TEKNILLINEN KORKEAKOULU
Tietotekniikan osasto
Tietojenkäsittelyopin laboratorio

Taavi Hupponen, taavi.hupponen@hut.fi
Oskar Ojala, omojala@cc.hut.fi

T-106.720 Ohjelmistotekniikan projekti

SAX:n hyödyntäminen www-julkaisujärjestelmän toteutuksessa

Espoo, 31. toukokuuta 2004

Sisältö

1 Johdanto **2**

1.1 HTML:stä, XML:stä ja XHTML:stä 2

1.2 Ohjelmointirajapinnat XML:n käsittelyyn 2

1.3 Käsiteltävien asioiden rajaus 4

1.4 Esimerkkiongelmat 4

2 Kontekstin kuvaus **5**

2.1 Projektin tausta 5

2.2 WebCMS-julkaisujärjestelmä 6

3 Vaihtoehtoisista ratkaisuista **7**

4 Omat ratkaisumme **8**

4.1 Yleistä SAX:sta 8

4.2 Ratkaisut esimerkkiongelmiin 8

5 Ratkaisujen ja SAX:n arviointia **11**

5.1 Ratkaisujen toimivuus 12

5.2 SAX:in edut 12

5.3 SAX:in ongelmat 13

6 Yhteenveto **14**

Tiivistelmä

Tässä raportissa esitellään XML:n käsittelyyn tarkoitetun SAX-ohjelmointirajapinnan käyttöä www-julkaisujärjestelmässä. SAX:n käyttöön perehdytään ratkaisemalla kolme esimerkkiongelmaa: HTML-dokumentin muuntaminen korrektiksi XHTML-dokumentiksi, XHTML-dokumentin DOM-esityksen sarjallistaminen HTML:ksi ja erityisen uutistekstin muuntaminen XHTML-dokumentin DOM-esitykseksi.

SAX:n todetaan soveltuvan hyvin ainakin tiettyjen julkaisujärjestelmään liittyvien ongelmien ratkaisuun ja esimerkkiongelmille esitelläänkin ratkaisut, joiden perustana toimivat valmiita ja itsetoteutettuja komponentteja sisältävät SAX-liukuhihnat.

Yleisesti SAX:in todetaan olevan toimiva ja selkeä tekniikka, jonka vahvuutena on komponenttien uudelleenkäyttö, liukuhihnamallin tarjoama joustavuus ja ainakin julkaisujärjestelmän kannalta valmiiden komponenttien saatavuus.

1 Johdanto

1.1 HTML:stä, XML:stä ja XHTML:stä

HTML

HTML on SGML-pohjainen [1] merkkauskieli, joka on saavuttanut dominoivan aseman www-sivujen merkkauskielenä. Ellei toisin mainita HTML:llä tarkoitetaan tässä raportissa W3C:n määrittelemää HTML 4.01 strictiä [2]. HTML sisältää joitakin SGML:n piirteitä, jotka tekevät siitä ihmisten kannalta helpompaa käsitellä, mutta jotka vaikeuttavat ohjelmallista käsittelyä.

XML

XML [3] on SGML:n kevennetty ja yksinkertaistettu versio. XML:ää määriteltäessä pohjaksi otettiin SGML-standardi, josta sitten karsittiin ominaisuudet joita tarvittiin vain harvoin tai jotka tekivät SGML-kielen käsittelyn ohjelmallisesti hankalaksi. XML onkin varsin selkeä ja yksinkertainen, mutta silti ilmaisuvoimainen ja ennen kaikkea sen käsittely ohjelmallisesti on huomattavasti SGML:ää helpompaa.

XHTML

XHTML [4] on XML-sovellutus HTML:stä. XHTML:n etuja HTML:än nähden ovat esimerkiksi laajennettavuus ja XML:n myötä ohjelmallisen käsittelyn helppous. Ylipäätään XHTML:ää pidetään HTML:ää parempana esimerkiksi www-sivujen merkkaukseen [6]. Tästä huolimatta edes kaikki yleisimmät www-selaimet eivät kuitenkaan vielä tue XHTML:ää. Tulevaisuudessa XHTML kuitenkin epäilemättä syrjäyttää HTML:n.

Ellei toisin mainita, tässä raportissa korrektilla XHTML:llä tarkoitetaan XML-hyvämuotoista [3], XHTML-1.0-Strict:n [5] toteuttavaa XHTML:ää.

1.2 Ohjelmointirajapinnat XML:n käsittelyyn

Yleisesti ottaen XML:n käsittelyyn tarkoitetut ohjelmointirajapinnat voidaan jakaa kahteen luokkaan sen mukaan miten ne XML-dataa käsittelevät. Nämä luokat ovat tapahtumapohjainen datan käsittely ja puumalleihin perustuva datan käsittely.

Tapahtumapohjaiset ohjelmointirajapinnat

Tapahtumapohjainen XML:n käsittely perustuu jäsennystapahtumien käsittelyyn funktiokutsujen avulla. Tapahtumapohjaisen mallin etuja ovat nopeus ja

vähäinen muistintarve: käsiteltävästä XML-datasta pidetään muistissa kerrallaan vain pieni, sillä hetkellä käsiteltävä osa. Tämän mallin suurin heikkous on hajasaannin (random access) puute: dataa käsitellään tapahtumavirtana, jossa ei ole mahdollista liikkua taaksepäin.

