Opinnäytetyö (AMK)

Tietotekniikka

Sulautetut järjestelmät

2010

Ilkka Helenius

XML-RAJAPINNAN TOTEUTUS JAVALLA

-määrittelytiedoston luku Pulu-sovellukseen

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietotekniikka | Sulautetut järjestelmät

Kesäkuu 2010| Sivumäärä

Ohjaaja: Jari-Pekka Paalassalo, tekn. lis., Yliopettaja

Ilkka Helenius

XML-RAJAPINNAN TOTEUTUS JAVALLA

Tässä työssä suunniteltiin ja kehitettiin uudenlainen tapa siirtää tiedot java-sovellukselle. Sovellus hyödyntää GPS-dataa paikan kohdentamiseen. Sovellukselle siirrettävät tiedot sisältävät koordinaatti tietoja, kohteiden nimiä sekä tiedostojen nimiä joita käytetään kohteilla esitettävissä esityksissä. Vanhan tiedonsiirtotavan ongelmat olivat sen sekavuus ja virheherkkyys.

Tiedot tuli esittää ymmärrettävässä muodossa, sekä tietoja piti pystyä muuttamaan helposti. Sovelluksessa piti kuitenkin säilyä vanha tiedon prosessointi tapa, jotta vanhat esitykset olisivat vielä käyttökelpoisia.

Työssä päädyttiin käyttämään XML:ää tietojen säilyttämiseen ja Javan DOM-parseria tietojen purkamiseen sovelluksen käyttöön. Päädyttiin DOM-parseriin, koska se purki tiedot hyvin samankaltaisesti kuin aikaisempi menetelmä. DOM-parseri ei myöskään vaatinut mitään ulkopuolisia kirjastoja, vaan pystyttiin käyttämään javan omia kirjastoja.

XML:n käyttöönoton jälkeen aukeni myös muita mahdollisuuksia. Sovellukseen oli helpompi lisätä uusia ominaisuuksia, kuten kohteilla esitettävät kysymykset. Tämä olisi ollut vaikea toteuttaa aikaisemmalla menetelmällä, koska aikaisempi menetelmä käytti CSV:tä eli pilkuilla eroteltuja arvoja.

Työstä on hyötyä myös sovelluksen kehittymisen kannalta. Ilman XML:n käyttöönottoa sovelluksen kehitys jäisi hyvin minimaaliseksi.

ASIASANAT:

DOM, GPS, Java, kehittäminen, parserointi, XML

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology | Embedded Systems

June 2010 | Total number of pages

Instructor: Jari-Pekka Paalassalo, Lic.Tech., Principal Lecturer

Ilkka Helenius

IMPLEMENTING XML INTERFACE WITH JAVA

In this thesis a new way of transferring information to the software was designed and implemented to simplify the data file containing the information needed. The old version of the file was way too complicated to quickly understand what information it held and where to apply new data.

Software uses GPS data to pinpoint the current location. The transferred information contains details about coordinates, target names and filenames. The most common problems of the old method were it was very error prone and confusing.

The goal was to make the data file more understandable. In addition, altering the information should be easier. There also had to be a support for the old method to keep old presentations usable.

XML was chosen for the data file format and DOM parser for parsing the file to the software. DOM parser was chosen because it functioned very similarly to the previous parser. DOM parser also worked with the existing libraries in Java, so no external libraries were needed.

After taking on XML, new possibilities opened to the software. New features were easier to apply to the software, like adding a question to a target presentation. This would have been difficult with the old method because the system used CSV (Comma Separated Values).

This thesis also has benefits for the future development of the software. Without applying XML the software would have had trouble evolving much further.

KEYWORDS:

development, DOM, GPS, Java, parsing, XML

SISÄLTÖ

KUVAT

Symbolit ja lyhenteet

API	Application Programming Interface, ohjelmointirajapinta
CSV	Comma Separated Values, pilkulla erotellut arvot
DOM	Document Object Model, XML-rajapinta
GPS	Global Positioning System, paikannusjärjestelmä
JTS	Java Topology Suite, Java geometria kirjasto
SAX	Simple API for XML, XML-rajapinta
XML	eXtensible Markup Language, merkintäkieli
Xpath	XML-rajapinta

1 Johdanto

Opinnäytetyön aiheena on Java-sovelluksen päivittäminen ja keskeiseksi teemaksi kyseissä sovelluksessa nousi XML ja sen hyödyntäminen Java-sovelluksessa. Sovellukseen tarvittiin XML-tuki laajentamaan sovelluksen mahdollisuuksia ja helpottamaan syötetyn tiedon ymmärtämisessä.

Tavoitteena oli saada syötetyistä tiedoista sellaista, että sitä opittaisiin tulkitsemaan lyhyen opastuksen jälkeen. Samalla myös pyrittiin vähentämään mahdollisten virheiden syntymistä ja olemassaolevien virheiden paikannallistamista.

Työssä käsitellään myös uusien osien lisäämistä sovellukseen hyödyntämällä uusia tiedonsyöttömenetelmiä.

2 Taustatiedot

2.1 PULU-hanke

PULU tulee sanoista ”Pulpetista ulos”. PULU:n idea on antaa uudenlainen oppimiselämys peruskoulun oppilaille. Tavallisen luokassa oppimisen sijaan viedään oppilaat ulos luontoon, jolloin saadaan pysyvämpi muisto oppilaalle kokemastaan.

