Jani Pajunen

Lähdekoodin dokumentointi ja Web-kehityksessä käytettävät tekniikat

Tapaustutkimus: Notta for Home -kodinohjausjärjestelmän Valvomo-sovelluksen lähdekoodin dokumentointi

Opinnäytetyö
Syksy 2009
Tekniikan yksikkö, Seinäjoki
Tietotekniikan koulutusohjelma
Ohjelmistotekniikan suuntautumisvaihtoehto

Seinäjoen ammattikorkeakoulu
SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

SEINÄJOEN AMMATTIKORKEAKOULU

Opinnäytetyön tiivistelmä

Koulutusyksikkö: Tekniikan yksikkö
Koulutusohjelma: Tietotekniikan koulutusohjelma
Suuntautumisvaihtoehto: Ohjelmistotekniikan suuntautumisvaihtoehto

Tekijä: Jani Pajunen

Työn nimi: Lähdekoodin dokumentointi ja Web-kehityksessä käytettävät tekniikat

Ohjaaja: Petteri Mäkelä

Vuosi: 2009 Sivumäärä: 38 Liitteiden lukumäärä: 0

Tämän opinnäytetyön tarkoituksena on esitellä yleisesti web-sovelluskehityksessä käytettäviä tekniikoita, sekä ohjelmistojen lähdekoodin dokumentointia ja lähdekoodin dokumentoinnissa käytettäviä työkaluja. Erityistä huomiota kiinnitetään Google Web Toolkit -sovelluskehykseen ja Javadoc-työkalun määritelmien mukaiseen lähdekoodin dokumentointiin.

Opinnäytetyössä on tapaustutkimuksena Notta for Home -kodinohjausjärjestelmän Valvomo-sovelluksen dokumentointiprojekti. Projektissa toteutettiin sovelluksen lähdekoodin dokumentointi Javadoc-työkalun määritelmien mukaisten dokumenttikommenttien avulla.

Opinnäytetyö on suunnattu kaikille, jotka ovat kiinnostuneita Web-kehityksessä käytettävistä tekniikoista. Työn lukija saa perustiedot eri tekniikoiden historiasta ja niiden käyttötarkoituksesta. Lisäksi lukija pääsee perehtymään Google Web Toolkit -ohjelmistokehyksen mahdollisuuksiin ja sovelluskehyksen käyttöön NetBeans-kehitysympäristön yhteydessä. Lukija saa myös lähtökohdan Javadoc-työkalun käyttöön.

Asiasanat: web-sovelluskehitys, Google Web Toolkit, kodinohjausjärjestelmä

SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

Thesis abstract

Faculty: School of Technology
Degree programme: Information Technology
Specialisation: Software Engineering

Author: Jani Pajunen

Title of thesis: Source documentation and the techniques used in web development
Supervisor: Petteri Mäkelä

Year: 2009 Number of pages: 38 Number of appendices: 0

The purpose of this thesis is to give an overview of the web development techniques and tools used for software source documentation. Special attention is paid to the Google Web Toolkit -framework and the Javadoc tool and its comment syntax specification.

Documenting the source code of the Notta for Home Control Center is used as a case study in this thesis. The documentation of the source was done by the specification of Javadoc doc comments.

This thesis is aimed at anyone who is interested in technologies used in web development. The reader will get basic knowledge of the history of web technologies and their usage. In addition the reader will get basic knowledge of the possibilities of Google Web Toolkit and its usage within the NetBeans integrated development environment. A starting point for using the Javadoc tool is also given.

Keywords: web development, Google Web Toolkit, home control system

SISÄLTÖ

Käytetyt termit ja lyhenteet

HTML	.HyperText Markup Language, web-sivujen merkintäkieli
XHTML	Extensible HyperText MarkupLanguage, xml-kielellä kuvattu web-sivujen merkintäkieli
CSS	Cascading Style Sheets, tyyliohje
Google Web Toolkit	Googlen kehittämä sovelluskehys

(Pfaffenberger, Schafer, White & Karow 2004, 5-6, 15-16.) (Dwyer 2008, 5.)

Kuvio- ja taulukkoluettelo

1 TYÖN KUVAUS

1.1 Työn lähtökohdat ja tavoitteet

Opinnäytetyön tavoitteena on esitellä Web-sovelluksissa käytettäviä tekniikoita, erityisesti Google Web Toolkit -sovelluskehystä, ja tarjota tietoa eri tekniikoiden tarjoamista mahdollisuuksista ja ominaisuuksista. Työssä kerrotaan myös Google Web Toolkit -sovelluskehyksen käytöstä NetBeans-kehitysympäristön osana ja selostetaan Javadoc-työkalun dokumenttikommentit ja itse työkalun käyttö NetBeans-kehitysympäristössä.

Tapaustutkimuksessa käsitellyssä projektissa on ollut tavoitteena dokumentoida Notta for Home -kodinohjausjärjestelmän Valvomo-sovelluksen lähdekoodit dokumenttikommenteilla.

1.2 Työn rakenne

Työn toisessa luvussa esitellään ohjelmistoprojektin dokumentointia, ja siinä käytettäviä työkaluja. Kolmannessa luvussa esitellään erilaisia web-sovelluskehitystekniikoita. Neljäs luku esittelee Google Web Toolkit -sovelluskehystä ja sen käyttöä NetBeans-kehitysympäristön kanssa. Viidennessä luvussa käsitellään tapaustutkimusta ja kootaan työn tulokset ja työn aikana tehdyt havainnot.

2 OHJELMISTOPROJEKTIN DOKUMENTOINTI

Ohjelmistoprojektissa tulisi olla vähintään projektisuunnitelma sekä toiminnallinen ja tekninen määrittely. Projektisuunnitelma kuvaa projektin tavoitteita, aikataulua ja käytettäviä resursseja. Toiminnallinen määrittely kuvaa sovellukselle asetettuja vaatimuksia, käyttöympäristöä, tietosisältöä sekä liittymiä muihin järjestelmiin. Tekninen määrittely kuvaa näiden ominaisuuksien teknistä toteutusta. (Haikala & Märijärvi, 2002, 50, 238, 78-81,83-84.)

Myös lähdekoodin dokumentointi on tärkeää, koska ilman dokumentointia sovelluksen toiminnan selvittäminen jälkikäteen on vaikeaa (Vesterholm, M. & Kyppö, J. 2006, 348). Tapaustutkimuksen Valvomo-sovelluksen tapauksessa lähdekoodin dokumentoinnin tärkeys korostuu, koska sovelluksen lähdekoodit on tarkoitus julkaista avoimena lähdekoodina.

2.1 Javadoc-työkalu ja dokumentointikommentit

Javadoc on työkalu, jolla tuotetaan lähdekoodiin kirjoitetuista dokumentointikommenteista sovelluksen toimintaa ja rajapintoja kuvaavia dokumentteja. Dokumentointikommenteista käytetään myöhemmin tässä työssä selkeyden vuoksi termiä kommentti. Oletuksena Javadoc-työkalu tuottaa HTML-muotoisia dokumentteja, mutta Javadoc-työkalua voidaan myös laajentaa Doclet API rajapinnan kautta tuottamaan dokumentaatiota myös muihin formaatteihin, kuten XML- tai RTF-tiedostoja. (Javadoc 5.0 Tool, [Viitattu 2.11.2009].)

Javadoc-työkalua voidaan käyttää sekä komentoriviltä annetuilla komennoilla, että sovelluskehyksen sisäänrakennetuilla työkaluilla. Työkalua käytetään komentoriviltä alla olevan esimerkin mukaisesti.

javadoc [optiot] [pakkaukset tai tiedostonimet.java]*

Tärkeimmät optiot on esitelty alla olevassa taulukossa.

TAULUKKO 1. Javadoc-työkalun optioita

Optio	*Option parametri*	*Kuvaus*
-public-		*Käsittelee vain kyseisen näkyvyysmääreen omaavat luokat ja attribuutit*
-private		
-protected		
-sourcepath	*Hakemistot tai tiedostot ;-merkillä erotettuna*	*Määrittelee hakemistot tai tiedostot, joissa olevat kommentit otetaan mukaan*
-d	*Kohdehakemisto*	*Hakemisto, johon dokumentaatio luodaan*

Kommenteilla voidaan dokumentoida Javan luokkia, metodeja ja attribuutteja. Kommentti sijoitetaan juuri ennen kutakin dokumentoitavaa kohdetta. Kommentti aloitetaan /**-merkinnällä. Kommentin jokaiselle riville merkitään *-merkki, joka asemoidaan aloitusmerkinnän ensimmäisen *-merkin kohdalle. Alkaen Javan versiosta 1.4 kommentin keskellä olevat *-merkit eivät ole pakollisia, mutta suositeltavia selkeyden vuoksi. Kommenttirivien pituus tulisi rajata 80 merkkiin, jotta rivit eivät mene sekaisin. Kommentti lopetetaan */-merkinnällä. (How to Write Doc Comments for the Javadoc Tool, [Viitattu 6.11.2009].)

```
/**
 * Esimerkki Javadoc-kommentin rakenteesta.
 */
```

Kommentin sisältö koostuu dokumentoitavan kohteen kuvauksesta (kuvausosio) sekä @-merkeillä alkavista tägeistä (tägiosio). Kuvausosion ja tägiosion välille jätetään yksi tyhjä rivi, ja kuvausosio päättyy ensimmäiseen @-merkkiin. Kuvauso-

sion sisälle voidaan kuitenkin merkitä tägejä, jotka on ympäröity aaltosuluilla (ks. TAULUKKO 2). Kuvauksessa voidaan tarpeen mukaan käyttää HTML-merkintäkielen tägejä muotoiluun. Esimerkiksi, mikäli kuvaukseen tulee useampia kappaleita, ne erotetaan <p>-HTML-tägillä. (How to Write Doc Comments for the Javadoc Tool, [Viitattu 6.11.2009].)