Yleensä tapahtumapohjaiset ohjelmointirajapinnat toimivat ns. push-mallin mukaan. Push-mallissa jäsennyksen aktiivisena osapuolena toimii jäsentäjä, joka läpikäy xml-datavirtaa ja raportoi jäsennystapahtumista rekisteröityjen funktiokutsujen avulla. Push-mallin mukaan toimivista tapahtumapohjaisista ohjelmointirajapinnoista SAX on noussut de facto -standardin asemaan. SAX on alunperin ainoastaan Javalle tarkoitettu avoin ohjelmointirajapinta, mutta nykyisin SAX-toteutuksia on saatavilla myös muille ohjelmointikielille. [7]

Harvinaisemmassa pull-mallissa jäsentäjä ei itse tee funktiokutsuja vaan aktiivisena osapuolena toimiva asiakasohjelma pyytää jäsentäjältä tiedon seuraavasta jäsennystapahtumasta. Pull-malli on hyödyllinen erityisesti, kun tiettyyn tapahtumaan liittyvä prosessointi riippuu jäsennyksen tilasta (esimerkiksi ollaanko <head>:n vai <body>:n sisällä). Pull-mallien toteutukset, kuten StAX ovat kuitenkin vielä melko tuoreita ja niitä pidetään usein epäluotettavina. [12]

Tapahtumapohjainen XML:n käsittely on omimmillaan erityisesti tilanteisssa, joissa dataa luetaan tai suodatetaan, mutta jossa sitä ei varsinaisesti muokata. SAX olikin keskeisin työkalu tässä raportissa esittelemiemme ongelmien ratkaisemisessa.

Puumalliin perustuvat ohjelmointirajapinnat

Puumalliin perustuvien ohjelmointirajapintojen ideana on muodostaa XML-dokumentista puuesitys, jossa voidaan liikkua ja jota voidaan muokata. Tämän lähestymistavan etuna on juuri dokumentissa navigoinnin ja dokumentin muokkaamisen helppous. Erityisesti suurten dokumenttien kohdalla esiin nousevana haittana on kuitenkin se, että koko dokumentti täytyy pitää samanaikaisesti muistissa. Puumalliin perustuvia ohjelmointirajapintoja ovat muun muassa DOM [8], JDOM [9] ja XOM [10].

DOM:ia käytetään julkaisujärjestelmän TemplateEngine-, WebUI- sekä osin myös asiakassovellusosissa, mutta tässä raportissa esitettyjen ongelmien ratkaisuun puumalliin perustuvat ohjelmointirajapinnat eivät erityisen hyvin sovellu.

JAXP

JAXP eli Java API for XML Processing on XML:ää käsitteleville Java-ohjelmille tarkoitettu rajapinta. JAXP yrittää tarjota ohjelmoijalle yhdenmukaisen rajapinnan eri XML-tekniikoihin. JAXP tukee esimerkiksi DOM:ia, SAX:ia ja XSLT:ä. [11]

GNU JAXP

GNU JAXP on Free Software Foundationin JAXP-, SAX- ja DOM-rajapintojen vapaa toteutus. [14]

1.3 Käsiteltävien asioiden rajaus

Ryhmämme toteuttamassa julkaisujärjestelmässä käytetään niin SAX:ia kuin muitakin XML-rajapintoja. Tässä raportissa keskitymme kuitenkin SAX:in käyttämiseen kolmen esimerkkiongelman avulla. Näihin esimerkkiongelmiin esittelemme SAX-pohjaiset ratkaisut. Esimerkkiongelmat esitellään seuraavaksi.

1.4 Esimerkkiongelmat

Esittelemme nyt julkaisujärjestelmän toteutuksessa kohtaamiamme ongelmia, jotka ratkaistiin SAX:n avulla. Ratkaisut on esitetty tarkemmin kappaleessa 4.

HTML-dokumentin muuntaminen korrektiksi XHTML-dokumentiksi.

Eräs julkaisujärjestelmän tarjoamista palveluista on käyttäjän valmiiden XHTML- ja HTML-dokumenttien sisällyttäminen julkaisujärjestelmän kautta tarjottaviin sivuihin. Koska julkaisujärjestelmä ei tue sisäisesti HTML-dokumentteja ja koska ulospäin tarjottavien sivujen tulee olla korrekteja, täytyy HTML-dokumentit muuttaa XHTML-muotoon.

Ongelmana oli siis muuntaa käyttäjältä saatu HTML-dokumentti korrektiksi XHTML-dokumentiksi. Ratkaisuna toteutettiin 7-osainen SAX-liukuhihna, jonka aloittaa SAX-jäsentäjänä toimiva sekalaismuotoista HTML:ää ymmärtävä TagSoup [13]. Muita liukuhihnan osia ovat muun muassa attribuuttien ja elementtien nimet korjaava NSFilter [15], muut kuin XHTML-elementit ja -attribuutit suodattava XHTMLCruftDropper sekä XHTML:n korrektiuden tarkistava RELAX NG -validaattori Jing [20].

XHTML-dokumentin DOM-esityksen sarjallistaminen HTML:ksi

Toteuttamamme julkaisujärjestelmä käsittelee sisäisesti dokumentteja XHTML-dokumentteina. TemplateEngine tuottaa sivuista DOM-esitykset, jotka sarjallistetaan edustapalvelimille. Vaikka useat nykyiset web-selaimet tukevatkin jo XHTML:ää, tietyt laajassakin käyttössä olevat selaimet (esimerkiksi Microsoft Internet Explorer) eivät kuitenkaan XHTML:ää ymmärrä. Julkaisujärjestelmämme täytyi siis tarjota valmiit sivut kaikkien selainten ymmärtämänä HTML:nä.