Hanketta on kokeiltu Eurassa sekä Naantalissa positiivisin tuloksin. Oppilaat ovat 2 - 4 hengen ryhmissä osallistuneet oppimiskierroksille, joissa kierretään kunnan historiallisesti tärkeitä kohteita.

Oppilailla on käytössään paneelitietokone tai minikannettava, johon on asennettu PULU-sovellus. PULU-sovellus käyttää hyväkseen joko laitteeseen integroitua tai ulkoista GPS-paikanninta, jonka avulla sovellus tietää oppilasryhmän sijainnin. Ryhmän saapuessa historialliselle kohteelle, joka kuuluu oppimiskierrokseen, sovellus käynnistää esityksen kohteesta.

Tällä tavalla oppilas pääsee lähemmin tutustumaan omaan paikallishistoriaansa ja saa paremman kuvan oppimistaan asioista. Sovellus on interaktiivinen, joka myös edistää oppimista. Kohteessa oppilaalle saatetaan esittää kysymyksiä kohteen esityksestä, jolloin oppiminen tapahtuu asioiden pohtimisen ja tekemisen yhteydessä.

Esitys voi sisältää tekstipohjaisen tietopaketin lisäksi myös videon sekä erillisen äänitteen. Kysymykset ovat yleisesti ottaen monivalintakysymyksiä, mutta myös tekstipohjaiset vastaukset ovat mahdollisia.

2.2 Keskiajan Saaristomeri

Keskiajan saaristomeri on turun saaristoon kehitetty sovellus, joka auttaa löytämään saariston historiallisesti merkittävät kohteet. Sovellusta käytettiin pohjana PULU-hankkeessa. Insinööri Jesse Huurteen opinnäytetyönään kehittämä alusta soveltui hyvin PULU:n käyttötarkoitukseen.

Sovellus ilmoittaa, kun lähestytään kohdetta ja halutessaan käyttäjä voi käynnistää kohteesta kertovan esityksen. Esitykseen kuului usein tekstipohjaisen infromaation lisäksi video- sekä kuvamateriaalit.

3 XML-merkintäkieli

XML on yleisessä käytössä oleva merkintäkieli. Kuten HTML, XML perustuu SGML-kieleen, jonka merkintätapa on rakenteellinen dokumentti. XML:ään voidaan sijoittaa tietoa, jota voidaan sitten hyödyntää laskutuksessa, tietokannoissa, internetsivuissa yms. XML on hyvin yksinkertainen tiedon talletustapa dokumenttiin. Tämä helpottaa useissa tapauksissa, joissa ei ole mahdollista käyttää erillistä tietokantaa tietojen tallettamiseen ja hakemiseen. XML on myös hyvin muistiystävällinen sillä se pystyy sisältämään paljon informaatiota hyvin pienessä muistiavaruudessa.

DOM ja SAX ovat XML:n ohjelmointirajapintoja, joita tarvitaan tiedon purkamiseen XML-dokumentista. Java ohjelmointiympäristössä XML-parseri on standardoitu, joten se tekeekin XML-dokumenttien parseroinnista ja käsittelystä erittäin helppoa.

3.1 SAX

SAX on eventtiperusteinen parseri, joka luo kutsuja elementtien alkuihin ja loppuhuihin. SAX sopii hyvin tilanteisiin, jolloin tarvittaessa hypätään tiettyyn kohtaan XML-dokumentissa ja luetaan tarvittu data. Tämän tyyppinen parserointi vaatii sitä, että ympäri sovellusta tulisi luoda kutsuja parserin kautta dokumenttiin tarvittavan tiedon puitteissa. [1][2]

Tämä on hyvin kevyt parseri ja nopeuttaisi sovellusta paljon, mutta tämän parserin käyttöönottoa varten tulisi muuttaa hyvin paljon sovelluksen rakennetta ja toiminnallisuutta. Jos on mahdollista, olisi ideaalista muuttaa sovelluksen parseri käyttämään SAX-parserointia juurikin sen keveyden takia. Samalla säästyisi tilaa muistiavaruudessa, kun ei tarvitsisi tallettaa kaikkea sitä dataa sovelluksen sisään.[1][2]

SAX-parseroinnin hyödyt,

- nopea parserointi ja
- kevyt.

SAX-parseroinnin haitat PULU:ssa,

- Sovelluksen koodin rakenne muuttuisi.

3.2 DOM

DOM-parseri toimii hyvin samalla periaatteella kuin CSV-parserikin. DOM-parseroinnissa iteroidaan koko dokumentti kerralla läpi ja otetaan kaikki data talteen sovelluksen muistiin. Tämä on hieman hitaampi vaihtoehto SAX-parseriin verrattuna, mutta koska DOM-parseri muistutti niin paljon CSV-

parserointi, soveltui se erittäin hyvin sovelluksen käyttötarkoituksiin. Hyvin vähän jouduttiin muokkaamaan itse sovellusta, mikä nopeutti XML-parseroinnin käyttöönottoa. Se myös mahdollisti CSV-parserin rinnakkaisen käytön silloin, kun XML-versiota tiedoista ei ollut saatavilla.