Kuvausosion ensimmäisen lauseen tulee olla tiivis yhteenveto kommentoitavan kohteen tarkoituksesta ja toiminnasta. Yhteenveto päättyy ensimmäiseen pisteeseen, jonka jälkeen on joko välilyönti, tabulaattori tai rivinvaihto. Mikäli on tarpeen merkitä piste keskelle yhteenvetoa, voidaan välilyönti korvata HTML-merkinnällä   tai HTML-kommenttimerkintöjen sisälle lisätyllä välilyönnillä. (How to Write Doc Comments for the Javadoc Tool, [Viitattu 6.11.2009].)

TAULUKKO 2. Listaus kommenteissa käytettävistä tägeistä

Tägi	*Dokumentoitava kohde*	*Käyttötarkoitus/sisältö*
@author	*Luokat*	*Tekijä*
{@code}	*Luokat, metodit ja attribuutit*	*Tägi, jolla voidaan korvata HTML-kielen <code>-tägit*
{@docRoot}	*Luokat, metodit ja attribuutit*	*Dokumentin suhteellinen polku*
@deprecated	*Luokat, metodit ja attribuutit*	*Merkitsee dokumentoinnin kohteen vanhentuneeksi*
@exception	*Metodit*	*Synonyymi @throws-tägille*
{@inheritDoc}	*Metodit*	*Perii dokumentoinnin ylemmältä tasolta*
{@link}	*Luokat, metodit ja attribuutit*	*Kuvausosion tai tägin kuvauksen sisällä oleva linkki*
{@linkplain}	*Luokat, metodit ja attribuutit*	*Kuvausosion tai tägin kuvauksen sisällä oleva muotoilematon linkki*
{@literal}	*Luokat, metodit ja attribuutit*	*Kuvausosion tai tägin kuvauksen sisällä oleva teksti, jota ei tulkita HTML-kieliseksi*
@param	*Metodit*	*Kuvaus metodin tai konstruktorin parametrille*
@return	*Metodit*	*Kuvaus paluuarvolle*
@see	*Luokat, metodit ja attribuutit*	*"See also"-kohtaan dokumenttia lisättävä linkki*
@serial	*Attribuutit*	*Sarjallistettavissa olevan attribuutin kuvaus*
@serialData	*Attribuutit*	*Sarjallistettavissa olevan attribuutin kuvaus*
@serialField	*Attribuutit*	*Sarjallistettavissa olevan attribuutin kuvaus*
@since	*Luokat, metodit ja attribuutit*	*JDK-versio, josta alkaen dokumentoinnin kohde on kuulunut luokkakirjastoon*
@throws	*Metodit*	*Kuvaus mahdollisesta poikkeuksesta*
{@value}	*Attribuutit*	*Staattisen vakion arvo*
@version	*Luokat*	*Versiomerkintä*

Taulukko 1 kuvaa dokumenttikommenteissa käytettäviä tägejä ja niiden käyttökohteita. Tägejä voidaan myös luoda lisää tarpeen mukaan Javadoc-työkalun-tag -komentoriviparametrilla. Selkeyden vuoksi käyttäjän luomat tägit on rajattu pois tämän työn aihepiiristä. Luokkien ja rajapintojen dokumentoinnissa voidaan käyttää tägejä @author, @version, @see, @since ja @deprecated. (How to Write Doc Comments for the Javadoc Tool, [Viitattu 6.11.2009].)

Javadoc-työkalu muodostaa @see-tägistä linkin dokumentaation ”See also”-kohtaan alla olevan esimerkin mukaisesti.

```
@see ExampleClass#exampleMethod(String) exampleMethod
```

Esimerkin tapauksessa dokumentin ”See also”-osiossa olisi linkki, jonka tekstinä on exampleMethod, ja joka osoittaa ExampleClass-luokan, parametrina String-tyyppisen muuttujan ottavan, exampleMethod-metodin dokumentaatioon. Työkalu muodostaa samalla syntaksilla myös {@link}- ja {@linkplain}-tägeistä linkit, sillä erotuksella, että nämä näytetään suoraan kuvauksen sisällä. (Javadoc tool reference guide. [Viitattu 8.11.2009].)

Metodeja ja konstruktoreita dokumentoitaessa voidaan käyttää @see-, @since-, @deprecated-tägejä. Lisäksi kaikki metodin tai konstruktorin parametrit tulee dokumentoida @param-tägillä alla olevan esimerkin mukaisesti.

```
@param length          a float value representing length of the object
```

@param-tägin jälkeen merkitään parametrin nimi, jonka jälkeen kirjoitetaan parametrin kuvaus. Kuvaus tulee aloittaa substantiivilla, joka kuvaa parametrin tietotyyppiä, int-tyyppisen parametrin tietotyyppi jätetään yleensä mainitsematta. Tietotyyppi kirjoitetaan pienellä alkukirjaimella. Kuvauksen eteen voidaan lisätä välilyöntejä, jotta kuvakset ovat samassa linjassa. Kuvaus päätetään pisteeseen vain jos se sisältää useampia lauseita. (How to Write Doc Comments for the Javadoc Tool, [Viitattu 6.11.2009].)

Jos metodi palauttaa arvon, tulee se dokumentoida @return-tägillä. @return-tägiä koskevat samat ohjeet kuin aiemmin käsiteltyä @param-tägiäkin. (How to Write Doc Comments for the Javadoc Tool, [Viitattu 6.11.2009].)

Mikäli metodi voi heittää poikkeuksia, ne täytyy dokumentoida @exeption- tai @throws-tägillä. @exeption- tai @throws-tägit muotoillaan seuraavan esimerkin mukaisesti.

@tägi poikkeuksen_nimi	kuvaus tilanteesta joka aiheuttaa poikkeuksen

Jos poikkeuksia on useita, ne luetellaan poikkeuksen nimen mukaan aakkosjärjestyksessä. (How to Write Doc Comments for the Javadoc Tool, [Viitattu 6.11.2009].)

3 WEB-TEKNIIKAT

Tapaustutkimuksen projektissa dokumentoinnin kohteena oleva Valvomo rakentuu monesta eri web-tekniikasta. Vaikka Google Web Toolkit -sovelluskehystä käytettäessä varsinainen ohjelmointi tehdään pääasiassa Java-kielellä, valmis sovellus rakentuu seuraavaksi esiteltävien tekniikoiden pohjalle.

3.1 HTML

Tim Berners Lee määritteli Standard Generalized Markup Language (SGML) -kielellä HyperText Markup Language (HTML) -merkkauskielen. SGML-kielen perusideana on erottaa tiedon rakenteen määrittely tiedon esityskerroksesta. Tiedon rakenteen määrittelyllä tarkoitetaan sitä, miten tiedon osat liittyvät toisiinsa, kun taas esityskerros määrittelee sen, kuinka tiedon rakenne näytetään. HTML-kielen alkuvaiheissa näitä kerroksia ei kuitenkaan erotettu selvästi toisistaan, ja kieleen lisättiin esitystapaan liittyviä standardoimattomia ominaisuuksia, jotka myöhemmin liitettiin HTML 3.2 standardiin. (Pfaffenberger, Schafer, White & Karow 2004, 5-6.)

Uudempi HTML 4.0 standardi kuitenkin määrittelee kaikki esitystapaan liittyvät elementit vanhentuneiksi. Kaikki muotoilu pitäisi nykyisien standardien mukaan siirtää CCS-tyyliohjeisiin (ks. Luku 3.4 CSS). (Pfaffenberger, ym. 2004, 5-6)

3.2 XML

Extensible Markup Language (XML) on merkintäkieli, jolla kuvataan tiedon tai dokumenttien rakennetta. XML-merkintäkieli on vapaasti laajennettavissa tarpeen mukaan standardien rajoissa. Toisin kuin HTML-merkintäkielessä, XML-merkintäkielen määritelmä ei määrittele tägejä, vaan on itsessään määritelmä, jonka mukaan tägejä voidaan määritellä. (Dykes & Tittel. 2005. 33.)

XML-merkintäkieli kuvaa vain tiedon rakennetta eikä määrittele millään tapaa tiedon esitystapaa. Tiedon esitystapa määritellään erillisessä tyyliohjeessa. Tyyliohjeita määritellään Cascading Style Sheets (CSS) ja Extensible Stylesheet Language Transformations (XSLT) -kielillä. Tyyliohjeita voidaan käyttää erikseen tai yhdessä määrittelemään tiedon esitystapa. (Dykes & Tittel. 2005. 34.)

