Semantic Web - Metadata editor

Ohjelmistotuotantoprojekti, kesä 2002
Ohjelmistotuotantoryhmä 1, Meedio

http://www.cs.Helsinki.FI/group/meedio

Mikko Apiola (M.A.)
Ari Inkovaara (A.I.)
Miikka Junnila (M.J.)
Justus Karekallas (J.K.)
Pekko Parikka (P.P.)

Helsinki 3. elokuuta 2002
Suunnitteludokumentti
Helsingin yliopisto
Tietojenkäsittelytieteen laitos

Versiohistoria

Versio	Pvm	Laatija	Kommentti
0.1	24.6.2002	Pekko Parikka	Alustava versio kommentoitavaksi
0.2	17.7.2002	Justus Karekallas	Tarkistettu ja korjattu versio
0.3	23.7.2002	Pekko Parikka	FTR:ssä esiin tulleiden virheiden korjaus.
0.4	1.8.2002	Pekko Parikka	Virheitä korjattu.
1.0	3.8.2002	Pekko Parikka	Hyväksytty lopullinen versio dokumentista.

Sisältö

1 Johdanto (P.P.)

Tässä dokumentissa suunnitellaan ohjelmisto, jonka ohjelmistotuotantoryhmä Meedio toteuttaa Helsingin yliopiston tietojenkäsittelytieteen laitoksen (TKTL) ohjelmistotuotantoprojektissa 29.5.2002 - 30.8.2002.

Meedio-ryhmän jäsenet ovat Mikko Apiola, Ari Inkovaara, Miikka Junnila, Justus Karekallas ja Pekko Parikka.

Luvussa 2 annetaan yleiskuvaus ohjelmasta. Luvussa 3 kuvataan muutama ohjelman loppukäyttäjien käyttötapaus. Luvussa 4 esitetään vaatimuksia järjestelmän tekniselle toteuttamiselle. Luvussa 5. selostetaan suunnitterajoituksia. Luvussa 6. kuvataan ohjelman osajärjestelmät yleisellä tasolla. Luvussa 7. kuvataan tarkemmin XML-käsittelijän toteutusta. Luvussa 8. kuvataan RDF-käsittelijän toteutusta. Luvussa 9. kuvataan käyttöliittymän toteutus. Luvussa 10. kuvataan tiedostonkäsittelijä-osion toteutus. Luvussa 11. kuvataan Controllerin eli ohjaimen toteutus. Liitteenä on tulostettuna luokkien metodikuvaukset, jotka on toteutettu Javadocilla.

2 Ohjelmiston yleiskuvaus (P.P.)

Metadataeditori on ohjelma ontologian luokkien ilmentymien eli instanssien luomiseen, ts. kyseessä on instanssieditori. Metadataeditorilla luodaan museoiden tietokannasta tuotettujen XML-muotoisten esinekuvausten (esinekorttien) pohjalta RDF-tietoa, eli RDF-skeeman mukaisia instansseja. Ohjelman on tarkoitus helpottaa museohenkilöiden työskentelyä Finnish Museum Online (FMO) projektissa, jossa eri museoiden esinetiedoista luodaan yksi yhteinen instanssitietokanta. Metadataeditorilla luetaan museoiden tietokannasta tuotettu XML-tieto, josta sitten luodaan halutun ontologian, eli RDF-skeeman, mukaisia ilmentymiä. RDF-skeema on siis vaihdettavissa ja muutettavissa.

Metadataeditori toimii paikallisesti koneella. Luodut instanssit voidaan tallentaa paikalliselle levyjärjestelmälle, josta ne jollain muulla ohjelmalla siirretään haluttuun tietokantaan.

Ohjelmisto toteutetaan Javalla käyttäen JSP Tag Libraries tekniikkaa Jakarta Tomcat ympäristössä. RDF-käsittely toteutetaan käyttäen Jenaa. XML-käsittely toteutetaan käyttämällä Apachen Xerces jäsentäjää. Java ympäristönä on JDK 1.3.1. Ohjelmisto toimii sekä Windows että Linux ympäristöissä ja sekä Netscape Navigator että MS Internet Explorer selaimilla, ks. luku 12. viitteet.

3 Ohjelmiston käyttötapauksia (M.J.)

Jotta ohjelman toimintaa olisi helpompi ymmärtää, on hyvä lähestyä sitä myös tyypillisten käyttötapausten kannalta. Käyttötapaukset kuvaavat jonkun ongelman, jonka käyttäjä yrittää ratkaista metadataeditorin avulla. Kun ohjelma on valmis, pystyvät myös tässä luvussa kuvatut henkilöt suoriutumaan tehtävistään.

3.1 Käyttötapaus I

Museotyöntekijä Matti on ollut työssä museossaan jo parikymmentä vuotta, ja tuntee esineistön paremmin kuin kukaan muu. Osaamisensa vuoksi Matti on valittu museonsa FMO-vastaavaksi, ja on juuri saanut ohjelman asennettua museon koneelle. Matti haluaa tutustua ohjelmaan, ja haluaa heti lähteä lajittelemaan museonsa kokoelmia.

Käyttäjän tavoite: Lajitella esineet mahdollisimman tehokkaasti.

Statustietoa:

- Matti ei tunne tietotekniikkaa paljoakaan, mutta on kiinnostunut oppimaan
- Metadataeditori toimii museon koneella, mutta Matti ei ole koskaan käyttänyt sitä
- Ontologia on tehty vastaamaan suunnilleen perinteisiä museo-luokitteluja

3.2 Käyttötapaus II

Matti on jo opetellut käyttämään metadataeditoria, kun häneltä pyydetään kansainväliseen näyttelyyn kaikkia heidän museonsa aseita. Matti haluaa löytää kaikki aseet heidän tiedoistaan ja haluaa liittää niihin luokittelut. Matti haluaa myös tarkastaa että aseiden tiedot ovat muutenkin ajan tasalla.

Käyttäjän tavoite: Saada aseet lajiteltua.

Statustietoa:

- Matti osaa käyttää ohjelmaa
- Kaikista aseista ei tiedetä onko niitä käytetty myös muihin tarkoituksiin (työkaluina, uskonnollisiin menoihin…)
- Useiden aseiden tiedot ovat puutteellisia (ei tiedetä ikää, joistain on kadonnut tieto löytäjästä)

3.3 Käyttötapaus III

Käyttäjän tavoite: Antti on Keltaiset sivut -projektiryhmän jäsen, ja haluaa tutkia miten heidän ryhmänsä voisi käyttää metadataeditoria hyödykseen projektissaan.

Statustietoja:

- Antti tuntee hyvin RDF(S)-tekniikan sekä muutakin metadataeditorissa käytettyä tekniikkaa
- Keltaiset sivut -projekti tarvitsisi työkalun, jolla ilmoittajat voivat laittaa keltaisille sivuille ilmoituksia RDF-muodossa helposti
- Anttia kiinnostaa erityisesti, miten museokäyttöön tarkoitettu editori toimii täysin uudella ja erityyppisellä ontologialla ja XML-datalla

4 Tärkeimmät vaatimukset järjestelmälle (J.K.)

Kappaleessa kuvataan ominaisuudet, jotka ovat ehdottoman tärkeitä järjestelmän toiminnan kannalta. Yhdenkin tällaisen ominaisuuden puuttuminen tekee järjestelmästä käyttökelvottoman.

4.1 Konfiguraatiotiedosto

Konfiguraatiotiedosto sisältää järjestelmän toiminnan kannalta oleelliset asiat. Ilman konfiguraatiotiedostoa järjestelmää ei voida käyttää.

Konfiguraatiotiedostoon kirjataan sekä RDF-skeeman että XML-skeeman nimi ja sijainti. Lisäksi konfiguraatiotiedostoon määritellään XML-muotoisen esinekortin alun ja lopun määrittelevät kentät ja XML-kortin yksittäisen kentän otsikon kertova attribuutti. Myös esineen kuvan sijainnin ilmaiseva XML-kenttä on määritelty konfiguraatiotiedostossa.

4.2 Laitteisto- ja ohjelmistovaatimukset

Ohjelmisto suunnitellaan käytettäväksi tavallisessa koti/toimisto-tietokoneessa. Käytettävässä koneessa pitäisi olla peruslaitteiden (näyttö, hiiri, näppäimistö ym.) lisäksi tietoliikenneyhteys.

Järjestelmä toteutetaan Java-kielellä ja se toimii Linux ja Windows käyttöjärjestelmissä.

Toimiakseen järjestelmä vaatii käytössä olevaan tietokoneeseen asennetun Jakarta Tomcat -palvelinohjelman. Myös jokin internet-selain on syytä olla asennettuna, toiminta taataan Ms. Internet Explorer ja Netscape Navigator selaimilla.