DOM-parseri lukee XML-dokumentin sovellukseen puumaisena mallina, jossa elementit ovat puun solmukohtia. Kaikki DOM-parserin käyttämät luokat, hyödyntävät *org.w3c.dom*-kirjastossa olevia luokkia.[3][4]

DOM-parseroinnin hyödyt,

- parseri ottaa koko dokumentin sovelluksen käyttöön ja
- sovelluksen muu koodi pysyy lähes muuttumattomana.

DOM-parseroinnin haitat PULU:ssa,

- vie paljon sovelluksen käytettävissä olevasta muistista.

3.3 Xpath

Xpath on hyvin yksinkertainen ns. XML-kyselykone. Xpath toimii hieman saman tyyppisesti kuin SQL tai muut kyselypohjaiset kielet. Xpathia ei alun perin ole suuniteltu Javaan ja aluksi olikin vain muutamia API:ja, jotka hyödynsivät Xpathia. Java 5:ssä tuo kuitenkin muuttui ja esiteltiin *java.xml.xpath*-paketti joka sisälsi Xpath API:n.[5]

Xpath soveltuu hyvin tiedonhakuun XML-dokumentista silloin, kun tiedetään mitä haetaan. PULU-sovellukseen Xpath ei kuitenkaan soveltunut niin hyvin. Xpath toimisi sovelluksessa, jos ensin iteroitaisiin DOM-parserilla kokonainen lista kohteiden nimistä, jonka jälkeen iteroitaisiin tarvittavat tiedot Xpath:lla nimilistaa käyttäen.

Xpathin kyselyt ovat yksinkertaisia, mutta Xpathin käyttäminen tässä tilanteessa olisi monimutkistanut parserointia huomattavasti enemmän, kuin oli tarpeen. Xpath osuu soveltuvuudeltaan tähän sovellukseen johonkin DOM:n ja SAX:n välimaastoon keveytensä takia. Monimutkaisten iterointien takia päätettiin

kuitenkin, että on parasta jättää Xpath kokonaan kokonaan pois tästä sovelluksesta.

Xpathin hyödyt,

- helpot hakukomennot.

Xpathin haitat PULU:ssa,

- koko dokumentti ei ole kerralla sovelluksen käytössä,
- sovelluksen rakenne muuttuisi ja
- kohteet pitäisi iteroida tapauskohtaisesti.

4 XML:n rakenne

Suunnitteluvaiheessa piti ottaa huomioon, että XML:n tulisi sisältää kaikki samat tiedot kuin CSV-mallikin ja tiedot tulisi olla helppo ohjata sovelluksessa eteenpäin oikeisiin paikkoihin.

Kaikki XML-elementit on suljettava juurielementin sisään, ja sovelluksen XML-dokumentti sisältää kohteiden tietoja. Juurielementille annettiin nimi ’*objects*’ ja yksittäiset kohteet suljettiin *’object’*-nimisen elementin sisään. Sovelluksen täytyy tietää kohteesta nimi, kohteen koordinaatit (longitude ja latitude), liipaisuetäisyys, videotiedoston polku, audiotiedoston polku, tekstitiedoston polku, taustakuvan polku sekä mahdollisen kartan karttatiedoston polku ja ylävasemman ja alaoiken kulman koordinaatit (longitude ja latitude).

Useiden eri tietokenttien takia on tärkeää, että XML on selkeästi ja oikealla tavalla muodostettua eli XML:n tulee olla ns. well-formed.

4.1 Well-formed XML

Well-formed tarkoitta käytännössä sitä, ettei XML dokumentissa saa olla virheitä ja elementtien tulee olla loogisesti nimettyjä. Elementit eivät myöskään saa mennä ristiin. Elementit ovat menneet ns. ristiin, kun elementti aukaistaan toisen elementin sisällä ja suljetaan saman elementin ulkopuolella.

Ristiinmenevä elementti aiheuttaa virheen dokumentissä, eikä dokumenttia voida silloin parseroida.

”Well-formedness” on tarkemmin määriteltynä: oikealla tavalla jäsennelty dokumentti, joka ei riko w3c:n asettamia lauseoppisääntöjä. Lauseoppisääntöjen lista on erittäin pitkä, mutta tärkeimmät ovat

- sisältää ainoastaan laillisia, merkistöstandardiin kuuluvia merkkejä
- erikoismerkkejä, kuten elementit sulkevat hakasulkeet, ei saa esiintyä XML-dokumentissa muualla kuin nille tarkoitetuilla paikoilla
- dokumentissa tulee olla juurielementin, joka sisältää kaikki muut elementit
- kaikkien elementtien tulee olla oikein suljettu oikein, yhtään elementtiä ei saa puuttua eikä elementit saa mennä ristiin sekä tai
- aloittavan ja lopettavan elementti tagin tulee olla identtiset.

Näitä ohjeita noudattaen tehtiin ensimmäinen versio XML-dokumentista.[6](Koodi 1.)