3.3 XHTML

Extensible HyperText Markup Language (XHTML) -merkintäkieli on HTML 4.0 -standardista kehitetty XML-merkintäkielellä kuvattu merkintäkieli. XHTML-merkintäkieli eroaa HTML-merkintäkielestä siten, että XHTML on kuvattu XML-kuvauskielellä HTML-kielen kuvaamisen käytetyn SGML-kielen sijaan. (Pfaffenberger, ym. 2004, 15.)

3.4 CSS

Cascading Style Sheets (CSS) tyyliohjeet suunniteltiin korvaamaan HTML 4.0 -standardin vanhentuneeksi määrittelemän HTML-esityskerroksen. CSS-tyylitiedostot tarjoavat tavan määritellä ja muotoilla dokumentin esitystapaa kajoamatta itse dokumentin rakenteeseen. CSS julkistettiin samaan aikaan HTML 4.0 -standardin julkaisun kanssa vuonna 2007. Koska HTML 4.0 standardi määritteli esitystapaan liittyvät elementit vanhentuneiksi, tarvittiin tapa määritellä tiedon esitystapa. (Pfaffenberger, ym. 2004, 13-15.)

3.5 Javascript

Useiden esiversioiden ja nimien jälkeen Javascript-nimeä alettiin käyttämään vuonna 1996. Vuonna 1997 JavaScript 1.1 -versiosta luotiin standardi nimellä ECMAScript Technical Commitee #39-ryhmän toimesta. ECMAScript ei ole kieli itsessään, vaan standardi, jonka pohjalta voidaan luoda skriptikieliä. Tällaisia kieliä

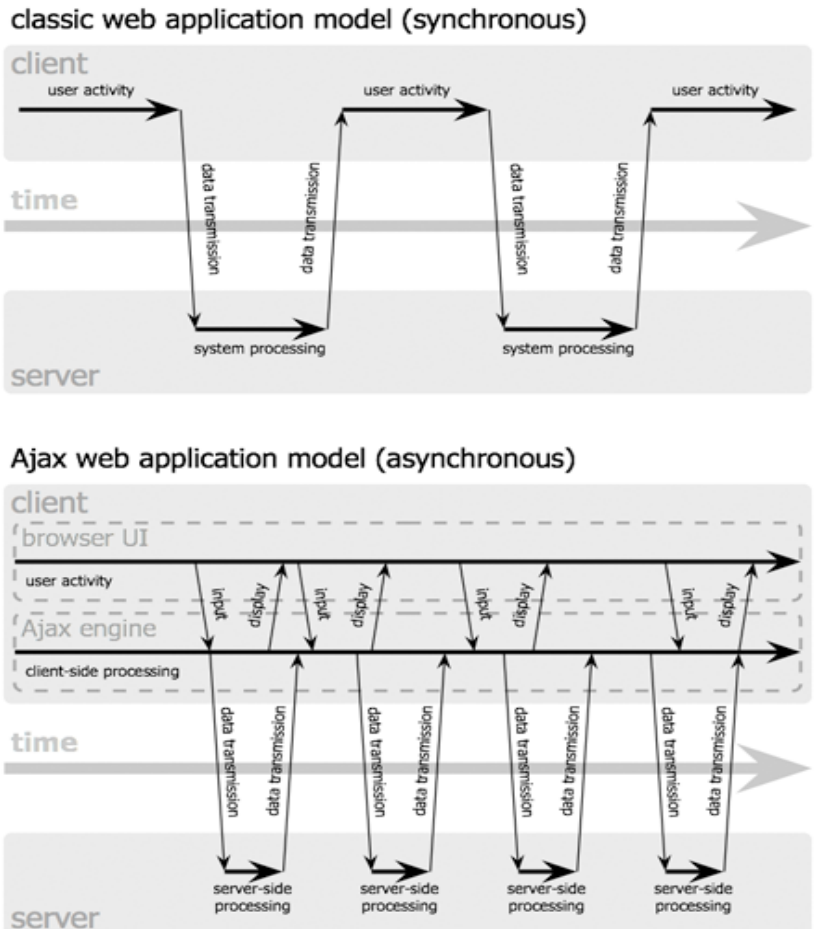
ovat nykyinen JavaScript, ActionScript ja ScriptEase. Nykyinen JavaScript-nimellä tunnettu kieli koostuu ECMAScript-standardin toteutuksesta sekä Document Object Model- ja Browser Object Model -rajapinnoista. (Zakas 2009, 39-42,45.)

XML-kielestä laajennettu Document Object Model on rajapinta, joka mallintaa HTML-dokumentin rakenteen hierarkkiseksi puurakenteeksi. Tämän puurakenteen avulla sivun sisältöä ja rakennetta voidaan muokata. DOM-rajapinnan loi W3C vuonna 1998, ja se on yleinen rajapinta, jota voidaan käyttää myös muissa kielissä. (Zakas 2009, 45-46.)

Browser Object Model (BOM) on rajapinta, jonka avulla voidaan vaikuttaa Internet-selaimen ikkunaan. BOM-rajapintaa ei ole standardoitu ja eri selaimet toteuttavat sen tavalla. (Zakas, 2009, 47-48.)

3.6 Ajax

Termi Ajax on lyhenne sanoista Asychronous JavaScript + XML, ja sillä tarkoitetaan yhdessä käytettäviä tekniikoita, joiden avulla voidaan päivittää web-sivun osiota ilman tarvetta ladata koko sisältöä uudelleen. Nämä tekniikat ovat XHML, CSS, Javascript, DOM ja XML. (Garret, 2005.)



KUVA 1. Perinteinen ja asynkroninen kommunikointi palvelimen kanssa. (Garret, 2005)

Kuvassa 1 on kuvattu perinteisen web-sivun (classic web application model) ja Ajax-tekniikalla (Ajax web application model) toteutetun sivun tiedonsiirtoja suhteessa aikaan ja käyttäjän syötteisiin. Kun käyttäjä antaa syötteen perinteisen mallin mukaan tehdyllä web-sivulla, tieto siirretään ensin palvelimelle, se prosessoidaan ja uusi tieto palautetaan käyttäjälle. Tiedon siirtojen ja käsittelyn ajan käyttäjän ei ole mahdollista käyttää sivua. Ajax-tekniikalla toteutetulla web-sivulla käyttäjä voi antaa syötteitä jatkuvasti, ja tiedon siirto tapahtuu samanaikaisesti taustalla. (Garret, 2005.)

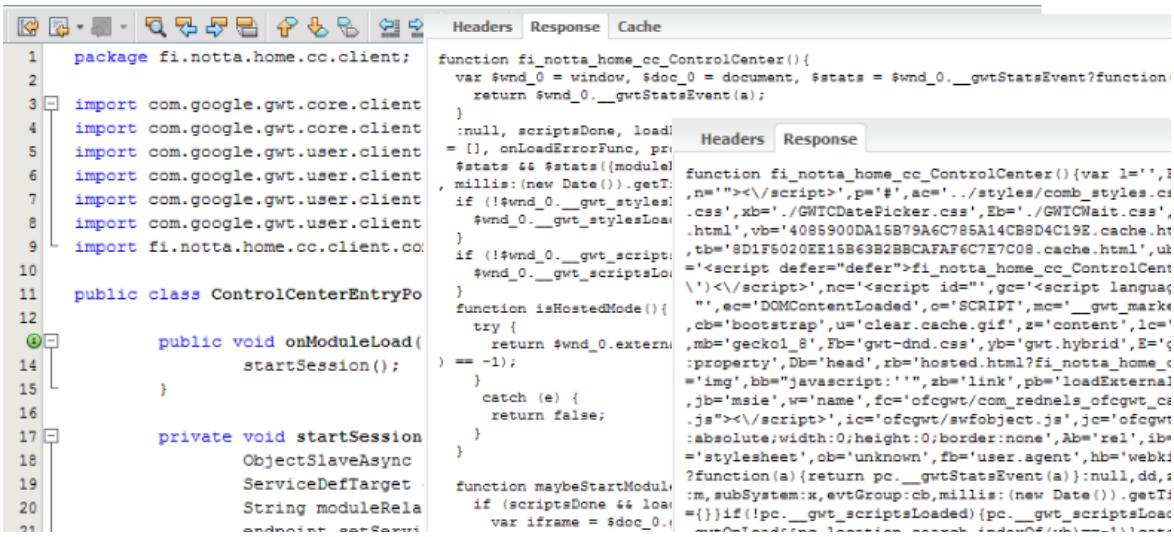
4 GOOGLE WEB TOOLKIT

Google Web Toolkit (GWT) on Googlen kehittämä avoimen lähdekoodin Java-pohjainen sovelluskehys. GWT julkistettiin vuonna 2006 alun perin suljetun lähdekoodin sovelluskehyksenä. Vuonna 2007 sovelluskehyksen 1.3-version julkaisun yhteydessä lähdekoodit julkaistiin avoimen lähdekoodin Apache 2.0 -lisenssillä. (Dwyer 2008, 5.)

