetyö antoi hyvää kokemusta työelämän ohjelmistoprojektin läpiviennistä, sen suunnittelusta ja toteutuksesta. Toiminnallisen määrittelyn avulla työn tekninen toteutus oli sujuvaa ja vastaan tulleet muutokset olivat helposti hallittavia. Määrittely saatiin lopulliseen kuntoon työn teknisen toteutuksen valmistuttua. Määrittelydokumenttia voidaan hyödyntää ominaisuuksien esittelyssä asiakkaille sekä seuraavien toteutusvaiheiden läpivientiin.

Ohjelmistoprojektin tuoman kokemuksen lisäksi opinnäytetyön tekeminen antoi minulle henkilökohtaisesti osaamista Microsoftin Internet-ohjelmointitekniikasta ja kehitysympäristöstä, syventävää tietoa relaatiotietokannoista sekä Web 2.0 -ajan toimintamalleista. Tietoa ja opastusta tarjosi Hypermedia Oy:n henkilökunta, jota täydensi itseopiskelu eri lähteitä käyttäen.

Työlle asetetut välitavoitteet pitivät ja työ saatiin kokonaisuudessaan valmiiksi aikataulun mukaisesti vuoden 2008 loppuun mennessä. Keskusteluominaisuuden ensimmäinen toteutusvaihe jää odottamaan ensimmäistä todellista testiään eli asiakasta, joka haluaa julkaisujärjestelmän osaksi keskusteluominaisuuden. Lisäksi ensimmäinen todellinen käyttötilanne tuo tullessaan varmasti useita parannusehdotuksia.

LÄHTEET

Esposito, D. 2005. Introducing Microsoft ASP.NET 2.0. Washington. Microsoft Press.

Hintikka, A. 2007. Web 2.0 - johdatus internetin uusiin liiketoimintamahdollisuuksiin. Helsinki. Tietoyhteiskunnan kehittämiskeskus ry.

Inkinen, V. 2003. ASP.NET. Jyväskylä. Docendo Finland Oy.

Lahtonen, T. 2002. SQL. Jyväskylä. Docendo Finland Oy.

Platt, D. 2001. Microsoft .NET - Uudet ominaisuudet. Helsinki. Edita Oyj.

Scoble, R. & Israel, S. 2008. Blogit ja bisnes. Helsinki. Basam Books Oy.

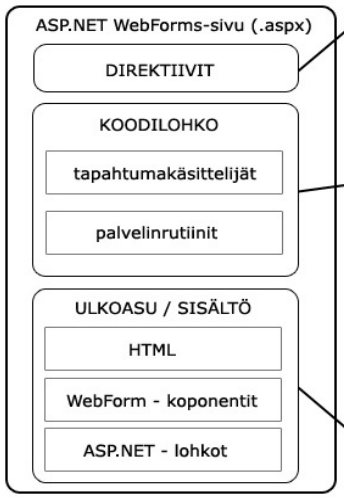
Tirronen, M. 2008. Web 2.0 Verkon numerologia. Helsinki. BTJ Finland Oy.

W3Schools. 2008. AJAX Tutorial [verkkodokumentti]. [Viitattu 29.11.2008]. Saatavissa: http://www.w3schools.com/Ajax/.

Wikipedia. 2008a. Microsoft SQL Server [verkkodokumentti]. [Viitattu 2.12.2008]. Saatavissa: http://en.wikipedia.org/wiki/Microsoft_SQL_Server.

Wikipedia. 2008b. Transact SQL [verkkodokumentti]. [Viitattu 2.12.2008]. Saatavissa: http://en.wikipedia.org/wiki/Transact-SQL.