Koodi 1. Esimerkki kohteen XML-muotoilusta.

```
<object>
  <name>Bårnholm</name>
  <actionLON>22.010851</actionLON>
  <actionLAT>60.205845</actionLAT>
  <TriggerDistance>2</TriggerDistance>
  <moviePath>/kohteet/Borgholm/borgholm_1.avi</moviePath>
  <textPath>/kohteet/Borgholm/teksti.htm</textPath>
  <audioPath></audioPath>
  <mapPath>/kohteet/Borgholm/na_barg_1.jpg</mapPath>
  <upLeftLON>21.98673576</upLeftLON>
  <upLeftLAT>60.23095319</upLeftLAT>
  <lowRightLON>22.08253689</lowRightLON>
  <lowRightLAT>60.18168158</lowRightLAT>
  <bgImagePath>/kohteet/Borgholm/borgholm.jpg</bgImagePath>
</object>
```

Elementit nimettiin CSV-tiedostosta löytyneiden englannin kielisten vastikkeiden mukaan. (Koodi 1.)

Koodi 2. Esimerkki karttakohteesta ennen muutoksia.

```
<object>
  <name>1_Nauvo-Rymättylä-Parainen</name>
  <actionLON>0</actionLON>
  <actionLAT>0</actionLAT>
  <TriggerDistance>0</TriggerDistance>
  <moviePath></moviePath>
  <textPath></textPath>
  <audioPath></audioPath>
  <mapPath>/kohteet/defaultmap/na_rym_pa.jpg</mapPath>
  <upLeftLON>21.83930393</upLeftLON>
  <upLeftLAT>60.33621986</upLeftLAT>
  <lowRightLON>22.17744769</lowRightLON>
  <lowRightLAT>60.16329721</lowRightLAT>
  <bgImagePath></bgImagePath>
</object>
```

Kartta tunnistettiin nimen alussa olevasta numerosta, joka myös määritti kartan skaalan sovelluksessa. Kartan koordinaatit, kuten myös liipaisuetäisyys, olivat aina 0.(Koodi 2)

Alun perin kartat ja kohteet esitettiin samalla tavalla, mutta kartat haluttiin myöhemmin erotella selvemmin kohteista. Karttojen erottelu muista kohteista selkeytti XML:n rakennetta ja teki XML-dokumentista helpommin ymmärrettävän.(Koodi 3.)

Koodi 3. Esimerkki karttakohteesta muutosten jälkeen.

```
<map>
  <name>Luistarin kalmiston arkeologinen kartta</name>
  <mapScale>1</mapScale>
  <mapPath>/eura/luistarikartta.jpg</mapPath>
  <upLeftLON>22.144923</upLeftLON>
  <upLeftLAT>61.112982</upLeftLAT>
  <lowRightLON>22.147239</lowRightLON>
  <lowRightLAT>61.111867</lowRightLAT>
  <bgImagePath></bgImagePath>
</map>
```

Karttakohteista poistettiin kaikki ylimääräiset elementit ja muutettiin elementtien nimet paremmin sopiviksi. Ensimmäisessä versiossa ei ollut juurikaan eroa kartta- ja normaalikohteen välillä, koska yritettiin pitää tiedonsiirto sovelluksella

mahdollisimman identtisenä vanhan tiedonsiirtomenetelmän kanssa. Sovelluksen kartoittamisen ja paremman ymmärtämisen jälkeen oli helpompaa muokata sisään menevää dataa, kun pystyttiin täysin kontrolloimaan tiedon siirtymistä oikeisiin paikkoihin.

Myös normaalien kohteiden rakennetta muutettiin, koska tiettyjen elementtien nimet eivät sopineet niiden tarkoitukseen. (Koodi 4.)

Koodi 4. Esimerkki kohteen XML-muotoilusta muutosten jälkeen.

```
<target>
  <name>Kauttuan linnavuori</name>
  <actionLON>22.15587</actionLON>
  <actionLAT>61.10544</actionLAT>
  <TriggerDistance>0.2</TriggerDistance>
  <moviePath></moviePath>
  <audioPath></audioPath>
  <textPath>/eura/linnavuoriA.html</textPath>
  <mapPath>/eura/kauttua_kartta1.jpg</mapPath>
  <upLeftLON>22.139466</upLeftLON>
  <upLeftLAT>61.121142</upLeftLAT>
  <lowRightLON>22.175862</lowRightLON>
  <lowRightLAT>61.102934</lowRightLAT>
  <bgImagePath>/eura/linnavuori1.jpg</bgImagePath>
</target>
```

Sovelluksen kehityttyä tarvittiin myös tiedot kohteiden kysymyksistä. Kaikissa kohteissa ei tietenkään ole kysymyksiä, mutta niissä kohteissa joissa on, tarvittiin tietynlainen esittelytapa kysymysten tietojen esittelyyn. Kysymyselementti sijaitsi aina sen kohde-elementin sisällä, johon se kuului. Näin parseroitaessa ei ollut epäselvää, mihin kysymys kuuluu ja milloin se pitää laukaista.(Koodi 5.)