4.1 Yleistä Google Web Toolkit -sovelluskehyksestä

GWT-sovelluskehyksen suunnittelun lähtökohtana on ollut yhteensopivuus eri selainten kanssa, testattavuus sekä kieliversioiden tukeminen. Lisäksi GWT-sovelluskehyksen avulla voidaan ohittaa AJAX-tekniikoista yleisesti vaivaavat ongelmat sivuhistorian ja kirjanmerkkien asettamisen kanssa. Sovelluskehys mahdollistaa sovellusten luomisen Java-kielellä ja sovelluksen osien kääntämisen JavaScript-kielelle (ks. Projektin rakenne). (Dwyer 2008, 5.)

Myös sovellusten skaalautuvuus on otettu huomioon sovelluskehystä suunniteltaessa. Skaalautuvuudella tarkoitetaan ohjelmistotuotannossa järjestelmän tai prosessin kykyä selvitä käyttäjämäärän kasvusta tai mahdollisuutta uusien ominaisuuksien lisäämiseen. Perinteisissä JavaScript- ja Ajax-sovelluksissa uusien ominaisuuksien lisääminen vaatii usein liian suuria työpanoksia saavutettavaan hyötyyn nähden. (Dwyer 2008, 6-7.)



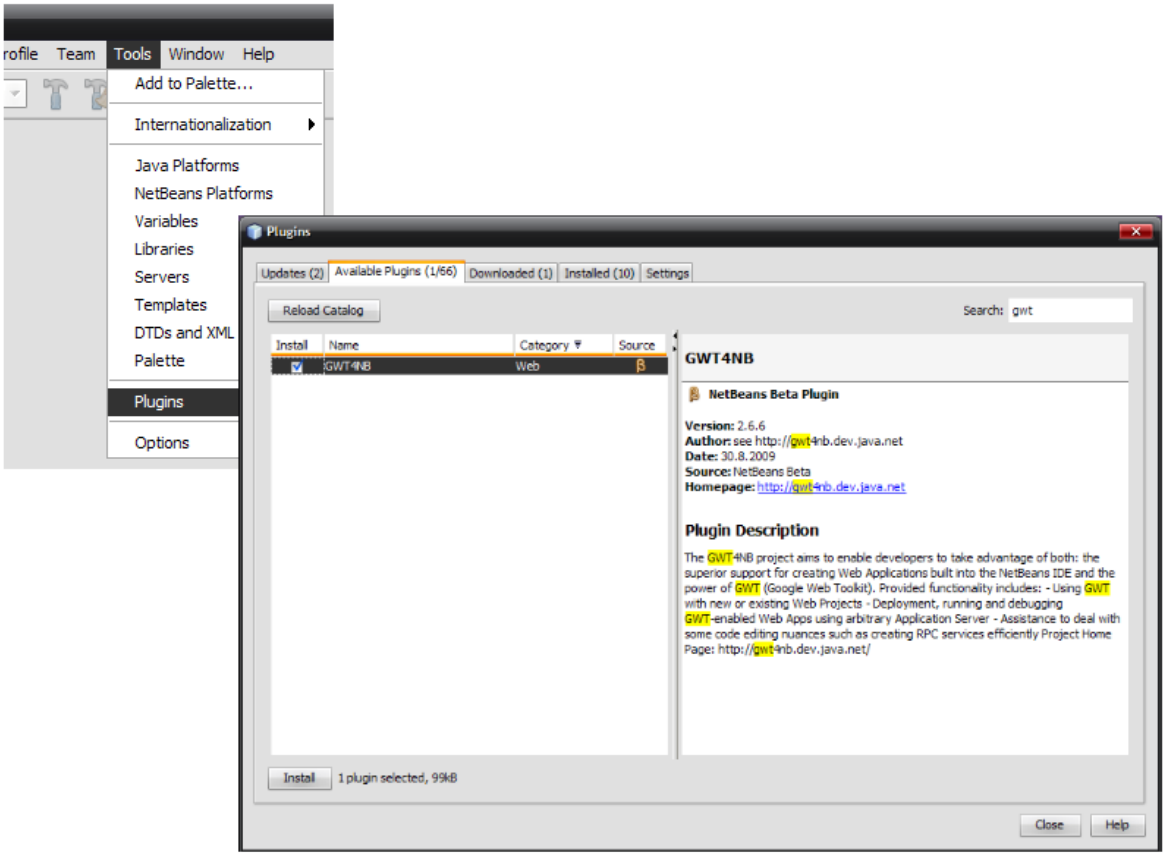
KUVA 2. Lähdekoodia Java-kielellä, JavaScript-kielellä ja optimoituna JavaScript-kielellä

GWT-kääntäjä optimoi käännettävän lähdekoodin asiakkaan kieliversion ja selaimen mukaan. Lisäksi kääntäjä osaa tarvittaessa poistaa lähdekoodista rivinvaihdot sekä käyttämättömän koodin. Lisähyötyä saavutetaan myös käyttämällä ”obfuscation”-ominaisuutta, jolloin kääntäjä lyhentää metodien ja muuttujien nimet lyhinpään mahdolliseen muotoonsa. Lisäksi kääntäjän luomat JavaScript-tiedostot pysyvät asiakkaan selaimen välimuistissa, joten niitä ei tarvitse ladata kuin ensimmäisellä sivulatauksella tai sovelluksen muuttuessa. (Dwyer 2008, 8-10.)

4.2 GWT-sovelluskehyksen integrointi NetBeans-kehitysympäristöön

GWT-sovelluskehys on saatavilla sekä Eclipse-kehitysympäristön lisäosana että erillisenä ladattavana zip-tiedostona muita kehitysympäristöjä varten. Google tarjoaa sovelluskehyksen asennus- ja käyttöohjeet mukana tuleville työkaluilla sekä Eclipse-kehitysympäristön lisäosalle sovelluskehyksen kotisivuilla. (Project Overview: Google Web toolkit, [Viitattu 8.11.2009].)

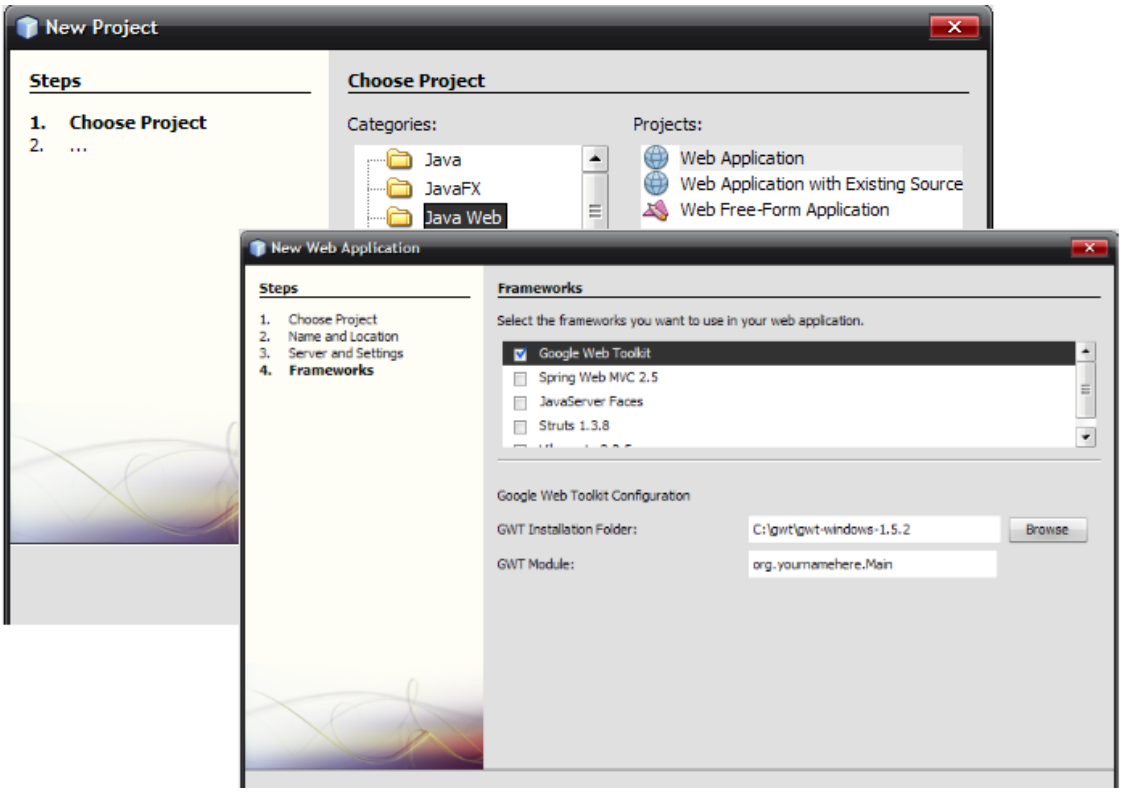
Sovelluskehyksen käyttöä varten NetBeans-kehitysympäristöön asennetaan GWT4NB-lisäosa. Lisäosa asennetaan NetBeans-kehitysympäristön Tools-valikon alta löytyvän Plugins-työkalun avulla. Available Plugins-välilehdeltä valitaan GWT4NB-lisäosan kohdalta Install-valintalaatikko ja painetaan Install-painiketta. Asennuksen valmistuttua sovelluskehystä voidaan käyttää kehitysympäristöstä.