5 Suunnittelurajoitukset (M.J.)

Ohjelmaa tehtäessä on pyrittävä rajaamaan tehtävä sopivaksi ajan ja resurssien mukaan. Näin joudutaan ohjelmasta jättämään pois useita sinänsä hyviä ja tärkeitä ominaisuuksia, jotta saataisiin metadataeditorin perustoiminnallisuus toimimaan moitteettomasti. Koska uusia ominaisuuksia jotka voidaan mahdollisesti myöhemmin toteuttaa on helppo keksiä, varaudutaan ohjelman laajentamiseen tulevaisuudessa niin, että näitä laajennuksia, muutoksia ja parannuksia olisi helpompi tehdä. Tässä kuvataan joitain ominaisuuksia, joiden lisäykseen pyrimme varautumaan niin, ettei ohjelmamme rakenne estä niiden jatkokehittelyä.

5.1 Käyttöliittymän ja tietorakenteiden riippumattomuus toisistaan

Koska metadataeditorin tarkoitus on toimia millä tahansa ontologialla, on tärkeää, että käyttöliittymän generoiva osa on irrotettu ontologiasta, jotta tietorakenteet voidaan vaihtaa niin ettei käyttöliittymä hajoa, ja vastaavasti myös käyttöliittymän generoiva moduuli tulee olla vaihdettavissa niin, että tietorakenteet sen alla pysyvät ennallaan, mutta ulkoasu saadaan vaihdettua, esimerkiksi kohderyhmän vaihduttua museohenkilöistä yrittäjiksi.

5.2 Tietoliikenne

Vaikka järjestelmässämme lähdetään siitä ajatuksesta, että suurin osa tiedosta sijaitsee paikallisella koneella, jossa editoria käytetään, on pidettävä mielessä mahdollisuus laajentaa ohjelmaa paremmin etäkäyttöön sopivaksi. Eli jos järjestelmää laajennetaan tähän suuntaan, voivat editorin käsittelemät tiedot sijaita jossain verkon takana. Ohjelman rakenne sallii myös sen, että ohjelma on laajennettavissa melko helposti siten, että käyttöliittymä ja muu ohjelma sijaitsevat eri paikoissa.

5.3 Tekoäly

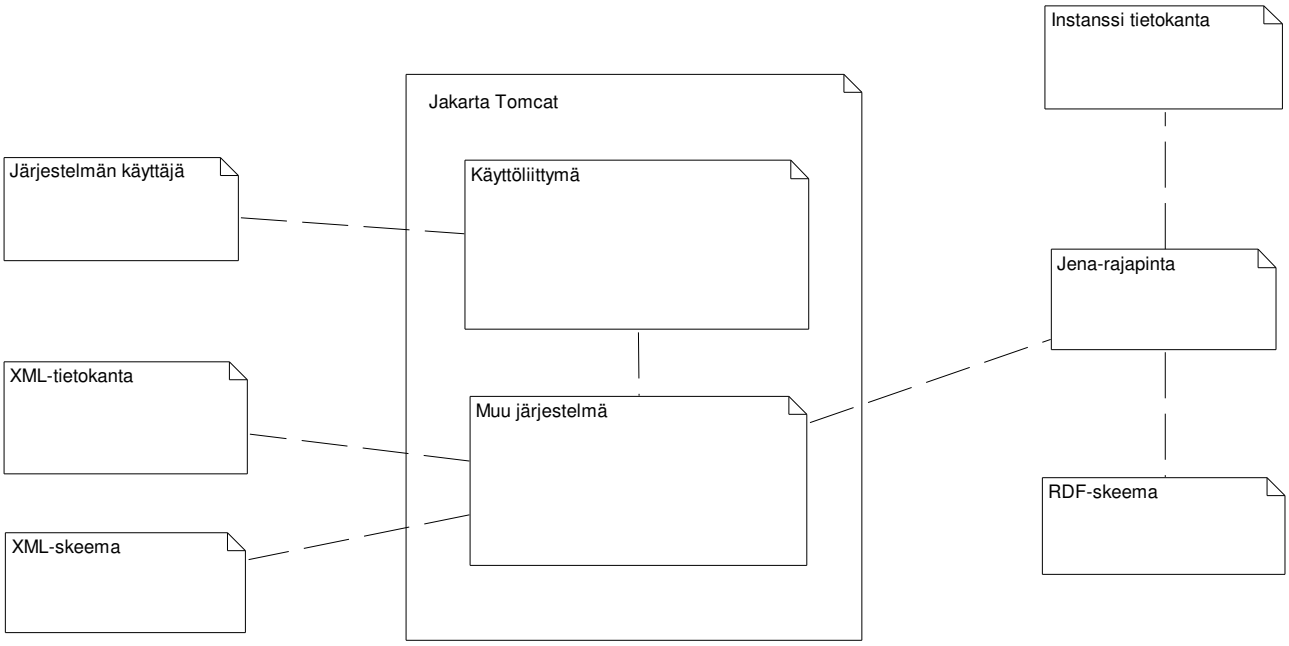
Järjestelmän älykkyyttä tulee voida lisätä kohtuullisella vaivalla niin, että luokkien päättely toimii paremmin. XML-tiedon perusteella ehdotettaessa luokkia käyttäjälle tulee tekoälyn tason olla nostettavissa. Myös usein toistuvat ominaisuuksien arvot tiettyjen instanssien kohdalla voitaisiin automaattisesti täyttää tulevaisuudessa. Instanssikyselyitä tulee voida laajentaa niin, että ne paremmin vastaisivat käyttäjän tarpeisiin, ja osaisivat ehdottaa käyttäjälle ensimmäisenä usein valittuja vaihtoehtoja.

6 Osajärjestelmät (J.K.)

Luvussa kuvataan järjestelmään kuuluvat osajärjestelmät sekä näihin liittyvät ulkoiset ja sisäiset oliot. Osajärjestelmät vastaavat kukin jonkin erityisen tehtäväalueen toimintojen suorittamisesta. Ulkoisilla olioilla tarkoitetaan järjestelmän ulkopuolisia resursseja (tiedostot, tietokannat) ja käyttäjiä. Sisäisiä olioita ovat järjestelmään sisällytetyt toiminnot kuten tiedostopalvelin ja käyttöliittymä.

6.1 Ulkoiset oliot

Järjestelmän ulkoiset oliot tuottavat järjestelmään tietoa ja toimivat tiedon tallennus- ja hakupaikkoina.



Kuva 6.1: Järjestelmän ulkoiset oliot

1) Järjestelmän käyttäjä

Käyttäjä muokkaa XML-muotoista tietoa sisältäviä kortteja annetun ontologian mukaiseksi, eli tekee niille ontologian mukaisen luokittelun, jonka perusteella RDF-tieto luodaan.

2) XML-tietokanta

XML-tietokanta sisältää esineiden kuvaukset XML-kortteina.

3) XML-skeema

XML-skeema sisältää kuvauksen, jonka perusteella validoidaan XML-tietokannasta luotu kaikki kortit sisältävä XML-tiedosto, kun tiedosto otetaan ensimmäisen kerran ohjelman käyttöön.

4) RDF-skeema

RDF-skeeman avulla tehdään korttien semanttinen luokittelu eli luodaan RDF-tietoa.

5) Jena-rajapinta

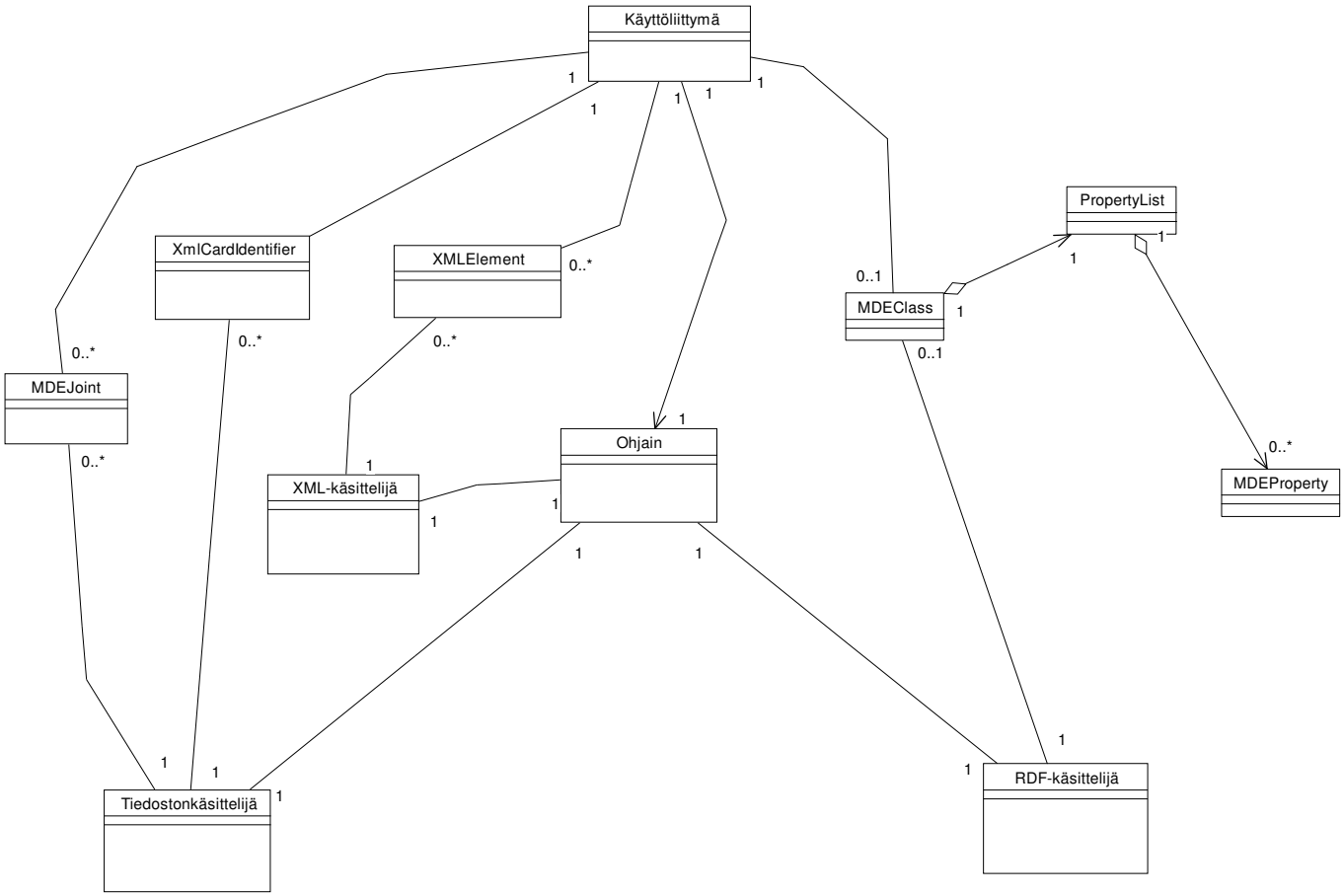
Jena-rajapinnan avulla tarkistetaan olemassa olevien ja uusien ilmentymien yhteensopivuus. Lisäksi Jena-rajapinnan avulla luodaan tietorakenteita, joiden perusteella käyttöliittymä piirtää näkymän.