LIITELUETTELO




```
<%@ Page Language="VB" %>

<script runat="server">
    Public strOtsikko As String
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
        'Alustetaan sivun elementit
        strOtsikko = "Tervetuloa ASP.NET-sivuille!"
        btnColor.Text = "Vaihda taustaväriä"
        body.Attributes.CssStyle.Add("background-color", ddlColors.SelectedValue)
        If Not Page.IsPostBack Then 'Värit pudotusvalikkoon
            ddlColors.Items.Add("white")
            ddlColors.Items.Add("blue")
            ddlColors.Items.Add("red")
        End If
    End Sub
    Private Sub btnChangeBackgroundColor_Click(ByVal sender As Object, ByVal e As System.EventArgs
        body.Attributes.CssStyle.Add("background-color", ddlColors.SelectedValue)
    End Sub
</script>
<html>
<head runat="server">
    <title>Tervetuloa!</title>
</head>
<body runat="server" id="body">
    <form id="form1" runat="server">
        <h1><%=strOtsikko%></h1>
        <asp:Button runat="server" ID="btnColor" />
        <asp:DropDownList runat="server" ID="ddlColors" AutoPostBack="true"/>
    </form>
</body>
</html>
```

LIITE 1

```
<%@ Application Language="VB" %>

<script runat="server">

    Sub Application_Start(ByVal sender As Object, ByVal e As EventArgs)
        ' Suoritetaan sovelluksen käynnistyksen yhteydessä
        Application("istunnot") = 0
        Application("lopetetut_istunnot") = 0
        Application("pyynnot") = 0
    End Sub

    Sub Application_End(ByVal sender As Object, ByVal e As EventArgs)
        ' Suoritetaan sovelluksen sammutuksen yhteydessä
        ' Esim. tallennetaan laskureiden arvot tietokantaan...
    End Sub

    Sub Session_Start(ByVal sender As Object, ByVal e As EventArgs)
        ' Suoritetaan jokaisen uuden istunnon alkaessa
        Application("istunnot") += 1
    End Sub

    Sub Session_End(ByVal sender As Object, ByVal e As EventArgs)
        ' Suoritetaan jokaisen istunnon päätteeksi
        Application("lopetetut_istunnot") += 1
    End Sub

    Protected Sub Application_BeginRequest(ByVal sender As Object, ByVal
        ' Suoritetaan jokaisen pyynnön yhteydessä
        Application("pyynnot") += 1
    End Sub
</script>
```

Global.asax

```
Sivuja ladattu yhteensä: <%=Application("pyynnot")%><br />
Istuntojen määrä: <%=Application("istunnot")%> <br />
Lopetetut istunnot: <%=Application("lopetetut_istunnot")%> <br />
```

laskurit.aspx

```
<?xml version="1.0" encoding="utf-8" ?>
<henkilosto>
  <henkilo sukupuoli="mies">
    <etunimi>Matti</etunimi>
    <sukunimi>Meikäläinen</sukunimi>
    <puhelinnumero>0401234567</puhelinnumero>
  </henkilo>
  <henkilo sukupuoli="nainen">
    <etunimi>Tarja</etunimi>
    <sukunimi>Turunen</sukunimi>
    <puhelinnumero>0501234567</puhelinnumero>
  </henkilo>
  <henkilo sukupuoli="nainen">
    <etunimi>Maija</etunimi>
    <sukunimi>Meikäläinen</sukunimi>
    <puhelinnumero>040654321</puhelinnumero>
  </henkilo>
  <henkilo sukupuoli="mies">
    <etunimi>Ville</etunimi>
    <sukunimi>Virtanen</sukunimi>
    <puhelinnumero>040111222</puhelinnumero>
  </henkilo>
</henkilosto>
```

henkilosto.xml

```
<style type="text/css">
   .otsikko {
      font-weight:bold;
      background-color:Gray;
      text-align:center;
   }
</style>
<asp:DataGrid runat="server"
              id="dgrPuhelinluettelo"
              AutoGenerateColumns="true">
</asp:DataGrid>
```

xml.aspx

```
Dim dsXML As New System.Data.DataSet
dsXML.ReadXml(Server.MapPath("henkilosto.xml"))
dgrPuhelinluettelo.DataSource = dsXML
dgrPuhelinluettelo.DataBind()
dgrPuhelinluettelo.HeaderStyle.CssClass = "otsikko"
```

