Semantic Web - Metadata Editor

Ohjelmistotuotantoprojekti
Ohjelmistotuotantoryhmä 1, Meedio

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

Helsinki 30. elokuuta 2002

Esimerkki dokumentti

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Versiohistoria

Versio	Päivämäärä	Laatija	Komentti
0.1	24.7.2002	Pekko Parikka	Dokumentin runko
0.2	29.8.2002	Ari Inkovaara	Dokumentin koostaminen ja oikoluku
0.3	30.8.2002	Justus Karekallas	oikoluku
1.0	30.8.2002	Ari Inkovaara	Valmis dokumentti

Sisältö

1 Johdanto (P.P.)

Tässä dokumentissa kuvataan Helsingin yliopiston tietojenkäsittelytieteen laitoksella ohjelmistotuotantoprojektissa 29.5.2002 - 30.8.2002 Meedio ryhmän tekemän ohjelmiston toteutus. Tämä dokumentti kuvaa järjestelmän toiminnan pääpiirteissään. Tärkeimpiä yksityiskohtia on myös kuvattu. Dokumentti on tarkoitettu helpottamaan lähdekoodin ymmärtämistä.

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

1.1 Ohjelmiston kuvaus

Meedio on metadataeditori, eli instanssieditori, jolla luodaan Resource Description Framework [RDF] -tyyppistä metatietoa Extensible Markup Language [XML] -tiedon pohjalta. Ohjelmalla luetaan tietokannasta tuotettua XML-tietoa, joka kuvaa esine- tai kuvakortteja, ja luodaan korteista RDF-skeeman [RDF(S)] -tyyppisen ontologian mukaisia ilmentymiä eli instansseja. Ohjelma on tarkoitettu Finnish Museum Online [FMO] projektin käyttöön museoiden esineiden luokittelemiseen. Ensimmäiseksi ohjelman on tarkoitus tulla Kansallismuseon käyttöön.

Ohjelmisto on toteutettu Java-kielellä käyttäen Jakartan Tomcat-palvelinympäristöä ja se toimii sekä Linux- että Windows-käyttöjärjestelmissä.

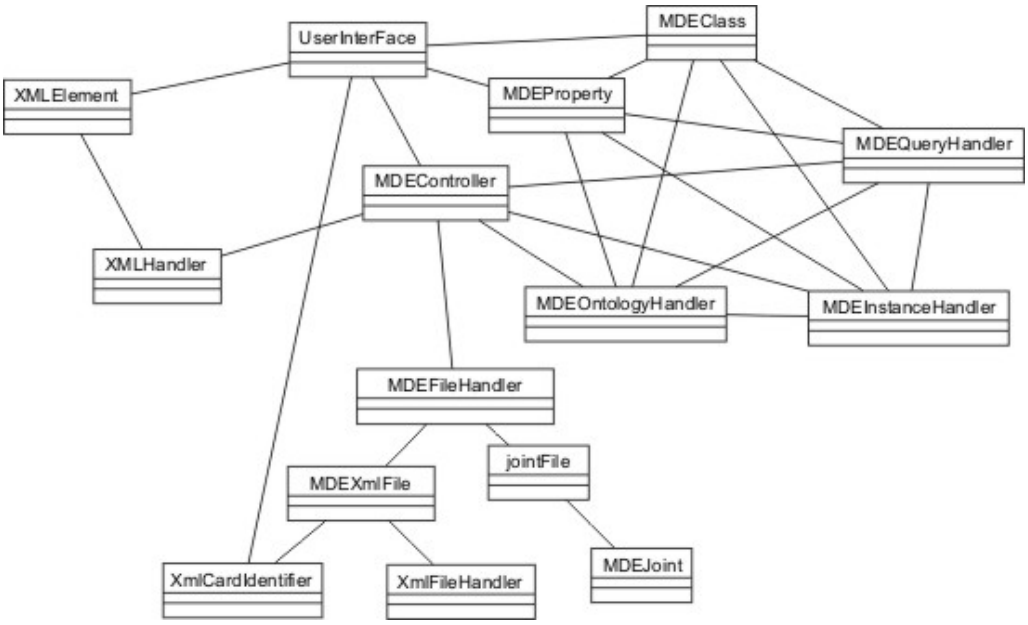
1.2 Tärkeimmät vaatimukset

Ohjelman tarkoitus on helpottaa museotyöntekijöitä tietokantojen esinekuvausten luokittelemisessa. Projektin tavoite on toimiva ja laadukas ohjelmisto vähintään perustoiminnoilla toteutettuna. Ohjelman on pystyttävä lukemaan RDF(S)-muotoisia ontologioita ja tuottamaan niiden mukaisia ilmentymiä, RDF-muotoisia semanttisia luokitteluita. Ohjelman on oltava mahdollisimman geneerinen, eli sen on kyettävä toimimaan erilaisilla ontologioilla ja XML-skeemoilla. Ohjelman käyttöliittymän on myös oltava helposti muokattavissa ja vaihdettavissa.

2 Järjestelmän yleiskuvaus (P.P.)

Meedio-ohjelmisto on jaettu viiteen osioon: Käyttöliittymään, Ohjaimeen, RDF-käsittelijään, XML-käsittelijään sekä tiedostonkäsittelijään. Kuvassa 1 esitetään kaikki osajärjestelmät ja niiden keskinäiset viittaukset.

Meedio ohjelmisto noudattaa asiakas-palvelin mallin kaltaista suunnittelumallia, jossa käyttöliittymä toimii asiakkaana pyytäen ohjaimelta palveluja. Tämän lisäksi ohjelmiston palvelinpuoli jakautuu kahteen tasoon, joista ylempänä on ohjain ja sen alla muut osajärjestelmät. Käyttöliittymä kutsuu vain ohjaimen palveluja. Näin käyttöliittymä on helposti vaihdettavissa. Käyttöliittymä on toteutettu Java-Server Pages Standard Tag Library [JSTL] -tekniikalla. Valitun tekniikan ansiosta käyttöliittymän ulkoasua on melko helppo muokata. Koska Meedio-ohjelmiston rakenne muistuttaa asiakas–palvelin -mallia, on ohjelmisto melko helposti muokattavissa sellaiseksi, että palvelinosiota käyttää yhtä aikaa monta käyttöliittymää. Tätä varten tarvitsee vain toteuttaa ohjaimen ja käyttöliittymän väliin palvelupyyntöjä käsittelevä moduuli, joka huolehtii palvelupyyntöjen jonottamisesta.



Kuva 1: Järjestelmän yleisrakenne

2.1 Hakemistorakenne

Järjestelmän hakemistorakenne on kuvattu taulukossa 1.

Tiedoston nimi	**Tiedoston käyttö**
classes/ui/tags	Hakemisto, johon käännetyt luokat sijoitetaan
systemfiles/	Hakemisto, jossa on kaikki ohjelman käyttämät tiedostot
systemfiles/MDEdata/	Hakemisto, jossa on konfiguraatiotiedosto, liitokset määrittelevä tiedosto, viimeisen käsitellyn tiedoston nimi
systemfiles/identifiers	Hakemisto, jossa on kaikkien avoinna olevien tiedostojen tilatiedot sarjallistettuina XmlCardIdentifier-taulukko-olioina. Tiedostojen pääte on .xci
systemfiles/rdffiles	Hakemisto, jossa on kaikki ohjelman käsittelyssä olevat RDF-tiedostot
systemfiles/xmlfiles	Hakemisto, jossa on kaikki ohjelman käsittelyssä olevat XML-tiedostot
index.html	Käyttöliittymän käynnistävä html-sivu
configuration.mde	Ohjelman konfiguraatiotiedosto
lastfile	Edellisen avatun tiedoston nimi sarjallistettuna String-oliona
liitokset.mde	Liitokset XML-elementtien ja RDF(S)-luokkien välillä

Taulukko 1: Järjestelmän hakemistorakenne

2.2 Käyttöliittymä

Käyttöliittymä on toteutettu Tomcat-webpalvelimen päälle ja sitä käytetään internetselaimella. Käyttöliittymä on toteutettu Hypertext Markup Languagella [HTML]

ja siinä on käytetty hyväksi Javascriptiä. Käyttöliittymä kutsuu pelkästään ohjaimen palveluja ja on siksi erittäin helposti korvattavissa uudella. Koska käyttöliittymän piirtäminen on toteutettu JSTL-tekniikalla, on sen ulkoasu melko helposti muutettavissa. Tarkempi kuvaus käyttöliittymästä on luvussa 7.

2.3 Ohjain

Ohjain tarjoaa korkeamman tason rajapinnan käyttöliittymälle kaikkiin ohjelman tarjoamiin palveluihin. Ohjain käyttää RDF-käsittelijää, XML-käsittelijää sekä tiedostonkäsittelijää. Myös muut osat järjestelmää, kuin käyttöliittymä, käyttävät ohjaimen tarjoamia palveluja. Ajatuksena on, että kaikki osajärjestelmien välinen kommunikointi tapahtuu ohjaimen kautta. Ohjain toimii eräänlaisena palvelimena käyttöliittymälle. Jos ohjelman toimintaa halutaan muuttaa, voidaan muuttaa pelkästään ohjainta. Myös minkä tahansa ohjaimen alla olevan osion voi vaihtaa ilman, että käyttöliittymälle tarvitsee tehdä mitään muutoksia. Ohjaimen tarkempi kuvaus on luvussa 3.

2.4 RDF-käsittelijä

RDF-käsittelijä hoitaa kaiken RDF:n ja RDF-skeemojen käsittelyn. RDF-käsittelijän avulla selataan ontologiaa, luodaan uusia instansseja ja kirjoitetaan ne levylle sekä tehdään instanssikyselyjä verkon yli instanssikantaan. RDF-käsittelijän toteutuksessa on käytetty Jena-rajapintaa. RDF-käsittelijä on muutettavissa käyttämään mitä tahansa muuta rajapintaa kuin Jena-rajapintaa, toteuttamalla Meedio-ohjelmiston käyttämät rajapinnat RDFOntologyHandler, RDFInstanceHandler ja RDFQueryHandler. RDF-käsittelijä käyttää tiedon välitykseen luokkia MDEClass ja MDEProperty, jotka kuvaavat ontologian luokan ja ominaisuuden eli propertyn. Tarkempi kuvaus RDF-käsittelijästä on luvussa 5.

2.5 XML-käsittelijä

XML-käsittelijä hoitaa XML:n validoinnin XML-skeemaa vasten ja XML:n parsinnan. Kun ohjelmassa avataan uusi tiedosto käyttöön, se validoidaan konfiguraatiotiedostossa määriteltyä XML-skeemaa vasten. Jos luettu XML-tiedosto ei ole validia, niin siitä aiheutuu poikkeus eikä tiedostoa avata. Kun avatusta tiedostosta valitaan jokin XML-kortti käsiteltäväksi, niin se annetaan parsittavaksi XML-käsittelijälle. XML-käsittelijän tarkempi kuvaus on luvussa 4.

2.6 Tiedostonkäsittelijä

Tiedostonkäsittelijä hoitelee kaiken levyltä lukemisen ja kirjoittamisen, paitsi RDF:n kirjoittamisen tiedostoon, jonka RDF-käsittelijä tekee. Tiedostonkäsittelijä kopioi uudet XML-tiedostot järjestelmän käyttöön, luo niille vastaavan RDF-tiedoston, johon instanssit kirjoitetaan RDF-käsittelijän toimesta, tiedostonkäsittelijä lukee uudet tiedostot läpi ja luo niistä tietorakenteen, jossa on tiedoston sisältämien korttien ID:t, otsikot ja tilatiedot (käsittelemätön / valmis). Tiedostonkäsittelijä tallentaa viimeksi käsitellyn tiedoston nimen ja se myös poistaa järjestelmästä käsittelyssä olevia tiedostoja. Tarkempi kuvaus tiedostonkäsittelijästä on luvussa 6.