KUVA 3. GWT4NB-lisäosan asennus.

4.3 GWT-projektin luominen NetBeans-kehitysympäristössä

GWT-projektia varten tarvitaan GWT-sovelluskehyksen tiedostot, jotka ovat saatavilla pakattuna tiedostona sovelluskehyksen kotisivuilta osoitteesta http://code.google.com/intl/fi/webtoolkit/download.html. Sovelluskehys on saatavilla Windows- Mac OS X- ja Linux-käyttöjärjestelmille. Kun sovelluskehys on ladattu, se puretaan hakemistoon, joka voidaan lukea NetBeans-kehitysympäristöstä.



KUVA 4. Uuden GWT-projektin luominen

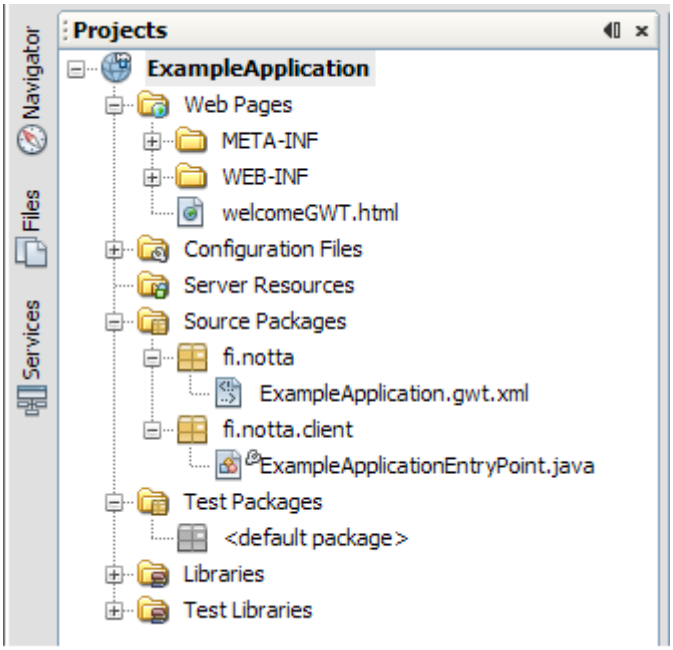
Uusi projekti luodaan valitsemalla NetBeans-kehitysympäristön New Project-valintaikkunasta Java Web-alihakemistosta Web Application -tyyppinen projekti. Projektiin valitaan Frameworks-kohdasta Google Web Toolkit ja GWT Installation Folder -kohtaan aiemmin purettu GWT-sovelluskehyksen hakemisto. GWT-module kohtaan merkitään luotavan moduulin nimi. Kun nämä tiedot on täytetty, kehitysympäristö luo tarvittavat hakemistot ja projektitiedostot sekä avaa sovellukseen liittyviä tiedostoja. (GWT4NB Flash based demo, [Viitattu 7.11.2009].)

4.4 Google Web Toolkit projektin rakenne

Projektin tiedostojärjestelmä rakentuu lähdekoodit sisältävästä src-hakemistosta. Kaikki Java-kieliset lähdekoodit tulisi pitää src-hakemiston java-alihakemistossa, testit tests-alihakemistossa ja konfiguraatio- ja resurssitiedostot public-alihakemistossa. (Dwyer, 2008, 18.)

Client-alihakemisto sisältää kaikki JavaScript-kielelle käännettävät Java-luokat. Client-hakemistoon tulevat luokat, jotka välittävät tietoa asiakassovelluksen ja palvelinsovelluksen välillä sekä käyttöliittymään liittyvät luokat. Server-alihakemisto sisältää kaikki muut Java-lähdekoodit. Koska Server-hakemiston lähdekoodeja ei ole tarpeen kääntää JavaScriptiksi, ei myöskään ole rajoituksia luokkien tai kirjastojen käytöstä. Public-alihakemistoon talletetaan kaikki tiedostot, jotka eivät sisällä lähdekoodia. Tällaisia tiedostoja ovat esimerkiksi CSS-tyyliohjeet ja kuvatiedostot. (Dwyer, 2008, 18-21.)

Alkaen GWT-sovelluskehyksen 1.6 versiosta public-alihakemistota ei enää käytetä, vaan kaikki resurssitiedostot talletetaan war-hakemistoon (What's New in GWT 1.6?, [Viitattu 5.11.2009]). NetBeans-kehitysympäristössä gwt4nb-lisäosaa käytettäessä tämä hakemisto on nimellä web. (GWT4NB Issue 76, [Viitattu 8.11.2009]).

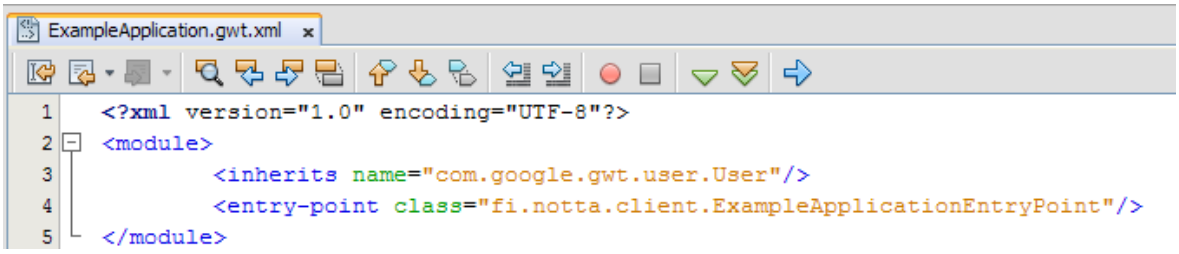


KUVA 5. GWT-projektin rakenne

Kuvassa 5 on luotu ExampleApplication-niminen moduuli. GWT4NB-lisäosa on luonut automaattisesti kuvassa näkyvät hakemistot ja tiedostot. Huomioitavaa kuvassa ovat muutamat erot aiemmin käsiteltyyn projektin rakenteeseen. Src-hakemisto näkyy kehitysympäristössä Source Packages -nimisenä ja web-hakemisto näkyy nimellä Web Pages. Lisäksi, server-alihakemistoa ei ole luotu,

koska projektissa ei ole palvelimella ajettavaa lähdekoodia. Nämä erot eivät kuitenkaan vaikuta itse sovelluskehitykseen.

GWT-moduuli määritellään XML-muotoisessa konfigurointitiedostossa, johon viitataan yleisesti termillä Module.xml, ja jonka tiedostonimi on muotoa moduulin_nimi.gwt.xml. Module.xml-tiedosto sijaitsee client- ja server-alihakemistoiden kanssa samassa hakemistossa. Module.xml-tiedostossa määritellään vähintään EntryPoint-rajapinnan toteuttava luokka, joka käynnistää sovelluksen. Lisäksi tiedostossa määritellään moduulin riippuvuudet inherits-tägeillä. Inherits-tägillä osoitetaan tarvittavien moduulien polku (paketti), josta kunkin moduulin Module.xml-tiedosto löytyy. NetBeans-kehitysympäristön GWT4NB-lisäosa luo konfigurointitiedoston automaattisesti, ja käyttäjä voi muokata sitä tarpeen mukaan. (Dwyer, 2008, 18,23-24.)



KUVA 6. GWT-moduulin määrittely.

Yllä olevassa kuvassa näkyy ExampleApplication-moduulin määrittely. Moduulilla on yksi riippuvuus, com.google.gwt.user-paketista löytyvä User-moduuli. ExampleApplication-moduulin Entry Point-luokaksi on määritelty fi.notta.client-paketista löytyvä ExampleApplicationEntryPoint-luokka.

4.5 Kommunikointi palvelimen kanssa Remote Procedure Calls -tekniikalla

GWT-sovelluskehyksessä on sisäänrakennettuina ominaisuuksina Remote Procedure Calls (RPC)- ja olioiden serialisointi -tekniikat. RPC-tekniikan avulla voidaan kommunikoida palvelimen kanssa. RPC-tekniikan toimintaperiaatteena on tehdä asynkroninen ja synkroninen rajapinta sekä palvelimella toimiva palvelu, joka toteuttaa synkronisen rajapinnan sekä GWT-sovelluskehyksen oman RemoteServi-

ceServlet-rajapinnan. Synkroninen rajapinta liittyy GWT-sovelluskehyksen sisäiseen toteutukseen ja asynkroninen rajapinta on asiakassovelluksen kommunikaatiota varten. Selkeyden vuoksi olisi hyvä lisäksi erottaa varsinainen palvelimella toimiva toteutus palvelimen RPC-rajapinnasta. Serialisointitekniikka puolestaan mahdollistaa palvelinsovelluksen Java-luokkien ja asiakassovelluksen JavaScript-olioiden vaihdon keskenään. (Dwyer 2008, 25-28.)

