Ylläpitodokumentti

Orava

Helsinki 5.5.2005

Ohjelmistotuotantoprojekti

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Kurssi

581260 Ohjelmistotuotantoprojekti (6 ov)

Projektiryhmä

Juhani Bergström
Peter von Etter
Teppo Känsälä
Olli Lyytinen
Jessika Penttinen
Mikko Waris

Asiakas

Eero Hyvönen

Johtoryhmä

Juha Taina
Hannu Räisänen

Kotisivu

`http://www.cs.helsinki.fi/group/orava/`

Versiohistoria

Versio	Päiväys	Tehdyt muutokset
0.1	9.4.2005	Ensimmäinen versio
0.2	21.4.2005	Tarkastettava versio
1.0	5.5.2005	Palautus

Sisältö

1 Johdanto

Tämä on Ohjelmistotuotantoprojekti-kurssin Orava-ryhmän tuottaman Orava-portaalin ylläpitodokumentti. Portaali on semanttisen webin [SemWeb] tekniikoilla toteutettu haku- ja suosittelukone YLE Opetusohjelmien Klaffi-videoarkistoon [Klaffi], jonka videot ovat nykysin verkossa käytettävissä Opinportti-palvelun [Opinportti] kautta. Portaali on rakennettu soveltaen MuseoSuomi-hankkeessa [MuseoSuomi] kehitettyä OntoViews-järjestelmää [OntoViews], jolla videot saadaan linkitettyä semanttisesti toisiinsa.

Projektin aikana on tuotettu myös käyttöohje [Käyttöohje], jossa on ohjeistettu muunmuassa järjestelmän asennus, käynnistys, XML-lähdetiedostojen lisääminen ja portaalin käyttö.

1.1 Terminologia

Ontologia Määrittelee käsitteitä ja niiden välisiä suhteita. Koko järjestelmässä ontologioita on useita: esimerkiksi paikkaontologia joka sisältää eri paikkojen määritelmät, ja aikaontologia joka määrittelee eri aikakäsitteitä.

(Ontologinen) käsite Ontologiassa yksiselitteisesti määritelty käsite. Käsitteitä on kahdenlaisia: luokkia ja yksilöitä. Luokkakäsite voi sisältää yksilökäsitteitä ja luokkia. Yksilökäsite taas viittaa johonkin yksilöolioon. Esimerkiksi yksilökäsite ”vuosi 2005” kuuluu luokkakäsitteeseen ”2000-luku”.

Kohde Yksilökäsite, joka viittaa tiettyyn tietokannan videoon tai muuhun kohteeseen.

Suositeltu kohde Käyttäjän parhaillaan tarkastelemaan kohteeseen semanttisesti liittyvä kohde, jota portaali suosittelee käyttäjälle nykyisen kohteen yhteydessä.

Kategoria Kohteiden järjestämiseen käytetty käsite.

Näkymä Hierarkisiksi taksonomioiksi järjestetyt kategoriat.

Valinta Käyttäjän tekemä rajaus näytettäviin kohteisiin. Käyttäjä voi lisätä rajauksia valitsemalla näkymistä kategorioita tai lisäämällä hakusanoja.

Moninäkymä-käyttöliittymä Käyttäjän valintojen mukaan näytettävä näkymien joukko.

LOM Lyhenne engl. sanoista *Learning Object Metadata*. Tietomalli opetukseen liittyvien objektien metadatalle. [LOM]

XML Lyhenne engl. sanoista *Extensible markup language*. Yleinen rakenteellinen kuvauskieli. [XML]

XSL Lyhenne engl. sanoista *Extensible stylesheet language*. XML-muotoinen kieli, jolla muunnetaan XML-dokumentteja eri muotoisiksi XML-dokumenteiksi. [XSL]

RDF Lyhenne engl. sanoista *Resource Description Framework*. Semanttisen webin tietomalli metadatalle. [RDF]

RDFS Lyhenne engl. sanoista *Resource Description Framework Schema*. Ontologioiden määrittelyyn käytetty RDF-tietomalli. [RDFS]

URI Lyhenne engl. sanoista *Universal resource identifier*. Yksilöllinen tunniste jollekin resurssille, mihin täytyy voida viitata esimerkiksi RDF-dokumentissa. [URI]

OntoViews MuseoSuomi-portaalin ohjelmisto. [OntoViews] Koostuu osajärjestelmistä Ontogator ja Ontodella.

(Prolog-)päättelysääntö Prolog-kielellä ilmaistu sääntö, jota käytetään kategorioiden tekemisessä ja suositeltujen kohteiden hakemisessa.

Metadata Yleisesti, tietoon liittyvää tietoa. Tässä dokumentissa tarkoittaa Klaffi-arkiston videoihin liittyvää tietoa videoista, esimerkiksi videon teknisiä tietoja, sisältöön liittyviä avainsanoja jne. Joissakin kohdissa tarkoitetaan myös MuseoSuomen metadataa, joka ei liity videoihin vaan MuseoSuomen materiaaliin. Sama metadata esiintyy järjestelmässä erimuodoissa eri komponentteja varten. Kuvasta 1 käy ilmi, missä muodossa mikäkin komponentti käyttää metadataa. Kuvaan on merkitty “Klaffi-materiaali” alkuperäisen metadatan kohdalle. Metadata voidaan esittää esimerkiksi XML- tai RDF-tiedostoilla.

1.2 Dokumentin rakenne

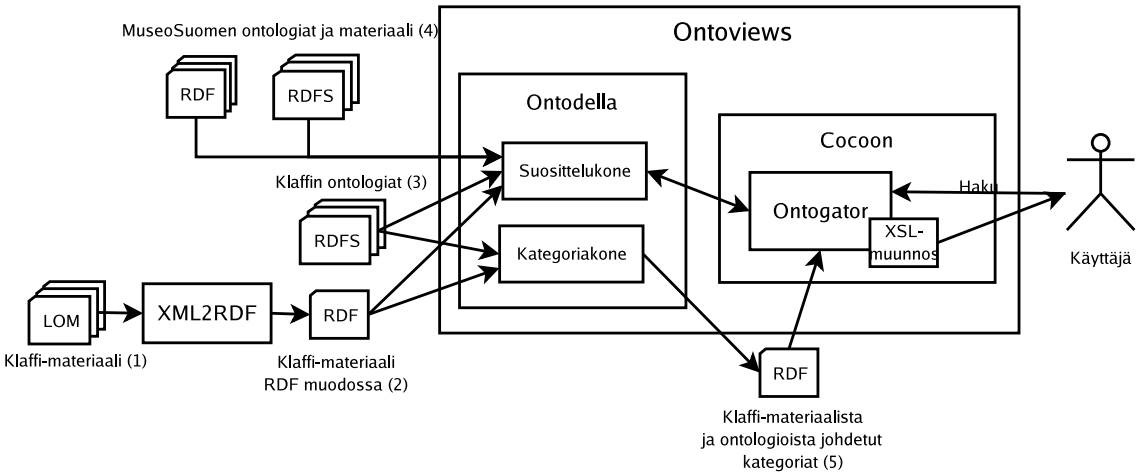
Luvussa 2 annetaan yleiskuva järjestelmän eri osajärjestelmistä ja niiden toiminnasta sekä järjestelmän hakemistorakenteesta. Luvussa 3 kunkin komponentin tarjoamat palvelut ja rajapinnat kuvataan tarkasti. Luvussa 4 kuvataan Ontodellan ymmärtämät predikaatit, joilla portaalin linkityksen määräävät päättelysäännöt laaditaan. Luvussa 5 on esitelty XML2RDF-työkalu. Tähän lukuun liittyy olennaisesti liite 1, jossa on XML2RDF-järjestelmän luokkakuvaukset. Luvussa 6 esitellään Cocoon-sovelluskehyksen toiminta.