xml.aspx.vb

etunimi	sukunimi	puhelinnumero	sukupuoli
Matti	Meikäläinen	0401234567	mies
Tarja	Turunen	0501234567	nainen
Maija	Meikäläinen	040654321	nainen
Ville	Virtanen	040111222	mies

```
<table>
 <tr>
  <th>Otsikko</th>
  <th>Ylläpito</th>
  <!-- jne... -->
 </tr>
 <asp:Repeater runat="server" ID="rptDiscussions">
  <ItemTemplate>
   <tr>
    <td>
     <asp:Label runat="server" ID="lblOtsikko" />
    </td>
    <asp:PlaceHolder Runat="server" ID="plhPainikkeet">
     <td>
      <asp:HyperLink Runat="server" ID="hprYllapito">
       <img src="/yllapito.gif" border="0" />
      </asp:HyperLink>
     </td>
     <!-- jne... -->
    </asp:PlaceHolder>
   </tr>
  </ItemTemplate>
 </asp:Repeater>
</table>
```

keskustelut.aspx

```
sqlCmd.CommandText = "SELECT * FROM keskustelut ORDER BY otsikko"
sqlCon.Open()
Dim rdrKeskustelut As SqlDataReader = sqlCmd.ExecuteReader
rptDiscussions.DataSource = rdrKeskustelut
rptDiscussions.DataBind()
sqlCon.Close()
```

keskustelut.aspx.vb (page load)

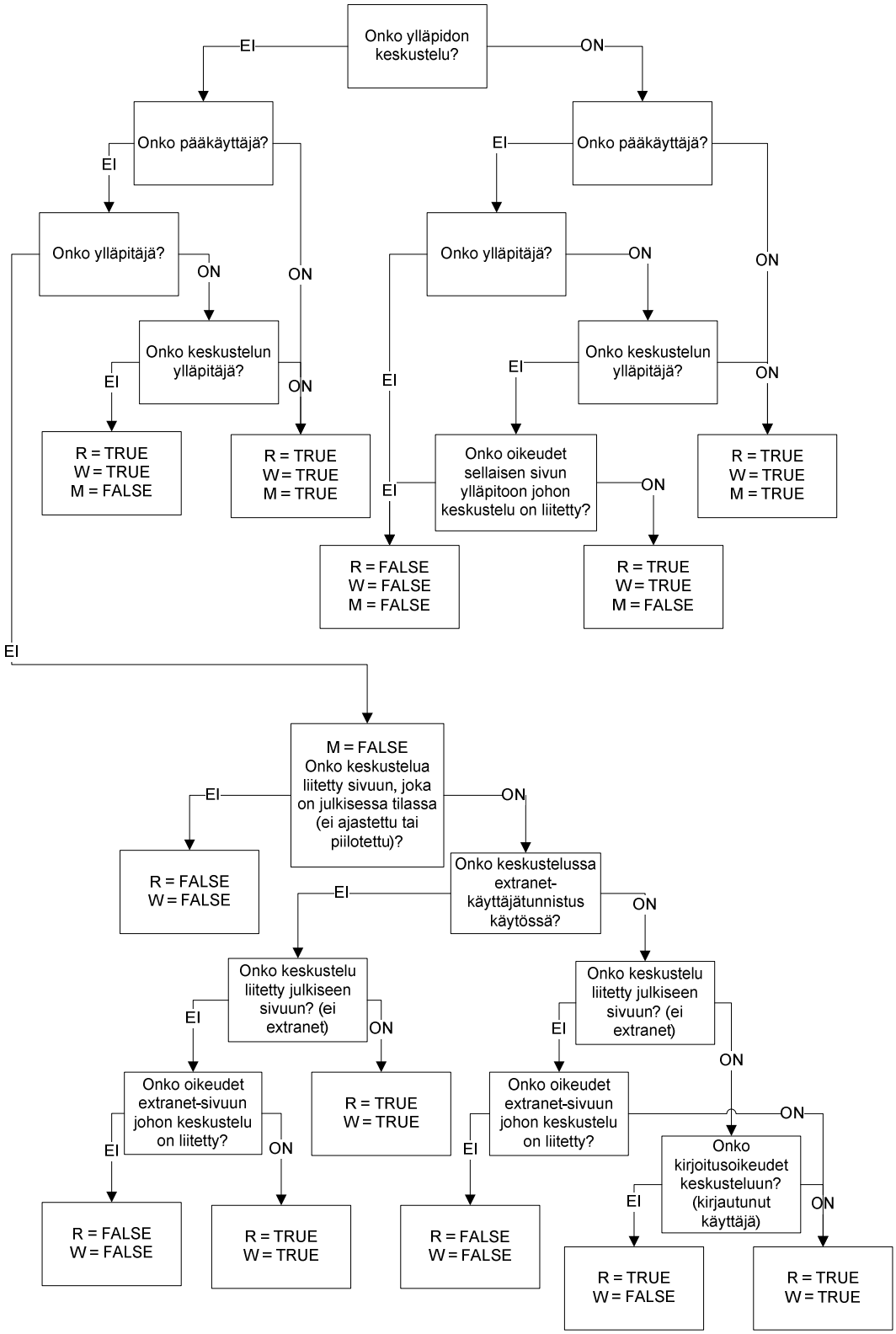
```
Protected Sub rptDiscussions_ItemDataBound(ByVal sender As Objec
    Dim strOtsikko As String = e.Item.DataItem("otsikko")
    Dim intID = e.Item.DataItem("keskusteluID")

    Dim lblOtiskko As Label
    lblOtiskko = e.Item.FindControl("lblOtsikko")
    Dim hprYllapito As HyperLink
    hprYllapito = e.Item.FindControl("hprYllapito")
    Dim plhPainikkeet As PlaceHolder
    plhPainikkeet = e.Item.FindControl("plhPainikkeet")

    lblOtiskko.Text = strOtsikko
    hprYllapito.NavigateUrl = "keskustelusivu.aspx?id=" & intID

    'Hallintapainikkeiden näkyvyys
    If keskustelu.getOikeudet(intKayttajaID, intID) Then
        plhPainikkeet.Visible = True
    Else
        plhPainikkeet.Visible = False
    End If
End Sub
```