Kaikkien palvelimen ja asiakkaan välillä välitettävien olioiden (luokkien) tulee toteuttaa GWT-sovelluskehyksen IsSerializable-rajapinta, jotta kääntäjä pystyy kääntämään ne JavaScriptiksi. Varsinainen IsSerializable-rajapinta ei määrittele toteutettavia metodeja, mutta kaikkien rajapinnan toteuttavien luokkien tulee määritellä parametriton konstruktori. Alkaen GWT-sovelluskehyksen versiosta 1.4 välitettävien on myös mahdollista toteuttaa Javan Serializable-rajapinta GWT-sovelluskehyksen vastaavan sijaan. Tämä muutos toteutettiin, koska IsSerializable-rajapinta on GWT-luokkakirjastosta riippuvainen, joten muualla sovelluksessa ei voida käyttää samoja datasiirto-olio-luokkia. Serializable-rajapinnan käyttö ei kuitenkaan poista tarvetta luoda parametritonta konstruktoria datasiirto-olio-luokkiin. (Cooper & Collins, 2009, 83-84.)

Muita rajoituksia JavaScriptiksi käännettävillä luokilla ovat ulkoisten kirjastojen puute, sekä joidenkin Javan ominaisuuksien ja luokkien käytön rajoitukset. Rajoitukset riippuvat GWT-sovelluskehyksen versiosta, esimerkiksi GWT 1.5 -versioon asti tuetaan vain Javan 1.4 -versiota.(Dwyer, 2008, 20.)

Uuden RPC-palvelun luonti NetBeans-kehitysympäristössä onnistuu valitsemalla File-valikosta New-valinta ja edelleen Google Web Toolkit -hakemistosta GWT RPC Service -kohta. NetBeans-lisäosa luo tarvittavat asiakas- ja palvelinpuolen rajapinnat, sekä lisäksi esimerkin RPC-palvelun käytöstä. (GWT4NB Flash based demo, [Viitattu 7.11.2009].)

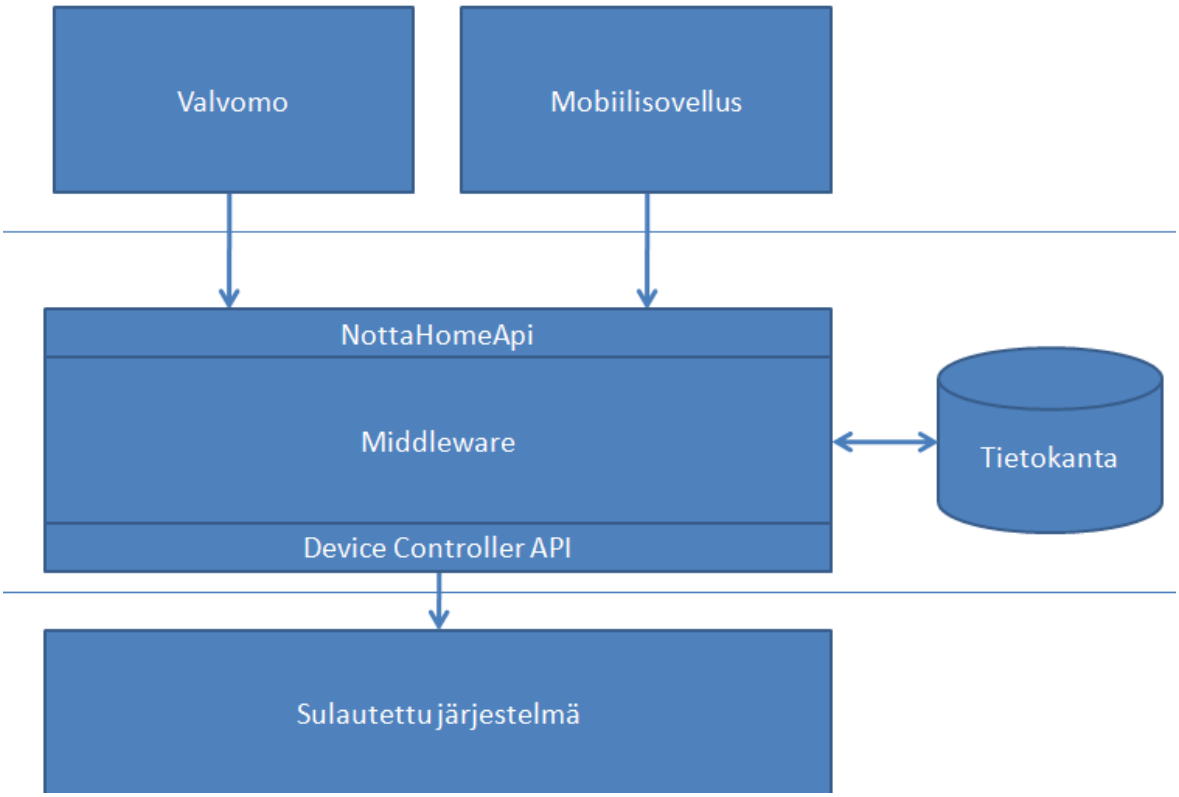
5 TAPAUSTUTKIMUS: VALVOMO-SOVELLUKSEN DOKUMENTOINTI

Opinnäytetyössä on käytetty tapaustutkimuksena syksyllä 2009 tehtyä Notta for Home -kodinohjausjärjestelmän Valvomo-sovelluksen dokumentaatioprojektia. Projektin tavoitteena oli toteuttaa lähdekoodin dokumentaatio Javadoc-työkalulla.

5.1 Notta for Home -kodinohjausjärjestelmän kuvaus

Notta for Home -kodinohjausjärjestelmän kehitys aloitettiin vuonna 2005, ja nykyään järjestelmä on käytössä noin 20 seurantakohteessa. Järjestelmän toiminnallisuus määritellään kunkin asiakkaan tarpeiden mukaisesti. Notta for Home -kodinohjausjärjestelmän avulla voidaan ohjata kaikkia kodin sähköpisteitä, kuten ilmastointia, valaistusta ja lämmitystä. Muita järjestelmän ominaisuuksia ovat muun muassa kulunvalvonta, erilaiset hälytykset sekä mahdollisuus toimintojen ajastamiseen. Lisäksi järjestelmään voidaan liittää erilaisia mittalaitteita, joiden avulla voidaan seurata kodin kulutustietoja, kuten veden- ja sähkön kulutusta. (Eteläaho, 2009).

Järjestelmää voidaan ohjata kotona paikalliskytkimillä, kosketuspinnalla varustetulta ohjauspaneelilla tai lähiverkon kautta tietokoneelta. Käyttö onnistuu myös Internet-yhteyden kautta kodin ulkopuolelta tietokoneella tai älypuhelimella. Järjestelmän mukana toimitetaan myös Java-tuella varustettujen puhelimien kanssa yhteensopiva erillinen Java-sovellus. Lisäksi järjestelmää on mahdollista ohjata tekstiviesteillä. (Eteläaho, 2009).



KUVA 7. Notta for Home -kodinohjausjärjestelmän rakenne

Yllä olevassa kuvassa on esitelty Notta for Home -kodinohjausjärjestelmän rakenne. Järjestelmä koostuu kolmesta kerroksesta: asiakassovellukset, Middleware ja sulautettu järjestelmä. Asiakassovellukset kommunikoivat järjestelmän muiden osien kanssa Middleware-kerroksen tarjoaman NottaHomeApi-rajapinnan kautta. NottaHomeAPI-rajapinta on tarkoitus julkaista Valvomon lähdekoodien julkaisun yhteydessä, jolloin järjestelmään voidaan vapaasti kehittää uusia asiakassovelluksia. Middleware-kerros tunnistaa asiakassovellukset ja tarvittaessa vaatii käyttäjää kirjautumaan järjestelmään. Middleware-kerros pitää yllä tietokantaa ja kommunikoi sulautetun järjestelmän kanssa Device Controller API -rajapinnan kautta.

5.1.1 Valvomo

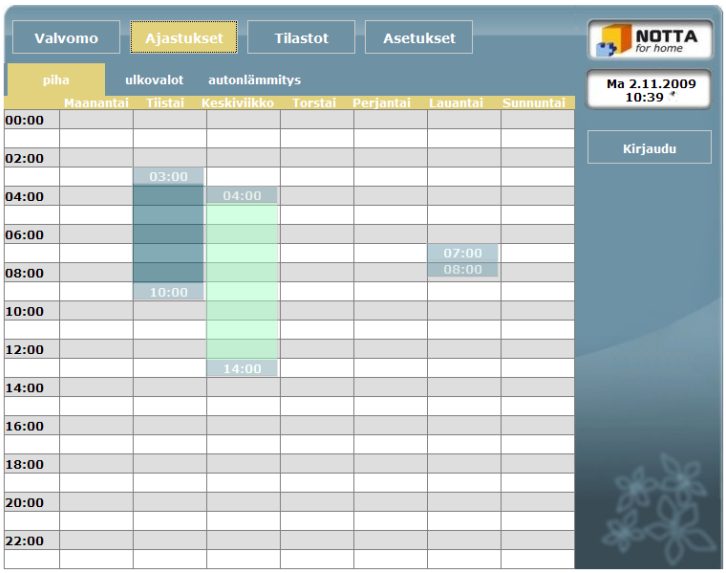
Notta For Home Valvomo on GWT-sovelluskehyksellä toteutettu web-sovellus. Valvomon avulla voidaan ohjata kodinohjausjärjestelmän toimintoja. Valvomon

käyttöliittymää suunniteltaessa on kiinnitetty erityistä huomiota sen helppokäyttöisyyteen.