3 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 niin, että käyttöliittymä kutsuu ohjainta tietoliikenneyhteyden yli.

3.1 MDEController

Luokan tarkoitus on toimia järjestelmässä muiden osioiden ohjaimena. Ohjaimen kutsumisesta vastaa pääasiassa käyttöliittymä.

Muuttujat

private final String CARD_START="card_start_attribute" Määrittelee konfiguraatiotiedoston kohdan, josta kortin aloittava merkkijono luetaan.

private final String CARD_ID="card_id_attribute" Määrittelee konfiguraatiotiedoston kohdan, josta kortin id:n määrittävä tieto luetaan.

private final String DB_ID_CLASS="db_id_class" Määrittelee konfiguraatiotiedoston kohdan, josta DBIDCLASS-muuttujaan luetaan arvo.

private final String DB_ID_PROPERTY="db_id_property" Määrittelee konfiguraatiotiedoston kohdan, josta DBIDPROPERTY-muuttujaan luetaan arvo.

private final String MUSEUM_ID="museum_id" Määrittelee konfiguraatiotiedoston kohdan, josta museon identifioiva merkkijono luetaan.

private final String CARD_HEADER="card_header_attribute" Määrittelee konfiguraatiotiedoston kohdan, josta kortin header-tiedon määrittävä merkkijono luetaan.

private final String XML_SKEEMA="xml-schema" Määrittelee konfiguraatiotiedoston kohdan, josta xml-skeeman nimen määrittävä merkkijono luetaan.

private final String CSTART, CID, CHEADER, DBIDCLASS,

DBIDPROPERTY, MUSEUMID Näihin muuttujiin konstruktori lukee arvot. CSTART:n luetaan kortin aloittavan kentän määrittävä merkkijono. CID:n luetaan kortin ID:n määrittävä merkkijono. CHEADER:n luetaan kortin header:n määrittävä merkkijono. DBIDCLASS:n luetaan tarvittava merkkijono. DBIDPROPERTY:n luetaan tarvittava merkkijono. MUSEUMID:n luetaan museon identifioiva merkkijono.

private MDEOntologyHandler rdfOH Viittaus ontologiakäsittelijään.

private MDEInstanceHandler rdfIH Viittaus instanssikäsittelijään.

private MDEQueryHandler rdfQH Viittaus instanssikyselykäsittelijään.

private MDEFileHandler mdeFH Viittaus tiedostonkäsittelijään.

private MDEXmlFile mdeXF Viittaus xml-tiedostoon (MDEXmlFile-luokka).

private XMLHandler xmlH; Viittaus XML-käsittelijään.

Konstruktorit

Luokassa on yksi konstruktori, jossa tarvittavat muuttujat alustetaan konfiguraatiotiedostosta luetuilla tiedoilla.

public MDEController() throws MDEException

Metodit

Tässä kuvataan luokan metodit.

Suunnitteludokumentissa määritellyt metodit:

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)

Toteutetut metodit:

public String[ ] **getFilenames(String subdirectory) throws MDEException** Palauttaa parametrina annetusta polusta XML-tiedostonimet.

public String[ ] **getFilesInProcess()** Palauttaa järjestelmän käsittelyssä olevien XML-tiedostojen nimet.

public XmlCardIdentifier[ ] **openXMLFile(String filename) throws MDEException** Avaa halutun nimisen XML-tiedoston ja palauttaa listan, joka sisältää avatun XML-tiedoston korttien ID-tiedot, käsittelyvaiheet ja otsikkotiedot. Jos parametrinä annettu tiedosto ei vielä ole järjestelmän käytössä, kopioidaan se ensin järjestelmän käyttöön ja luodaan sitä vastaava tyhjä RDF-tiedosto. Tämän jälkeen palautetaan lista korttien tiedoista.

public XmlCardIdentifier[ ] **OpenLastXMLFile() throws MDEException** Avaa viimeksi käsitellyn XML-tiedoston. Palauttaa listan joka sisältää avatun XML-tiedoston korttien ID-tiedot, käsittelyvaiheet sekä otsikkotiedot

public LinkedList openXMLCard(String ID, MDEClass m) throws MDEException Avaa halutun XML-kortin, palauttaa kortin tiedot XMLElement-olioista koostuvana linkitettynä listarakenteena sekä viiteparametrin arvona tiedot kortin ID:tä vastaavasta RDF-instanssista, jos sellainen löytyy.

public void createInstance(MDEClass m, String ID) throws MDEException Luo ilmentymän kortin semanttisesta luokituksesta. Parametrinä MDEClass-olio, josta generoidaan RDF RDFInstanceHandlerin avulla ja kortin ID, joka tarvitaan Uniform Resource Identifiers:n [URI] generoimiseen.

public Vector getRootClassURIs() Palauttaa ontologiasta kaikki juuriluokkien URIt. Paluuarvona String-olioita sisältävä Vector-olio. Toiminta: kutsuu RDF-käsittelijän getRootClassNames()-metodia, ja palauttaa sen paluuarvon kutsujalle.

public Vector getSubClassURIs(String classURI) Palauttaa halutun luokan aliluokkien URIt String-olioista koostuvana Vectorina. Toiminta: kutsuu RDF-käsittelijän getSubClassNames()-metodia, ja palauttaa sen paluuarvon kutsujalle

public Vector getSuperClassURIs(String classURI) Palauttaa luokan yliluokkien nimet.Toiminta: kutsuu RDF-käsittelijän getSuperClassNames()-metodia, ja palauttaa sen paluuarvon kutsujalle.

public MDEClass getClass (String classURI) throws MDEException Palauttaa halutun luokan MDEClass-oliona, joka sisältää kaikki luokan ominaisuudet. Toiminta: kutsuu RDF-käsittelijän getClass()-metodia, ja palauttaa sen paluuarvon kutsujalle

public Vector doInstanceQuery (MDEClass m) throws MDEException Suorittaa instanssikyselyn parametrinään MDEClass-olio. Toiminta: kutsuu RDF-instanssikäsittelijän doInstanceQuery()-metodia, ja palauttaa sen paluuarvon kutsujalle

public String getElement(String class_, String property) throws MDEException Palauttaa luokan URIin ja luokan propertyn URIin liitetyn elementin.

public boolean addJoint(String key, String _class, String predicate)

throws MDEException Lisää liitosparin, parametreinä 3 arvoa: key, joka on xml-kenttä joka liitetään, sekä Class, eli domainluokan URI ja predikaatti, joihin

kahteen avainarvo liitetään. Avainarvo liitetään siis aina kahteen arvoon: Luokkaan ja johonkin siihen liittyvään propertyyn. Property on siis predikaatti, esim. hasname, joka annetaan parametrinä URIna.

public MDEJoint getJoint(String key) throws MDEException Hakee liitosta avaimella. Palauttaa MDEJoint:n joka sisältää domainclass URI + propertyn URI eli predikaatti.

public boolean deleteJoint(String key) Poistaa liitoksen.

public boolean saveRDFFile(String pathName) throws MDEException Tallentaa käsittelyssä olevasta XML-tiedostosta luodut instanssit sisältävän RDF-tiedoston haluttuun sijaintiin.

public boolean eraseCurrentXMLFile(String filename) throws MDEException Poistaa halutun käsittelyssä olevan XML- ja RDF-tiedoston levyjärjestelmästä. Metodi poistaa vain ohjelman käsittelyssä olevia tiedostoja. Jos parametrinä antaa XML-tiedoston nimen, niin ohjelma poistaa automaattisesti sitä vastaavan RDF-tiedostonkin levyjärjestelmästä.

public String getConfigItem(String key) Palauttaa halutun tiedon konfiguraatiotiedostosta. Kutsuu tiedostonkäsittelijän konfiguraatiotiedoston käsittelystä vastaavaa metodia.

Tietorakenteet

Luokassa käytetään Javan tietorakenneluokkia HashMap ja Vector.

4 XML-käsittelijä (A.I.)

XML-käsittelijän tehtävänä on validoida annettu XML-ilmentymä annettua XML-Skeemaa vasten ja tuottaa annetusta XML-kortista XMLElement-olioita.

4.1 XMLHandler

Luokan tehtävänä on validoida annettu XML-ilmentymä annettua XML-Skeemaa vasten ja tuottaa annetusta XML-kortista XMLElement-olioita. XML:n parsinnassa ja validoinnissa käytetään Apachen Xerxcesin Simple API For XML [SAX] -rajapintaa.

Muuttujat

LinkedList elements lista XMLElement-olioita

String xml Käsiteltävä XML-Skeeman ilmentymä

String schemauri XML-Skeeman URI

String xmlFile XML-tiedoston osoite

String cardElement XML-kortin alku / loppu tagi

String cardHeaderAttribute XML-kortin otsikon kertovan attribuutin nimi

String fieldHeaderAttribute Yksittäisen XML-kortin elementin nimen kertovan attribuutin nimi

String cardHeader XML-kortin otsikko

boolean output Ollaanko XML-kortin sisällä.

boolean processingElement onko SAX-parseri parhaillaan elementin sisällä vai ulkona

boolean validating Onko validointi käynnissä, jos ei käsitellään XML-korttia

boolean valid Onko XML-Skeeman ilmentymä validi

XMLElement element Parsittava elementti.

Konstruktorit

XMLHandler(MDEController controller) parametrina MDEController-olio

Metodit

void setXMLInstance(String xml) Asettaa validoitavan XML-skeeman instanssin.

void setXMLSchema(String uri) Asettaa validointiin käytettävän XML-skeeman osoitteen

void parse() Käynnistää yksittäisen XML-kortin prosessoinin.

void setXMLFile(String uri) Asettaa validoitavan XML-tiedoston osoitteen.

void validate() Validoi XML-tiedoston annettua XML-skeemaa vasten.

LinkedList getElements() Palauttaa listan XMLElement-olioita.

String getHeader() Palauttaa XML-kortin otsikon.

Lisäksi luokassa on muutama SAX-parserin kutsuma metodi, joita ei tässä esitetä.

4.2 XMLElement

XMLElement-luokka säilöö yksittäisen XML-elementin tietoja.

Muuttujat

String name String elementin nimi

String value String elementin arvo

String header String elementin otsikko

HashMap attributes name, value -pari elementin attribuuteista

Konstruktorit

XMLElement(String name) parametrina elementin nimi

Metodit

String getName() palauttaa elementin nimen

void setValue(String value) asettaa elementin arvon

String getValue() palauttaa elementin arvon

void setHeader(String header) asettaa elementin otsikon

String getHeader() palauttaa elementin otsikon

void setAttribute(String name, String value) asettaa elementille attribuutin ja sille arvon.