2 Järjestelmän yleiskuva

Tuotettava ohjelmisto käyttää hyväkseen lukuisia valmiita ohjelmia ja tekniikoita. Tässä luvussa esitellään kukin osajärjestelmä. Kuvassa 1 on graafisesti esitetty järjestelmän osajärjestelmät ja tiedon kulku eri komponenttien väleillä.

2.1 Ontoviews

Projektissa on käytetty Ontoviews [OntoViews] kehitysalustaa. Ontoviewsin tarkoitus on helpottaa semanttisen portaalin toteuttamista. Se tuottaa annetusta aineiston metadatasta



Kuva 1: Järjestelmän osajärjestelmät ja niiden väliset yhteydet. Metadatat ja ontologiat on numeroitu, samaa numerointia käytetään kuvassa 2.

moninäkymäkäyttöliittymän, jolla käyttäjän on helppo hakea materiaalia tietokannassa. Ontoviews koostuu pääasiassa kahdesta eri komponentista: Ontodella ja Ontogator.

Ontodella

Ontodella on SWI-Prolog:lla [SPr] toteutettu sovellus, jolla on kaksi eri tarkoitusta Ontoviewsissa. Jotta moninäkymän esittäminen käyttäjälle olisi tehokasta, esikäsitellään metadata Ontodellalla, jonka tuloksena saadaan näkymien kategoriapuut. Tämä vaihe saattaa kestää melko pitkään (jopa tunteja), joten se on parasta suorittaa vain silloin, kun alkuperäinen metadata muuttuu.

Ontodella toimii myös http-palvelimena, jota Ontogator käyttää käyttöliittymän tuottamiseen: Ontogator tiedustelee Ontodellalta käyttäjän valitsemaan käsitteeseen liittyviä linkkejä. Ontodellan ei siis välttämättä tarvitse sijaita edes samalla koneella kuin käyttäjälle näkyvä Ontogator.

Ontodellan päättelysääntöjä muokkaamalla portaaliin voidaan tehdä uusia näkymiä tai uusia linkkisuosituksia, sekä portaalin sisäisiä että muihin vastaaviin järjestelmiin osoittavia ulkoisia suosituksia.

Ontogator

Ontogator on Apache Cocoon [Coc] -sovelluskehykselle kehitetty järjestelmä. Se toimii http-palvelimena, joka tarjoaa järjestelmän käyttöliittymän käyttäjälle. Ontogator tuottaa moninäkymäkäyttöliittymän Ontodellan tuottamista kategoriapuista ja käyttäjän valinnoista. Se pitää kirjaa käyttäjän tekemistä valinnoista liittämällä nämä käyttäjälle näytettävän sivuston koodiin, josta ne puolestaan välittyvät jälleen Ontogatorille rajoitettaessa näkymää lisää uudella valinnalla. Tarpeen mukaan se tiedustelee Ontodellalta käsitteisiin

liittyviä linkkejä, joita se näyttää käyttäjälle.

Cocoon tarjoaa hyvät mahdollisuudet tehdä muutoksia järjestelmään liukuhihna (pipeline) -arkitehtuurinsa ansiosta. Yksinkertaisia muutoksia voi tehdä pelkästään XSL-tyylitiedostojen avulla ja monimutkaisempia tekemällä sopivia transformaatio-luokkia Javalla. Cocoonin toimintaperiaatetta on selvitetty luvussa 6.

2.2 XML2RDF

XML2RDF osajärjestelmä on projektissa tuotettu sovellus, joka muuntaa XML muotoisia dokumetteja RDF-muotoon. Klaffi-materiaalia varten se on tehty muuntamaan LOM-tietomallin mukaisessa muodossa olevia XML-dokumentteina RDF-muotoon, jotta niitä voidaan käyttää OntoViews-järjestelmässä. XML2RDF-työkalua voidaan mukauttaa myös muille XML-tietomalleille. XML2RDF:n muokkaus on ohjeistettu luvulla 5.

2.3 Hakemistorakenne

Kuvassa 1 esitettiin järjestelmän eri osajärjestelmät. Kuvassa 2 on esitetty järjestelmän normaali hakemistorakenne ja tärkeimmät tiedostot. Kummassakin kuvassa on käytetty samaa numerointia ontologia- ja metadatatiedostoille, jotta tiedonkulun ja tiedostojen sijaintien välinen yhteys selviäisi.

On huomattava, että osa tiedostoista esiintyy useassa eri hakemistossa. Erityisesti ontologiat ja metadata eri muodoissa on usein kopioituna aina sen osajärjestelmän hakemistorakenteeseen, joka sitä tarvitsee. Lisäksi Ontoviewsin build-hakemisto rakennetaan kääntämällä Ontoviews Apache Ant-työkalulla.