KUVA 8. Valvomon valvomo-välilehti

Valvomo-välilehdellä näkyvät kodin eri alueet välilehdiksi ryhmiteltynä. Jokaiselle alueelle on määritelty asennusvaiheessa oma pohjakuvansa, ja alueen laitteiden kuvakkeille paikka pohjakuvan päällä. Kuvake ilmaisee kunkin laitteen tilan reaaliaikaisesti. Laitteiden ohjaaminen tapahtuu klikkaamalla halutun laitteen kuvaketta.

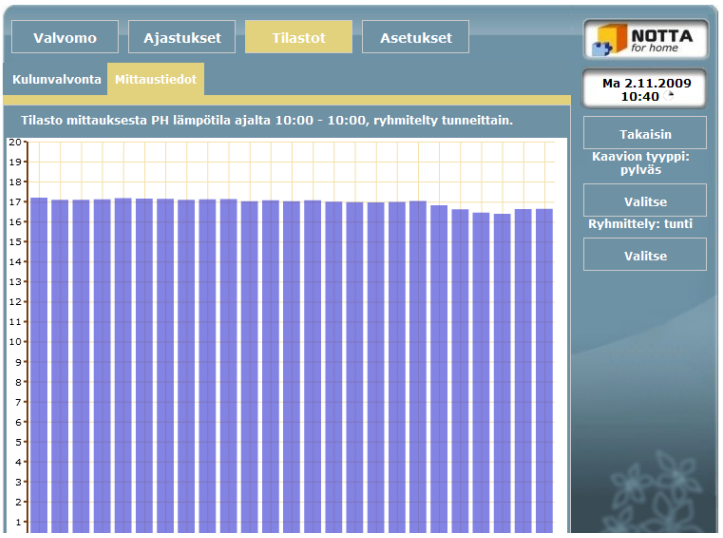
Valvomo-välilehdeltä on myös pääsy eri hälytyslaitteiden ja ilmastoinnin hallintaan, pikatoiminnot laitteiden ohjaukseen sekä järjestelmään asennettujen antureiden reaaliaikaiset arvot.



KUVA 9. Valvomon ajastukset-välilehti

Ajastukset-välilehdellä voidaan tarkastella ja asettaa viikkokalenterilla uusia ajastuksia järjestelmään kytketyille laitteille. Ajastettavat laitteet on ryhmitelty omiin välilehtiinsä. Aiemmin asetetut ajastukset esitetään viikkokalenterilla värillisinä laatikoina.

Laitteen ajastaminen tapahtuu maalamalla halutun päivän kohdalta haluttu aikaväli. Kun aikaväli on valittu, aukeaa automaattisesti uusi valintaikkuna, josta käyttäjä voi tarkentaa aloitus- ja lopetusaikoja ja tarvittaessa asettaa aikaohjelman toistumaan päivittäin. Asetuksista riippuen laite asetetaan päälle tai pois päältä ajastuksen mukaisesti viikoittain haluttuna päivänä haluttuun aikaan. Ajastettavia laitteita voidaan lisätä asetusnäkymän kautta.



KUVA 10. Valvomon tilastot-välilehti

Tilastot-välilehdellä voidaan tarkastella järjestelmään kytkettyjen antureiden keräämiä tietoja kuvaajien muodossa. Tilastot-välilehdeltä voidaan myös selata kulunvalvonnan tallentamia tietoja. Tietoa voidaan esittää kerätyn tiedon tyypistä riippuen eri lailla ryhmiteltynä ja kuvattuna. Kuvaajien esitys on toteutettu flash-tekniikalla toimivalla Open Flash Chart -grafiikkakirjastolla.

KUVA 11. Valvomon asetukset-välilehti

Asetukset-välilehdellä käyttäjä voi asettaa kodinohjausjärjestelmän asetuksia. Ylläpitäjän oikeudet omaava käyttäjä voi esimerkiksi asettaa muille käyttäjille käyttöoikeuksia laitteilla, luoda uusia pikatoimintoja sekä ajastettavia toimintoja.

5.1.2 Middleware-palvelinsovellus

Middleware on Java API for XML Web Services (JAX-WS) -ohjelmointirajapinnalla toteutettu palvelinsovellus, jonka avulla asiakassovellukset voivat kommunikoida sulautetun järjestelmän kanssa. Middleware huolehtii myös järjestelmän tietokannasta ja yhteyksistä sinne.

5.1.3 Sulautettu järjestelmä

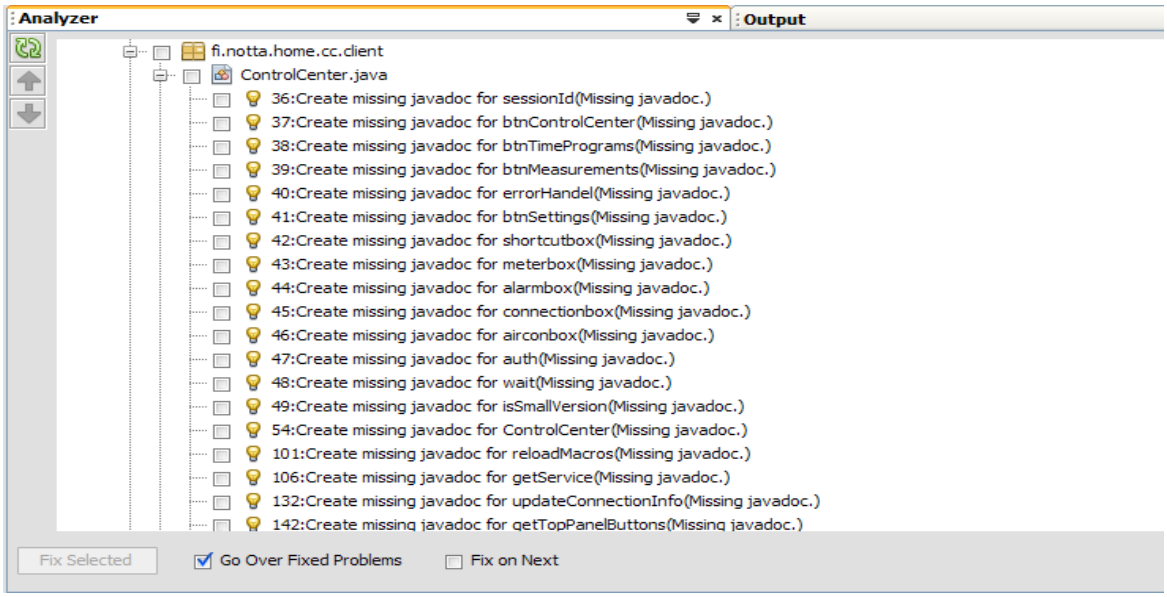
Sulautettu järjestelmä koostuu isäntäkortista ja laajennuskorteista. Kukin laajennuskortti sisältää 12 digitaalista lähtöä, 4 analogista lähtöä sekä 16 digitaalista tuloa ja 4 analogista tuloa. Yksinkertaistettuna sanottuna isäntäkortilla oleva ohjelmisto seuraa tuloja ja asettaa lähtöjä ohjelmallisesti toteutetun ohjelmoitavan logiikan avulla. Järjestelmä ohjelmoidaan erillisellä Java-kielellä toteutetulla ohjelmointityökalulla.

5.2 Projektissa käytetyt työkalut

Projektissa käytettiin NetBeans-kehitysympäristöä ja sen työkaluja. NetBeans on ilmainen avoimen lähdekoodin kehitysympäristö, jossa on tuki muun muassa kielille: Java, C/C++, PHP ja Ruby. NetBeans-kehitysympäristö on saatavissa Windows- ja Linux-, Mac OS X- ja Solaris-käyttöjärjestelmille. Kirjoitushetkellä NetBeans-kehitysympäristön uusin vakaa versio on 6.7.1, ja se on saatavilla NetBeans-kehitysympäristön kotisivulta osoitteesta http://www.netbeans.org. (NetBeans IDE [Viitattu 6.11.2009].)

5.2.1 Analyze Javadoc -työkalu

Javadoc-työkalun ymmärtämien kommenttien kirjoitukseen käytettiin NetBeans-kehitysympäristön sisäänrakennettua Analyze Javadoc -työkalua, jonka avulla voitiin käydä järjestyksessä läpi dokumentoimattomat osiot. Työkalu etsii puuttuvat kommentit ja listaa ne. Työkalulla voidaan myös luoda automaattisesti kommentteja, jotka sisältävät pakolliset tägit, mutta itse kommenttien sisältö luodaan käsin. Alla olevassa kuvassa näkyy Analyze Javadoc -työkalun käyttöliittymä, jossa on näkyvissä ContoCenter-luokan analysoinnin tulos.



KUVA 12. Analyze Javadoc -työkalu

5.2.2 UML-työkalu

UML-kaavioiden luontiin suoraan kehitysympäristöstä on tarjolla sekä maksullinen Visual Paradigm: SDE for NetBeans ja ilmainen avoimen lähdekoodin NetBeans UML-lisäosa (Unified Modeling Language (UML) Plugin, [Viitattu 6.11.2009]). Projektia varten valittiin ilmainen lisäosa, koska sen käytöstä oli aiempia kokemuksia.