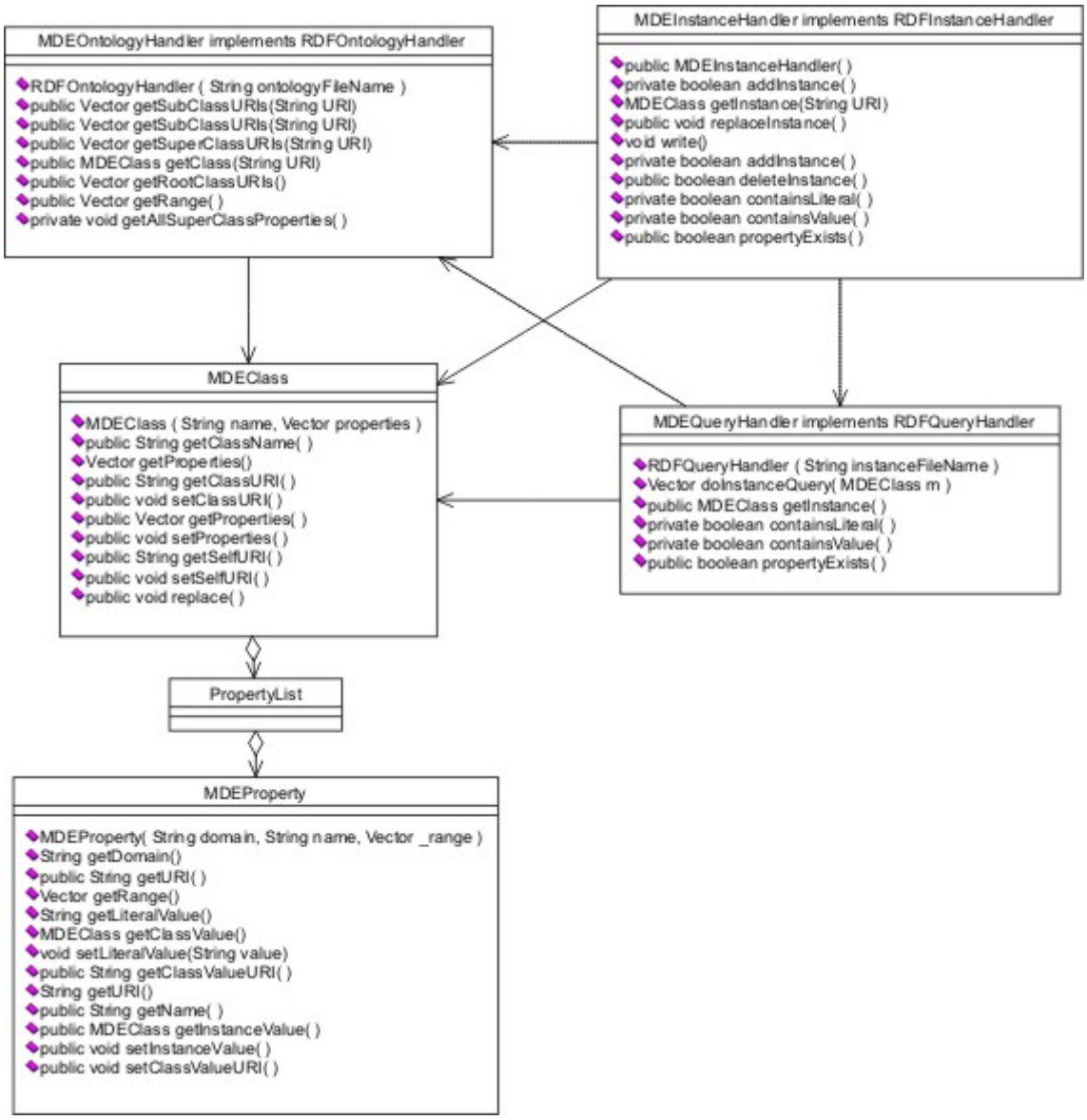
HashMap getAttributes() palauttaa HashMap:in elementin attribuutteja. String name, String value -pareina.

5 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-instanssien käsittelyyn sekä instanssikyselyn tekemiselle RDF-tietokantaan. RDF-käsittelijä koostuu seuraavista luokista, joiden suhteet käyvät ilmi kuvasta 2: MDEOntologyHandler (vastaa RDF(s)-muotoisen ontologian käsittelystä), MDEInstanceHandler (vastaa instanssien käsittelystä), MDEQueryHandler (vastaa instanssikyselyn tekemisestä rdf-tietokantaan), MDEClass (pitää sisällään tiedot yhdestä RDF-luokasta), MDEProperty (pitää sisällään tiedot yhdestä RDF-luokan ominaisuudesta) .

5.1 MDEClass

Luokan tarkoitus on säilyttää tietoa yksittäisestä RDF-luokasta (ja siitä luotavasta instanssista).



Kuva 2: RDF-käsittelijän rakenne

Muuttujat

private String classURI Muuttuja sisältää luokan URI:n.

private String selfURI Muuttuja sisältää luokasta luodun/luotavan instanssin URI:n.

private Vector propertyList Muuttuja sisältää MDEProperty-olioista koostuvan

Vector-taulukon

Konstruktorit

Luokassa on yksi konstruktori, joka saa ensimmäisenä parametrinaan luokkaURI:n ja toisena MDEProperty-olioita sisältävän Vector-taulukon.

public MDEClass(String _classURI, Vector properties) throws MDEException

Metodit

Seuraavaksi esitellään luokan metodit.

Suunnitteludokumentin mukaiset metodit:

public String getClassName() Palauttaa luokan nimen, eli URI pätkäistynä "#-merkin kohdalta, palauttaa null:n, jos #-merkkiä ei löydy.

public String getClassURI() Palauttaa luokan URI:n

public Vector getProperties() Palauttaa luokan propertyt MDEProperty:istä koostuvana Vector:na.

Seuraavat metodit on lisätty toteutusvaiheessa:

public void setClassURI(String _classURI) Asettaa luokan URI:n.

public void setProperties(Vector props) Asettaa luokan ominaisuudet.

public String getSelfURI() Palauttaa luokasta luotavan/luodun instanssin URI:n.

public void setSelfURI(String URI) Asettaa luokasta luotavan instanssin URI:n.

public void replace(MDEClass m) Korvaa luokan muuttujien arvon parametrina annettavan MDEClass-olion arvoilla.

Tietorakenteet

Ainoat luokassa käytettävät tietorakenteet ovat Java:n Vector-taulukot ja String-oliot.

5.2 MDEProperty

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.

Muuttujat

private String domain Domain-luokan URI, eli tieto siitä mihin luokkaan kyseinen ominaisuus kuuluu.

private String name Sisältää ominaisuuden nimen, eli URI:n josta on katkaistu namespace-osa pois.

private String URI Ominaisuuden URI.

private Vector range String-olioita sisältävä Vectori, joka sisältää ominaisuuden sallimat arvot.

private String literalValue Ominaisuudelle asetettu literaaliarvo.

private MDEClass instanceValue Ominaisuudelle asetettu instanssiarvo.

private String classValueURI Ominaisuudelle asetettu luokka-arvo.

Konstruktorit

Luokassa on yksi konstruktori, joka saa parametrinaan domain-luokan nimen, ominaisuuden URI:n ja sallitut arvot sisältävän range-Vectorin.

public MDEProperty(String _domain, String _URI, Vector _range)

throws MDEException

Metodit

Suunnitteludokumentissa suunnitellut luokan metodit:

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)

Toteutetut metodit:

public String getDomain() Palauttaa ominaisuuden domainluokan URI:n. Palauttaa siis luokkamuuttujan domain arvon.

public String getName() Palauttaa ominaisuuden nimen. Palauttaa siis luokkamuuttujan name arvon.

public String getURI() Palauttaa ominaisuuden URI:n. Palauttaa siis luokkamuuttujan URI arvon.

public Vector getRange() Palauttaa ominaisuuden sallimien arvojen Vector-taulukon. Palauttaa siis luokkamuuttujan range.

public String getLiteralValue() Palauttaa ominaisuudelle asetetun literaaliarvon. Palauttaa luokkamuuttujan literalValue.

public MDEClass getInstanceValue() Palauttaa ominaisuudelle asetetun instanssiarvon. Palauttaa luokkamuuttujan instanceValue arvon.

public void setLiteralValue(String value) throws MDEException Asettaa ominaisuudelle literaaliarvon. Asettaa luokkamuuttujan literalValue arvon parametrina annetuksi arvoksi.

public void setInstanceValue(MDEClass value) throws MDEException Asettaa ominaisuudelle instanssiarvon. Asettaa luokkamuuttujan instanceValue arvon parametrina annetuksi arvoksi.

public void setClassValueURI(String _uri) Asettaa ominaisuudelle luokka-arvon. Asettaa luokkamuuttujan classValueURI arvoksi parametrina annetun arvon.

public String getClassValueURI() Palauttaa ominaisuudelle asetetun luokka-arvon. Palauttaa siis luokkamuuttujan classValueURI arvon.

Tietorakenteet

Luokassa käsitellään Javan Vector-taulukoita ja String:ejä. Muita tietorakenteita luokassa ei käytetä.

5.3 MDEOntologyHandler

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

Muuttujat

private Model model Jena-oliomalliin kuuluva Model-luokan ilmentymä, johon ontologia luetaan.

private HashMap classMap Hajautustaulu, jossa avaimena on luokan URI, ja arvona Vector-taulukko, joka sisältää kyseisen URI:n aliluokat ontologiassa.

private HashMap rangeMap Hajautustaulu, jossa avaimena on ominaisuuden URI, ja arvona Vector-taulukko, joka sisältää ominaisuudelle sallitut arvot.

private HashMap domainMap Hajautustaulu, jossa avaimena on ominaisuuden URI, ja arvona Vector-taulukko, joka sisältää ominaisuuteen liittyvät kantaluokat.

private HashMap superclassMap Hajautustaulu, jossa avaimena on luokan URI, ja arvona Vector-taulukko, joka sisältää kyseistä URI:a vastaavat yliluokat ontologiassa.

private String ontologyName Ontologiatiedoston nimi.

Konstruktorit

Luokassa on yksi konstruktori, jossa luetaan RDF(s)- muotoinen ontologia ensin Jena:n Model-luokan ilmentymään. Tämän ilmentymän kaikki lausekkeet käydään läpi ja muodostetaan niiden perusteella luokan sisäiset hajautusrakenteet (classMap, rangeMap ,domainMap sekä superclassMap. Konstruktori aiheuttaa MDEException-poikkeuksen tapauksessa, jossa parametrina annettu RDF(s)-muotoisen ontologian nimi on virheellinen, tai RDF(s)-tiedosto on väärän muotoinen tai sitä ei saada esim. puutteellisten levyjärjestelmän oikeuksien vuoksi auki.

public MDEOntologyHandler(String ontologyFileName) throws MDEException

Metodit

Tässä kuvataan luokan metodit.

Suunnitteludokumentissa määritellyt metodit:

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)

Toteutetut metodit:

public Vector getRootClassURIs() Palauttaa ontologian juuriluokkien nimet String-olioista koostuvana Vector-taulukkona. Hakee luokan sisäisestä luokat sisältävästä hajautusrakenteesta (classMap) RDF-määritelmän mukaisen yliluokan aliluokat kutsumalla getSubClasses()- funktiota parametrilla "..#Resource".

public Vector getSubClassURIs(String URI) Palauttaa ontologiasta halutun luokan aliluokkien nimet String-olioista koostuvana Vector-taulukkona. Aliluokkien nimet löytyvät hajautusrakenteesta className-avaimen takaa.

public Vector getSubClassURIs(MDEClass c) Sama kuin edellinen, mutta ottaa parametrinaan MDEClass-olion.

public Vector getSuperClassURIs(String URI) Palauttaa luokan yliluokkien nimet. Yliluokkien nimet löytyvät superClassMap-hajautustaulusta.

public MDEClass getClass (String URI) throws MDEException Palauttaa halutun luokan MDEClass-oliona, joka sisältää luokan ominaisuudet (Properties). Toimii siten, että hajautusrakenteesta domainMap haetaan parametrin

nimeä vastaava domain:it sisältävä Vector-taulukko. Tämä taulukko käydään läpi hakien jokaista alkiota vastaava range-tieto Vectoriin. Näistä tiedoista muodostetaan MDEProperty-olioita, jotka liitetään palautettavaan MDEClass-olioon.

public Vector getRange(String URI) Palauttaa ominaisuuden sallimat arvot, eli rangen vector-taulukkona. Palauttaa siis luokkamuuttujan range arvon.

private void getAllSuperClassProperties(String URI, Vector results, HashMap h) throws MDEException Tätä metodia käytetään apuna getClass:ssa. Palauttaa luokan yliluokkien ja niiden yliluokkien propertyt. Parametrina kuljetettava HashMap on duplikaattien poistoa varten, eikä metodin kutsuja tarvitse sitä.