6) Jakarta Tomcat ympäristö

Jakarta Tomcatin avulla luodaan käyttöliittymäelementit Java Server Pages(JSP) -sivuina.

6.2 Sisäiset oliot

Sisäiset oliot tuottavat muokkaavat ja tallentavat tietoa sekä hoitavat kommunikoinnin käyttäjien kanssa.



Kuva 6.2: Järjestelmän sisäiset oliot

1) Käyttöliittymä

Käyttöliittymän avulla käyttäjä luokittelee kortit semanttisesti rikastettuun muotoon. Käyttöliittymän rakenteen yksityiskohdat kuvataan luvussa yhdeksän.

2) Ohjain

Ohjain toimii käyttöliittymän ja muiden järjestelmän olioiden välissä jakaen järjestelmän toiminnot eri osioiden suoritettaviksi. Ohjaimen rakenne ja rajapinnat kuvataan tarkemmin luvussa 11.

3) XML-käsittelijä

XML-käsittelijä tarkistaa muokattavan XML-tiedoston kelpoisuuden verraten sitä XML-skeemaan ja parsii käsittelyyn otettavan esinekortin. XML-käsittelijän rakenteelliset ominaisuudet rajapintoineen kuvataan seitsemännessä luvussa.

- **XMLElement**

  XML-käsittelijä välittää ohjaimen kautta käyttöliittymälle elementtejä XML-tiedostosta linkitettynä listana.

4) RDF-käsittelijä

RDF-käsittelijä luo ontologian, eli RDF-skeeman, mukaisia RDF-ilmentymiä. RDF-käsittelijä tuottaa myös Jena-rajapinnan avulla käyttäjälle esitettävän tietorakenteen ontologiasta. RDF-käsittelijän tekninen toteutus kerrotaan luvussa kahdeksan.

- **MDEClass**

  RDF-käsittelijä välittää ohjaimen avulla RDF-skeeman luokkia kuvaavia olioita käyttöliittymälle.

- **MDEProperty**

  RDF-käsittelijä välittää käyttöliittymälle ohjaimen avulla RDF-skeeman luokkien ominaisuuksia kuvaavia olioita.

5) Tiedostonkäsittelijä

Tiedostonkäsittelijä huolehtii XML-korttien hakemisesta luokiteltavaksi käyttöliittymälle. Tiedostonkäsittelijä huolehtii myös mm. liitosten lukemisesta tiedostosta sekä konfiguraatiotiedoston lukemisesta. Tarkka suunnitelma tiedostonkäsittelijän toteuttamiseksi selvitetään luvussa kymmenen.

- **MDEJoint**

  Tiedostonkäsittelijä välittää käyttöliittymälle ohjaimen avulla tietoa liitettyjen ominaisuuksien tiedoston sisällöstä. Tieto välitetään MDEJoint-olioina.

- **XMLCardIdentifier**

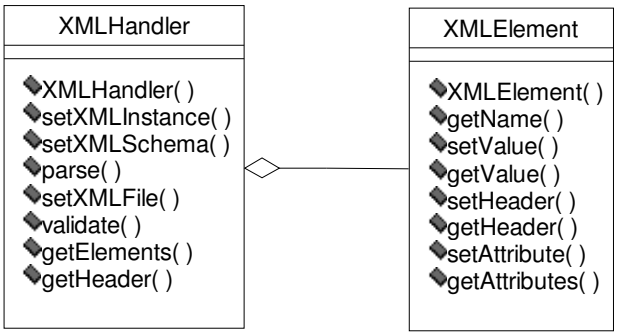
  Tiedostonkäsittelijä välittää ohjaimen avulla käyttöliittymälle XML-kortteja kuvaavia XMLCardIdentifier-olioita. Oliot välitetään taulukossa.

7 XML-käsittelijä (A.I.)

XML-käsittelijän tehtävänä on validoida annettu XML-dokumentti annettua XML-skeemaa vasten. XML-skeeman ja sen ilmentymän käsittelyssä käytetään Javan Simple Api for XML -rajapintaa. Validoinnissa käytetään Apachen Xerces-jäsentäjää.

7.1 Luokat

XML-käsittelijässä on kaksi luokkaa (kuva 7.1) XMLHandler ja XMLElement.



Kuva 7.1: XML-käsittelijän luokat

7.2 Rajapinta

Tämä luku esittelee lyhyesti XML-käsittelijän käskyrajapinnat. Tarkemmat kuvaukset ovat tämän dokumentin liitteenä Javadoc-muodossa.

7.2.1 Luokka XMLHandler

Luokan tehtävänä on validoida annettu XML-skeeman ilmentymä.

Luokan aksessorit:

--XMLHandler()

--public setXMLInstance(String XML)

--public setXMLSchema(String uri)

--public parse()

--public setXMLFile(String uri)

--public validate()

--public LinkedList getElements()

--public String getHeader()

7.2.2 Luokka XMLElement

Luokka säilöö tietoa yksittäisestä elementistä.

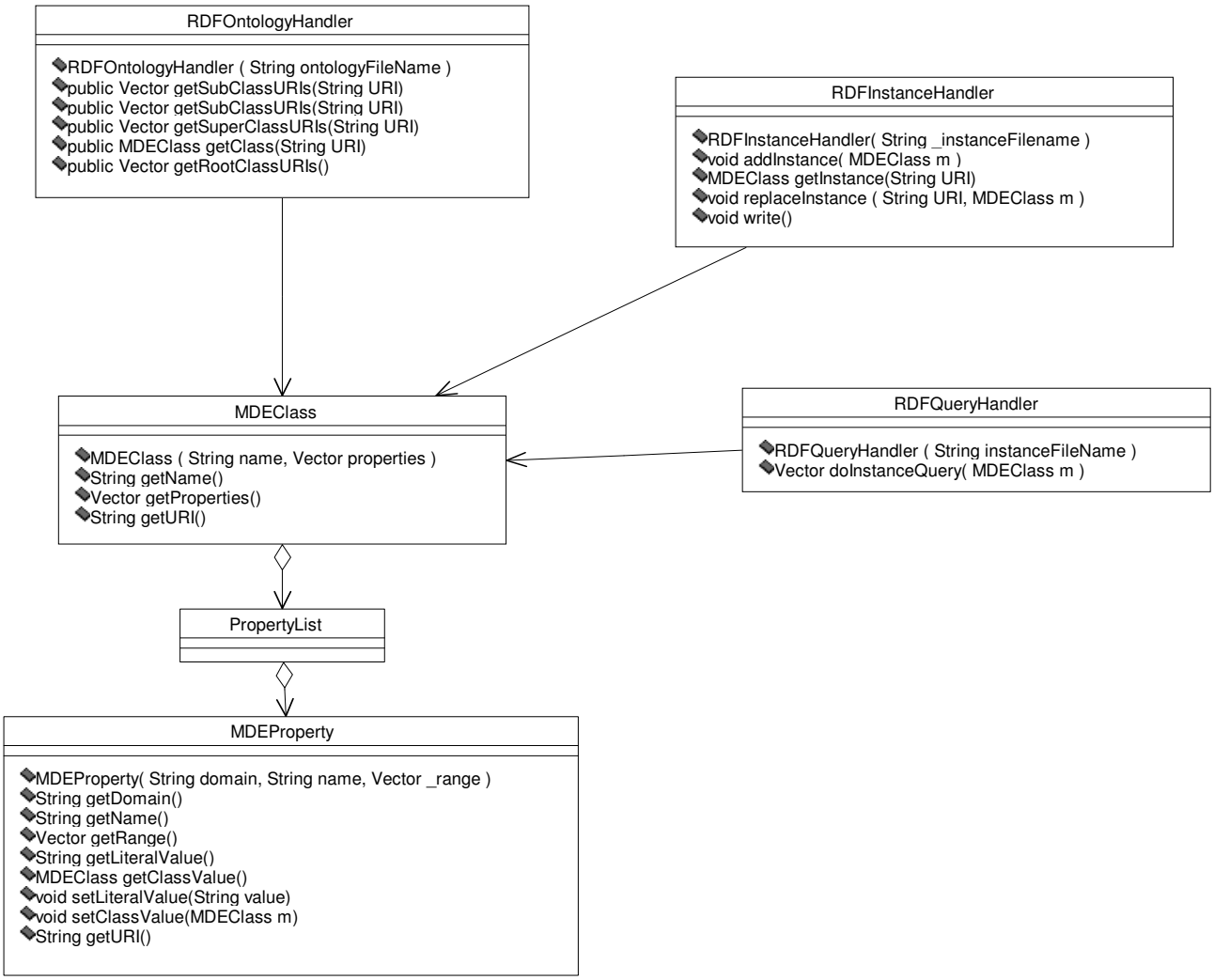
Luokan aksessorit:

--XMLElement(String name)
--public String getName()
--public setValue(String value)
--public String getValue()
--public setHeader(String header)
--public String getHeader()
--public setAttribute(String name, String value)
--public LinkedList getAttributes()

8 RDF-käsittelijä (M.A.)

RDF-käsittelijän tehtävänä on käsitellä RDF(s)-muotoista ontologiaa ja tuottaa sen mukaista RDF-tietoa. RDF-käsittelijä tarjoaa toiminnot joilla saadaan järjestelmän tarvitsemat tiedot ontologiasta esille, toiminnot RDF:n tuottamiseen sekä instanssikyselyn tekemiselle.

8.1 Luokat



Kuva 8.1: RDF-käsittelijän luokat

RDF-käsittelijä koostuu seuraavista luokista:

- MDEOntologyHandler
- MDEInstanceHandler
- MDEQueryHandler
- MDEProperty
- MDEClass

8.1.1 Rajapintaluokat

Näiden rajapintaluokkien toteutus voidaan myöhemmin vaihtaa käyttämään esim. ontogator-mediatoria (ks. luku 8.2).

- RDFOntologyHandler
- RDFInstanceHandler
- RDFQueryHandler

Tämä luku esittelee lyhyesti RDF-Käsittelijän käskyrajapinnat. Tarkemmat kuvaukset ovat tämän dokumentin liitteenä Javadoc-muodossa.

8.1.2 Luokka MDEOntologyHandler

MDEOntologyHandler-luokka toteuttaa RDFOntologyHandler-rajapintaluokan, ja vastaa RDF(S)-muotoisen ontologian lukemisesta. Järjestelmässä MDEController-luokka kutsuu MDEOntologyHandler-luokan metodeja saadakseen tietoja ontologiasta.

Luokan aksessorit:

-- public RDFOntologyHandler(String ontologyFileName)

-- RDFOntologyHandler(String ontologyFileName)

-- public Vector getSubClassURIs(String className)

-- public Vector getSubClassURIs(MDEClass c)

-- public Vector getSuperClassURIs(String classURI)

-- public MDEClass getClass (String className)

8.1.3 Luokka MDEInstanceHandler

MDEInstanceHandler-luokka toteuttaa RDFInstanceHandler-rajapintaluokan, ja vastaa RDF-instansseja sisältävän tiedoston käsittelystä. Järjestelmässä MDEControllor-luokka kutsuu MDEInstanceHandler-luokan metodeja halutessaan lukea, tai kirjoittaa RDF-instansseja.

-- public RDFInstanceHandler (String _instanceFilename)

-- public void addInstance(MDEClass m)

-- public MDEClass getInstance(String URI)

-- public void replaceInstance(String URI, MDEClass m)

-- public void write()

8.1.4 Luokka MDEQueryHandler

Luokka toteuttaa RDFQueryHandler-rajapinnan. Luokan tarkoitus on huolehtia instanssikyselyiden suorituksesta. RDFQueryHandler-luokan kutsuminen on järjestelmässä MDEController-luokan tehtävä.

-- public RDFQueryHandler (String instanceFileName)

-- public Vector doInstanceQuery(MDEClass m)

8.1.5 Luokka MDEProperty

Luokan tarkoitus on säilyttää tietoa yksittäisestä RDF-luokan ominaisuudesta. Luokkaa käytetään tiedonvälitykseen järjestelmän RDF-osion, ohjaimen ja käyttöliittymän välillä. Järjestelmä varastoi MDEProperty-luokan olioita MDEClass-luokan properties-vektoriin.

Luokan aksessorit:

-- public MDEProperty(String domain, String name, Vector _range)

-- public String getDomain()

-- public String getName()

-- public String getURI()

-- public Vector getRange()

-- public String getLiteralValue()

-- public MDEClass getClassValue()

-- public void setLiteralValue(String value)

-- public void setClassValue(MDEClass value)

8.1.6 Luokka MDEClass

Luokan tarkoitus on säilyttää tietoa yksittäisestä RDF-luokasta. Luokkaa käytetään tiedon välitykseen RDF-käsittelijän ja ohjaimen välillä. MDEClass-luokka sisältää tietoja luokasta; luokan nimen ja listan ominaisuuksista, jotka ovat MDEProperty-olioina.

Luokan aksessorit:

-- public MDEClass(String name, Vector properties)

-- public String getName()

-- public String getURI()

-- public Vector getProperties()

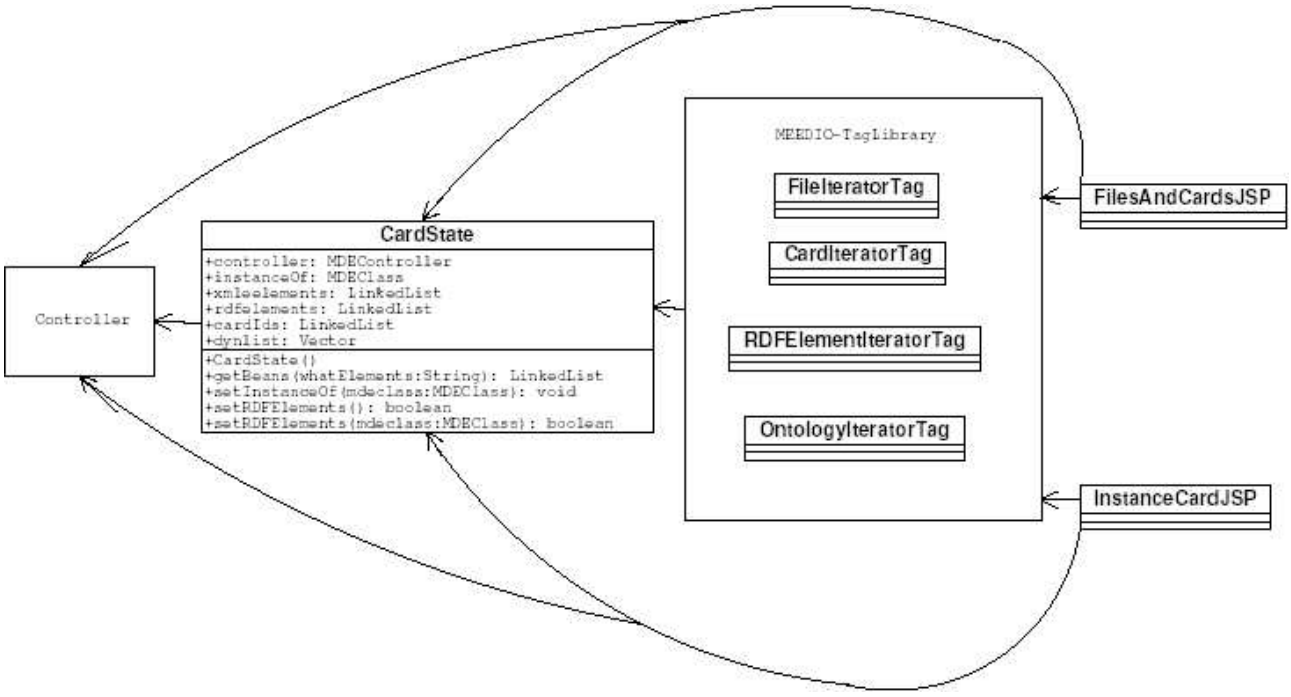
8.2 RDF-käsittelijän siirrettävyys ontogator-mediatorille

Järjestelmän RDF-käsittelijä on siirrettävissä ontogator-mediatorille siten, että laaditaan rajapintaluokkien RDFInstanceHandler, RDFOntologyHandler sekä RDFQueryHandler toteuttavat ontogator-mediatoria käyttävät luokat.