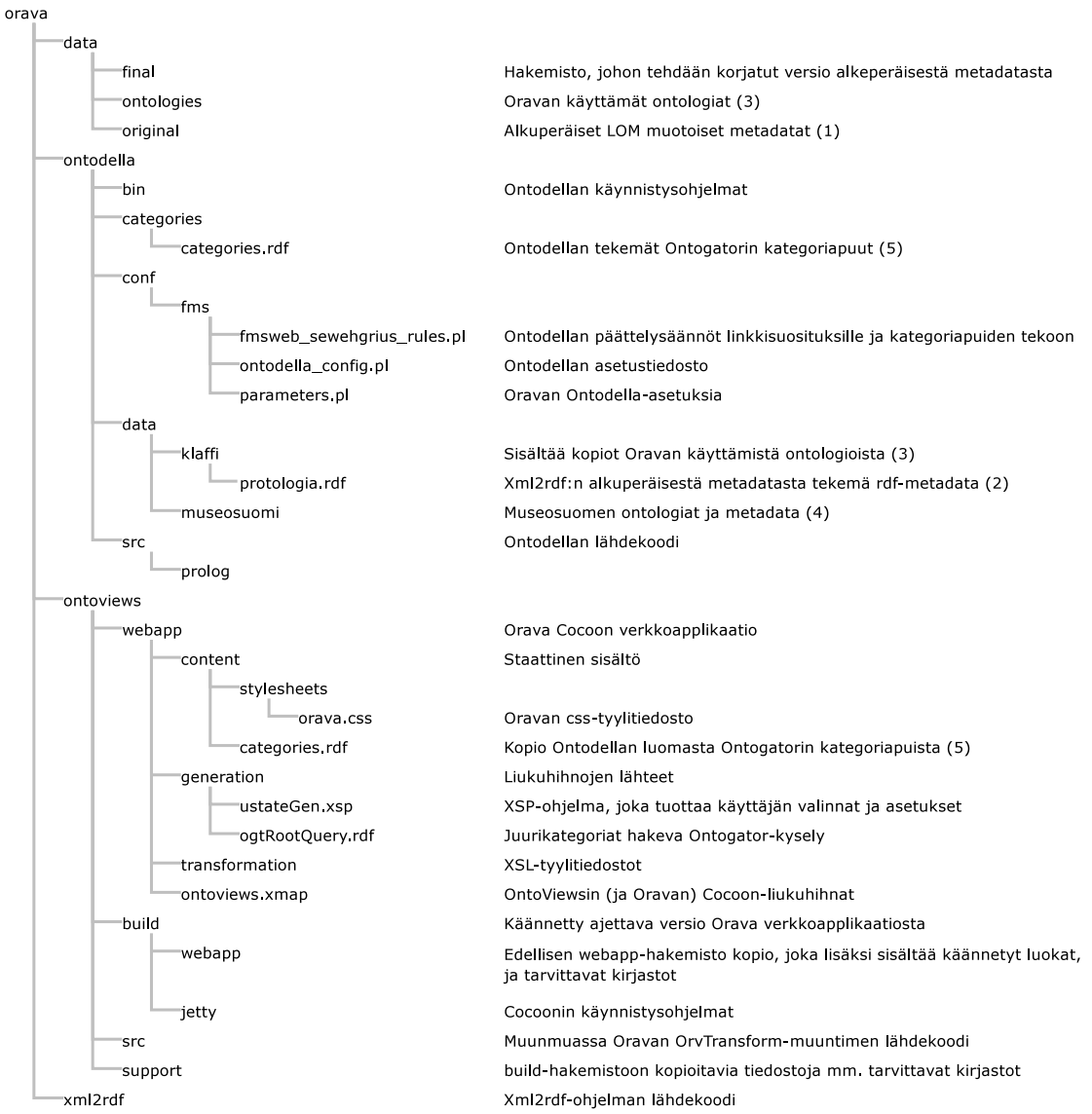
Järjestelmä voidaan konfiguroida myös erilaiseen hakemistorakenteeseen. Erityisesti osajärjestelmien sijainti voi vaihdella ja esimerkiksi Ontodellan voi sijoittaa jopa kokonaan eri koneelle kuin Ontogator.

3 Rajapinnat

Tässä luvussa on kuvattu järjestelmän eri osajärjestelmien komponenttien tarjoamat palvelut ja niiden rajapinnat. OntoViewsin sisäisistä komponenteista rajapinnat on kuvattu vain sillä tarkkuudella, että järjestelmän toiminnasta saa selkeän kuvan.

3.1 Ontodella

Ontodella on SWI-Prologilla toimiva ohjelmisto, jolla tehdään kyselyitä järjestelmän pohjana toimivaan metadataan ja sen ontologioihin. Ontodella tarjoaa kaksi erillistä palvelua, joiden kuvaukset ovat alla. Kummatkin käyttävät Prolog-päättelysääntöjä, jotka on kuvattu tarkemmin luvussa 4.



Kuva 2: Järjestelmän hakemistorakenne. Metadatat ja ontologiat on numeroitu, samaa numerointia käytetään kuvassa 1.

Kategorioiden teko

Kuvaus

Ontodella luo Ontogatorin käyttämät kategoriapuut. Kyseessä on siis OntoViewsin sisäinen rajapinta.

Syötteet

Konfiguraatiotiedostossa *ontodella/conf/fms/fmsweb_sewehgrius_rules.pl* annetaan Prolog-päättelysäännöt, joiden avulla kategoriat tehdään.
Konfiguraatiotiedostossa *ontodella/conf/fms/ontodella_config.pl* annetaan RDF- ja RDFS-tiedostojen sijainnit, mistä luetaan käytettävä metadata ja ontologiat.

Tulos

RDF-muotoinen tiedosto *ontodella/categories/categories.rdf* , joka on Ontogatorin käyttämän ontologian mukainen ja joka sisältää Ontogatorin tarvitseman tiedon metadatasta ja ontologioista, jotta se voi rakentaa moninäkymäkäyttöliittymän.

Suositeltavat linkit

Kuvaus

Ontodella palvelee Ontogatoria antamalla linkkisuosituksia. Ontodella-palvelin on http-palvelin ja sitä käytetään suorittamalla http-kyselyjä.

Syötteet

Konfiguraatiotiedostossa *ontodella/conf/fms/fmsweb_sewehgrius_rules.pl* annetaan Prolog-päättelysäännöt, joiden perusteella suosituksia tehdään. Katso myös luku 4.
Konfiguraatiotiedostossa *ontodella/conf/fms/ontodella_config.pl* annetaan RDF- ja RDFS-tiedostojen sijainnit, mistä luetaan käytettävä metadata ja ontologiat.
Http-kysely sisältää tiedon käsitteestä, johon halutaan suosituksia.

Tulos

Palvelin vastaa http-kyselyyn RDF-dokumentilla, joka sisältää suositellut linkit Ontogatorin käyttämän ontologian mukaisesti.

4 Ontodellan päättelysäännöt

Portaalin näkymät ja esitettävät linkkisuositukset määritellään Ontodellan päättelysäännöillä. Päättelysäännöt ovat *fmsweb_sewehgrius_rules.pl* -tiedostossa sijaitsevia Prolog-kielisiä predikaatteja.

Seuraavassa luvussa 4.1 kuvataan näkymän tekemiseen tarvittavat säännöt. Luvussa 4.2 käsitellään linkkisuositusten tekemiseen tarvittavat säännöt.

Sääntöjen parametrien kuvaamisessa on käytetty yleistä Prolog-predikaattien notaatiota, jossa kaksoispisteellä : merkitään termiä, joka on predikaatin tai vastaavan nimi. Plusmerkillä + merkitään termiä, jonka tulisi olla sidottu predikaattia käytettäessä, ja miinusmerkillä – termiä, jonka predikaatti itse sitoo. Lisäksi merkitään kysymysmerkillä ? termiä, jota voidaan käyttää molempiin tarkoituksiin.

4.1 Näkymän teko

sewhgrius_category(–JuuriURI, :JuurenNimi, :Alakategoriat, :Lehdet)

Kuvaus

Tällä säännöllä määrätään portaalin käyttöliittymässä esitettävät näkymät.

Parametrit

–JuurenURI Näkymän identifioiva URI.

:JuurenNimi Näkymän nimen kertova predikaatti, kuvaus jäljempänä.

:Alakategoriat Näkymän alakategoriat määräävä predikaatti, kuvaus jäljempänä.

:Lehdet Kategoriapuun lehdet määräävä predikaatti, kuvaus jäljempänä.

Näkymän nimen kertova predikaatti *p(+JuurenURI, –NimiLista)*

Kuvaus

Tällä säännöllä määritellään näkymän nimi.

Parametrit

+JuurenURI Näkymän identifioiva URI.

–NimiLista Assosiatiivinen lista nimistä, jossa avaimina ovat kielten lyhenteet ja arvoina näkymän nimet. Esimerkki: *[fi:'Kiva näkymä', en:'Nice Facet']*.