Tietorakenteet

Luokassa käytetään Javan tietorakenneluokkia HashMap ja Vector. Lisäksi käytetään jena-oliomallin Model-tietorakennetta.

5.4 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.

Muuttujat

Tässä kuvataan MDEInstanceHandler-luokan muuttujat.

private Model model Jena-oliomallin Model-luokan ilmentymä, jota luokka käsittelee.

private String instanceFilename Instanssitiedoston nimi.

private MDEOntologyHandler ontologyHandler Luokassa tarvitaan tietoja ontologiasta, jotka saadaan tämän viitteen kautta.

private MDEQueryHandler queryHandler Luokassa tarvitaan tietoja rdf-tietokannasta. queryHandler Vastaa kyselyistä rdf-tietokantaan.

private String museum_ID Museon yksilöivä merkkijono. Tämä liitetään uusien instanssien URI:n.

private long instanceCounter Luotavien instanssien URI:in lisättävä laskuri.

Konstruktorit

Luokassa on yksi konstruktori, jossa luetaan RDF- muotoinen instanssitiedosto Jena:n Model-luokan ilmentymään ja asetetaan luokan tarvitsemat muuttujat kohdalleen. Konstruktori aiheuttaa MDEException-poikkeuksen tapauksessa, jossa parametrina annettu RDF-muotoisen instanssitiedoston nimi on virheellinen, tai RDF-tiedosto on väärän muotoinen tai sitä ei saada esim. puutteellisten levyjärjestelmän oikeuksien vuoksi auki.

public MDEInstanceHandler (String _instanceFilename, MDEOntologyHandler mdeO, MDEQueryHandler mdeQ, String _museum_ID) throws MDEException

Metodit

Tässä kuvataan MDEInstanceHandler-luokan metodit.

Suunnitteludokumentissa määritellyt metodit:

public RDFInstanceHandler (String _instanceFilename)

public void addInstance(MDEClass m)

public MDEClass getInstance(String URI)

public void replaceInstance(String URI, MDEClass m)

public void write()

Toteutetut metodit:

private boolean addInstance(MDEClass m, String ID) throws MDEException Luo parametrina annetun MDEClass-olion perusteella ilmentymän luokasta Jenan model-malliin.

private boolean addInstance(MDEClass m) throws MDEException Vastaava, kuin edellinen, muttei generoi URI:a. Tämä on replaceInstance-metodia varten.

public MDEClass getInstance(String URI) throws MDEException Palauttaa halutun instanssin MDEClass-oliona.

public void replaceInstance(MDEClass m, String ID) throws MDEException Korvaa instanssin uudella.

public boolean deleteInstance(String URI) throws MDEException Poistaa parametrina annetun URI:n mukaisen instanssin.

private boolean containsLiteral(Vector v) Testaa sisältääkö annettu range-vektori Literaalimääritystä.

private boolean containsValue(Vector v, String search) Testaa sisältääkö annettu range-vektori parametrina annettua arvoa.

public boolean propertyExists(String URI) throws Exception Testaa sisältyykö parametrina annettua URI:a vastaava ominaisuus instansseihin (Jena-oliomalliin).

Tietorakenteet

Luokassa käytetään Javan tietorakenneluokkia HashMap ja Vector. Lisäksi käytetään jena-oliomallin Model-tietorakennetta.

5.5 MDEQueryHandler

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

Muuttujat

private ModelRDB model Jenan oliomallin mukainen Model-luokan ilmentymä, jota luokka käsittelee.

private MDEOntologyHandler ontologyHandler Viittaus ontologiaselaimeen. Luokka tarvitsee myös ontologiasta tietoja.

Konstruktorit

Luokassa on yksi konstruktori, jossa luodaan yhteys instanssikantaan. Parametrit: RDF-instanssitiedoston nimi, käyttäjänimi, salasana, tietokannassa olevan mallin nimi sekä viittaus ontologiaselaimeen (MDEOntologyHandler).

public MDEQueryHandler (String connString, String userName, String passWord, String modelName, MDEOntologyHandler mdeO) throws MDEException

Metodit

Suunnitteludokumentissa kuvatut metodit:

public RDFQueryHandler (String instanceFileName)

public Vector doInstanceQuery(MDEClass m)

Toteutetut metodit:

public Vector doInstanceQuery(MDEClass m) throws MDEException Kyselyn suorittava metodi. Parametrina annetaan MDEClass-olio, joka sisältää haettavat kentät. Paluuarvona saadaan MDEClass-olioista koostuvan Vector-taulukko. Tämä taulukko sisältää löydetyt instanssit. Jos ei löydetty mitään, palautetaan null.

public MDEClass getInstance(String URI) throws MDEException Palauttaa halutun instanssin MDEClass-oliona.

private boolean containsLiteral(Vector v) Testaa sisältääkö annettu range-vektori Literaalimääritystä.

private boolean containsValue(Vector v, String search) Testaa sisältääkö annettu range-vektori parametrina annettua arvoa.

public boolean propertyExists(String URI) throws Exception Testaa sisältyykö parametrina annettua URI:a vastaava ominaisuus instansseihin (Jena-oliomalliin).

Tietorakenteet

Luokassa käytetään Javan tietorakenneluokkia HashMap ja Vector. Lisäksi käytetään jena-oliomallin Model-tietorakennetta.

5.6 Siirrettävyys

Järjestelmän RDF-käsittelijä on siirrettävissä ontogator-mediatorille siten, että laaditaan rajapintaluokkien RDFInstanceHandler, RDFOntologyHandler sekä RDFQueryHandler toteuttavat ontogator-mediatoria käyttävät luokat. Jälkikäteen ajateltuna järjestelmä olisi ollut miellyttävä toteuttaa siten, että ontogator-mediator-rajapinta olisi ollut käytettävissä, jolloin olisi voitu välttää monia toteutuksen kuoppia.

6 Tiedostonkäsittelijä (J.K.)

Tiedostonkäsittelijä avaa ja sulkee käyttäjän käsittelemät xml-tiedostot. Tiedostonkäsittelijä tarjoaa käyttäjälle mahdollisuuden kopioida tiedostoja järjestelmän ulkopuolelta järjestelmän omaan xml-tiedostoja sisältävään hakemistoon. Avatun xml-tiedoston pohjalta tiedostonkäsittelijä muodostaa XmlCardIdentifier-olioita, joiden avulla pidetään kirjaa käsitellyistä ja käsittelemättömistä xml-korteista ja niiden ominaisuuksista. Kyseiset olio tallennetaan järjestelmän hakemistoon sarjallistettuina olioina. Tiedostonkäsittelijän avulla järjestelmä ylläpitää xml-korteista luotuja rdf-ilmentymiä sisältäviä rdf-tiedostoja, kunnes käyttäjä siirtää ne järjestelmän ulkopuoliseen hakemistoon. Tiedostonkäsittelijä pitää myös huolen tiedostojen poistamisesta järjestelmän omasta hakemistosta. Jokaista xml-tiedostoa vastaa yksi XmlCardIdentifier- tiedosto ja yksi rdf-tiedosto. Kun xml-tiedosto poistetaan, poistetaan samalla sitä vastaava XmlCardIdentifier-tiedosto ja rdf-tiedosto. Järjestelmän käytössä on lisäksi konfiguraatiotiedosto sekä liitettyjen ominaisuuksien tiedosto. Nämä tiedostot ovat tarkoitettu ainoastaan järjestelmän luettaviksi, joten niihin tehtävät muutokset on tehtävä järjestelmän ulkopuolisilla editoreilla.

6.1 MDEFileHandler

Luokka huolehtii tiedostonkäsittely-palvelujen tarjoamisesta järjestelmän muille osille. Palvelut välitetään muuhun järjestelmään ohjaimen avulla. Luokka käyttää Javan valmiiden luokkien lisäksi luokkia XmlCardIdentifier, MDEJoint, MDEJointFile ja MDEXmlFile toimintojensa toteuttamiseen.

Muuttujat

Luokkan on lisätty vakiomuuttujat. Muut luokan muuttujat ovat samat, jotka suunnitteludokumentissa määriteltiin.

Seuraavat muuttujat on lisätty toteutusvaiheessa:

private static final String SYSTEM Muuttuja määrittelee hakupolun järjestelmätiedostoille.

private static final String XMLFILES Muuttuja määrittelee hakupolun xml-tiedostoille.

private static final String CONFIG Muuttuja määrittelee konfiguraatiotiedoston nimen.

private static final String LAST Muuttuja määrittelee viimeksi käsitellyn tiedoston nimen.

Alkuperäisen suunnitelman mukaan toteutetut muuttujat:

private String lastFile Muuttujaan sijoitetaan viimeksi käsitellyn tiedoston nimi.

private File configFile Muuttujaan sijoitetaan konfiguraatiotiedosto.

protected MDEXmlFile activeFile Muuttuja sisältää käsittelyssä olevan xml-tiedoston sekä sitä vastaavan rdf-tiedoston ja XmlCardIdentifier-olion.

private String [ ] **xmlFilesInUse** Muuttujaan sijoitetaan järjestelmän xml-tiedostoja sisältävän hakemiston sisältö.

private MDEJointFile joints Muuttuja sisältää liitettyjen ominaisuuksien tiedoston.

Konstruktorit

Luokassa on kaksi konstruktoria, joista toinen on parametriton. Alunperin suunnitteludokumentissa määriteltiin ainoastaan yksi konstruktori, joten toteutuksessa on jouduttu lisäämään toinen konstruktori. Toinen konstruktori lisättiin helpottamaan järjestelmän hakemistossa olevien xml-tiedostojen avaamista käsiteltäviksi.

Toteutuksen yhteydessä lisätty konstruktori:

public MDEFileHandler(String fileName) Luo MDEFileHandler-olion avaten parametrina annetun xml-tiedoston ja sitä vastaavan rdf-tiedoston sekä XmlCardIdentifier-olion. Avaa myös konfiguraatiotiedoston ja liitettyjen ominaisuuksien tiedoston.

Alunperin suunniteltu konstruktori:

public MDEFileHandler() Avaa konfiguraatiotiedoston ja liitettyjen ominaisuuksien tiedoston.

Metodit

Suunnitteludokumentissa määritellyistä toiminnoista jouduttiin joitakin toimintoja jättämään pois. Toisaalta joitakin alunperin puuttuvia toimintoja jouduttiin lisäämään toteutuksen yhteydessä.

Suunnitteludokumentissa määritellyistä metodeista seuraavia ei toteutettu:

public boolean openConfigurationFile() Hyödytön, sillä konfiguraatiotiedosto avataan aina, kun luodaan ilmentymä luokasta.

public boolean closeConfigurationFile() Konfiguraatiotiedostoon ei kirjoiteta mitään, joten sitä ei tarvitse erikseen sulkea.