9 Käyttöliittymämoduuli (M.J.)

Käyttöliittymämoduuli tulee olemaan keskeinen osa järjestelmää, ja sen haasteisiin kuuluu kuvata sekä XML- että RDF-tiedot selkeästi käyttäjälle niin ettei tämän tarvitse vaivata päätään alla toimivien tekniikoiden erityispiirteistä tai rajoituksista. Käyttöliittymän tulee myös olla mahdollisimman dynaaminen, jotta se pystyisi piirtämään järkevän käyttöliittymän hyvin erilaistenkin ontologioiden pohjalta.

Käyttöliittymän pohjana tulee olemaan jako, jossa on yksi osa tiedostonhallinnalle ja toinen yhden XML-kortin RDF-tietojen täyttämiseen ja muokkaamiseen. Jotta kokonaisuus toimisi, on käyttöliittymään sisällytettävä tietorakenteita, joita JSP-sivut voivat yhdessä käyttää. (Kuva 9 a)



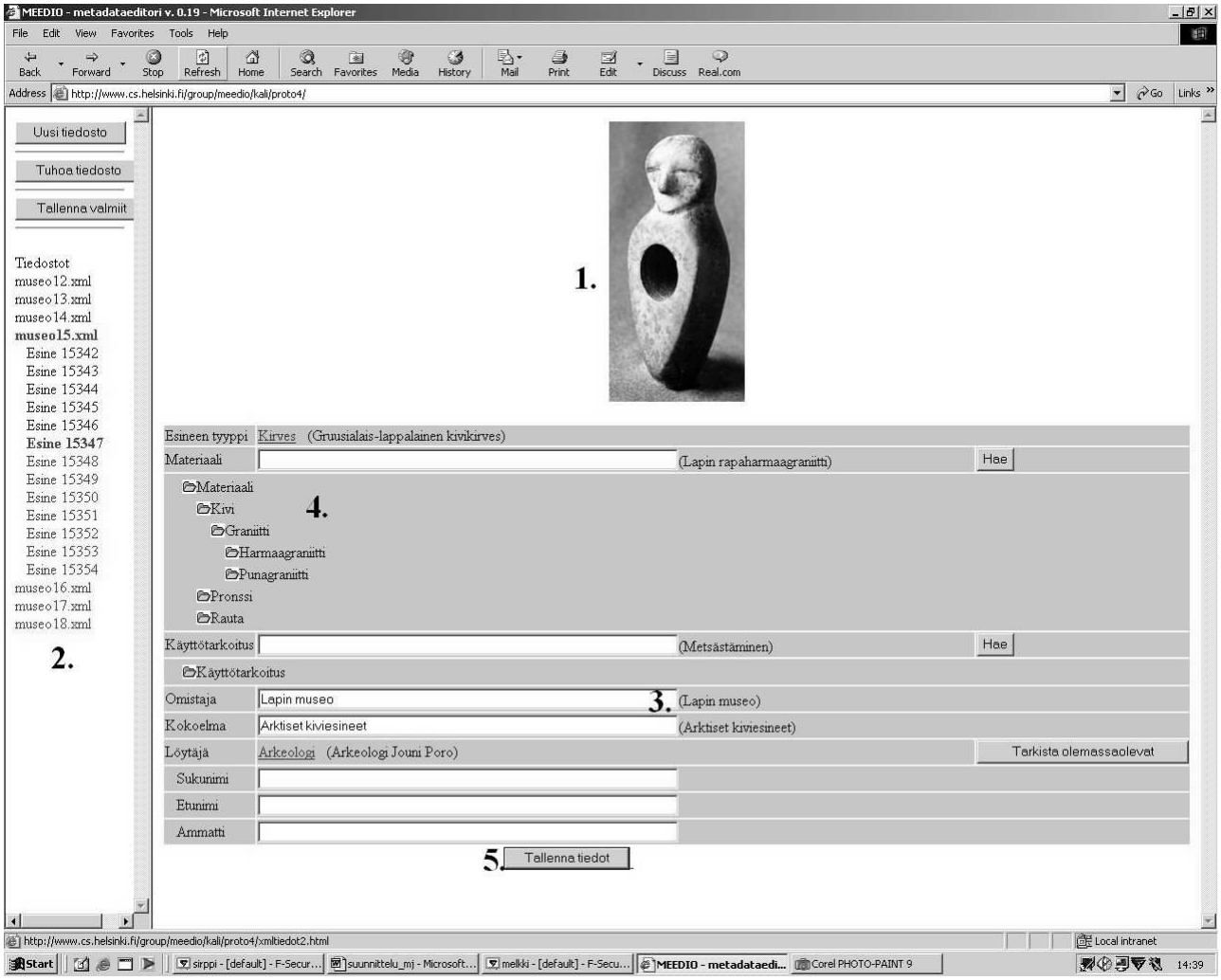
Kuva 9 a: Käyttöliittymän luokkakaavio

9.1 Rajapintojen kuvaus

Käyttöliittymämoduuli käyttää ainoastaan MDEController-moduulia, ja saa siltä käyttöliittymän piirtämiseen tarvittavat tiedot. Eli mikään ohjelman moduuleista ei käytä Käyttöliittymämoduulin palveluita, vaan ainoastaan käyttäjä, eli moduulin rajapinta on käyttöliittymä itse.

9.2 Käyttöliittymäkuvaus

Tässä kuvataan miten käyttöliittymä tulee toimimaan. Logiikkana on se, että tässä käydään läpi käyttöliittymälle asetetut vaatimukset ja viitataan kuvaan 9.2 a käyttöliittymästä, jossa näkyy toiminnon suorituspaikka.



Kuva 9.2 a: Kuva käyttöliittymästä

-- Kuvan näyttäminen, kohta 1

Kuva näytetään käyttöliittymässä tässä kohtaa, mikäli sellainen on saatavilla.

-- Tiedostojen ja sen sisältämien korttien näyttäminen havainnollisesti, kohta 2

Tiedostot ja kortit näytetään omassa kehyksessä sivun vasemmassa laidassa. Väreistä voi päätellä korttien tilan. Värikoodauksen sijasta/rinnalla voidaan käyttää myös V=valmis ja K=kesken koodausta.

-- XML-kortin tietojen havainnollinen näyttäminen, kohta 3

Literaaliarvoisissa ominaisuuksissa tieto täytetään automaattisesti suoraan kenttään. Tämän lisäksi ominaisuutta vastaava XML-tieto on aina kentän perässä sulkeissa.

-- Ontologian esittäminen dynaamisena hierarkkisena listana sekä semanttisen tiedon lisääminen XML-korttiin, kohta 4

Ontologiasta avataan aina täytettävään kohtaan sopiva ontologia-puun haara, josta käyttäjä pääsee valitsemaan sopivan arvon. Näin liitetään semanttista tietoa korttiin. Valitun luokan mukaan avautuu taulukkoon myös sen attribuutteja. Esimerkiksi kuvassa on valittu kirves listasta, jonka jälkeen lista häviää, mutta taulukkoon tulee uusia täytettäviä tietoja kirves-luokan ominaisuuksien mukaan. Ne ominaisuudet, joiden arvo on luokka, saavat alleen uuden listanpätkän, kun taas literaaliattribuutit saavat tyhjän kentän, ilman listaa. Luokan voi täyttää listasta valitsemisen sijasta myös kirjoittamalla luokan nimi tyhjään kenttään ja painamalla nappia Hae, jolloin lista häviää ja arvo lukkiutuu kuten listasta valittaessakin, jos kentän arvo vastaa jonkun luokan nimeä. Mikäli ominaisuuden arvoksi tulee instanssi, voidaan arvot syöttää tyhjiin kenttiin, kuten kuvan Löytäjä-attribuutin tapauksessa, ja painaa ”Tarkista olemassa olevat”, jolloin syötettyihin arvoihin sopivat instanssit tulevat vaihtoehdoiksi näkyviin listana. Niitä voi selata valitsemalla listasta, jolloin instanssien tiedot tulevat näkyviin kenttiin. Jos mikään instanssi ei kelpaa, voidaan täyttää tiedot käsin, jolloin luodaan uusi instanssi.

-- Kortin tallennus, kohta 5

Kortin muuttuneet ja lisätyt tiedot tallennetaan jatkuvasti skripteillä käyttäjän toimien lomassa, mutta kohdan viisi napista saa tiedot lopullisesti talteen, kun käsittely lopetetaan tai kortti on täysin valmis.

-- XML-tiedoston tallennus

XML-tiedostoa päivitetään kortteja tallennettaessa, joten tiedoston tallentamiseen ei ole erillistä nappia.

-- Käsiteltävän XML-tiedoston valmiiden korttien tallennus

Kortit tallennetaan tiedostonhallintaikkunan ”Tallenna valmiit” -napista. Tällöin tallentuvat nimenomaan valmiit RDF-instanssit.

9.3 Luokkakuvaus

Tässä kuvaukset JSP-sivuista ja siitä mitä niillä voi tehdä.