Alakategoriat määräävä predikaatti *p([?]+YläKategoria, [?]–AlaKategoria)*

Kuvaus

Tällä säännöllä määritellään kategoriat ja alakategoriat yhteen näkymään.

Yksinkertainen tapa tuottaa ontologian määrittämää käsitepuuta vastaava kategoriapuu on määritellä tämä todeksi silloin, kun *YläKategoria* on ontologiassa *AlaKategoria*:n yläluokka.

Näkymän korkeimman tason kategorioille yläkategoriana on näkymän identifioiva URI.

Parametrit

+YläKategoria Yläkategorian URI.

–AlaKategoria Alakategorian URI.

Lehdet identifioiva predikaatti *p(?KohteenURI, ?KategorianURI)*

Kuvaus

Tällä säännöllä määritellään, miten kohteet sijoitetaan lehdiksi kategoriaan.

Haluttaessa määritellä ontologian mukainen kategoriapuu, määritellään tämä todeksi silloin, kun KohteenURI on ontologiassa tyypiltään KategorianURI.

Parametrit

?KohteenURI Kohteen URI.

?KategorianURI Kategorian URI.

4.2 Linkkisuositukset

sewehgrius_relation_rule(?RelaatioURI, ?RelaationNimi, :RelaatioSääntö)

Kuvaus

Tällä säännöllä määritellään linkkisuositustyypit. Varsinaiset suosittelusäännöt määritellään tämän säännön parametreilla.

Parametrit

?RelaatioURI Linkkisuositustyypin identifioimiseen käytetty URI.

?RelaationNimi Käyttäjälle näytettävän relaation nimet eri kielillä assosiatiivisena listana jossa avaimena on kielen lyhenne ja arvona näytettävä nimi. Esimerkki: *[fi:'Läheisesti liittyviä käsitteitä', en:'Closely Related Concepts']*.

:RelaatioSääntö Predikaatti, joka määrää käsitteet, joiden välillä linkki on.

Relaation säännön määräävä predikaatti *p(+SubjektiURI, –RelaationKuvaus, –KohdeURI)*

Kuvaus

Tällä säännöllä määritellään yhden linkkisuositustyypin kohteiden linkit. Tyypillisesti tämä on tosi, jos kohteet liittyvät toisiinsa jonkin tietyn (tai joidenkin tiettyjen) ontologian relaatioiden kautta.

Parametrit

+SubjektiURI Sen käsitteen URI, johon halutaan linkki.

–RelaationKuvaus Käyttäjälle näytettävä kuvaus yhteydestä. Muoto: [commonResources(*K*), label(fi:'*L*')], jossa *K* on subjektia ja kohdetta "yhdistävän" käsitteen URI, ja *L* yhteyden kuvaus suomeksi. *Eri kielisiä kuvauksia voinee liittää jatkamalla listaa label-faktoilla.*

–KohdeURI Sen käsitteen URI, johon linkki menee.

4.3 Muut päättelysäännöt

sewehgrius_labels(+URI, –Nimet)

Kuvaus
Predikaatilla määritellään käsitteiden käyttäjälle näytettävät nimet eri kielillä.

Parametrit

+URI Käsitteen URI, jolle haetaan nimiä.

–Nimet Assosiatiivinen lista nimistä eri kielillä, jossa avaimina ovat kielten lyhenteet ja arvoina näytettävät nimet.

sewehgrius_category_is_viewable(–URI)

Kuvaus
Predikaatilla määritellään, mitkä kategoriat näytetään käyttäjälle.

Parametrit

–URI Kategorian URI.

sewehgrius_annotation(+KäsiteURI, –Nimiavaruus, –Ominaisuus, –Arvo)

Kuvaus
Predikaatilla määritellään eri käsitteiden annotaatioita.

Parametrit

+KäsiteURI Käsitteen URI, johon haetaan annotaatioita

–Nimiavaruus Haettavan ominaisuuden nimiavaruus, yleensä 'http://localserver.org/deduced/annotations#'

–Ominaisuus Haettava ominaisuus, esim 'title' tai 'pictureurl'

–**Arvo** Haettavan ominaisuuden arvo

Ontodella_config.pl -tiedostossa on predikaatit, joilla annetaan tarvittavien RDF- ja RDFS-tiedostojen sijainnit, sekä muita järjestelmän alustamiseen tarvittavia tietoja. Samassa tiedostossa on myös *ontodella_preprocess*-predikaatti, joka käydään läpi järjestelmän käynnistyksen yhteydessä. Mikäli käytetään esiprosessointia vaativia sääntöjä, esiprosessoinnin tekevät predikaatit lisätään tähän.

Ontodella_request_handler.pl -tiedostossa on *get_categories*-predikaatissa määritelty, mitä käsitteitä ylipäänsä käsitellään kohteina. Ennen riviä "http_print_bookmark(SubjectURI)"olevalla rivillä suodatetaan käsiteltävät kohteet. Orava-projektissa on käyttää predikaattia, joka tarkistaa, että käsite on tyyppiä video.

parameters.pl -tiedostossa on Oravan päättelysääntöjen käyttämiä parametreja, joita muuttamalla voi vaikuttaa Oravan näkymiin ja suositussääntöihin.

5 XML2RDF

XML2RDF-työkalu muuntaa XML-dokumentteja RDF-muotoon. Orava-projektissa lähdetiedostot olivat LOM-tietomallin mukaisessa XML muodossa, mutta muunnosprosessi suunniteltiin siten, että sitä voidaan käyttää myös muun mallisille XML-tiedostoille. Tässä luvussa kerrotaan, XML2RDF:n toiminta periaate ja miten sitä voidaan käyttää muille XML-tiedostoille. Liitteessä 7.2 on javadoc-dokumentaatio koko XML2RDF-osajärjestelmästä. XML2RDF käyttää XPath-tekniikkaa, joka on Java:n vakiokirjastoissa versiosta 1.5 lähtien. Lisäksi se tarvitsee Jena [Jena] -kirjaston RDF-mallien käsittelyä varten.

5.1 Toimintaperiaate

XML2RDF koostuu eristäjäluokista (Java-luokka *InfoExtractor*), joista kukin hakee tietoa lähdetiedostoista. Kukin eristäjä hakee lähteestä jonkin tietyn tiedonpalasen, esimerkiksi videon nimen tai videon avainsanat, tai prosessoi haettua tietoa jollakin tavalla. Eristäjiä liitetään yhteen siten, että toisen tulos toimii toisen syötteenä. Näin muodostetut eristäjäketjut voivat tehdä monimutkaisiakin operaatioita vaikka jokainen eristäjäluokka itsessään on hyvin yksinkertainen. Eristäjät välittävät tietoa toisilleen kokoelma luokissa (*java.util.Collection*), jolloin on mahdollista erotella syötteestä useita erikseen käsiteltäviä osia. Eristäjät käyttävät Java version 1.5:n geneerisyys-ominaisuutta tyyppimääritysten tarkastamiseen käännösvaiheessa. Tämän takaa sen, että eristäjiä ei voi yhdistää sillä tavalla, että toisen tulos ei sovi syötteeksi toiselle.