Seuraavat toiminnot on lisätty luokkaan toteutusvaiheessa:

public boolean closeFiles() Sulkee avoinna olevat tiedostot.

public MDEXmlFile getActiveFile() Palauttaa käsittelyssä olevan xml-tiedoston.

private String readConfigurationFile(String item) Palauttaa parametrina annettua arvoa vastaavan toiminnon konfiguraatiotiedostosta.

public String getJoint(MDEJoint key) Palauttaa parametrina annettua avainta vastaavan xml-elementin.

public String getLastFile() Palauttaa viimeksi käsitellyn tiedoston nimen.

public boolean closeLastFile() Sulkee viimeksi käsitellyn tiedoston ja tallentaa sen järjestelmän hakemistoon systemfiles/MDEdata.

Seuraavat toiminnot on toteutettu, mutta niitä ei ole tarkoitettu käytettäviksi järjestelmässä:

public boolean addJoint(String key, String domainUri, String propertyUri) Luo uuden liitetyn ominaisuuden. Toimintoa ei käytetä, koska liitettyjen ominaisuuksien tiedosto toimitetaan järjestelmän ulkopuolelta.

public boolean deleteJoint(String key) Poistaa parametrina annettua avainta vastaavan liitoksen. Toimintoa ei käytetä, koska liitettyjen ominaisuuksien tiedosto toimitetaan järjestelmän ulkopuolelta.

Seuraavat toiminnot on määritelty suunnitteludokumentissa:

public MDEXmlFile openNewXmlFile(String fileName) Kopioi järjestelmän ulkopuolelta parametrina annetun xml-tiedoston järjestelmän hakemistoon. Avaa tiedoston ja luo tiedostoa vastaavat rakenteet.

public MDEXmlFile openCurrentXmlFile(String fileName) Avaa järjestelmän hakemistosta xml-tiedoston.

public MDEXmlFile openLastFile() Avaa viimeksi käsitellyn xml-tiedoston.

public boolean saveRdfFile(String path) Tallentaa rdf-ilmentymätiedoston parametrin osoittamaan hakemistoon.

public boolean eraseCurrentXmlFile(String fileName) Poistaa parametrina annetun tiedoston.

public String openXmlCard(String id) Avaa käsittelyssä olevasta xml-tiedostosta parametrina annetulla id-tunnuksella varustetun xml-kortin.

public String [ ] **getFilesInProcess()** Palauttaa järjestelmän hakemistossa olevien tiedostojen nimet.

public String [ ] **getFileNames(String subDirectory)** Palauttaa parametrina annetun hakemiston sisältämät tiedostot ja hakemistot.

public MDEJoint getJoint(String key) Palauttaa parametrina annettua avainta vastaavan liitetyn ominaisuuden.

public String getConfigurationInfo(String configurationFileItem) Palauttaa parametrina annetun konfiguraatiotiedon.

Tietorakenteet

Luokan tietorakenteet ovat samat kuin suunnitteludokumentissa määritellyt.

6.2 MDEXmlCardIdentifier

Luokka kuvaa xml-korttien ominaisuuksia järjestelmässä. Korteista säilytetään seuraavat tiedot: kortin id-tunnus, otsikko, kortista muodostetun rdf-ilmentymän uri ja kortin tilatieto(valmis, kesken).

Muuttujat

Luokan muuttujat ovat samat kuin suunnitteludokumentissa määritellyt.

private String cardHeader Kortin otsikko.

private String cardID Kortin tunniste.

private String cardUri Kortista tehdyn rdf-ilmentymän osoite.

private boolean cardState Kortin tila.

Konstruktorit

Konstruktorit vastaavat suunnitteludokumentin määritelmiä.

public XmlCardIdentifier() Luo tyhjän olion.

public XmlCardIdentifier(String id) Luo parametrina annetulla tunnisteella varustetun olion.

Metodit

Luokan metodit ovat pääosin samat kuin suunnitteludokumentissa.

Luokkaan on lisätty seuraavat toiminnot:

public void setCardHeader(String header) Asettaa parametrina annetun arvon kortin otsikoksi.

public String stateAsString() Palauttaa kortin tilatiedon String-muodossa(valmis/kesken)

public String toString() Palauttaa merkkijono esityksen olion attribuuteista.

Suunnitteludokumentissa määritellyt toiminnot:

public void setCardID(String id) Asettaa parametrina annetun arvon kortin tunnisteeksi.

public void setCardUri(String uri) Asettaa parametrina annetun arvon kortista tehdyn rdf-ilmentymän osoitteeksi.

public void setCardState(boolean state) Asettaa parametrina annetun arvon kortin tilaksi.

public void getCardID() Palauttaa kortin tunnistetiedon.

public void getCardUri() Palauttaa korttia vastaavan rdf-tiedoston osoitteen.

public void getCardState() Palauttaa kortin tilatiedon.

public void getCardHeader() Palauttaa kortin otsikon.

Tietorakenteet

Luokassa on käytetty ainoastaan Javan perustyyppejä.

6.3 MDEJoint

Luokka liittää kaksi rdf-ominaisuutta toisiinsa. Liitettävät ominaisuudet ovat: tietyn luokan uri-osoite ja kyseistä luokkaa vastaavan ominaisuuden uri-osoite.

Muuttujat

Luokan muuttujat vastaavat suunnitteludokumentin määritelmiä.

private String jointDomain Luokan osoite.

private String jointProperty Luokan ominaisuuden osoite.

Konstruktorit

Luokan konstruktorit ovat samat kuin suunnitteludokumentissa määritellyt konstruktorit.

public MDEJoint() Luo tyhjän olion.

public MDEJoint(String domainUri, String propertyUri) Luo parametreina annetuilla arvoilla varustetun olion.

Metodit

Luokkaan on lisätty toteutusvaiheessa kaksi metodia, jotka puuttuvat suunnitteludokumentista.

Luokkaan lisätyt toiminnot:

public String toString() Palauttaa merkkijono esityksen luokan attribuuteista.

public boolean equals(Object o) Vertaa kahta ilmentymää.

Suunnitteludokumentissa määritellyt toiminnot:

public void setDomainUri(String domainUri) Asettaa luokan osoitteen.

public void setPropertyUri(String propertyUri) Asettaa luokan ominaisuuden osoitteen.

public String getDomainUri() Palauttaa luokan osoitteen.

public String getPropertyUri() Palauttaa luokan ominaisuuden osoitteen.

Tietorakenteet

Luokassa on käytetty ainoastaan Javan perustyyppejä.

6.4 MDEXmlFile

Luokassa määritellään tiedostonkäsittelijän xml-tiedon muokkauksessa käyttämät tiedostot. MDEXmlFile-olio sisältää käsiteltävän xml-tiedoston, xml-tiedostoa vastaavan rdf-ilmentymätiedoston ja xml-tiedostoa vastaava XmlCardIdentifier-taulukon. Luokkaa käytetään MDEFileHandler-luokassa tiedon välittämiseen järjestelmän muille osille. Luokka on muuttunut melko radikaalisti suunnitteludo-

kumentissa tehdyistä määritelmistä. Kaikki määritellyt toiminnot on toteutettu, mutta toimintoja on lisätty runsaasti toteutusvaiheessa.

Muuttujat

Muuttujat vastaavat suunnitteludokumentissa määriteltyjä muuttujia, mutta vakiomuuttujat on lisätty toteutuksen yhteydessä.

Lisätyt muuttujat:

private static final String CARDDIR Hakemisto, jossa on xml-korttien tunnistetiedot.

private static final String RDFDIR Hakemisto, jossa on rdf-tiedostot.

private static final String XMLDIR Hakemisto, jossa on xml-tiedostot.

Suunnitteludokumentissa määritellyt muuttujat:

protected File xmlFile Muuttuja sisältää xml-tiedoston.

protected File rdfFile Muuttuja sisältää xml-tiedostoa vastaavan rdf-tiedoston.

protected XmlCardIdentifier [ ] **xmlCards** Muuttuja sisältää xml-tiedostoa vastaavan XmlCardIdentifier-taulukon.

Konstruktorit

Luokkaan on lisätty toteutusvaiheessa yksi konstruktori.

Luokkaan lisätty konstruktori:

public MDEXmlFile(String xmlFileName, String idField, String headerField, String idAttribute)

Konstruktori avaa xmlFileName-parametrin osoittaman xml-tiedoston ja luo sitä vastaavan XmlCardIdentifier-taulukon muiden parametrien avulla. Luo myös uuden rdf-tiedoston, joka vastaa avattua xml-tiedostoa.

Suunnitteludokumentissa määritelty konstruktori:

public MDEXmlFile(String xmlFileName) Avaa parametrina annetun xml-tiedoston ja sitä vastaavat muut tiedostot.

Metodit

Luokkaan on lisätty useita suunnitteludokumentista puuttuvia toimintoja. Suunnitteludokumentissa määrittellyt toiminnot on kuitenkin kaikki toteutettu.

Luokkaan toteutusvaiheessa lisätyt toiminnot:

private void setCardInfo(String cardStart, String headerStart, String idAttribute) Asettaa xml-tiedoston korttien tiedot muuttujaan.

public String getXmlFileName() Palauttaa xml-tiedoston nimen.

public String getRdfFileName() Palauttaa rdf-tiedoston nimen.

public File getRdfFile() Palauttaa rdf-tiedoston.

public XmlCardIdentifier [ ] **getCardIdentifiers()** Palauttaa korttien tiedot sisältävän taulukon.

public String [ ] **getFileNames()** Palauttaa järjestelmän hakemistossa olevien tiedostojen nimet.

Suunnitteludokumentissa määritellyt toiminnot:

public MDEXmlFile openFile (String xmlFileName, String idField, String headerField, String idAttr) Avaa ensimmäisenä parametrina annetulla nimellä

varustetun tiedoston. Luo muiden parametrien avulla xml-tiedostoa vastaavan XmlCardIdentifier-taulukon. Avaa xml-tiedostoa vastaavan rdf-tiedoston. Metodi on muuttunut suunnitellusta siten, että parametreja on neljä yhden sijasta. Alunperin ainoaksi parametriksi suunniteltiin xmlFileName.

public boolean closeFile() Sulkee käsittelyssä olevat tiedostot.

public boolean eraseFile(String fileName) Poistaa parametrina annetulla nimellä varustetun xml-tiedoston ja sitä vastaavan rdf-tiedoston sekä XmlCardIdentifier-tiedoston.

public String getXmlCard (String cardId, String cardStart, String idAttribute) Palauttaa ensimmäisenä parametrina annettua tunnistetta vastaavan xml-kortin. Käyttää muita parametreja apunaan etsiessään korttia. Metodi on muuttunut alunperin suunnitellusta siten, että cardStart ja idAttribute parametrit on lisätty vasta toteutusvaiheessa.

Tietorakenteet

Luokassa käytetään suunnitteludokumentissa määriteltyjä tietorakenteita.

6.5 XMLFileHandler (A.I)

Luokan tehtävänä on palauttaa annetusta XML-tiedostosta kortit identifioivien attribuuttien arvot sekä palauttaa XML-kortti annetun id:n perusteella. Luokkaa ei ole määritelty suunnitteludokumentissa. Luokan tarpeellisuus havaittiin vasta integrointitestauksen yhteydessä. Suunnitteludokumentissa luokan toiminnot oli määritelty MDEXmlFilen toiminnoiksi. Luokka käyttää Apachen Xerxces ja Xalan -rajapintoja. Kortit ja niiden id:t etsitään Document Object Model [DOM] -puusta XML Path Languagella [XPath].