Ongelmana oli siis sarjallistaa web-sivun XHTML-muotoinen DOM-esitys HTML:ksi. Ongelma ratkaistiin muuntamalla DOM-esitys GNU JAXP:n [14] DomParser:in [18] avulla SAX-tapahtumiksi, joita vastaanottamaan toteutettiin HTML-sarjallistin, joka muunsi XHTML:n HTML-yhteensopivaksi.

Käyttäjän syöttämän uutistekstin muuntaminen korrektiksi XHTML-dokumentin DOM-esitykseksi

Eräs julkaisujärjestelmän käyttäjille tarjoamista palveluista on uutispalvelu. Uutispalvelun avulla käyttäjä voi erillisen web-käyttöliittymän avulla lisätä varsinaisella sivustolla näkyvään uutisipalveluun uutisia. Uutisteksteihin haluttiin mukaan kappalejaot ja linkit. Uutisten kirjoittajien ei kuitenkaan voitu olettaa osaavan kirjoittaa HTML:ää, saatika XHTML:ää. Avuksi otettiin syntaksi, jossa yksi tai useampi tyhjä rivi tulkitaan kappalevaihdoksi ja linkit kirjoitetaan muodossa [Linkin nimi osoite]. Hakasulkeiden sisällä viimeinen välilyönti siis erottaa linkin nimen osoitteesta. Tästä syntaksista käytetään myöhemmin nimeä uutissyntaksi. Seuraavassa esimerkki uutissyntaksilla kirjoitetusta uutistekstistä ja sitä vastaavasta XHTML-tekstistä.

> Activision ja id Software vahvistivat [Doom III http://www.doom3.com]-räiskinnän ilmestyvän Yhdysvalloissa kesäkuun puolivälissä. Euroopassa virallinen julkaisupäivä on edelleen vuoden lopussa.
>
> Uskoo ken tahtoo.

```
<body>
<p>Activision ja id Software vahvistivat <a href=”http://www.doom3.com”>Doom
III-räiskinnän ilmestyvän Yhdysvalloissa kesäkuun puolivälissä. Euroopassa virallinen julkaisupäivä on edelleen vuoden lopussa. </p>
<p>Uskoo ken tahtoo.</p> </body>
```

Varsinaisen uutistekstin ja muiden uutiseen liittyvien tietojen, kuten uutisen otsikon, yhdistäminen yhdeksi XHTML-dokumentiksi tehtiin DOM:n avulla, joten ongelmana oli siis muuntaa uutissyntaksilla kirjoitettu uutisteksti korrektiksi XHTML-dokumentin DOM-esitykseksi.

Ratkaisimme ongelman toteuttamalla oman uutisjäsentäjän, joka skannaa uutistekstiä ja lähettää siitä XHTML:n mukaisia SAX-tapahtumia DomConsumerille, joka puolestaan rakentaa näiden tapahtumien perusteella uutisen XHTML-DOM-esityksen.

2 Kontekstin kuvaus

Tässä kappaleessa esitellään lyhyesti projektin tausta ja projektin tuloksena syntynyt WebCMS-julkaisujärjestelmä.

2.1 Projektin tausta

Kuuden hengen ryhmätyönä toteutettu julkaisujärjestelmä on osa Teknillisen korkeakoulun T-106.720 Ohjelmistotekniikan projekti -kurssia. Tämän kurssin

pääpaino on valmiin tuotteen toteuttamisen sijaan ohjelmistotekniikan oppimisessa, mikä vaikutti osaltaan joihinkin tekemiimme ratkaisuihin. Erityisesti toinen kirjoittajista oli innokas opettelemaan SAX:n käyttöä, joten SAX:ia saatettiin käyttää myös tilanteissa, joissa toiset ratkaisut olisivat voineet olla parempia.

2.2 WebCMS-julkaisujärjestelmä

WebCMS-julkaisujärjestelmä on web-sivustojen julkaisemiseen tarkoitettu järjestelmä. WebCMS helpottaa sivustojen ylläpitäjien tehtäviä tarjoamalla ylläpitäjien käyttöön erilaisia valmiiksi toteutettuja palveluja, jotka ylläpitäjän on helppo lisätä sivustoille. Tämä mahdollistaa ylläpidon keskittymisen itse palvelujen sisällön tuottamiseen. Tarjottavia palveluja ovat esimerkiksi uutispalvelu, valmiin XHTML:n upottaminen sivuun ja haku-palvelu.

Teknisesti yksi WebCMS:n johtoajatuksista on koostaa web-sivut yhdistelemällä ne eri palveluihin liittyvien XHTML-dokumenttien puuesityksistä. Ylipäätään sisäisesti järjestelmä käsittelee dokumentteja (uutistekstejä, sivupohjia jne.) XHTML-muodossa. HTML:ää tuetaan ainoastaan järjestelmän ulkopuolisissa liittymissä, eli palvelujen sisällön syöttämisen yhteydessa ja valmista sivua ulospäin palveltaessa. Näihin kohtiin keskittyvät myös tässä raportissa esitetyt esimerkkiongelmat.