XML2RDF käsittelee lähdetiedostot yksi kerrallaan. Vaihtoehtoisesti, jos koko lähdemateriaali on yhdessä tiedostossa, se voitaisiin myös käsitellä elementti kerrallaan, mikäli se koostuu kokoelmasta saman muotoisia elementtejä. Tiedostot (tai elementit) ladataan yksi kerrallaan DOM-muistitietorakenteeseen. Kukin näistä käsitellään *CollectedInfoExtractor* -luokalla, joka käyttää kokoelmaa eristäjiä tiedon hakemiseen. Ensiksi tuotetaan RDF-resurssi yhdellä eristäjäketjulla. Tämän jälkeen tähän resurssiin lisätään ominaisuuksia

muiden eristäjäketjujen avulla. *CollectedInfoExtractor* -luokassa käytetään kuvausta (*java.util.Map*), jolla kuvataan RDF-ominaisuuksia eristäjäketjuille. Tämä kuvaus määrittelee ominaisuudet, joita resurssiin liitetään ja eristäjät, jotka tuottavat arvot niille. Kukin eristäjä tuottaa tuloksena kokoelman joko RDF-resursseja tai literaali-arvoja. Kunkin eristäjäketjun syötteenä taas toimii XML-tiedostoa vastaava DOM-elementti.

Kun muunnetaan XML-tiedostoja, ketjun ensimmäinen eristäjä saa syötteeksi itse XML:n. Orava-projektissa ensimmäinen eristäjä hakee tietoa XPath-tekniikalla (Java-luokka *XPathInfoExtractor*). Tämä tuottaa XML-dokumentista poimitun merkkijonon, jota jatkokäsitellään muilla eristäjillä esimerkiksi pilkkomalla se osiin pilkkujen kohdalta tai suorittamalla haku ja korvaus -operaatioita säännöllisillä lausekkeilla. Lopuksi eristetty arvo muutetaan RDF-resurssiksi tai literaaliksi. Resurssin tapauksessa tässä vaiheessa eristetty arvo otetaan talteen ja sitä käytetään pian luotavan resurssin nimenä. Nimestä tehdään URI, jonka avulla voidaan tehdä resurssi. Tämä tehdään luokassa *TermCollectorExtractor* .

5.2 Muokkaus

XML2RDF-työkalun muokkaus jollekin muulle XML-muodolle sopivaksi onnistuu muokkaamalla pääluokkaa *KlaffiExport* tai tekemällä uusi vastaava pääluokka. Pääluokka on melko yksinkertainen. Se toimii seuraavasti:

- lataa eristyksessä tarvittavat ontologiat
- alustaa eristäjät ja rakentaa niistä eristäjäketjut
- lataa XML-lähdetiedostot yksi kerrallaan ja suorittaa eristyksen niille
- tulostaa eristetysta tiedosta tuotetut RDF-mallit tiedostoihin

Näistä oleellisin on eristäjäketjujen luonti, sillä ne määräävät sen, miten tietoa haetaan ja käsitellään. Nämä toimivat yleensä seuraavasti:

- Haetaan XPath-lausekkeella XML-tiedostosta haluttu tieto (luokka *XPathInfoExtractor*).
- Pilkotaan tieto osiin säännöllisellä lausekkeella (luokka *SplitInfoExtractor*).
- Suoritetaan tiedon siistimistä poistamalla osa merkkijonosta tai korvaamalla osa siitä toisella merkkijonolla. Nämä voidaan tehdä säännöllisillä lausekkeilla (luokka *RegexInfoExtractor*).
- Suoritetaan ontologioiden yhteensovitusta (luokka *MapInfoExtrator*).
- Jos eristetään RDF-resurssia, loppu tapahtuu luokan *TermCollectorExtractor* sisällä. Aluksi muunnetaan arvo URI:ksi (luokka *URIInfoExtractor*).
- URI:sta tehdään RDF-resurssi.

- Resurssin nimeksi asetetaan eristetty arvo ennen URI-muunnosta.

Vaiheiden järjestys voi hieman vaihdella ja kaikkia kohtia ei tarvita aina. Kun yksittäiset eristäjät alustetaan sopivasti, esimerkiksi antamalla sopiva XPath- ja säännöllinen lauseke, voidaan pelkästään näillä eristäjillä eristää erittäin suuri osa halutusta informaatiosta. Monimutkaisemmissa tapauksissa voi olla tarpeen tehdä uusia eristäjä-luokkia.

Jos tietoa eristetään erimuotoisista tiedostoista tai elementeistä, joudutaan tekemään useita *CollectedInfoExtractor* -ilmentymiä ja käyttämään kullekin tyypille sopivaa ilmentymää. Joissakin tapauksissa saattaa myös olla hyödyllistä suorittaa eristys useassa vaiheessa käymällä tiedostot useaan kertaan läpi esimerkiksi jos kustakin tiedostosta saadaan eristettyä useita erityyppisiä kohteita tai jos edellisten vaiheiden tulosta halutaan hyödyntää seuraavissa vaiheissa. Orava-projektissa ohjelmasarjat tuotetaan erillisellä läpikäynnillä.

6 Cocoon

Apache Cocoon on Apache projektin tuottama sovelluskehys, jonka tarkoitus on helpottaa verkkosovellusten kehittämistä ja tehdä niistä modulaarisia. Tässä luvussa esitetään miten Orava-portaali käyttää Cocoonin liukuhihnatekniikkaa. Tarkka Cocoonin dokumentaatio on saatavilla sen kotisivuilta. Lisäksi esitetään URL-parametrit, joita Orava-portaalille voi antaa.