Muuttujat

String cardId Attribuutinnimi, jossa kortin id:t

String cardElement Elementinnimi, joka erottelee kortit.

String uri XML-tiedoston osoite.

String cardHeader Attribuutinnimi, jonka arvo kortin otsikko.

Konstruktori

XMLFileHandler (String uri, String cardElement, String cardIdAttribute, String cardHeaderAttribute), String cardElement Parametrina XML-tiedoston osoite, kortin alku / loppu -tagi, id-attribuutinnimi ja otsikko-attribuutinnimi.

Metodit

XmlCardIdentifiers [ ] getCardIdentifiers()] Palauttaa taulukon XmlCardIdentifiers-oliota.

String getXMLCard(String id) Palauttaa XML-kortin. Parametrina kortin id.

6.6 MDEJointFile

Luokan avulla luodaan liitettyjen ominaisuuksien tiedostosta MDEJoint-olioita, jotka sijoitetaan hajautustauluihin järjestelmän käytettäväksi. Luokka ei kaikilta osiltaa vastaa suunnitteludokumentin määrittelyjä, vaan luokkaan on lisätty toimintoja. Alunperin luokan oli tarkoitus luoda vain yksi hajautustaulu, mutta toteutusvaiheessa lisättiin toinen hajautustaulu. Nyt luokka hajauttaa liitettyjen ominaisuuksien tiedostoon molempiin suuntiin. Toiseen hajautustauluun xml-elementit hajautetaan MDEJoint- olioiden perusteella ja toiseen MDEJoint-oliota hajautetaan xml-elementtien perusteella.

Muuttujat

Muuttujiin on lisätty toteutusvaiheessa vakiomuuttujat sekä toinen hajautustaulu. Muilta osin muuttujat vastaavat suunnittelu- dokumentin määritelmiä.

Luokkaan lisätyt muuttujat:

private static final String JSTART Liitettyjen ominaisuuksien tiedoston liitoksen alun osoittava kenttä.

private static final String JEND Liitettyjen ominaisuuksien tiedoston liitoksen lopun osoittava kenttä.

private static final String FILEEND Liitettyjen ominaisuuksien tiedoston loppumisen osoittava kenttä.

private static final String ELEMENT Xml-elementin nimi liitettyjen ominaisuuksien tiedostossa.

private static final String DOMAIN Luokan nimi liitettyjen ominaisuuksien tiedostossa.

private static final String PROPERTY Luokan ominaisuuden nimi liitettyjen ominaisuuksien tiedostossa.

private HashMap jointObjects Liitettyjen ominaisuuksien tiedostosta muodostettu hajautustaulu, jossa elementit on järjestetty MDEJoint-olioiden perusteella. Muuttuja on lisätty luokkaan toteutusvaiheessa, koska käyttöliittymässä tarvitaan kyseistä tietoa.

Suunnitteludokumentissa määritellyt muuttujat:

private File jointFile Liitettyjen ominaisuuksien tiedosto.

private HashMap jointElements Liitettyjen ominaisuuksien tiedostosta muodostettu hajautustaulu, jossa MDEJoint-oliot on järjestetty elementtien perusteella.

Konstruktorit

Luokan ainoa konstruktori muutettiin toteutusvaiheessa parametrittomasta yhden parametrin saavaksi konstruktoriksi.

public MDEJointFile(String fileName) Konstruktori avaa liitettyjen ominaisuuksien tiedoston, jonka perusteella luo molemmat hajautustaulut.

Metodit

Suunnitteludokumentissa määritellyt toiminnot on toteutusvaiheessa muokattu täysin uusiksi. Luokka ei juurikaan vastaa alunperin suunniteltua, vaan lähes kaikki toiminnot on tehty toteutuksen yhteydessä. Luokan metodeista suuri osa on jätetty kokonaan pois ja uusia on tullut paljon lisää. Luokassa on myös totetettu toimintoja, joita ei järjestelmässä ole tarkoitus käyttää.

Toiminnot, joita ei toteutettu:

private Static String readLine() Hyödytön, koska rivien lukeminen tehdään metodien sisällä.

private Static boolean writeToFile() Metodi on toteutettu nimellä writeAttributeToFile.

Luokkaan lisätyt toiminnot:

public String getAttribute(MDEJoint object) Palauttaa parametrina annettua MDEJoint-oliota vastaavan xml-elementin.

public HashMap getJointObjects() Palauttaa hajautustaulun, jossa xml-elementit on hajautettu MDEJoint-olioiden perusteella.

public HashMap getJointElements() Palauttaa hajautustaulun, jossa MDEJoint-oliot on hajautettu xml-elementtien perusteella.

Suunnitteludokumentissa määritellyt toiminnot:

public MDEJoint getAttribute(String element) Palauttaa parametrina annettua xml-elementtiä vastaavan MDEJoint-olion.

public boolean setAttributes(String element, String domainUri, String propertyUri) Luo uuden toisen ja kolmannen parametrin perusteella MDEJoint-olion ja hajauttaa sen kumpaankin hajautustauluun.

private void fileToHashMap() Lukee liitettyjen ominaisuuksien tiedoston ja luo sen perusteella hajautustaulut.

Luokkaan laaditut toiminnot, joita ei käytetä järjestelmässä:

Näitä toimintoja ei käytetä järjestelmässä, koska liitettyjen ominaisuuksien tiedostoa on tarkoitus muokata editoimalla tiedostoa järjestelmän ulkopuolisella tekstieditorilla.

public boolean deleteAttribute(String element) Poistaa parametrina annettua elementtiä vastaavat tiedot hajautustaulusta.

public boolean deleteAttribute(MDEJoint objects) Poistaa parametrina annettua MDEJoint-oliota vastaavat tiedot hajautustaulusta.

private void writeAttributeToFile(String element, String domain, String property) Kirjoittaa liitettyjen ominaisuuksien tiedostoon rivin, joka sisältää parametreina saadut tiedot.

Tietorakenteet

Luokassa käytetyt tietorakenteet vastaavat suunnitteludokumentissa tehtyjä määritelmiä.

7 Käyttöliittymämoduuli (M.J.)

Käyttöliittymä on metadataeditorin käyttäjäystävällisin osa.

7.1 FilesAndCardsJSP

FilesAndCardsJSP-sivu kuvaa käyttäjälle tämän käytössä olevat tiedostot ja valitun tiedoston kortit, sekä näiden tilat. Apuna käytetään iterateFiles- ja iterateCards-tageja.

Käyttäjän valitseman tiedoston tai kortin id välitetään parametrina get-metodilla. Luo tiedosto -nappi avaa uuteen ikkunaan sivun newFile.jsp. Poista tiedosto -nappi avaa uuteen ikkunaan sivun deleteFile.jsp. Tallenna valmiit -nappi lähettää oikeanpuoleisen kehyksen lomakkeen sivulle InstanceCard.jsp, joka latauduttuaan lähettää vasemmanpuoleisen kehyksen lomakkeen FilesAndCards.jsp-sivulle. Näin varmistutaan siitä, että kaikki käyttäjän tekemät muutokset tallentuvat.

7.2 InstanceCardJSP

InstanceCardJSP-sivu kuvaa käyttäjälle valitun XML-kortin tiedot, sekä antaa käyttäjälle mahdollisuuden luoda RDF-skeeman pohjalta instanssi. Apuna käytetään iterateRDFElements- ja iterateDynlist-tageja.

JSP-sivun alussa alustetaan cardstate-muuttuja, johon luodaan uusi CardState-olio. Sitten iteroidaan RDF-elementtejä. JSP-sivulla tarkastetaan RDFElementBeanin muuttujista minkälainen tapaus on kyseessä, ja piirretään sen mukaan oikeanlainen osa käyttöliittymään. Mikäli beanissä on dynaaminen hierarkkinen lista, iteroidaan sen osat sisemmässä silmukassa(iterateDynlist), ja piirretään lista sen RDFElementBeanin yhteyteen, johon se kuuluu. RDFElementBeaneja iteroidessa tarkastetaan sen muuttujia, ja muuttujien yhdistelmien mukaan piirretään oikeanlaiset tiedot käyttöliittymään. Tapaukset ovat

1. Literaalirange-property ilman arvoa
2. Literaalirange-property jolla on arvo
3. Instancerange-property ilman arvoa
4. Instancerange-property jolla on arvo
5. Classrange-property ilman arvoa
6. Classrange-property jolla on arvo
7. Ensimmäinen property, eli se jonka arvoksi luotava instanssi pistetään, ilman arvoa
8. Ensimmäinen property jolla on arvo

(Tapauksien ymmärtäminen vaatii RDF:än ymmärtämistä.)

Näiden tapauksien mukaan valitaan JSP-sivussa oikea haara, ja piirretään oikeat HTML-elementit.

JSP-sivu myös käsittelee parametrit, jotka se saa HTML-taulukosta. Aina kun sivu ladataan uudestaan, muutetaan ohjelman tietojen tila vastaamaan käyttäjän tekemiä muutoksia.

Lomakkeen kentät

Lomakkeen kentät on nimetty kyseisen beanin id:n perusteella. Alla *n* kuvastaa beanin id:tä.

search_class Teksti, jonka perusteella etsitään luokkaa.

search_case Mitä tapausta etsitään.

instanceURI Käyttäjän valitseman instanssin URI.

instance Käyttäjän valitseman Beanin, joka sisältää instanceURI:n, id.

search_text Teksti, jonka perusteella etsitään instanssia.

search Beanin id, johon etsitään

reloadFileControl Ladataanko vasemmanpuoleinen kehys uudelleen.

meedio_*n* Käyttäjän syöttämä tieto *n*:nteen beaniin.

dectree_meedio_*n* Mistä kohtaa *n*:nnen beanin hierarkkista listaa pienennetään.

inctree_meedio_*n* Mistä kohtaa *n*:nnen beanin hierarkkista listaa kasvatetaan.

dynlist_meedio_*n* Onko *n*:nnen beanin hierarkkinen lista auki.

7.3 newFileJSP

NewFile.jsp-sivulta käyttäjä voi kopioida uuden XML-tiedoston järjestelmän käyttöön. Tiedoston nimi välitetään newFile.jsp-sivulle post-metodilla. Sulje-nappi sulkee ikkunan ja uudelleen lataa Pääikkunan vasemman puoleisen kehyksen.

7.4 deleteFileJSP

DeleteFile.jsp-sivulta käyttäjä voi poistaa järjestelmän käytössä olevia XML-tiedostoja. Poistettavan tiedoston nimi välitetään get-metodilla deleteFile.jsp-sivulle. Sulje-nappi sulkee ikkunan ja uudelleen lataa Pääikkunan vasemman puoleisen kehyksen.

7.5 CardState

Luokka CardState sisältää tietorakenteita käyttöliittymän senhetkisestä tilasta. JSP-sivut piirtävät käyttöliittymän tietorakenteissa olevien olioiden mukaan tagien avustuksella. CardStatessa on myös metodeita joilla pidetään rdfBeans-listaa ajan tasalla, sekä apumetodeita joita tarvitaan käyttöliittymän toiminnassa. Näiden lisäksi CardStatessa on myös MDEController-olio, jonka kautta saadaan käyttöön muiden ohjelman osien tarjoamat palvelut.

Muuttujat

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(MDEPropertyt).