KUVA 13. ControlCenter-luokan luokkakaavio

Yllä olevassa kuvassa on UML-lisäosan reverse engineering -toiminnolla toteutettu ControlCenter-luokan luokkakaavio. Kuvassa näkyvät luokan attribuutit ja metodit, sekä niiden näkyvyysmäärittelyt. UML-lisäosan avulla voidaan tuottaa sekä UML-kaavioita lähdekoodista reverse engineering -toiminnolla, sekä lähdekoodia UML-kaavioista forward engineering -toiminnolla. UML-lisäosalla luotua kaaviota voidaan myös muokata ja muutokset voidaan päivittää takaisin lähdekoodin. Esimerkiksi muuttujan nimi voidaan muuttaa lisäosalla luodusta kaaviosta käsin, ja päivit-

tää muutos takaisin lähdekoodiin. (Unified Modeling Language (UML) Plugin, [Viitattu 6.11.2009].)

5.3 Dokumentointi

Tapaustutkimuksen projektissa luotiin Javadoc-työkalun määritelmien mukaiset dokumentointikommentit Valvomo-sovelluksen lähdekoodeille ja tuotettiin tarpeellisiksi katsotuista osioista UML-kaavioita. Lisäksi parannettiin lähdekoodin rakennetta refaktoroinnilla. Refaktorointi tarkoittaa ohjelmiston rakenteen muuttamista ilman ohjelmiston ulkoisten liittymien muuttamista (Fowler, 1999, 9).

Dokumenttikommenttien tuottaminen sujui nopeasti NetBeans-kehitysympäristön työkalujen avulla. Kommentit luotiin kehitysympäristön Analyze Javadoc -työkalulla ja kommenttien sisältö kirjoitettiin aiemman järjestelmätuntemuksen pohjalta. Koska lähdekoodit on tarkoitus julkistaa myöhemmin avoimena lähdekoodina, dokumenttikommentit kirjoitettiin englanniksi.

Tapaustutkimuksen projektissa oli tavoitteena dokumentoida kaikki lähdekoodi myöhemmin tapahtuvaa avoimen lähdekoodin julkistusta varten. Analyze Javadoc -työkalu ei kuitenkaan ilmoita private-näkyvyysmääreellä määriteltyjä metodeja tai attribuutteja, joten nämä jouduttiin etsimään ja luomaan ilman automaatiota.

Kun kommentit oli saatu valmiiksi, varsinaiset HTML-muotoiset tiedostot generoitiin kehitysympäristön Generate Javadocs -komennolla. Komento ajaa asennetusta Javan versiosta riippuen Javadoc-työkalun oletusasetuksilla. Komennolle voidaan myös asettaa parametreja Project properties -ikkunan kautta, mutta tätä ominaisuutta ei tarvittu tätä projektia varten.

Projektin aikana tuotetut Valvomon Javadoc-dokumentit ovat tämän opinnäytetyön liitteenä olevalla cd-levyllä. Valvomon lähdekoodeja ei lisätty liitteeksi, koska lisenssistä, jolla lähdekoodit julkaistaan ja julkaisuajankohdasta ei ole vielä päätetty.

6 YHTEENVETO

Opinnäytetyön tarkoituksena oli esitellä eri web-tekniikoita ja erityisesti Google Web Toolkit -sovelluskehystä. Google Web Toolkit soveltuu erityisen hyvin suuriin projekteihin, joissa sovelluksen suorituskyky ja yhteensopivuus eri internet-selainten kanssa on tärkeää. Lisäksi sovelluskehyksen avulla kehittäjät voivat keskittyä sovelluksen ominaisuuksien kehittämiseen selainkohtaisten ongelmien ratkaisemisen sijaan. Toisaalta Google Web Toolkit soveltuu myös yksittäisten web-sovellusten tekoon tietyissä tilanteissa. Lisäksi esiteltiin Javadoc-työkalu ja työkalun ymmärtämien kommenttien syntaksi. Javadoc-työkalun avulla voidaan tuottaa helposti ymmärrettävää dokumentaatiota sovelluskehittäjien tarpeisiin.

Tapaustutkimuksen Valvomo-sovelluksen dokumentointiprojektissa tuotettiin Javadoc-työkalulla standardien mukainen HTML-muotoinen dokumentaatio Valvomo-sovelluksen lähdekoodista. Dokumentaatiosta voidaan helposti tarkastella sovelluksen toimintaa ja rakennetta. Dokumentaation avulla projektia voidaan tulevaisuudessa jakaa osiin, ja tietotaidon jakaminen uusille kehittäjille helpottuu. Lisäksi luotiin NetBeans-kehitysympäristön refaktorointi-työkalujen avulla voidaan jatkokehitysvaiheessa helposti optimoida sovellusta.

Projektin aikana saatiin myös uusia ideoita sovelluksen arkkitehtuurin ja rakenteen yksinkertaistamiseen. Näitä ideoita tullaan hyödyntämään Valvomon seuraavaa versiota toteutettaessa. Dokumentoinnin kirjoittaminen onnistui hyvin, ja sen tarjoamasta informaatiosta on hyötyä myöhemmin, erityisesti julkistettaessa Valvomo-sovelluksen lähdekoodit avoimen lähdekoodin projektina.

LÄHTEET

Cooper, R. T. & Collins, C. E. 2008. GWT in Practice. Manning Publications.

Deitel, P. J. & Deitel, H. M. 2008. AJAX, Rich Internet Applications, and Web Development For Programmers. Indiana: Pearson Education.

Dwyer, J. 2008. Pro Web 2.0 Application Development with GWT. Apress.

Eteläaho, J. 2009. Notta for Home. [Henkilökohtainen sähköpostiviesti]. Vastaanottaja: Jani Pajunen. [Viitattu 24.11.2009]

Fowler, M. 1999. Refactoring. Improving the Design of Existing Code. Addison-Wesley.

Garret, J. J. 2005. Ajax: A New Approach to Web Applications. [Verkkojulkaisu]. [Viitattu 4.11.2009]. Saatavissa: http://www.adaptivepath.com/ideas/essays/archives/000385.php

GWT4NB Flash based demo. Ei päiväystä. [Flash-video]. Java.net GWT4NB project. [Viitattu 7.11.2009]. Saatavissa: https://gwt4nb.dev.java.net/manual/tutorials/AnagramDemo/gwt4nbDemo.htm

GWT4NB Issue 76. 2009. [Keskusteluryhmän viesti]. Java.net GWT4NB project. [Viitattu 8.11.2009]. Saatavissa: https://gwt4nb.dev.java.net/issues/show_bug.cgi?id=76

Haikala, I. & Märijärvi, J. 2002. Ohjelmistotuotanto. Helsinki:Satku.

How to Write Doc Comments for the Javadoc Tool. Ei päiväystä. [Verkkojulkaisu]. Sun Microsystems. [Viitattu 6.11.2009]. Saatavissa: http://java.sun.com/j2se/javadoc/writingdoccomments/

Javadoc 5.0 Tool. Ei päiväystä. [Verkkojulkaisu]. Sun Microsystems. [Viitattu 2.11.2009]. Saatavissa: http://java.sun.com/j2se/1.5.0/docs/guide/javadoc/index.html

Javadoc tool reference guide. Ei päiväystä. [Verkkojulkaisu]. Sun Microsystems. [Viitattu 8.11.2009]. Saatavissa: http://java.sun.com/j2se/1.5.0/docs/tooldocs/windows/javadoc.html#{@link}

NetBeans IDE. Ei päiväystä. [Verkkojulkaisu]. Sun Microsystems. [Viitattu 6.11.2009]. Saatavissa: http://www.netbeans.org/features/

Pfaffenberger, B., Schafer, S. M., White, C. & Karow, B. 2004. HTML, XHTML, and CSS Bible. 3. p. Indianapolis, Indiana: Wiley Publishing, Inc.

Pilone, D. & Pitman, N. 2005. UML 2.0 in a Nutshell. O'Reilly Media.

Project Overview: Google Web toolkit. Ei Päiväystä. [Verkkojulkaisu]. Google. [Viitattu 8.11.2009]. Saatavissa: http://code.google.com/intl/fi/webtoolkit/overview.html

Unified Modeling Language (UML) Plugin. Ei päiväystä. [Verkkojulkaisu]. NetBeans. [Viitattu 6.11.2009]. Saatavissa: http://www.netbeans.org/features/uml/index.html

Vesterholm, M. & Kyppö, J. 2006. Java-ohjelmointi. Helsinki: Talentum.

Dykes, L. & Tittel, E. 2005. XML For Dummies, 4th Edition. Indianapolis, Indiana: Wiley Publishing, Inc.

What's New in GWT 1.6? Ei päiväystä. [Verkkojulkaisu]. Google. [Viitattu 5.11.2009]. Saatavissa: http://code.google.com/intl/fi/webtoolkit/doc/1.6/ReleaseNotes_1_6.html

Zakas, N. C. 2009.Professional JavaScript for Web Developers. 2. p. . Indianapolis, Indiana: Wiley Publishing, Inc.