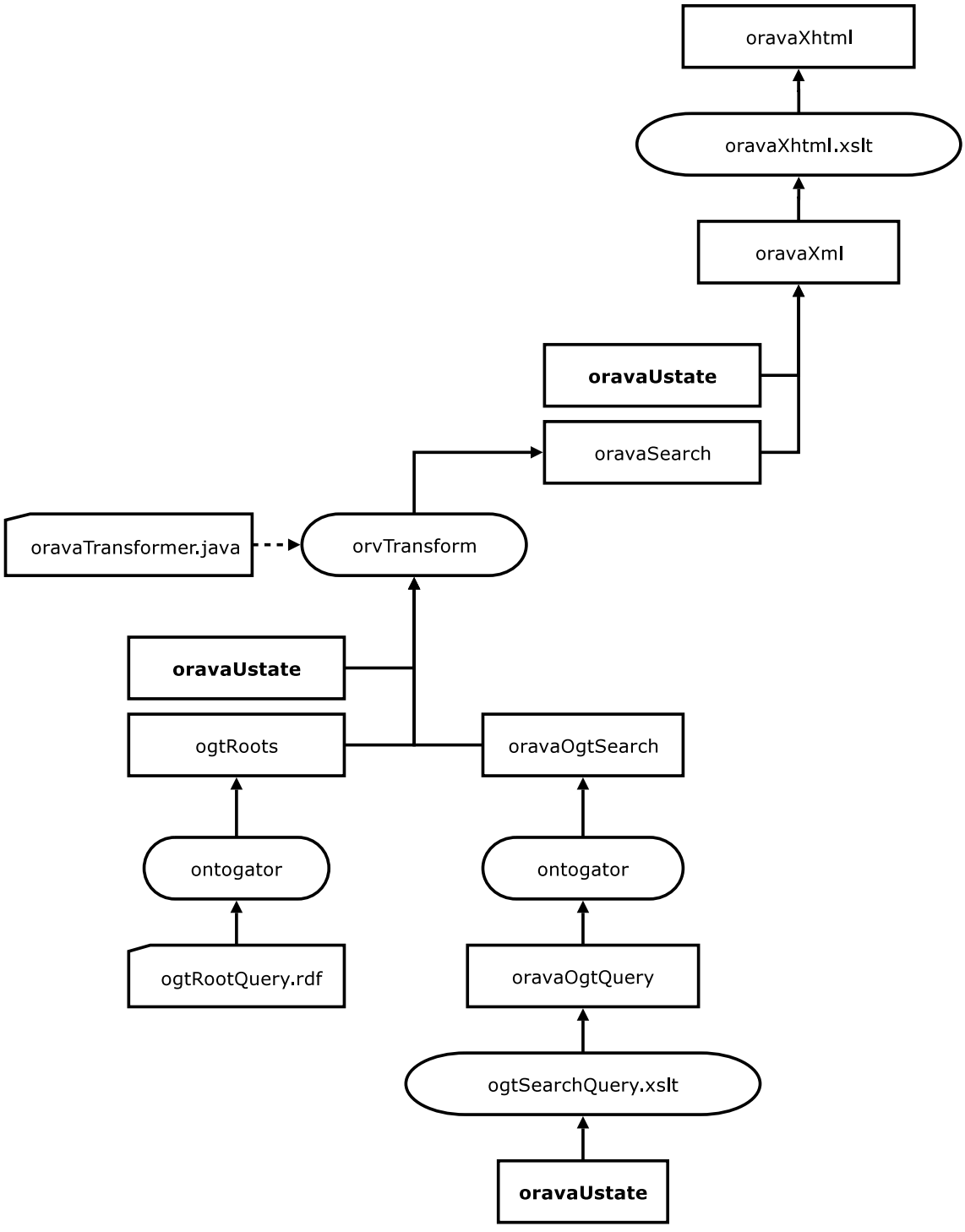
6.1 Liukuhihnat

Cocoon käyttää niinsanottua liukuhihnatekniikkaa sovellusten tuottamiseen. Sovellukselle määritellään resursseja, joista jokainen tuottaa jonkin xml-dokumentin. Resurssit tehdään yhdistämällä muiden resurssien tuloksia ja muuntamalla yhdistetty dokumentti toiseen muotoon joko Java-ohjelmalla tai xsl-tyylitiedostolla. Ketjun ensimmäisen resurssin sisältö voidaan lukea tiedostosta tai generoida Cocoonin xsp-kielellä. Lopuksi jokin resurssi muuntaa aikaisemman välituloksen xhtml-muotoon, mikä voidaan lähettää käyttäjän selaimelle.

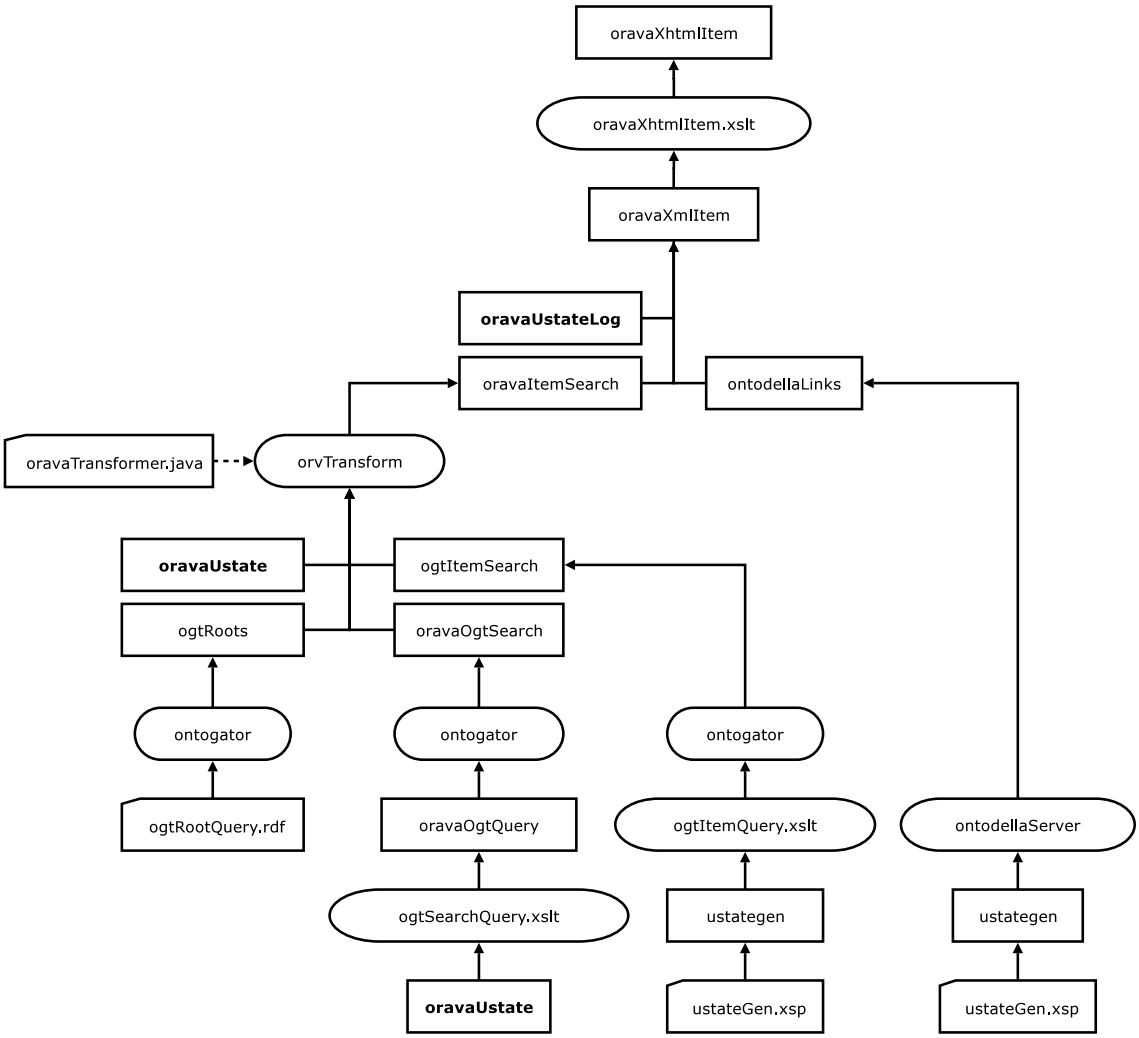
Kuvissa 3, 4 ja 5 on esitetty kaaviona resurssien riippuvuus toisistaan. Kuvissa suorakulmiot esittävät resursseja, soikiot muuntimia ja suorakulmiot, joista on leikattu kulma pois ovat tiedostoja. Suurin osa muuntimista on xsl muuntimia, jolloin soikion sisälle on merkitty xsl-tiedoston nimi, joka päättyy ".xslt". Alla on kuvattu kukin resurssi erikseen.

ustategen Tuottaa käyttäjän valinnat sekä kieli- ja muut asetukset suoraan http-pyynnöstä ustateGen.xsp-ohjelmalla. ustateGen.xsp:n ymmärtämät URL-parametrit on dokumentoitu luvussa 6.2.