LinkedList rdfBeans joissa on MDEPropertyjen ympärille lisätty myös käyttöliittymätietoja. Näistä piirretään varsinainen käyttöliittymä.

LinkedList files jossa on StringBeanit joista tiedostolista piirretään

LinkedList cardIds johon tulee auki olevien korttien XmlCardIdentifier-oliot

boolean newProperties kertoo pitääkö rdfBeans-listaan lisätä uusia beaneja

boolean removeProperties kertoo pitääkö rdfBeans-listasta poistaa beaneja

boolean newInstances kertoo pitääkö rdfBeans-listaan lisätä uusia instansseja

HashMap ontologyURIs täällä on kaikki käytetävän ontologian luokat (nimi,URI)-pareina

Konstruktorit

public CardState() luo uuden CardState-olion

Metodit

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

public void setInstanceOf(MDEClass mdeclass)

public MDEClass getInstanceOf()

public void setFiles(LinkedList files)

public LinkedList getFiles()

public boolean changeFileOpen(String filename) Tällä metodilla vaihdetaan auki olevaa tiedostoa. Kun yksi tiedosto avataan, muut suljetaan.

public void setNewProperties(boolean maybe)

public void setRemoveProperties(boolean maybe)

public void setNewInstances(boolean maybe)

public MDEController getController()

public boolean setRDFElements() Tämä metodi päivittää rdfBeans-listaa aina kun kortin tiedot näyttävä JSP-sivu ladataan uudestaan. Alussa tehdään ns.varjoproperty, jonka arvoksi voidaan laittaa joku luokka, jonka instanssi halutaan luoda. Sen jälkeen metodi vain tarkistaa, pitääkö propertyjä tai instansseja poistaa tai lisätä, ja kutsuu alimetodeita jotka tekevät nämä asiat.

public boolean setNewRDFelements() Tämä metodi käy läpi rdfBeans-listaa, kunnes löytää paikan jonka alle halutaan lisätä uusia propertyjä, ja lisää ne sitten oikeaan kohtaan listaa. MDEPropertyt myös muutetaan RDFElementBeaneiksi ennen lisäystä.

public boolean removeRDFElements() Tämä metodi poistaa rdfBeans-listasta propertyt, jos luokka jonka propertyjä ne olivat muutetaan toiseksi.

private void removeRemovables() Apumetodi, laittaa kaikkien RDFElementtejen hasRemovableProperties:in arvon falseksi.

public StringBean getPictureFile() Tämä metodi palauttaa StringBeanin, jonka string-muuttujassa on käsittelyssä olevan kortin kuvan hakemistopolku ja tiedostonimi.

public boolean loadOldRDFelements() Tällä metodilla ladataan rdfBeans-listaan avattu instanssi, eli tehdään uuden ladatun InstanceOf:in propertyistä RDFElementBeaneja, jotta ne voitaisiin piirtää.

private boolean loadOldRDFelements(MDEClass mdeclass, int depth) Parametrittoman samannimisen metodin rekursiivinen alimetodi.

private HashMap initOntologyURIs() Tämä metodi hajauttaa ontologian URIt hajautustauluun nimensä perusteella, josta voidaan sitten myöhemmin katsoa sisältyykö joku käyttäjän syöttämä luokka tähän ontologiaan.

private HashMap initOntologyURIs(HashMap hash, Vector roots) Tämä metodi on saman nimisen metodin rekursiivinen aliohjelma.

public String getURIfromName(String className) Tämä metodi tarkistaa onko ontologiassa parametrin nimistä luokkaa, ja jos on, palauttaa sen URIn.

7.6 Käyttöliittymän käyttämät beanit

JSP-sivut piirtävät käyttöliittymän bean-olioissa olevien tietojen perusteella. Tässä kuvataan lyhyesti beanien tietosisältö, ja get- ja set- metodeista poikkeavat ylimääräiset metodit.

Jokaisessa beanissa on siis julkisina metodeina jokaista muuttujaa vastaavat metodit getMuuttujanimi ja setMuuttujanimi. Näitä ei kuvata erikseen.

7.6.1 RDFElementBean

Tämä bean on suurin ja monimutkaisin käyttöliittymän käyttämistä beaneista.

Muuttujat

private boolean first kertoo onko olio ensimmäinen rdfBean- listan olioista, eli ainoa, jota ei ole tehty oikeasta MDEPropertystä. Sitä käsitellään JSP-sivulla eri tavalla kuin muita.

private boolean isMultirange kertoo onko tällä oliolla monta rangea

private boolean hasLiteralRange kertoo onko tällä oliolla literaali-arvoista rangea

private boolean hasClassRange kertoo onko tällä oliolla luokka-arvoista rangea

private boolean hasInstanceRange kertoo onko tällä oliolla instanssi-arvoista rangea

private String chosenRange kertoo tällä hetkellä valitun rangen

private int depth ominaisuuden syvyys ominaisuushierarkiassa

private boolean hasNewProperties kertoo onko tälle oliolle tullut uusia aliominaisuuksia, jotka pitäisi lisätä rdfBeans-listaan

private boolean hasRemovableProperties kertoo onko beanilla poistettavia propertyjä, jotka pitäisi poistaa CardStaten rdfBeans-listasta

private boolean hasNewInstances kertoo onko tällä beanilla uusi instanssi valittuna, jonka tiedot pitäisi lisätä CardStaten rdfBeans-listaan

private boolean hasValue kertoo onko tällä oliolla arvo

private Vector dynlist tässä Vectorissa on DynlistElement-olioita, joista koostuu dynaaminen hierarkkinen lista. Null jos listaa ei ole.

private boolean dynlistVisible kertoo onko olion dynaaminen hierarkkinen lista näkyvissä käyttöliittymässä

private MDEProperty property täällä on varsinainen rdf-tieto, jonka ympärille RDFElementBean on kääritty, jotta saataisiin käyttöliittymätieto pysymään tallessa

private MDEController controller Viite CardStatessa olevaan MDEController-olioon, jonka kautta myös bean saa käyttöönsä koko ohjelman toiminnallisuuden

private int id beanin id-numero

private Vector instances Vector jossa on tämän beanin luokkaa vastaavat tietokannasta haetut instanssit

private static int counter staattinen apumuuttuja jota käytetään beanien id-numeron yksiselitteisyyden takaamiseksi

Konstruktorit

public RDFElementBean() luo uuden beanin

public RDFElementBean(MDEProperty property) luo uuden beanin MDEPropertyn ympärille

Metodit

Get ja set-metodit löytyvät kaikille edellä mainituille muuttujille, paitsi muutama epäoleellinen metodi on jätetty pois, esimerkiksi staattista counter-muuttujaa ei voi metodein muuttaa. Näiden metodien lisäksi täältä löytyy dynaamisen hierarkkisen listan muuttamismetodit sekä muutaman apumetodin ohjelman tiettyjä toiminnallisuuksia tukemaan.

public void initDynlist() Tämä metodi alustaa dynaamisen hierarkkisen listan beanin muiden tietojen perusteella.

public boolean addToDynlist(String superURI) Tämä metodi lisää dynaamiseen hierarkkiseen listaan parametrina annetun URIn suorat alaluokat.

public boolean addInstanceToDynlist(MDEClass mclass) Tämä metodi lisää MDEClass-olion kuvaaman instanssin dynaamiseen hierarkkiseen listaan.

public boolean removeFromDynlist(String superURI) Tämä metodi poistaa dynaamisesta hierakkisesta listasta kaikki parametrina annetun URIn alaluokat ja instanssit.

public boolean checkClass(MDEClass mdeclass) Tämä metodi tarkistaa onko käyttäjän ehdottama luokka hyväksyttävä arvo kyseisen propertyn arvoksi. Eli käytännössä haetaan luokan kaikki yläluokat ontologiassa ja verrataan niitä sitten propertyn rangeihin.

private Vector findRootClasses(MDEClass mdeclass) Tämä metodi on checkClass-metodin apumetodi, joka hakee kaikki luokan yläluokat ja panee niiden URIt Vectoriin.

private Vector findRootClasses(Vector roots, Vector superClasses) Tämä metodi on samannimisen metodin rekursiivinen alimetodi.

public static String cutURI(String URI) Tämä metodi on staattinen apumetodi, joka leikkaa Stringistä pois #-merkkiä edeltävän osan. Käytetään siis jotta saataisiin URIsta luokan nimi.

7.6.2 DynlistElement

Tämä bean kuvaa yhtä riviä dynaamisessa hierarkkisessa listassa, mutta sisältää myös hiukan ylimääräistä tietoa tallessa sisällään, josta se otetaan esiin tarvittaessa.

Muuttujat

private int depth tämän elementin syvyys listassa

private String name tämän elementin nimi

private String URI tämän elementin URI

private boolean isInstance kertoo onko tämä elementti instanssi vai luokka

private boolean hasSubClasses kertoo onko tällä elementillä aluluokkia ontologiassa

private boolean visibleSubclasses kertoo onko tällä elementillä näkyviä aluluokkia hierarkkisessa dynaamisessa listassa

private boolean hasChosenInstance kertoo onko elementin instanssi valittu piirrettäväksi käyttöliittymään

private MDEClass instance mikäli tämä rivi dynaamisessa hierarkkisessa listassa kuvaa instanssia, on instanssi-olio tässä

Konstruktorit

public DynlistElement(String name,String URI,int depth) konstruktori jolla voi antaa osan muuttujien arvoista parametrinä.

public DynlistElement (String name,String URI,int depth, boolean isInstance, boolean hasSubClasses) konstruktori jolla voi antaa osan muuttujien arvoista parametrinä.

public DynlistElement(MDEClass instance,int depth) instanssikonstruktori, jonka tiedot otetaan parametrina annetusta MDEClass-oliosta.

7.6.3 StringBean

StringBean on bean jonka avulla voidaan käsitellä Stringejä JSP-sivuilla beanmaisesti. Periaatteessa tätä käytetään kuvaamaan tiedostoja FilesAndCards- JSP:lle, mutta sen lisäksi sitä käytetään myös yleisenä Stringin wrapperina.

Muuttujat

private String string täällä on tallessa merkkijono

private boolean open kertoo onko tiedosto auki

Konstruktorit

public StringBean() luo uuden beanin

public StringBean(String string) luo uuden beanin jonka string-arvo on string

7.6.4 XmlCardIdentifier

XmlCardIdentifier-olioita käytetään käyttöliittymässä beaneina, mutta ne sisältävät myös muuta tietoa. XmlCardIdentifier-olioista löytyy enemmän tietoa luvus-

sa Tiedostonkäsittelijä.

7.7 Tagikirjasto Meedio

Tässä kuvataan Meedio-tagikirjaston tagit. Koska tagien toimintalogiikka on näissä tageissa hyvin samantyyppinen, niitä ei kuvata samalla tarkkuudella kuin muita java-luokkia, vaan niistä kuvataan niiden tehtävä ja miten tagi sen toteuttaa.

7.7.1 FileIteratorTag

FileIteratorTag toteuttaa iterateFiles-tagin, joka käy läpi MDEControllerilta saadut tiedostonimet ja tiedostojen tilat, ja antaa ne yksi kerrallaan JSP-sivun käyttöön.

7.7.2 CardIteratorTag

CardIteratorTag toteuttaa iterateCards-tagin, joka käy läpi CardStatesta saadut XmlCardIdentifier-oliot, ja antaa ne yksi kerrallaan JSP-sivun käyttöön.