Koodi 5. Esimerkki kohteesta, jossa on kysymys.

```
<target>
  <name>Käräjämäki</name>
  <actionLON>22.14042</actionLON>
  <actionLAT>61.128026</actionLAT>
  <TriggerDistance>0.1</TriggerDistance>
  <moviePath></moviePath>
  <audioPath></audioPath>
  <textPath>/eura/käräjä1.txt</textPath>
  <mapPath>/eura2008/nauris1.jpg</mapPath>
  <upLeftLON>22.129795</upLeftLON>
  <upLeftLAT>61.126397</upLeftLAT>
  <lowRightLON>22.144635</lowRightLON>
  <lowRightLAT>61.122974</lowRightLAT>
  <bgImagePath>/eura2008/käräjämäki2.jpg</bgImagePath>

  <questions>
    <!-- Huom: tekstirivin pituus määrää kysymysikkunan leveyden! -->
    <question>Kuka mahtaa omistaa arkeologien maasta kaivamat aarteet</question>
    <type>3</type>
    <numberOfOptions>5</numberOfOptions>

    <option>arkeologeille itselleen</option>
    <returnee></returnee>
    <points>-5</points>

    <option>maan omistajalle</option>
    <returnee></returnee>
    <points>-1</points>

    <option>ohikulkijoille</option>
    <returnee></returnee>
    <points>-1</points>

    <option>paikalliselle museolle</option>
    <returnee></returnee>
    <points>1</points>

    <option>museovirastolle</option>
    <returnee></returnee>
    <points>5</points>
  </questions>

</target>
```

5 XML:n käyttöönotto

XML:ää käyttöönotettaessa pidettiin vielä CSV-parserointi osana sovellusta vanhempien versioiden hyödyntämiseksi, mutta sovellus kuitenkin aina ensisijaisesti käytti XML-versiota reitin tiedoista, mikäli se oli saatavilla.

Suurin haaste XML:ää käyttöönotettaessa oli saada tietorakenteet yhtenäiseksi vanhan tallennusmuodon kanssa ja saada oikeat tiedot oikeaan muistipaikkaan sovelluksessa. Ensimmäisissä versioissa oli huomattavia puutteita joidenkin tietojen katoamisten takia.

Tietojen lisääminen osa kerrallaan sovellukseen auttoi paremmin hahmottamaan tiedon kulun parserilta sovelluksen muuttujiin ja sieltä kyselyn kautta tarvittaessa ruudulle. Kohta kohdalta etenimen antoi myös paremman kuvan sovelluksesta kokonaisuutena ja siitä miten tietyt luokat toimivat keskenään.

5.1 Toteutus

Toteutuksessa liikkeelle lähdettiin tutustumalla sovellukseen ja sen ominaisuuksiin. Tarkasteltiin sovelluksen eri toimintoja ja sitä miten eri kohteiden informaatio näytetään käyttäjälle. Pienen selailun jälkeen, kun sovelluksen toiminnat alkoivat selkeytyä, tutustuttiin CSV:hen. CSV oli teksti-tiedosto muodossa ja sisältö oli ensikatsomalta melko sekava. Alussa oli kommentoitu eri kohtien sisältöjen tarkoitukset, mutta itse tiedon sisällöstä oli vaikea sanoa mikä tieto merkitsi mitäkin. Lisäksi tietorakenteessa oli useita samankaltaisia tietoja, jotka helposti sekoitti keskenään.

Mikäli CSV:stä puuttui yksi pilkku tai jotkin tiedot olivat väärässä järjestyksessä, koko sovellus kaatuisi saman tien. Oli siis löydettävä uudenlainen tietojen säilytystapa, joka olisi selkeä sekä mahdolliset virheet tietorakenteissa pitäisi olla helposti käsiteltävissä ja mahdollisesti jopa ennalta vältettävissä.

XML oli ensimmäinen varteenotettava vaihtoehto korvaamaan CSV:n juurikin sen selkeyden ja helppokäyttöisyyden takia.

5.1.1 Tutustuminen XML:ään

Tutustuttaessa XML:ään saatiin parempi kuva XML:n rakenteista ja iterointitavoista. Selvisi myös, että XML-dokumenttien käsittely Javalla on melko yksinkertaista.

Tehtiin pieni testisovellus, jonka avulla saatiin tuntuma XML:n käsittelyyn. Sovellusta varten tehtiin pieni XML-dokumentti ja käytettiin ensin Xerces nimistä parseria JAVA:ssa dokumentin parsimiseen. Muutaman kokeilun jälkeen huomattiin, että Xerces on vanhentunut XML-parseri ja JAVA:n omista kirjastoista löytyisi standardoitu yksinkertaisempi toteutus XML-parserista. Päätettiin kokonaan luopua Xerces kirjastosta.

Muutamien testiajojen jälkeen huomattiin, että Xercesistä luopuminen oli järkevä ratkaisu. Kokeiltiin myös Xpathia, mutta se ei sopinut sovelluksen tarpeisiin. Sovelluksen alkuperäinen periaate oli tallettaa kaikki tieto globaaleihin muuttujiin, joita voidaan sitten käyttää missä vain, kun taas Xpathia usein käytettiin määrätyn tiedon hankkimiseen tietyssä paikassa iteroinnin sijaan. Iterointi oli paljon järkevämpi sillä siten ei varsinaista sovellusta tarvinnut muokata kokonaan uudestaan.

Ensin olikin tarkoitus vain tehdä XML-parserointi palikka, joka olisi melko huomaamaton sovelluksen kokonaisuuden keskellä ja tarvittaessa toimisi rinnakkain CSV-parserin kanssa.