WebCMS on toteutettu Javalla, dokumenttien puuesityksissä käytetään DOM:ia.

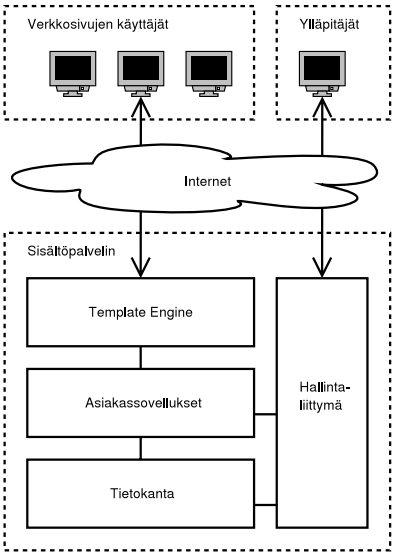
WebCMS:n toteutus on jaettu useampaan eri osaan, rakenne on esitelty kuvassa 1. Rakennekuva, kuten myös seuraavat eri osien toimintaa kuvaavat tekstit on lainattu suoraan projektin välimuistia ja edutapalvelimia käsittelevästä raportista [22]. Tarkemmin eri osioiden toimintaa on kuvattu erillisissä raporteissa [21], [22], [23] ja [24].

Template Engine näkyy ulospäin HTTP-palvelimena. Sisäisesti Template Engine toimii template-ohjattuna sisältöintegraattorina, joka yhdistelee asiakassovellusten tarjoaman sisällön sivukokonaisuuksiksi. [22]

Asiakassovelluksiin on toteutettu julkaisujärjestelmään liitettyjen palveluiden logiikka. Asiakassovellukset suorittavat Template Engineltä saamiensa pyyntöjen perusteella kyselyt tietokantaan ja muotoilevat tietokannasta saadut vastaukset Template Enginen ymmärtämään muotoon. Kutakin palvelua kohden tarvitaan erillinen asiakassovellus. [22]

Tietokanta toimii taltiona, johon voidaan tallentaa sekä staattisia tiedostoja (esim. template-tiedostot ja kuvat) että sovelluskohtaisia sisältöalkioita (mm. uutispalvelun uutiset tai tapahtumakalenterin tapahtumat). Taltion sisältöä voidaan käsitellä vain ja ainoastaan tietokantajärjestelmän rajapinnan kautta. [22]

Hallintajärjestelmällä hallitaan WebCMS-julkaisujärjestelmän toimintaa. Hallintajärjestelmä jakautuu eri sidosryhmille tarkoitettuihin osiin, joista kullekin tarjotaan rajatut mahdollisuudet julkaisujärjestelmän tietokantasisällön käsittelyyn. [22]



Kuva 1: WebCMS-julkaisujärjestelmän rakenne

3 Vaihtoehtoisista ratkaisuista

Jokaisen kolmen esimerkkiongelman ratkaisemiseksi käytettiin SAX:ia. SAX ei luonnollisestikaan ollut ainut, eikä aina kenties paraskaan ratkaisuvaihtoehto. Mielenkiinto oppia SAX:n käyttöä vähensi kenties intoa etsiä vaihtoehtoisia ratkaisuja, mutta loppujen lopuksi voidaan todeta, että SAX:illa toteutetut ratkaisut ovat varsin toimivia ja elegantteja. Seuraavassa kuitenkin muutamia vaihtoehtoisia ratkaisutapoja.

XHTML-DOM:in sarjallistaminen HTML:ksi olisi voitu toteuttaa esimerkiksi navigoimalla puu läpi syvyysjärjestyksessä ja sarjallistamalla sitä läpikäynnin edetessä. Sitä kumpi ratkaisu on parempi on vaikea sanoa.

Uutistekstin muuntamisessa olisi kenties voitu jättää SAX kokonaan pois muuttamalla uutisjäsentäjää siten, että se olisi suoraan tuottanut XHTML-merkkijonon, josta sitten olisi muodostettu DOM-esitys jollakin merkkijonoja syövällä DOM-rakentajalla.

HTML:n muuntamiseen XHTML:ksi olisi löytynyt valmiitakin työkaluja. Osa näistä ei kuitenkaan ole vapaita ohjelmia ja kaikkien työkalujen integrointi Java-ohjelmaan ei olisi välttämättä sujunut ongelmitta.

4 Omat ratkaisumme

4.1 Yleistä SAX:sta

SAX [7] eli Simple API for XML on siis alunperin vain Javalle tarkoitettu XML-ohjelmointirajapinta, joka perustuu jäsennystapahtumien raportointiin funktiokutsujen avulla. Tapahtumapohjaisena rajapintana sen etuna on pieni muistintarve: kerrallaan käsiteltävästä dokumentista on vain juuri käsiteltävänä oleva kohta. Toisaalta suurimpana puutteena on tapahtumapohjaiselle rajapinnalle tyypillinen dokumentissa navigoituvuuden puute: SAX-jäsentäjä scannaa dokumentin läpi alusta loppuun. Rajapintana SAX on varsin selkeä ja yksinkertainen. Alunperin Javalle tarkoitettuna se myös toteuttaa olio-ohjelmoinnin periaatteita hyvin.