7.7.3 RDFElementIteratorTag

RDFElementIteratorTag toteuttaa iterateRDFElements-tagin, joka käy läpi RDF-elementeistä tehdyt oliot. Kun tag ajetaan, päivitetään aluksi CardState-olion rdfBeans-muuttujan tila, ja sen jälkeen annetaan oliot yksi kerrallaan JSP-sivun käyttöön.

7.7.4 OntologyIteratorTag

OntologyIteratorTag toteuttaa iterateDynlist-tagin, joka käy läpi hierarkkisen dynaamisen listan, ja antaa ne yksi kerrallaan JSP-sivun käyttöön.

7.7.5 SetCardStateTag

SetCardStateTag toteuttaa setCardState-tagin, jossa laitetaan CardState- muuttuja kuntoon. Jos muuttujaa ei ole, se luodaan, jos se on, ei tehdä mitään.

8 Jatkokehittelymahdollisuudet

Koska metadataeditorin perustoiminnallisuus osoittautui niin työlääksi, jäi monta hyvää ideaa ajanpuutteen vuoksi toteuttamatta. Tässä luvussa kuvataan sekä jokunen itse huomaamamme puute, ja ajatuksia niiden korjaamisesta tulevaisuudessa, että muita jatkokehitysideoita.

8.1 'Multirange' - eli monen rangen propertyt

Koska RDFS-määrittelyissä sallitaan monen rangen laittaminen yhdelle propertylle, pyrimme toteuttamaan ohjelman niin, että tällaisetkin tapaukset toimisivat. Ajan loppuessa emme ehtineet lisätä JSP-sivuille sellaista haaraa, missä käsiteltäisiin monen rangen tapaukset, joten tällä hetkellä se ei toimi. Tästä huolimatta MDEPropertyn range-muuttuja on vektori, jossa voi siis olla monia arvoja, ja MDEPropertyistä RDFElementBeaneja luotaessa laitetaan RDFElementBeaniin tieto siitä, onko kyseinen property multirange. Myös RDFElementBeanin dynaamisen hierarkkisen listan alustus ottaa huomioon sen, että multirange-tapauksissa tulee listan juuriksi oikeat luokat.

Jos JSP-sivuille lisättäisiin multirangea koskevat tapaukset, voisi mahdollistaa esimerkiksi sen, että jos käyttäjä kirjoittaa tekstikenttään jotain mutta sitä ei löy-

dy ontologiasta Hae-nappia painettaessa, arvo voisi silloin mennä literaaliarvoksi MDEPropertyyn. Sellainen tapaus jossa olisi sekä class-range että instanssirange voisi tuottaa hieman päänvaivaa, koska dynaamiseen hierarkkiseen listaan joudutaan piirtämään kaikki luokat, ja näin ei välttämättä tiedettäisi, haluaako käyttäjä lisätä instanssirangeen kuuluvan luokan instanssin, vai ainoastaan sen luokan. Nämä ongelmat lienevät kuitenkin ratkaistavissa JSP-sivulla tavalla tai toisella.

8.2 Instanssien nimeäminen järkevämmin kuin sen URIn perusteella

Tällä hetkellä dynaamiseen hierarkkiseen listaan lisättävät instanssit piirretään listaan URInsa loppuosalla, koska ei ole mahdollista tietää minkä alipropertyn arvo olisi tähän järkevää laittaa täysin geneerisessä järjestelmässä, jossa propertyt vaihtelevat. Olisi kuitenkin mahdollista kehittää järjestelmä joka vastaisi ohjelmassamme jo toteutettua liitoskäytäntöä jossain määrin, eli tehtäisiin erillinen tiedosto jossa sidottaisiin tietty luokan URI tiettyyn sen luokan propertyyn, niin että sen luokan instanssien kohdalla piirrettäisiinkin URIn sijasta tämän määritellyn propertyn arvo. Jos tällaista tietoa ei löytyisi, piirrettäisiin sitten se URI.

8.3 Kortin otsikon liittäminen sen id:n ohelle

Tällä hetkellä korteista piirretään FilesAndCards.jsp:hen sekä InstanceCard3.jsp:n taulukon ekalle riville ainoastaan id-numero. Se ei ole kovin kuvaavaa, ja voisi olla viisasta liittää myös kortin otsikkotieto näihin paikkoihin, mikäli sellainen on konfiguraatiotiedostossa xml-skeemalle määritelty.Näin saataisiin lisää selkeyttä, kun epämääräisen numeron ohella olisi myös jotain kuvaavampaa tietoa(vrt. edellinen kohta).

8.4 RDF:n tallentaminen suoraan tietokantaan

Tulevaisuudessa metadataeditorin ollessa käytössä paikallisesti eri puolilla Suomea tulee pakosti vastaan datan monistuminen. Koska kaikki tieto ei ole tietokannassa, eikä sieltä näin ollen voida tarkastaa tiedon mahdollista olemassaoloa ennen sen luontia, syntyy duplikaatteja. Tämä taas aiheuttaa päänvaivaa niille jotka yrittävät pitää tietokannan tiedot yksiselitteisinä. Siksi olisi viisasta pitää ohjelmaa vain yhdessä paikassa, josta tiedot myös suoraan talletettaisiin tietokantaan, ja paikallisilla koneilla olisi vain käyttöliittymä. Tämä malli vaatisi tietoliikennemoduulin lisäämistä, niin että käyttöliittymä kutsuisi verkon yli MEEDIO-palvelimen ohjainta, joka taas lähettäisi tiedot takaisin. Kaikki tiedot talletettaisiin suoraan tietokantaan, ja näin tiedot olisivat koko ajan ajan tasalla.

8.5 Hakuominaisuuksien monipuolistaminen

Tällä hetkellä Hae-nappi hakee HashMapista URIa käyttäjän teksti avaimenaan, eli sen hyöty aloittelevalle käyttäjälle joka ei tunne ontologiaa on hyvin vähäinen. Jos teksti ei ole tarkalleen identtinen, ei tulosta saada. Älykkyyttä olisi mukava saada lisää, joten HashMapin sijasta ontologian URIt voisi tallentaa esimerkiksi AVL-puuhun nimensä perusteella, jolloin jo luokan nimen alun kirjoittaminen voisi riittää, mikäli se olisi yksiselitteinen, eli AVL-puussa voitaisiin haun lopussa tarkastaa, täsmääkö käyttäjän merkkijono solmun avaimen alkuun, ja jos näin on, se olisi ainut mahdollinen luokka ja näin ollen luultavasti käyttäjän tarkoittama. Jos käyttäjän merkkijono täsmäisi useampaan solmuun, voitaisiin tietysti jotenkin antaa lista vaihtoehdoista käyttäjälle.

AVL-puu-tekniikan lisäksi voitaisiin varmasti myös käyttää jonkinlaista SubString-hakua, jolloin useampien vaihtoehtojen näyttäminen pitäisi myös jotenkin ratkaista käyttöliittymässä. Eli haulla 'Koira' löytyisi myös 'Koiranruoka' ja 'Saksanpaimenkoira'. Tällainen vaatisi kuitenkin luultavasti merkkijonon vertaamista kaikkiin ontologian luokkiin, ja olisi näin ollen huomattavasti hitaampi kuin HashMap tai AVL-puu.

8.6 Tietokantahakujen monipuolistaminen

Samalla lailla kuin Hae-napin toiminnallisuutta voitaisiin parantaa, olisi myös tietokantahaussa älykkyyden lisäämisen varaa. Jos käyttäjälle sattuu kirjoitusvirhe, ei oikeaa instanssia löydetä, ja kirjoitusasun pitää muutenkin olla identtinen isoja ja pieniä kirjaimia myöten. Ongelma on tietysti pitkälti RDQL:ssä, mutta mikäli monimutkaisemmat kyselyt saataisiin toteutettua, tulisi järjestelmästä paljon käytettävämpi.

8.7 Luokkien ja xml-arvojen sitominen toisiinsa

Edelleen voitaisiin järjestelmän älykkyyttä lisätä sitomalla tiettyjä xml-arvoissa esiintyviä sanoja tiettyihin luokkiin, niin että esimerkiksi jos xml:stä löytyisi vaikka tagi <tyyppi>, niin sen arvo voitaisiin tarkastaa, ja jos siinä esiintyisi merkkijono 'kirves', voitaisiin automaattisesti valita kentän arvoksi Kirves-luokka, tai ainkin ehdottaa sitä käyttäjälle. Tämä helpottaisi luokittelijan työtä huomattavasti tapauksissa jossa luokitellaan hyvin samantyyppisiä esineitä suurissa määrin.

8.8 Uusien skriptien lisääminen käyttöliittymään

Tällä hetkellä käyttäjä joutuu klikkailemaan paljon nappeja käyttäessään käyttöliittymää. Voisi olla hyvä, jos esimerkiksi Hae-nappi korvattaisiin skriptille joka tekee saman asian kun hiiri siirretään kyseisen taulukon solun päältä pois. Myös instanssien hakeminen tietokannasta on nyt hieman epäkäytännöllistä, ja museohenkilö ei välttämättä edes muista käyttää koko ominaisuutta, joka entisestään lisää duplikaattien määrää tietokannassa. Paras vaihtoehto olisi varmaan synkronoitu tietokantakysely joka käyttäjän valintojen mukaan hakee aina tiedot tietokannasta, ja kun käyttäjä lisää tietoa, tarkentuisi samalla dynaamisen hierarkkisen listan instanssivaihtoehtojen määrä. Tällainen saattaa tosin käytänössä olla hyvin vaikea toteuttaa.

Näiden skriptien lisäksi voisi myös olla viisasta toteuttaa tallennusskripti, joka oli tarkoitus alunperin tehdä, mutta lopulta päädyimme kiireessä Tallenna-nappiin,

joka on sinänsä helppokäyttöinen, mutta lisää kuitenkin käyttäjän klikkailua ja mahdollistaa tietojen häviämisen kun käyttäjä ei muista tallentaa. Jos tallennus tapahtuisi automaattisesti aina kun korttia vaihdetaan, ei tätä riskiä olisi.

8.9 CSS-tiedoston tekeminen

Tällä hetkellä ohjelmamme ulkoasun viilailuun jäi hyvin vähän aikaa, ja CSS-tekniikalla voisi saada järjestelmän ulkoasua paranneltua. Ulkoasua voi tietysti parannella myös ilman CSS-tekniikkaa, mutta CSS olisi yksi hyvä vaihtoehto.

8.10 Luokkien pathin näyttäminen

Tästä ominaisuudesta oli puhetta joissain käyttöliittymäpalavereissa, mutta jäi lopulta toteuttamatta. Kaikki luokan pathit saisi kuitenkin varmasti esiin käyttämällä hyväksi RDFELementBean-luokan checkClass-metodin alimetodia findRootClasses, joka hakee kaikki luokan yläluokat. Tämän rekursion sisällä voisi varmaan jotenkin myös tehdä stringejä poluista luokasta juureen, jotka sitten voisi laittaa esimerkiksi linkin title-attribuutiksi, jolloin ainakin IE ja Mozilla osaisivat näyttää pathit luokkanimen päällä tooltippinä. Asian voi varmasti hoitaa toisinkin, mutta tässä yksi vaihtoehto.

8.11 XML-tietojen näyttäminen

Tällä hetkellä ohjelma hakee xml-elementeistä ensimmäisen elementin arvon, jos samannimisiä elementtejä on monta. Jatkossa voisi laittaa sulkuihin käyttöliittymässä vaikka kaikki arvot peräkkäin, mikäli ei haluta minkään tiedon jäävän pois luokittelijalta.

9 Viittaukset lähteisiin (P.P.)

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/

- Apchen 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/