-- FilesAndCardsJSP

Tämä JSP-sivu mahdollistaa käyttäjälle tiedostojen sekä tiedostojen sisältämien korttien selaamisen. Sivulla näkyy ohjelman käyttöön kopioidut XML-tiedostot, ja kun yksi valitaan, tulee näkyviin sen sisältämät kortit. Tiedostojen sekä korttien tila (Käsittelemätön, Käsitelty, ja mahdollisesti Kesken) käy myös ilmi korttien ja tiedostojen nimien väristä (tai muulla koodauksella, esim. Valmis-Kesken).

Perusnäkymän lisäksi käyttäjä voi myös avata ruudun, josta saa lisättyä uusia tiedostoja ohjelman käsiteltävien tiedostojen listaan.

Tiedostojen ja korttien avaamisen lisäksi tähän ruutuun tulee myös napit joilla kortin ja tiedoston voi tallentaa, valmiin tiedoston poistaa, ja valmiit kortit tallentaa valmiiden korttien hakemistoon.

Toiminnallisuuden aikaansaamiseksi ollaan yhteydessä MDEController-luokan rajapintaan. MDEController-luokasta löytyvät metodit joilla päästään tekemään yllämainitut toiminnot. Sivun piirtäminen menee taas tagien ja CardState-luokan kautta.

-- InstanceCardJSP

Tällä JSP-sivulla käyttäjä lisää XML-korttiin semanttista tietoa. Kortin tiedot näyttävä taulukko laajenee kun siihen lisätään RDF-tietoa, niin että valitun luokan ominaisuudet (property) tulevat tämän alle taulukkoon. Mikäli ominaisuus on arvoltaan literaali, voi sitä muokata suoraan, ja jos sen arvoksi tulee luokka tai instanssi, avautuu taulukkoon hierarkkinen dynaaminen lista, joka kuvaa mahdollisia valittavia luokkia. Lista häviää kun luokka on valittu. Luokkia voi listan lisäksi myös hakea kirjoittamalla luokan nimi tekstikenttään ja painamalla ”hae”-nappia.

Mikäli RDF-ominaisuuksia on sidottu tiettyihin XML-elementteihin, tulee literaaliarvo suoraan XML:n elementin sisällöstä kenttään. XML-elementin arvo tulee joka tapauksessa suluissa kentän perään, oli ominaisuuden range mikä tahansa.

InstanceCardJSP:n toiminnallisuus kulkee osittain CardStaten ja osittain MDEControllerin kautta. Kun kortti avataan, pitää käyttäjän aluksi valita minkä luokan instanssi tehdään. Tämän jälkeen tämän luokan ominaisuudet avautuvat taulukkoon, mahdollisine XML-instanssin mukaisine tietoineen. Tiedot ovat erilaisten käyttöliittymäelementtiolioiden koosteena CardStatessa, josta tagien avulla haetaan tieto. Tietoja muutettaessa pysyvät literaalitiedot taulukossa, joka lähetetään sivulle itselleen aina kun sivu ladataan uudestaan. Välillä ne myös tallennetaan elementtiolioihin.

-- Luokka CardState

JSP-sivujen lisäksi käyttöliittymään liittyy myös luokka CardState, joka sisältää tietorakenteita käyttöliittymän tämänhetkisestä tilasta. Tagit piirtävät käyttöliittymän tietorakenteissa olevien olioiden mukaan. CardStatessa on myös metodeita, joilla voi vaihtaa hierarkkisten dynaamisten listojen tilaa, sekä lisätä RDF-olioita kortissa olevan instanssin ominaisuuksien arvoiksi.

Muuttujia:

MDEController controller, johon tulee viite ohjaimeen, jota käyttöliittymä käyttää

MDEClass instanceOf, johon tulee se luokka, jonka instanssi kortin esineen halutaan olevan

LinkedList XMLelements, johon tulee korttikohtainen XMLElement-olioiden lista.

LinkedList RDFelements, johon tulee korttikohtaiset RDF-tiedot.

LinkedList cardIds, johon tulee auki olevien korttien XMLCardIdentifier-oliot

Vector dynlist, johon tulee RDF-elementteihin liittyvät dynaamisten listojen tilat.

Metodeja:

public CardState() luo uuden CardState-olion

public LinkedList getBeans(String whatElements) palauttaa JSP-sivuille niiden tarvitsemia tietoja oikeassa muodossa

public void setInstanceOf(MDEClass mdeclass) laittaa luokan instanceOf:in arvoksi

public boolean setRDFElements() tekee instanceOf:ista ja sen propertyistä sekä niiden arvoista ja alipropertyistä RDFelements-listan

public boolean setRDFElements(MDEClass mdeclass) edellisen metodin rekursiivinen apumetodi

public void setDynlist(Vector Dynlist, …) muuttaa tietyn dynaamisen hierarkkisen listan tilaa

9.4 Tagikirjasto Meedio

Tässä osiossa pohditaan alustavasti millaisia tageja on toteutettava, jotta käyttöliittymä toimisi halutulla tavalla.

-- FileIteratorTag

Tämä tag käy läpi tiedostot, joiden tiedot se on saanut MDEControllerilta getFilesInUse-metodilla, ja kirjoittaa tiedostonhallintaruutuun niiden nimet, tilan määräämällä värillä tai kirjainkoodilla.

-- CardIteratorTag

Tämä tag käy läpi halutun tiedoston kortit samalla logiikalla kuin FileIteratorTag käy tiedostot. Sille annetaan parametrina tiedosto, jonka kortit olisi tarkoitus käydä läpi.

-- RDFElementIteratorTag

Tämä tag käy läpi RDF-elementit, antaen tarvittavia tietoja JSP-sivulle. Koska erilaisia RDF-elementtejä on paljon, ja ne voivat olla eri tiloissa, on tämä tag luultavasti huomattavan monimutkainen.

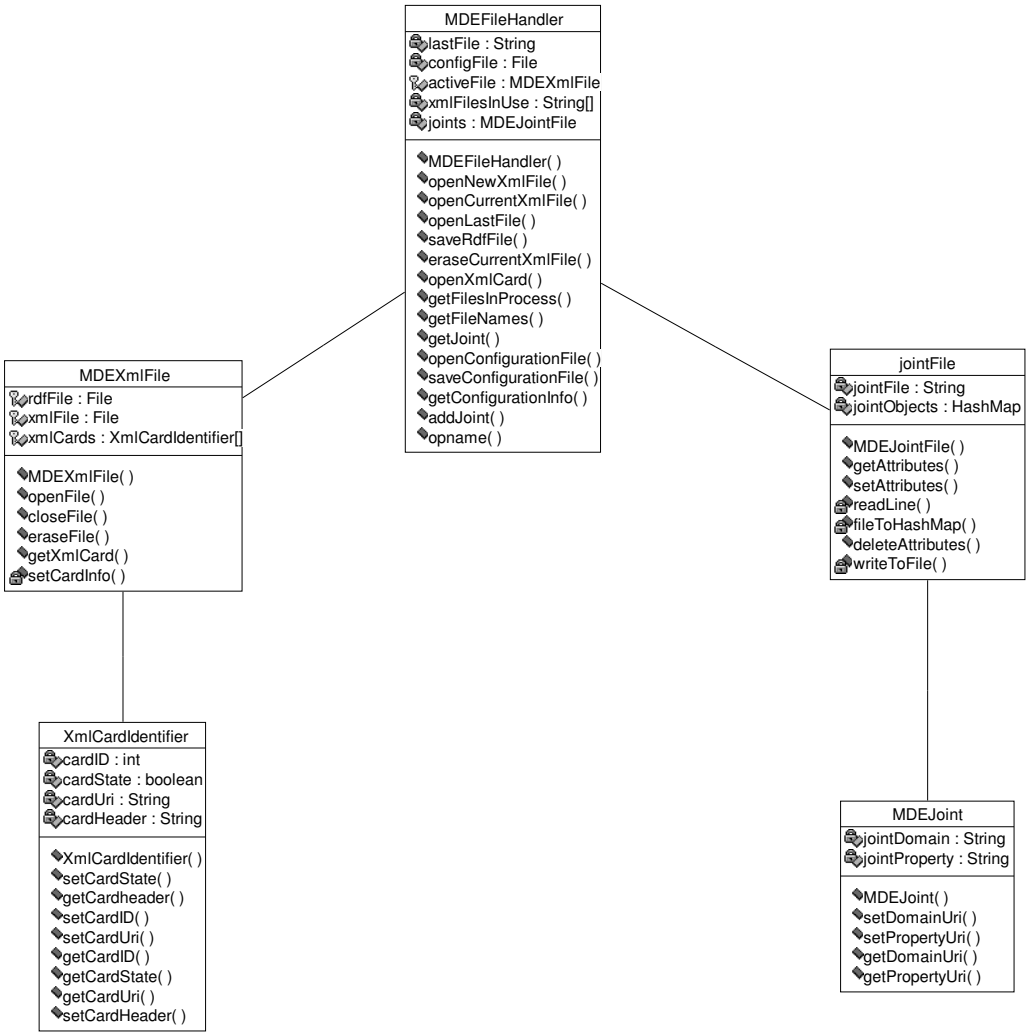
-- OntologyIteratorTag

Tämä tag käy läpi dynaamisen hierarkkisen listan, ja antaa sen tietoja JSP-sivulle.