5.1.2 XML-parserin rakentaminen

Tavoitteena oli siis ensin saada XML-parseri toimimaan XML-tiedostojen kanssa aivan samalla tavalla, kuin CSV-parserikin toimi CSV-tiedostojen kanssa. XML-parserin tuli kerätä tiedot XML-dokumentista ja lähettää ne edelleen *TargetActionEntity* luokkaan, josta tietoja pystyttäisiin käyttämään *ActionEntityHandler*-luokan avulla.(Liite 1.)

Tietojen lukeminen onnistui ensin laskemalla kaikki kohde-elementit ja sen jälkeen yksitellen käymällä elementit läpi FOR-silmukalla. Silmukassa pystyttiin yksitellen käsittelemään kohde-elementin sisällä olevat elementit. Elementtien data luettiin ja tallennettiin muuttujaan tarvittavalla muuttuja tyypillä.(Koodi 6.)

Kun kaikki elementit kohde-elementin sisällä oltiin käyty läpi ja otettu talteen muuttujiin, voitiin lähettää tiedot eteenpäin. Kohteen tunnisteena toimi aina

kohteen nimi. Silmukan mentyä kokonaan läpi oli syntynyt kohde taulukko, joka sisälsi kaikkien kohteiden tiedot eli *TargetActionEntity*-taulukko.

Koodi 6. *XMLFileParser* dokumentin luonti ja iteroinnin aloitus.

```
    fileName = file;
try {
    // parse the XML as a W3C Document
    DocumentBuilder builder = DocumentBuilderFactory.newInstance().newDocumentBuilder();
    Document document = builder.parse(new File(fileName));

    NodeList list = document.getElementsByTagName("object");
    int len = list.getLength();
    int[] tmpNumberOfOptions = new int[len];
    int maxNumberOfOptions = 0;
    for(int i = 0; i<len; i++){
        Element object_e = getElement(document, "object", i);
        NodeList optionsLst = object_e.getElementsByTagName("option");
        tmpNumberOfOptions[i] = optionsLst.getLength();
        if(maxNumberOfOptions < tmpNumberOfOptions[i]){
            maxNumberOfOptions = tmpNumberOfOptions[i];
        }
    }
    questions = new String[len];
    types = new int[len];
    numberOfOptions = new int[len];

    options = new String[len][maxNumberOfOptions];
    returnees = new String[len][maxNumberOfOptions];
    points = new double[len][maxNumberOfOptions];

    Element version_e = getElement(document, "version", 0);
    try{
        version = Double.valueOf(getElementValue(version_e));
    }catch(NumberFormatException nfe){

    }catch(NullPointerException npe){

    }
    for(int i = 0; i<len;i++){
        Element object_e = getElement(document, "object", i);
        Element name_e = getElementE(object_e, "name", 0);
        Element actionLON_e = getElementE(object_e, "actionLON", 0);
        Element actionLAT_e = getElementE(object_e, "actionLAT", 0);
        Element triggerDistance_e = getElementE(object_e, "TriggerDistance", 0);
        Element moviePath_e = getElementE(object_e, "moviePath", 0);
        Element audioPath_e = getElementE(object_e, "audioPath", 0);
        Element textPath_e = getElementE(object_e, "textPath", 0);
```

5.2 Testaaminen

Sovellusta ja sen uusia ominaisuuksia testattiin käyttöpohjaisella testausmenetelmillä. Testaus suunnitelmaa ei luotu, vaan kaikki testaaminen tapahtui kokeilemalla sekä käytössä.

6 Hyödyt

Parserointi helpottui huomattavsti XML:n käyttöönoton jälkeen. Ohjelma ei myöskään kaatunut, vaikka XML-datassa olikin virheitä. Elementti virheet pystyttiin käsittelemään ja sovellus ilmoittaa elementeissä tai niiden nimissä olevat virheet, mikäli niitä on.

Kohdetietojen kerääminen ja kasaaminen sovellusta varten helpottui, kun ei enään tarvinnut laskea pilkkujen määriä ja kohteet erottui selkeästi toisistaan. Rakenne on selvemmin hahmotettavissa ja ymmärrettävissä. Jokaisen tiedon vieressä luki mitä kyseinen tieto edustaa, josta selviää mihin kyseistä tietoa tarvitaan.

Tietojen oikeellisuuden tarkastaminen vei huomattavasti vähemmän aikaa ja siten säästi paljon työtunteja.

7 Sovelluksen kehittyminen

Sovellusta kehitettiin muutenkin kuin vain lisäämällä siihen XML-tuki. Alkuperäisessä sovelluksessa ei ollut oikeastaan minkäänlaista interaktiivisuutta. Ainoa asia johon käyttäjä pystyi vaikuttamaan oli se, haluaako katsoa esityksen kohteesta vai ei.

Oppimisympäristössä tarvitaan enemmän virikkeitä, jotta oppilas pysyy virkeänä ja kiinnostuneena koko kierroksen ajan. Sovellukseen oli myös tarve saada jonkinlainen mittari, jolla pystyttäisiin seuraamaan oppimista. Sovellukseen piti siis saada rasteille kysymyksiä ja jokin menetelmä, jolla saada vastaukset opettajalle palautteen tekemistä varten.