Yleensä SAX-sovellukset koostuvat kahdesta osasta: XML-jäsentäjästä, joka lukee XML-dokumenttia ja lähettää siitä SAX-jäsennystapahtumia ja ContentHandleristä, joka vastaanottaa SAX-tapahtumia. ContentHandler on SAX:n määrittelemä rajapinta joka SAX-tapahtumia vastaanottavan olion täytyy toteuttaa. Tärkeimämt ContentHandlerin määrittelemät metodit on esitetty taulukossa 1.

Taulukko 1: ContentHandler:n tärkeimmät metodit

characters(char[] ch, int start, int length)	Receive notification of character data.
void endDocument()	Receive notification of the end of a document.
void endElement(java.lang.String uri, java.lang.StringlocalName, java.lang.String qName)	Receive notification of the end of an element.
void startDocument()	Receive notification of the beginning of a document.
void startElement(java.lang.String uri, java.lang.String localName,java.lang.String qName, Attributes atts)	Receive notification of the beginning of an element.

Ketjuttamalla ContentHandlereitä saadaan muodostettua SAX-liukuhihna tai tarvittaessa SAX-verkko. SAX-liukuhihnan ensimmäisenä osana toimii usein SAX-jäsentäjä, joka lukee XHTML-dokumenttia. Liukuhihnojen muodostamiseen on tarjolla valmiita apuluokkia. Näiden heikkoutena on kuitenkin se, että ne eivät tue komponettejä joiden konstruktorit tarvitsevat rajapintojen ulkopuolisia parametrejä.

4.2 Ratkaisut esimerkkiongelmiin

Tässä kappaleessa esittelemme SAX-pohjaisen ratkaisun kuhunkin edellä mainittuun esimerkkiongelmaan.

HTML-dokumentin muuntaminen korrektiksi XHTML:ksi

Ongelmana oli siis muuntaa käyttäjältä saatu HTML-dokumentti korrektiksi XHTML-dokumentiksi. Ongelman ratkaisemiseksi toteutimme 7-osaisen SAX-liukuhihnan, joka on esitetty kuvassa 2.

Käsiteltäessä HTML-dokumentteja SAX:n avulla ongelmaksi muodostuu usein se, että useimpia SAX-jäsentäjiä varten jäsennettävän dokumentin tulee olla hyvämuotoista XML:ää. Käyttäjiltä saatavien HTML-dokumenttien ei voida olettaa täyttyvän tämän ehdon. Ratkaisuksi löysimme TagSoup-nimisen SAX-yhteensopivan jäsentäjän. TagSoup on SAX-jäsentäjä, joka lukee hyvämuotoisen XML-dokumentin sijaan sekalaismuotoisia HTML-dokumentteja ja mikäli vain mahdollista, tuottaa niistä hyvämuotoista XML:ää vastaavia SAX-tapahtumia [13]. TagSoup:n avulla pääsemme kätevästi heti suoraan käsittelemään SAX-tapahtumia ja se siis aloittaa ratkaisun SAX-liukuhihnan.

Seuraavana liukuhihnassa on NSFilter, joka on GNU JAXP-pakettiin [14] kuuluva SAX-suodatin. Sen tehtävänä on varmistaa, että elementtien ja attribuuttien nimillä on pätevät etuliitteet ja että nämä etuliitteet on määritelty oikein. [15]

NSFilter:iä seuraa XHTMLCruftDropper, joka suodattaa elementeistä ja attribuuteista pois ne, jotka rikkovat XHTML:n korrektiuden.

Jotta voitiin olla varmoja, että TagSoup:n, NSFilter:n ja XHTMLCruftDropper:n tuottama XHTML on korrektia, halusimme varmistaa korrektiuden käyttämällä erillistä XHTML-validaattoria. Kätevimmäksi validaattoriksi osoittautui Thai Open Source Software Center Ltd:n RELAX NG[19]-validaattori nimeltä Jing [20]. Jing ottaa vastaan SAX-tapahtumia ja validoi niitä mukana seuranneen XHTML RELAX NG -scheman mukaisesti. Korrektiusvirheet Jing raportoi SAX-virheenkäsittelijän avulla.

Jingin pienenä puutteena on se, että sitä ei voi käyttää SAX-liukuhihnan keskellä SAX-suodattimena, vaan ainoastaan SAX-liukuhihnan päättävänä osana. Tämän takia Jingiä edeltää liukuhihnassa GNU JAXP:n [14] TeeConsumer [16], joka haaroittaa SAX-liukuhihnan kahtia lähettämällä vastaanottamansa SAX-tapahtumat edelleen kahdelle eri vastaanottajalle.