10 Tiedostonkäsittelijä (J.K.)

Tässä luvussa kuvataan tiedostonkäsittelijä-osion toteutus. Tiedostonkäsittelijä tehtävänä on lukea XML-tiedostosta XML-kortteja, lukea konfiguraatiotiedostoa ja lukea liitokset sisältävää tiedostoa.

10.1 Luokat



Kuva 10.1: Tiedostonkäsittelijän luokat

10.1.1 XMLCardIdentifier

Luokka kuvaa XML-tiedoston sisältämän XML-kortin tietoja. Luokan avulla luodaan olio, joka sisältää tiedot kortin tunnistenumerosta, kortin tilasta (valmis, kesken), kortin perusteella luodusta RDF-ilmentymän URI:sta ja kortin otsikosta. Luokan tarjoamilla välineillä voidaan luoda korttia vastaavia olioita, muokata jo luotuja olioita ja tiedustella olion kenttien sisältämiä tietoja. Luokkaa käytetään MDEXMLFile-luokassa XMLCardIdentifier-olioita sisältävän taulukon luomiseen. Taulukossa olevien korttien id-tietoja välitetään edelleen käyttöliittymälle ohjaimen kautta.

Aksessorit:

-- public XMLCardIdentifier( )

-- public XMLCardIdentifier(int id)

-- public int getCardID( )

-- public boolean getCardState( )

-- public String getCardUri( )

-- public String getCardHeader( )

-- public void setCardID(int id)

-- public void setCardState(boolean state)

-- public void setCardUri(String cardUri)

Tietorakenteet: Luokassa käytetään ainoastaan Javan perustyyppejä.

10.1.2 MDEJoint

MDEJoint-luokan avulla luodaan RDF-skeeman luokan ja luokan ominaisuutta kuvaavia arvopareja, joita käytetään apuna MDEJointFile-luokassa elementin liittämiseksi tietyn luokan tiettyyn ominaisuuteen. Luokassa määriteltävät arvoparit sisältävät URI-osoitteita, joissa varsinaiset RDF-skeeman määrittelemät luokat ja luokan ominaisuudet sijaitsevat. Luokka tarjoaa välineet arvoparien luomiseksi ja niiden käsittelemiseksi. Luokkasta luotuja olioita sijoitetaan hajautustauluun luokassa MDEJointFile. MDEJoint-olioiden tietoja välitetään tarpeen vaatiessa käyttöliittymälle ohjaimen avulla.

Aksessorit:

-- public MDEJoint( )

-- public MDEJoint(String domainUri, String propertyUri)

-- public String getDomainUri( )

-- public String getPropertyUri( )

-- public void setDomainUri(String domainUri)

-- public void setPropertyUri(String propertyUri)

Tietorakenteet: Luokassa käytetään vain Javan perustyyppejä.

10.1.3 MDEXMLFile

Luokassa määritellään yhtä XML-kortteja sisältävää tiedostoa vastaavat rakenteet, joiden avulla korttien muokkaus RDF-muotoon suoritetaan. Luokka sisältää käsiteltävän XML-tiedoston, XML-tiedoston perusteella luodun XMLCardIdentifier-taulukon sekä tiedoston, johon tallennetaan RDF-ilmentymät. MDEXMLFile-luokka tarjoaa MDEFileHandler-luokalle palveluinaan tiedostojen

käsittelyyn vaadittavat välineet. Luokan ilmentymiä välitetään ohjaimen avulla muille järjestelmän osille. Luokka itse käyttää XMLCardIdentifier-luokan tarjoamia palveluja.

Kentät:

-- protected File XMLFile

-- protected File RDFFile

-- protected XMLCardIdentifier[] XMLCards

Aksessorit:

-- public MDEXMLFile(String XMLFileName)

-- public MDEXMLFile openFile(String XMLFileName)

-- public boolean closeFile( )

-- public boolean eraseFile(String fileName)

-- public String getXMLCard(int id)

Tietorakenteet:

- Luokassa käytetään XMLCardIdentifier-olioista muodostettua taulukkoa XML-korttien tietojen ylläpitämiseksi.
- Käsiteltävät tiedostot kuvataan File-olioina.

10.1.4 MDEJointFile

Luokka käsittelee järjestelmän ulkopuolisen tahon ylläpitämää liitettyjen ominaisuuksien tiedostoa. Luokan avulla muutetaan kyseisen tiedoston sisältämät liitokset hajautustauluksi, jota järjestelmä käyttää etsiessään halutulla avainarvolla varustettua liitosta. Luokan määrittelemä hajautustaulu sisältää tiettyä avainarvoa vastaavia MDEJoint-olioita. Avainarvoina toimivat museoesineiden elementit, joita vastaavia luokkia ja luokkien ominaisuuksia haetaan hajautustaulusta. MDEJointFile tarjoaa MDEFileHandler-luokalle liitosten haku palvelun. Luokka itse käyttää MDEJoint-luokan tarjoamia toimintoja apunaan. Ohjain käsittelee luokasta luotuja ilmentymiä välittäessään tietoa käyttöliittymälle.

Aksessorit:

-- public MDEJointFile( )

-- public MDEJoint getAttributes(String key)

-- public void setAttributes(String key, String domainUri, String propertyUri)

-- private static String readLine( )

-- private static void fileToHashMap( )

-- private static boolean writeToFile( )

Tietorakenteet:

- Hajautustaulua käytetään avainarvon yhdistämiseksi MDEJoint-olioon.
- Liitokset sisältävä tiedosto kuvataan File-oliona.

10.1.5 MDEFileHandler

MDEFileHandler tarjoaa ohjaimen välityksellä muulle järjestelmälle työkalut tiedostojen käsittelemiseksi. Luokka käyttää MDEXMLFile-luokan tarjoamia palveluja XML- ja RDF-tiedostojen käsittelemiseksi. MDEJointFile-luokan tarjoamilla välineillä MDEFileHandler käsittelee liitettyjä ominaisuuksia sisältävää tiedostoa. Luokka huolehtii myös konfiguraatiotiedoston lukemisesta.

Aksessorit:

-- public MDEFileHandler( )

-- public MDEXMLFile openNewXMLFile(String fileName)

-- public MDEXMLFile openCurrentXMLFile(String fileName)

-- public MDEXMLFile openLastFile( )

-- public String openXMLCard(int id)

-- public boolean openConfigurationFile( )

-- public boolean closeConfigurationFile( )

-- public boolean eraseCurrentXMLFile(String fileName)

-- public String getConfigurationInfo(String configurationFileItem)

-- public String[] getFileNames(String subDirectory)

-- public String[] getFilesInProcess( )

-- public String[] getJoints(String key)

-- public boolean saveRDFFile(String path)

Tietorakenteet:

- String-arvoja sisältävä taulukko järjestelmän hakemistossa sijaitsevien tiedostojen nimien kuvaamiseksi.
- MDEXMLFile-oliolla kuvataan käsiteltävä XML-tiedosto ja sen perusteella luotu XML-kortteja identifioiva XMLCardIdentifier-taulukko. Lisäksi olio sisältää viitteen tiedostoon, johon RDF-ilmentymät tallennetaan.
- MDEJointFile-olio, joka kuvaa liitettyjä ominaisuuksia.
- Konfiguraatiotiedoston sisältävä File-olio.

10.2 Konfiguraatiotiedosto

Konfiguraatiotiedostoa käsitellään tekstitiedostona ja se tallennetaan järjestelmän omaan käyttöön tarkoitettujen tiedostojen hakemistoon. Konfiguraatiotiedoston sisältämät komennot tulee sijoittaa kukin omalle rivilleen. Komennon alkaminen ilmoitetaan merkillä: < ja komennon loppuminen ilmoitetaan merkillä: >. Komento ja toimenpide erotetaan toisistaan yhtäkuin-merkillä (=). Kaikki muut merkit tulkitaan kommenteiksi. Konfiguraatiotiedostoa ei voi editoida järjestelmässä.