7.1 Kysymysten suunnittelu

Suunnittelu aloitettiin palaverilla, jossa ideoitiin, mitä tarvitaan ja miten toteutetaan. Päädyttiin siihen, että tarvitaan kolmenlaisia kysymyksiä: monivalintakysymyksiä, joissa on useita oikeita vastauksia, monivalintakysymyksiä, joissa on vain yksi oikea vastaus, ja kysymys, johon voi

vastata vapaasti kirjoittamalla. Päädyttiin myös siihen, että paras ja yksinkertaisin toteutus olisi uusi aukeava ikkuna kohteella, jossa olisi kysymys ja mahdolliset vastausvaihtoehdot tai tekstilaatikko.

Vastauksen tehtyä oppilas painaisi ’Vastaa’- tai ’Hyväksy’-nappia, jolloin vastaukset tallentuisivat jonnekin. Ei vielä ollut varmaa, mihin vastaukset tallentuisivat, mutta vaihtoehtoina oli XML-dokumentti tai tekstitiedosto.

7.2 Kysymysten toteutus

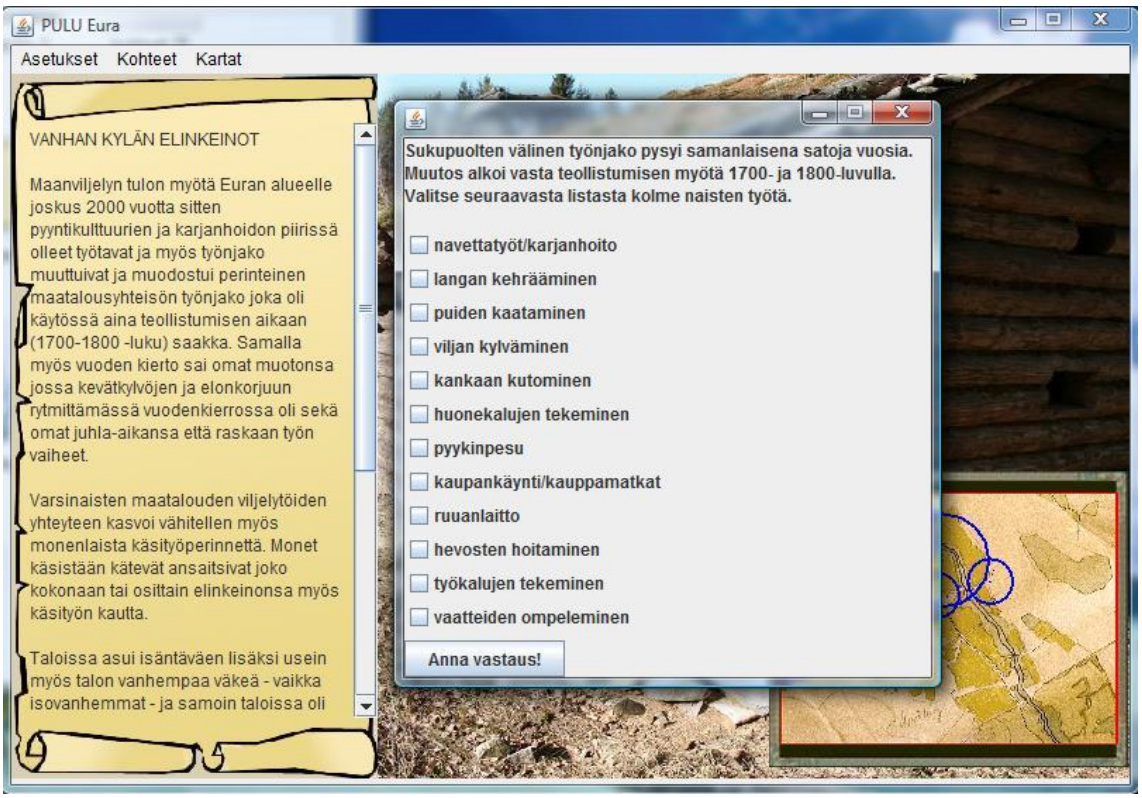
XML auttoi paljon kysymysten toteutuksessa. Kysymysten tiedot pystyttiin sisällyttämään samaan tiedostoon kohteiden muiden tietojen kanssa. Kysymykset pystyttiin upottamaan kohde elementin alle omaksi elementiksi, joka sisälsi kysymyksen parametrit omissa elementeissään. Kysymysten tietojen läpikäynti onnistui samalla periaatteella, kuin normaaleissa kohteissa. Tiedot, jotka saatiin läpikäymällä kaikki elementit, lähetettiin *QuestionActionEntity*-luokkaan. *QuestionActionEntity*-luokka toimii hyvin samankaltaisesti *TargetActionEntity*-luokan kanssa. Luokat toimivat ns. tietopankkeina sovellukselle, XML:stä saaduille tiedoille.(Liite 1.)

Aina kohteelle savuttaessa sovellus käy tarkistamassa, onko kohteella kysymystä. Mikäli kysymys löytyy kohteelta, kutsutaan *Questions*-luokkaa ja lähetetään luokalla *QuestionActionEntity* taulu kysymyksestä. *QuestionActionEntity* pitää sisällään kysymyksen nimen, kysymyksen, tyypin, mahdolliset vaihtoehdot, mahdolliset vastaukset ja pisteet siitä kuinka monta pistettä kustakin vastausvaihtoehdosta saa.(Liite 1.)