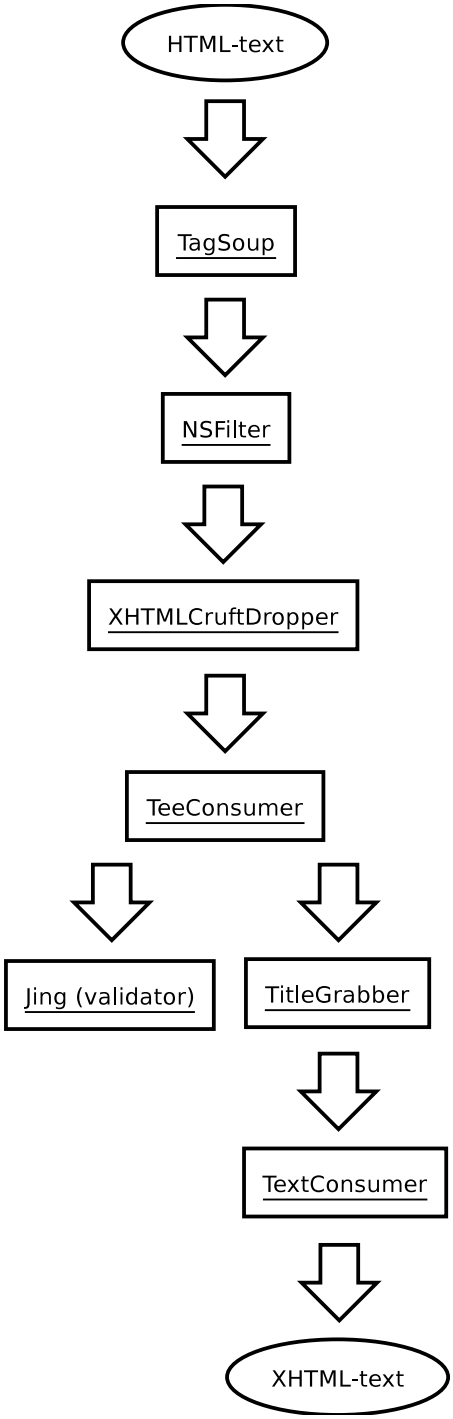
Viimeistä edellisenä liukuhihnassa on TitleGrabber, jonka tehtävänä on ainoastaan poimia dokumentin title-elementin sisältö talteen. Otsikon poimiminen ei varsinaisesti liity tämän liukuhihnan muuhun toimintaan, mutta ohjelman toiminnan kannalta se oli kätevää liittää juuri tähän kohtaan.

Liukuhihnan päättää TextConsumer, joka muodostaa saamiensa SAX-tapahtumien mukaisen XHTML-dokumenttitekstin.

XHTML-dokumentin DOM-esityksen sarjallistaminen HTML:ksi

Ongelmana oli siis sarjallistaa web-sivun XHTML-muotoinen DOM-esitys HTML:ksi.

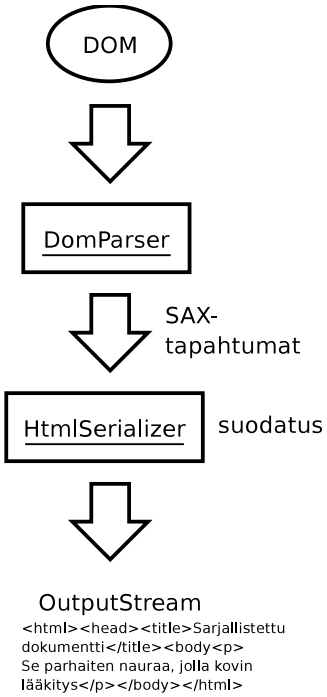
Ongelman ratkaisemiseksi toteutimme SAX-tapahtumia vastaanottavan sarjallistimen, joka poimii tapahtumavirrasta HTML:n kanssa epäyhteensopivat XHTML-elementit ja muuntaa ne HTML-yhteensopiviksi. Samalla varmistetaan, ettei



Kuva 2: HTML:n muuntaminen XHTML:ksi

julkaisujärjestelmän template-kielestä jää jälkia loppulliseen HTML-dokumentiin. DOM-esityksen muuntaminen SAX-tapahtumiksi tehtiin valmiin GNU JAXP:n

[14] DomParser:n [18]. DomParser siis läpikäy DOM:n muodostaen siitä samalla SAX-tapahtumia. Sarjallistusta on havainnollistettu kuvassa 3.



Kuva 3: XHTML-DOM:in sarjallistaminen HTML:ksi

Käyttäjän kirjoittaman uutistekstin muuntaminen korrektiksi XHTML:ksi

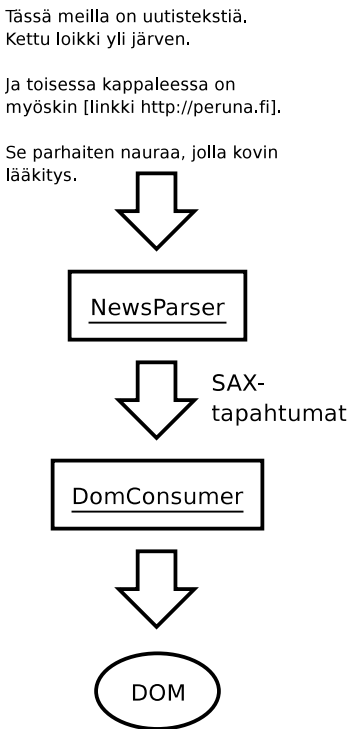
Ongelmana oli siis muuntaa uutis-syntaksilla kirjoitettu uutisteksti korrektiksi XHTML-dokumentin DOM-esitykseksi.

Ongelman ratkaisemiseksi toteutimme yksinkertaisen uutisjäsentäjän, joka skannaa uutistekstin läpi, tulkitsee erikoistapaukset kuten linkit ja kappaleenvaihdot ja luo samalla uutistekstistä XHTML:n mukaisia SAX-tapahtumia. SAX-tapahtumat vastaanottaa GNU JAXP:n [14] DomConsumer [**?**], joka rakentaa niiden perusteella uutistekstiä vastaavan XHTML-DOM-esityksen. Ratkaisua on havainnollistettu kuvassa 4.

Uutisparseri vastaa syntyneen XHTML:n korrektiudesta, joten erillistä validaattoria ei tarvittu.