Esimerkki konfiguraatiotiedostosta:

Metadata-editor;

konfiguraatiotiedosto (meedio.cfg);

<RDF-SCHEMA=www.skeema.fi/RDFskeema.RDFs>
<XML-SCHEMA=www.skeema.fi/XMLskeema.XMLs>
<TOMCAT=c:\tomcat\tomcat.bat>
<TOMCAT_OFF=c:\tomcat\tomcat_off.bat>
<XML-CARD=card>
<CARD_HEADER_ATTRIBUTE=card_header>
<FIELD_HEADER_ATTRIBUTE=field_header>
<PICTURE_ATTRIBUTE=picture_element>

10.3 Liitettyjen ominaisuuksien tiedosto

Liitettyjen ominaisuuksien tiedostoa käytetään XML-elementtien automaattiseen muuttamiseen RDF:n ominaisuuksiksi. Liitettyjen ominaisuuksien tiedostoon tallennetaan kolmikoita, jotka koostuvat XML-korteissa esiintyvistä esineiden nimistä sekä RDF-skeemassa esiintyvistä luokista ja luokkien ominaisuuksista. Liitettyjen ominaisuuksien tiedosto toimitetaan tekstitiedostona järjestelmän ulkopuolelta. Tiedosto tulee laatia siten, että tiedoston alkaminen ilmoitetaan <JOINTS>-määreellä ja loppuminen </JOINTS>-määreellä. Yksittäisen liitetyn ominaisuuden alkaminen kuvataan <ONEJOINT>-määreellä ja loppuminen </ONEJOINT>-määreellä. Yksittäinen liitos kuvataan kolmikkona, johon kuuluvat xml-elementin nimen kertova kenttä, rdf-skeeman em. elementtiä vastaavan luokan kertova kenttä sekä em. luokkaan kuuluva luokan ominaisuutta kuvaava kenttä. XML-elementin automaattinen muuttaminen RDF:n ominaisuudeksi tapahtuu siten, että XML-elementin arvo sijoitetaan liitettyjen ominaisuuksien tiedostossa määritellyn elementtiä vastaavan luokan ominaisuuden arvoksi. Kyseiset kentät niihin sijoitettavine arvoineen kuvataan alla olevan esimerkin osoittamalla tavalla.

Esim.

<JOINTS>
    <ONEJOINT>
        <ELEMENT=Kokoelma>
        <DOMAINCLASS=http://protege.stanford.edu/museoesineet#Kokoelma>
        <PROPERTY= http://protege.stanford.edu/museoesineet#kokoelman_nimi>

```
    </ONEJOINT>
</JOINTS>
```

Järjestelmä lukee tiedoston sisältämät kolmikot ja muokkaa ne omaan käyttöönsä MDEJoint-olioksi, jotka sijoitetaan hajautustauluun. Alkuperäistä tiedostoa ei muokata.

10.4 Käsittelyssä olevien XML-korttien tilatiedot sisältävä tiedosto

Tiedosto sisältää tiedot käyttäjän avaaman XML-tiedoston sisältämistä korteista. Tiedoston avulla järjestelmä pitää kirjaa valmiista ja kesken olevista korteista sekä kortin perusteella luodun RDF-ilmentymän osoitteesta. Tiedosto tallennetaan sarjallistettuna oliona järjestelmän hakemistoon. Jokaisella XML-tiedostolla on oma korttien tilaa kuvaava tiedostonsa. Tiedosto nimetään liittämällä sitä vastaavan XML-tiedoston nimeen .xci-liite. XML-kortin tilatiedot kuvataan XMLCardIdentifier-olioina.

10.5 RDF-ilmentymät sisältävä tiedosto

RDF-ilmentymät tallennetaan omaan tiedostoonsa, johon ne kirjoitetaan tekstimuotoisina XML-merkkauksia käyttäen. Tiedosto tallennetaan sitä vastaavan XML-tiedoston nimisenä .RDF -liitteellä varustettuna.

11 Ohjain (M.A.)

Ohjaimeen kuuluu yksi luokka, jonka välityksellä käyttöliittymämoduli kommunikoi järjestelmän muiden osioiden kanssa. Tämän luokan nimi on MDEController. Kaiken toiminnan keskittäminen ohjaimen hallintaan mahdollistaa käyttöliittymän vaihtamisen järjestelmään mahdollisimman vaivattomasti. Kommunikointi käyttöliittymän ja ohjaimen välillä noudattaa suurin piirtein asiakas-palvelin-mallin mukaista rakennetta. Järjestelmä on siis helposti muutettavissa myös siten, että käyttöliittymä kutsuu ohjainta tietoliikenneyhteyden yli.

Luokan metodien tarkemmat kuvaukset ovat dokumentin liitteenä JavaDoc-muodossa.

11.1 Rajapinta

Luokan MDEController aksessorit:

-- public MDEController()

-- public String[] getFilenames(String subdirectory)

-- public String[] getFilesInProcess()

-- public String[][] OpenXMLFile(String filename)

-- public String[][] OpenLastXMLFile()

-- public LinkedList openXMLCard(String ID, MDEClass m)

-- public boolean createInstance(MDEClass m)
-- public Vector getRootClassURIs()
-- public Vector getSubClassURIs(String classURI)
-- public Vector getSuperClassURIs(String classURI)
-- public MDEClass getClass (String className)
-- public Vector doInstanceQuery (MDEClass m)
-- public boolean addJoint(String key, String _class, String predicate)
-- public MDEJoint getJoint(String key)
-- public boolean deleteJoint(String key)
-- public boolean saveRDFFile(String filename)
-- public boolean eraseCurrentXMLFile(String filename)
-- public String getCFGItem(String key)

12 Viittaukset lähteisiin

Tässä luvussa kuvataan dokumentissa esiintyvät viittaukset eri tekniikoihin.

- XML (Extensible Markup Language), lisätietoa: http://www.w3.org/XML/
- XML-Skeema, lisätietoa: http://www.w3.org/XML/Schema
- RDF (Resource Description Framework), lisätietoa: http://www.w3.org/RDF/
- Java Server Pages, lisätietoa: http://java.sun.com/products/jsp/index.html
- JSP+tag libraries, lisätietoa: http://java.sun.com/products/jsp/taglibraries.html
- Jakarta Tomcat, lisätietoa: http://jakarta.apache.org/tomcat/index.html
- Sax (Simple API for XML), lisätietoa: http://www.saxproject.org/
- Apachen Xerces, lisätietoa: http://XML.apache.org/xerces-j/index.html
- Dom (Document object model), lisätietoa: http://www.w3.org/DOM/
- Jena, lisätietoa: http://www.hpl.hp.com/semweb/jena-top.html
- HTML (Hypertext Markup Language), lisätietoja: http://www.w3.org/MarkUp/
- CSS (Cascading Style Sheets), lisätietoja: http://www.w3.org/Style/CSS/
- Java Applet, lisätietoja: http://java.sun.com/
- Java Servlet, lisätietoja: http://java.sun.com/

Liite 1: Muutokset määrittelydokumenttiin nähden

Ohjelmiston tärkeimmät muutokset määrittelydokumenttiin nähden ovat seuraavat:

Ohjelman toiminta on muuttunut siten, että nykyään ohjelmalla ei enää editoida XML:ää, vaan metadataeditori on nykyään puhtaasti instanssieditori. Metadataeditorilla luodaan pelkästään RDF-tietoa XML-tiedon pohjalta.

Muutoksia eri moduuleissa:

XML-käsittelijästä (määrittelydokumentissa XML-validaattori) on poistettu kokonaan XML:n editointi ja elementtikohtainen validointi. XML-käsittelijä validoi nykyään koko XML-tiedoston kerralla ja, jos tiedosto ei ole validi, niin sitä ei avata lainkaan. Näin ollen kaikki XML-käsittelijän määrittelydokumentissa määritellyt XML:n editointiin liittyvät toiminnot poistuvat. XML-tieto ei myöskään tallenneta eikä RDF-tietoa lisätä XML-kortin viimeiseksi elementiksi, kuten ennen. Joten myös XML:n lisäämiseen liittyvät toiminnot poistuvat.

Tiedostonkäsittelijä on muuttunut siten, että nykyään luotu RDF-tieto tallennetaan uuteen tiedostoon eikä samaan, jossa alkuperäinen XML-tieto on. Näin ollen tiedostonkäsittelijä joutuu käsittelemään nykyään kahta tiedostoa yhden sijasta. RDF-tiedon tallentaminen tiedostoon tapahtuu nykyään RDF-käsittelijässä, joten tiedostoon kirjoittamiseen liittyvä toiminnallisuus on siirtynyt nykyään tiedostonkäsittelijältä RDF-osiolle. Samaten se on muuttunut siten, että käsittelyssä olevaan XML-tiedostoon ei enää tallenneta mitään, vaan kaikki uusi tieto tallennetaan yhteen uuteen RDF-tiedostoon.

Tietoliikenne-osio on nykyään liitetty RDF-osioon, joten kaikki siihen liittyvä toiminnallisuus liittyy nykyään RDF-osioon. Tietoliikenne-osiolle määritelty toiminnallisuus on pysynyt kuitenkin suurin piirtein ennallaan. Tietoliikenne-osiota ei näin ollen enää ole olemassa.

RDF-käsittelijä hoitelee nykyään myös RDF-tiedon kirjoittamisen tiedostoon, kuten edellä tiedostonkäsittelijän yhteydessä on kuvattu, joten RDF-käsittelijän toiminnallisuus lisääntyy määrittelydokumenttiin nähden tältä osalta. Myös tietoliikenne-osion toiminnallisuus liittyy nykyään RDF-osioon, kuten edellä tietoliikenne-osion kuvauksen yhteydessä on kuvattu.

Liite 2: Metodikuvaukset Javadoc-muodossa

Metodikuvaukset javadoc-muodossa löytyvät Internetistä osoitteesta:
www.cs.helsinki.fi/group/meedio/dokumentit/suunnittelu/javadoc/