Questions luokka käytti näitä tietoja luodessaan *JFramen*. *JFrame* on javan oma API, jolla voidaan rakentaa graafisia ikkunoita sovelluksessa. On kolme vaihtoehtoa, miltä ikkuna näyttää, kun kysymys ponnahtaa käyttäjälle. Jokaisessa vaihtoehdossa on aina ensimmäisenä kysymys. Kysymyksen jälkeen tulee joko

1. teksti-laatikko, jolloin vastauksen voi antaa vapaassa sanamuodossa
2. eri vaihtoehtoja joiden edessä on ruutu, jolloin vastauksia voi olla useampi kuin yksi(Kuva 1.) tai
3. eri vaihtoehtoja joiden edessä on ympyrä, jolloin vastauksia voi olla vain yksi.

Vastausvaihtoehtojen alapuolella on aina ”Anna Vastaus”-nappi. Kysymyksestä pääsi eteenpäin vain antamalla jonkin vastauksen ja painamalla nappia. Aina kun kysymykseen vastattiin, tallentui ryhmän nimiseen XML-tiedostoon vastaukset kansioon kysymyksen tiedot ja vastaus, sekä mahdolliset pisteet.



Kuva 1. Esimerkki kysymyksen toteutuksesta.

8 Tulevaisuuden näkymät

Sovelluksella on erittäin suuret potenttiaalit kehittyä paremmaksi oppimissovellukseksi. Kysymysikkunoiden uudelleenmuotoilu toisi sovellukselle

siistimmän ja ammattimaisemman kuvan. Ulkoasullisia muokkauksia voi tosin aina toteuttaa myös käyttäjän osalta vaihtamalla taustakuvia.

Toiminnallisuuden kehittyminen XML:n myötä on nyt myös todennäköisempää, kun sovelluksen välinen kommunikointi onnistuu paremmin XML-viestinnän avulla. Kohteiden luonnille pystytään myös rakentamaan oma editori monimutkaisempien XML-tiedostojen varalta.

JTS avaa aivan uuden maailman sovellukselle. JTS mahdollistaa reittikohteet, sekä oppilaan ohjaamiseen oikeaan suuntaan. Pisteiden sijaan kohteet voivat olla minkä muotoisia tahansa. JTS mahdollistaa myös liikkumalla kysymyksiin vastaamisen. Karttaan voidaan myös merkitä vilkkaasti liikennöidyt tiet, joista sovellus osaisi varoittaa oppilaita maastossa liikuttaessa.

9 Yhteenveto

Työssä paneuduttiin XML:n käyttämiseen lähes tietokannan tavoin Java-sovelluksessa ja seurattiin, kuinka paljon XML helpottaa sisällön tuottoa. Hyvin varhain selvisi, että XML auttaa ymmärtämään sisältöä paljon paremmin ja sisällöntuottaminen on paljon helpompaa ja selkeämpää. Oikein nimetyt elementit auttavat sisällöntuottajaa laittamaan oikeat tiedot oikeisiin kenttiin.

Jouduttiin hieman poikkeamaan alkuperäisestä suunnitelmasta, jossa olisi haettu tietoa tarpeen mukaan XML-dokumentistä. Sen sijaan ladataankin koko paketti suoraan sovelluksen käyttömuistiin ohjelman käynnistyessä. Tämä prosessi hieman hidastaa sovelluksen käynnistymistä ja saattaa tehottomammilla koneilla aiheuttaa lievää tehonpuutetta sovelluksen ollessa käynnissä.

Näistä tuloksista kuitenkin pystytään päättelemään, että XML on varteenotettava tiedon tallennusmuoto eikä rajoitu pelkästään tietojen vaihtamiseen tai laajennettuun HTML-kieleen. Pienissä tietokantaratkaisuissa voisi olla jopa järkevämpää hyödyntää XML:ää SQL-tietokantojen sijaan.

XML on jo hyvin yleisessä käytössä monissa eri ympäristöissä. Tästä tulee olemaan suuri etu tulevaisuudessa, kun tiedot pystytään siirtämään muiden alustojen kesken suuren XML tuen myötä.

Lähdeluettelo

[1] Janert, Philipp K., ”Simple XML Parsing with SAX and DOM”, [www-dokumentti] Saatavilla: http://onjava.com/pub/a/onjava/2002/06/26/xml.html?page=2 (Luettu: 08.06.2010)

[2] Wikipedia, “Simple API for XML”, [www-dokumentti] Saatavilla: http://en.wikipedia.org/wiki/Simple_API_for_XML (Luettu 08.06.2010)

[3] Janert, Philipp K., ”Simple XML Parsing with SAX and DOM”, [www-dokumentti] Saatavilla: http://onjava.com/pub/a/onjava/2002/06/26/xml.html?page=3 (Luettu: 08.06.2010)

[4] McLaughlin, Brett, Java & XML 2nd Edition. 101 Morris Street, Sebastopol, CA 95472: O’Reilly & Associates, Inc. 2001

[5] Harold, Eliiotte Rusty, “The Java XPath API”, [www-dokumentti] Saatavilla: http://www.ibm.com/developerworks/library/x-javaxpathapi.html (Luettu: 08.06.2010)

[6] Wikipedia, ”XML”, [www-dokumentti] Saatavilla: http://en.wikipedia.org/wiki/XML#Well-formedness_and_error-handling (Luettu 08.06.2010)

Sovelluksen luokkakaavio