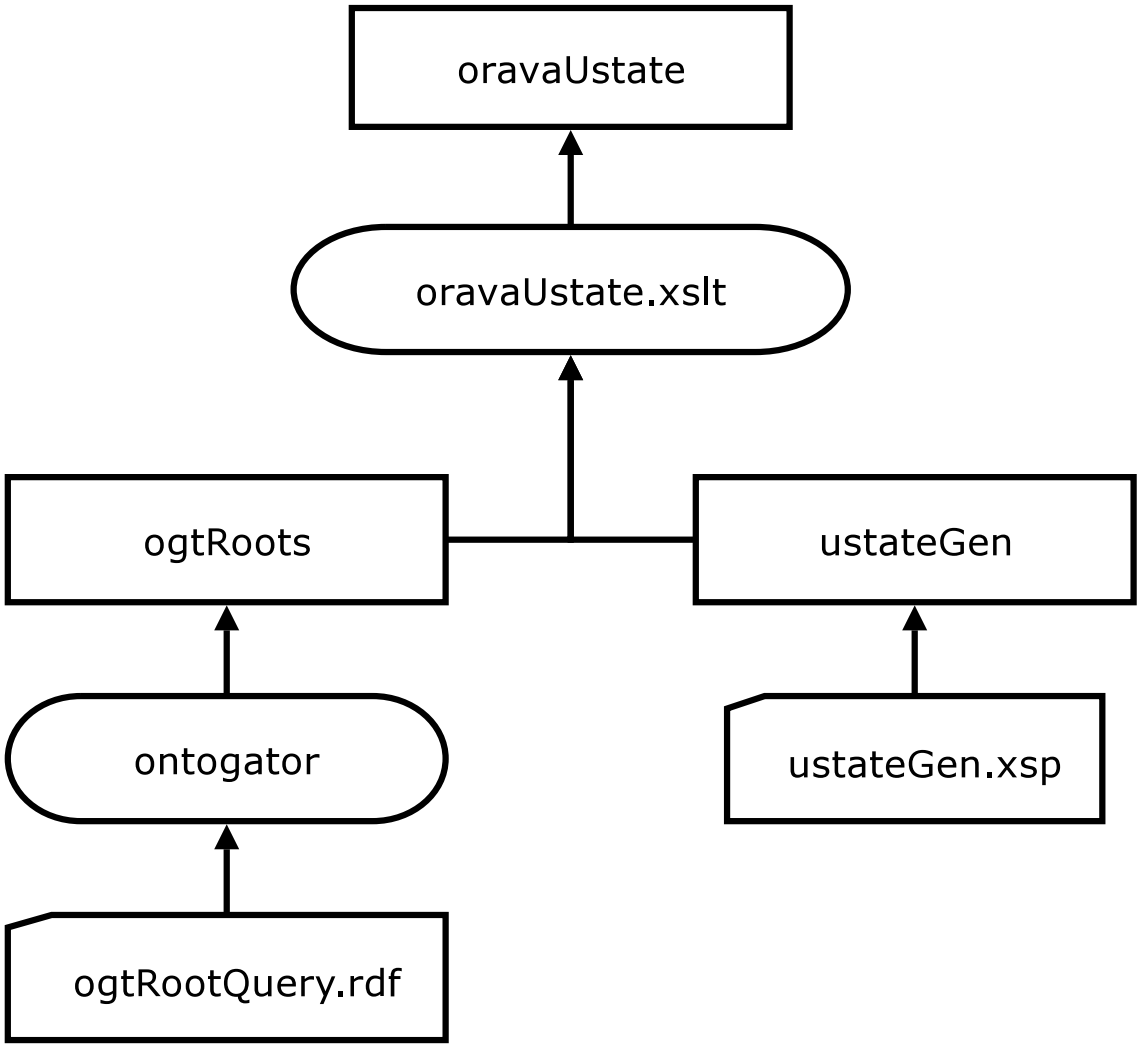
oravaUstate Tuottaa käyttäjän kategoria-valinnat sekä kieli- ja muut asetukset.



Kuva 3: oravaXhtml-resurssin riippuvuus muista resursseista.



Kuva 4: oravaXhtmlItem-resurssin riippuvuus muista resursseista.



Kuva 5: oravaUstate-resurssin riippuvuus muista resursseista.

oravaUstateLog Muuten sama kuin oravaUstate, mutta kirjaa muistiin kohteen, jonka käyttäjä haki. Tämän avulla voidaan tehdä lista suosituimmista kohteista ja viimeksi katsotuista kohteista.

ogtRootQuery.rdf Ontogator-kysely, joka hakee juurikategoriat.

ontogator Itse Ontogator ohjelma, joka suorittaa XML/RDF-muodossa olevan Ontogator-kyselyn ja tuottaa kyselyn tuloksen.

ogtRoots Suorittaa Ontogator-kyselyn, joka tuottaa juurikategoriat.

ogtSearchQuery.xslt Muuntaa käyttäjän valinnat Ontogator-kyselyksi, joka hakee kategorioita ja kohteita käyttäjän valintojen mukaan niin, että tulosten perusteella voidaan tuottaa moninäkymäkäyttöliittymä.

oravaOgtQuery Ontogator-kysely, joka hakee moninäkymäkäyttöliittymän tarvitsemat kategoriat ja resurssit käyttäjän valintojen mukaan.

oravaOgtSearch Käyttäjän valintojen mukaan haetut kategoriat ja kohteet, joista voidaan tuottaa moninäkymäkäyttöliittymä.

ogtItemQuery.xslt Tekee Ontogator-kyselyn, joka hakee tiedot käyttäjän valitsemasta kohteesta.

ogtItemSearch Tiedot käyttäjän valitsemasta kohteesta.

orvTransform Muuntaa Ontogator-kyselyn tulokset muotoon, jota on helpompi käsitellä xsl-tyylitiedostoissa.

oravaSearch Käyttäjän valintojen mukaan haetut kategoriat ja kohteet orvTransform-muodossa.

oravaItemSearch Juurikategoriat ja käyttäjänvalitseman kohteen kategoriat ja tiedot orvTransform-muodossa.

ontodellaServer Suorittaa ontodella kyselyn, joka tuottaa linkkisuositukset.

ontodellaLinks Ontodellalta saadut linkkisuositukset käyttäjän valitsemaan kohteeseen.

oravaXml Käyttäjän valintojen mukaan haetut tiedot orvTransform-muodossa sekä itse valinnat ja käyttäjän asetukset.

oravaXhtml.xslt Muuntaa oravaXml-resurssin sisällön xhtml-muotoon.

oravaXhtml Käyttäjän valintojen mukaan tehty moninäkymäkäyttöliittymä xhtml-muodossa.

oravaXmlItem Valitun kohteen tiedot XML-muodossa.

oravaXhtmlItem.xslt Muunta oravaXmlItem-resurssin sisällön xhtml-muotoon.

oravaXhtmlItem Valitun kohteen tiedot xhtml-muodossa.