keskustelut.aspx.vb (item data bound)


Onko ylläpidon keskustelu?
EI
ON
Onko pääkäyttäjä?
EI
ON
Onko ylläpitäjä?
EI
ON
Onko keskustelun ylläpitäjä?
EI
ON
R = TRUE
W = TRUE
M = FALSE
R = TRUE
W = TRUE
M = TRUE
Onko pääkäyttäjä?
EI
ON
Onko ylläpitäjä?
EI
ON
Onko keskustelun ylläpitäjä?
EI
ON
Onko oikeudet sellaisen sivun ylläpitoon johon keskustelu on liitetty?
EI
ON
R = FALSE
W = FALSE
M = FALSE
R = TRUE
W = TRUE
M = FALSE
R = TRUE
W = TRUE
M = TRUE
EI
M = FALSE
Onko keskustelua liitetty sivuun, joka on julkisessa tilassa (ei ajastettu tai piilotettu)?
EI
ON
R = FALSE
W = FALSE
Onko keskustelussa extranet-käyttäjätunnistus käytössä?
EI
ON
Onko keskustelu liitetty julkiseen sivuun? (ei extranet)
EI
ON
Onko oikeudet extranet-sivuun johon keskustelu on liitetty?
EI
ON
R = FALSE
W = FALSE
R = TRUE
W = TRUE
R = TRUE
W = TRUE
Onko keskustelu liitetty julkiseen sivuun? (ei extranet)
EI
ON
Onko oikeudet extranet-sivuun johon keskustelu on liitetty?
EI
ON
R = FALSE
W = FALSE
Onko kirjoitusoikeudet keskusteluun? (kirjautunut käyttäjä)
EI
ON
R = TRUE
W = FALSE
R = TRUE
W = TRUE