5 Ratkaisujen ja SAX:n arviointia

Tässä kappalessa arvioimme esitettyjen ratkaisujen toimivuutta. Lisäksi tuomme esiin ratkaisujen toteutuksessa ilmenneitä SAX:in käyttöön liittyviä etuja ja ongelmakohtia.



Kuva 4: Uutistekstin muuntaminen DOM:ksi

5.1 Ratkaisujen toimivuus

Esitetyt ratkaisut esimerkkiongelmiin osoittautuivat varsin toimiviksi. SAX-komponenttien toteuttaminen onnistui suuremmitta ongelmitta ja joitakin komponentteja pystyttiin hyödyntämään useammassa kohtaa julkaisujärjestelmää.

Ylipäätään alun opetteluvaiheen jälkeen SAX:n kanssa toimiminen sujui hyvin ja toteutuksista tuli mielestämme selkeitä ja ymmärrettäviä. Tämä oli ennakko-odotuksiin nähden pieni yllätys, koska usein esimerkiksi DOM:ia ja SAX:ia vertailtaessa mainitaan, että DOM:ia käyttävät ohjelmat olisivat SAX:ia käyttäviä selkeämpiä [12]. Havaintomme olivat tältä osin melkeinpä päinvastaisia, tähän tosin saattaa vaikuttaa myös se, että Javan DOM-toteutukset eivät ole erityisen 'Javamaisia'. Toisaalta liian tarkkaa vertailua ei tältä osin kannata edes yrittää tehdä, koska DOM ja SAX ratkaisevat yleensä erilaisia ongelmia ja ratkaisun monimutkaisuus riippuu yleensä juuri ongelmasta.

5.2 SAX:in edut

Kaikenkaikkiaan SAX-osoittautui siis varsin toimivaksi tekniikaksi esimerkkiongelmia ratkaistessamme. Seuraavassa on lueteltu projektimme kannalta SAX:n parhaita puolia.

- **Nopeus ja pieni muistin tarve**
  SAX:n havaittiin olevan nopea ja muistin tarve on tapahtumapohjaisille SAX-ohjelmointirajapinnoille tyypillisen pieni.

- **Toimiva liukuhihnamalli**
  SAX:n tapa toteuttaa monimutkainen XML:n käsittely kokoamalla yksinkertaisemmista komponenteistä SAX-liukuhihna mahdollistaa komponenttien joustavan yhdistelyn ja uudelleenkäytön.

- **Valmiit komponentit**
  Toimivista liukuhihna- ja komponenttimalleista johtuen erilaisia valmiita suodattimia, jäsentäjiä ja muita komponentteja on hyvin saatavilla. Tässäkin projektissa käytettiin runsaasti valmiita komponentteja kuten Jing[20], NSFilter[15] ja TagSoup[13].

- **Valmiit adapteriluokat**
  Yleisimmille SAX-komponenteille, kuten SAX-suodattimille on tarjolla valmiita adapteriluokkia, joista perimällä on kätevä toteuttaa omia komponentteja.

- **Rajapintojen selkeä määrittely**
  Kenties Java-taustastaan johtuen SAX:n määrittelemät rajapinnat ovat ainakin Javalla ohjelmoitaessa varsin selkeitä ja toimivia. Hyvänä vertailukohtana toimi projektin aikanä käytetty DOM:n Java-toteutus, jota yleisemminkin pidetään olio-ohjelmoinnin kannalta rumana. [12]

5.3 SAX:in ongelmat

SAX:n myönteisten puolten lisäksi kohtasimme myös joitakin ongelmia. Nämä ongelmat olivat kuitenkin lähinnä epäkäytännöllisyyksiä, jotka oli useimmiten suhteellisen helppo kiertää. Ongelmia on listattu seuraavassa:

- **Vaatimus XML-hyvämuotoisuudesta**
  Useimmat SAX-jäsentäjät vaativat, että jäsennettävä dokumentti on hyvämuotoista XML:ää. Käytännössä tämä aiheutti pieniä ongelmia esimerkiksi tilanteissa, joissa dokumenttiin piti liittää keinotekoinen juurielementti hyvämuotoisuuden varmistamiseksi.

- **Tavallisen tekstin käsittely**
  SAX:n tapa raportoida tagien väliset merkkijonot useampina characters()-tapahtumina, jotka ohjelmoijan on itse yhdistettävä on tietyissä tapauksissa varsin epäkäytännöllinen.

- **Funktiokutsujen määrä**
  Verrattuna suoraan merkkijonomanipulaatioon, SAX:ia käytettäessä funktiokutsujen määrä saattaa olla huomattavasti suurempi. Tämä luonnollisesti hidastaa ohjelman toimintaa, mutta usein

kuten tämänkin raportin esimerkkiongelmissa, tällä hidastumisella ei ole käytännössä mitään merkitystä. Lisäksi verrattuna esimerkiksi Javan DOM toteutuksiin lukuisine olioineen ja metodeineen SAX on hyvinkin nopea ja suoraviivainen.

6 Yhteenveto

Esimerkkiongelmien kannalta SAX osoittautui toimivaksi tekniikaksi: kaikkiin kolmeen ongelmaan saatiin toteutettua toimivat ja suht elegantit ratkaisut. SAX:in liukuhihnamalli mahdollisti valmiiden ja itse toteutettujen SAX-komponenttien joustavan yhdistelyn.