6.2 URL-parametrit

Ontoviews, ja siten myös Orava-portaali, pitää kaiken tiedon käyttäjän valinnoista http-pyynnössä GET-tyyppisinä parametreina URL:n sisällä. Alla on dokumentoitu Orava-portaalin käyttämät parametrit.

Kategoriavalinnoissa ja avainsanahaussa on käytäntönä, että yhdessä parametrissa on vanhatvalinnat, yhdellä parametrilla lisätään valintoja ja yhdellä poistetaan. Tämä helpottaa parametrien rakentamista XSL-tyylitiedostoissa.

help Jos 1 näytetään ohjesivu, muuten normaali hakusivu. Oletus 0.

l Käytettävä kieli. Oletus fi.

m Näytetäänkö moninäkymäkäyttöliittymässä kategoriat, joissa ei ole lainkaan haettuja kohteita. Orava-portaalissa tämän täytyy olla aina 1.

sb Mahdollisesti valitun kohteen URI.

k Aikaisemmat avainsanahaut, kukin avainsana välilyönnillä erotettuna.

nk Uusi avainsanahaku.

rk Poistettava avainsanahaku.

refine Jos “true” niin lisätään uusi avainsanahaku aikaisempiin, muuten korvataan vanhat haut uudella. Orava-portaalissa käytetään arvoa “true”.

c Valittu kategoria. Tämä paramerti voi esiintyä moneen kertaan, kukin valinta omana parametrinaan.

n Uusi kategoria valinta.

rc Poistettava kategoria valinta.

g Kategoria, jonka mukaan ryhmitellään kohteet.

skb Kategoriassa yli hypättävien kohteiden lukumäärä. Tämän avulla voidaan selata kategorian kohteita, jos sivulla ei näytetä kaikkia kohteita kerralla. Parametrissa on ensin kategorian tunniste sitten kaksoispiste ja sitten lukumäärä. Parametri voi esiintyä useamman kerran.

nskb Uusi hypättävien kohteiden lukumäärä. Formaatti kuten skb-parametrissa.

mail Jos “true” kirjataan valittu kohde ylös suosituimpien kohteiden listaa varten.

cb Valittu kohde muodossa: kohteen kategoria, kaksoispiste, kohteen järjestysnumero kategoriassa. Käytetään kohteiden selailuun tarkoitettujen “seuraava” ja “edellinen” -linkkien tekemiseen, mutta näitä linkkejä ei Orava-portaalissa ole, joten tätä parametria ei tarvitse käyttää.

cc Valitun kohteen kategoria muodossa: kategorian yläkategoria, kaksoispiste, kategorian järjestysnumero yläkategoriassa. Kuten parametria cb, tätäkin käytetään selailulikkien tekemiseen, joita Orava-portaalissa ei ole ja siten tätäkään parametria ei tarvitse käyttää.

7 Järjestelmän mukauttaminen

Järjestelmään saatetaan haluta lisätä uutta materiaalia, ja joissakin tapauksissa voi olla aiheellista mukauttaa järjestelmä hyödyntämään uuden materiaalin ominaisuuksia paremmin. Käyttäjälle tämä saattaisi ilmetä esimerkiksi uutena näkymänä tai entisen näkymän valittavana kategoriana.

Minkä tahansa muutoksen jälkeen järjestelmä voidaan päivittää ajamalla `./build build`, joka tekee kaiken tarvittavan, mutta saattaa olla joskus hieman hidas.

7.1 Näkymät ja kategoriat

Järjestelmä osaa hyödyntää LOM-muotoisia XML-tiedostoja ja niiden lisääminen on kuvattu käyttöohjeessa. Mikäli kuitenkin halutaan luoda näkymä tai entiseen näkymään alakategoria joidenkin XML-tiedoston kenttien perusteella, voidaan tämä tehdä muokkaamalla ontologioita ja päättelysääntöjä.

Oppiaine- ja teema-näkymiin uusien kategorioiden lisääminen käy yksinkertaisesti muokkaamalla näitä vastaavia ontologioita. Näissä ja muissa yksinkertaisissa tapauksissa näkymät ovat suoraan ontologian mukaisia ja ontologian käsitteitä vastaavat näkymien kategoriat. Kategoriaan kuuluminen selvitetään tutkimalla löytyykö LOM-metadatan avainsanoista kategoriaa vastaavaa ontologia käsittettä. Esimerkiksi LOM-metadan avainsanasta "tähtitiede"voitaisiin luoda uusi kategoria lisäämällä "tähtitiede"ontologiaan.

Uusi näkymä luodaan luvussa 4 esitellyllä tavalla. Yksinkertaisessa tapauksessa näkymä luodaan ontologian perusteella. Ontologiat ovat hakemistossa `data/ontologies/` olevia RDF(S)-tiedostoja.

Lopuksi järjestelmä on päivitettävä ajamalla `./build prepare_data` (tai `./build build`).

7.2 Metadatan uudet ominaisuudet

Mikäli halutaan käyttää metadatasta uusia tietoja, joita nykyinen XML2RDF-muunnostyökalu ei huomioi on XML2RDF-työkalua muokattava asianmukaisesti.

Esimerkiksi nykyinen työkalu ei huomioi ollenkaan LOM-metadatan kenttää `lom/general/structu` jota ei ole käytetty nykyisissä Klaffin metadatoissa, mutta on LOM-määrityksen mukainen.

XML2RDF-työkalun muokkaamisen jälkeen on ajettava `./build build`.

Lähteet

Jena — Jena – a semantic web framewor for java. `http://jena.sourceforg.net`. [9.4. 2005]

Klaffi — Yleisradio oy:n klaffi-opetusvideoarkisto. `http://www.yle.fi/klaffi`. [14.2. 2005]

LOM — Learning object metadata. `http://ltsc.ieee.org/wg12/`. [14.2. 2005]

OntoViews — Ontoviews. `http://www.cs.helsinki.fi/group/seco/museums/dist/`. [14.2. 2005]

Käyttöohje — Orava, Käyttöohje.

SemWeb — Semantic web. `http://www.w3.org/2001/sw/`. [19.2. 2005]

SPr — Swi-prolog. `http://www.swi-prolog.org`. [20.2. 2005]

URI — Uniform resource identifier. `http://www.ietf.org/rfc/rfc3986.txt`. [10.3. 2005]

Coc — Apache cocoon. `http://cocoon.apache.org/`. [14.2. 2005]

XML — Extensible markup language. `http://www.w3c.org/XML/`. [14.2. 2005]

XSL — Extensible stylesheet language. `http://www.w3c.org/XSL`. [20.2. 2005]

MuseoSuomi Museosuomi. http://museosuomi.cs.helsinki.fi/. [14.2. 2005]

RDFS Rdf schema. http://www.w3.org/TR/rdf-schema/. [10.3. 2005]

RDF Resource description framework. http://www.w3c.org/RDF/. [14.2. 2005]

Opinportti Yle opinportti. http://www.yle.fi/opinportti. [21.2. 2005]

This page intentionally not left blank due to the used LaTeX document package bug.