Yleisesti ottaen SAX:in vahvuudeksi osoittautui myös komponenttien uudelleenkäytön helppous: muutamia komponentteja pystyttiin hyödyntämään julkaisujärjestelmän eri osissa ja ylipäätään SAX-komponenttien todettin usein olevan luonteeltaan itsenäisiä kokonaisuuksia, joita voidaan käyttää hyvinkin erilaisissa tilanteissa. Samoin osoittautui, että ainakin julkaisujärjestelmän kannalta hyödyllisiä valmiita SAX-komponentteja on tarjolla varsin paljon.

SAX:in pienet puutteet eivät oikeastaan rajoittaneet toiminnallisuutta vaan olivat lähinnä epäkäytännöllisyyksiä, jotka voitiin kiertää kohtuullisen pienellä vaivalla.

Mikään yleisratkaisu kaikkeen XML:n käsittelyyn SAX ei tietenkään ole, mutta tiettyjen ongelmien, kuten julkaisujärjestelmämme esimerkkiongelmat, ratkaisemiseen se on oiva työkalu.

Viitteet

[1] Information Processing – Text and Office Systems – Standard Generalized Markup Language (SGML), ISO 8879:1986. Viitattu 2004-05-25] URL: http://www.iso.ch/cate/d16387.html.

HTML 4.01 Specification. D. Raggett, A. Le Hors, I. Jacobs, 24 December 1999. Viitattu [2004-05-25] URL: http://www.w3.org/TR/1999/REC-html401-19991224.

[2] HTML 4.01 Specification. D. Raggett, A. Le Hors, I. Jacobs, 24 December 1999. Viitattu [2004-05-25] URL: http://www.w3.org/TR/1999/REC-html401-19991224.

[3] Extensible Markup Language (XML). W3C. Viitattu [2004-05-25] URL: http://www.w3.org/XML/.

[4] XHTML 1.0 The Extensible HyperText Markup Language (Second Edition). W3C. Viitattu [2004-05-25] URL: http://www.w3.org/TR/xhtml1/.

[5] XHTML-1.0-Strict. W3C. Viitattu [2004-05-25] URL: http://www.w3.org/TR/xhtml1/dtds.html#a_dtd_XHTML-1.0-Strict.

[6] The benefits of XHTML. Daniel M. Frommelt. 2003-02-13. Viitattu [2004-05-25] URL: http://www.uwplatt.edu/news/testbed/13/archive/2003/387.html.

[7] About SAX. David Meggison, David Brownell (sivuston ylläpitäjät). Viitattu [2004-05-25] URL: http://www.saxproject.org.

[8] Document Object Model (DOM). W3C. Viitattu [2004-05-25] URL: http://http://www.w3.org/DOM/

[9] JDOM. Viitattu [2004-05-25] URL: http://http://www.jdom.org//

[10] XOM. Viitattu [2004-05-25] URL: http://www.cafeconleche.org/XOM

[11] Java API for XML Processing (JAXP). Sun Microsystems. Viitattu [2004-05-25] URL: http://java.sun.com/xml/jaxp/.

[12] What's Wrong with XML APIs (and how to fix them). Elliotte Rusty Harold. 2003. URL: http://www.cafeconleche.org/XOM/whatswrong/text0.html.

[13] TagSoup - Keep On Truckin'. John Cowan. Viitattu[2004-05-25 URL: http://mercury.ccil.org/~cowan/XML/tagsoup/.

[14] The GNU JAXP Project. Free Software Foundation Inc. Viitattu[2004-05-25] URL: http://www.gnu.org/software/classpathx/jaxp/.

[15] gnu.xml.pipeline.NSFilter JavaDoc. Viitattu[2004-05-25] URL: http://xmlconf.sourceforge.net/java/apidoc/gnu/xml/pipeline/NSFilter.html.

[16] gnu.xml.pipeline.TeeConsumer JavaDoc. Viitattu[2004-05-25] URL: http://xmlconf.sourceforge.net/java/apidoc/gnu/xml/pipeline/TeeConsumer.html.

[17] gnu.xml.pipeline.DomConsumer JavaDoc. Viitattu[2004-05-25] URL: http://xmlconf.sourceforge.net/java/apidoc/gnu/xml/pipeline/DomConsumer.html.

[18] gnu.xml.util.DomParser JavaDoc. Viitattu[2004-05-25] URL: http://xmlconf.sourceforge.net/java/apidoc/gnu/xml/util/DomParser.html.

[19] RELAX NG home page. James Clark (sivuston ylläpitäjä). Viitattu[2004-05-25] URL: http://www.relaxng.org/.

[20] Jing, A RELAX NG validator in Java. James Clark (sivuston ylläpitäjä). Viitattu[2004-05-25] URL: http://www.thaiopensource.com/relaxng/jing.html.

[21] Assembling Web Pages Using Document Trees. Henri Sivonen, Yrjö Kari-Koskinen & Jussi Koskela. 2004

[22] Dynaamisen HTTP-sisällön tarjoaminen välimuistipalvelimelta. Jussi Koskela & Henri Sivonen. 2004

[23] WWW-käyttöliittymän rakentaminen: DOMin ja MVC:n käyttö WebUI:ssa. Yrjö Kari-Koskinen, Henri Sivonen & Taavi Hupponen. 2004.

[24] Tietokanta WWW-julkaisujärjestelmälle. Oskar Ojala & Antti Saarinen. 2004.