Suunnitteludokumentti

WWW-museo

Tekijät:

Kim Josefsson (KJ)
Samppa Saarela (SS)
Eetu Mäkelä (EM)
Taneli Rantala (TR)
Robert Holmberg (RH)
Pasi Lehtimäki (PL)

Helsinki 3. kesäkuuta 2002

Versio 1.0

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Sisältö

1 Profiilimoduulin sisäinen rajapinta.

1 Johdanto (KJ)

Tämä dokumentti on Helsingin yliopiston museon WWW-näyttelyn suunnitteludokumentti. Dokumentissa kerrotaan, miten järjestelmä totetutetaan ja mitä metodeja ja luokkia järjestelmä sisältää. Luvussa 2 annetaan yleiskuva toteutettavasta ohjelmistosta. Luvussa 3 esitetään erilaisia käyttötapauksia, joita järjestelmän käyttäjillä on. Luvussa 4 esitetään vaatimuksia järjestelmän tekniselle toteutukselle. Luvussa 5 puhutaan suunnittelurajoituksista ja lisäominaisuuksista, joita ei toteuteta tässä projektissa. Luvussa 6 annetaan yleiskuva ohjelmiston osajärjestelmistä. Luvussa 7 esitellään tietokantamoduuli. Luvussa 8 käydään läpi käyttöliittymämoduuli. Luvussa 9 tarkastellaan profiilimoduulia. Luvussa 10 käydään läpi suodatinmoduulin rakenne. Luvussa 11 pohditaan testausta, eli miten siihen varaudutaan ohjelmoinnin aikana. Liitteessä 1 esitetään tietokantamoduulin sisäinen rajapinta. Liitteessä 2 esitetään käyttöliittymämoduulin sisäinen rajapinta. Liitteessä 3 on suodatinmoduulin sisäinen rajapinta.

1.1 Versiohistoria

Versio	Päivämäärä	Tehdyt muutokset
0.1	20.3.2002	Alustava versio. Sisältää perusrakenteen.
0.5	22.3.2002	Kuvia ja tekstiä paranneltu. Arviointiversio, jota muutetaan ryhmäläisten parannusehdotusten pohjalta.
0.9	27.3.2002	Lopullista hyväksymistä vaille valmis versio. Tässä versiossa on kaikki ryhmän mielestä oleellinen asia.
1.0	5.4.2002	Versio joka annetaan asiakkaalle.

2 Ohjelmiston yleiskuvaus (KJ)

Toteutettava järjestelmä on Helsingin yliopiston museon WWW-valokuvanäyttely, Valokuvien aiheena on promootio. Järjestelmä jakaantuu viiteen osajärjestelmään: käyttöliittymään, suodattimeen, profiilin, tietorakenteisiin ja tietokantaan. Nämä kuvataan tarkemmin seuraavissa luvuissa. Valokuvien tiedot on koodattu XML-tiedostoihin. Näitä tiedostoja muokataan käsin siten, että ne ovat RDF-syntaksin mukaisia. Nämä RDF-tiedostot ripustetaan Avril Styrmannin Protege-ohjelmalla luomaan Promootio-ontologiaan. Ontologiaa voidaan muuttaa ja muokata, mikäli muutokset katsotaan tarpeellisiksi. Käyttöliittymän ohjelmoinnissa käytetään JSP:tä. Java-koodit erotetaan sivuista Java-papujen ja JSP-merkkausten, tagien, avulla. Tekoäly osaa ehdottaa ohjelman käyttäjälle mielenkiintoisia linkkejä (katso Ongelmakenttädokumentti), jotka johtavat käyttäjää mahdollisesti kiinnostaviin kuviin. Järjestelmä toteuteaan Tomcat-ympäristössä yliopiston DB-palvelimelle.

3 Ohjelmiston käyttötapauksia

Seuraavassa esitetään erilaisia käyttötapauksia:

Persoonia

Teppo on 25 vuotias tietojenkäsittelytieteen opiskelija. Hän valmistuu maisteriksi ensi keväänä. Teppo on kiinnostunut vanhoista purjelaivoista.

käyttötapauksia

Teppo haluasi jonkinlaisen kuvan promootiotilaisuudesta. Ja mielellään joitain kuvia tapahtumista edellisltä vuosilta.

Teppo haluaa tietää, että onko proomootio risteilyä koskaan järjestetty Suomen joutsen aluksella. Ja jos on, niin hän haluaa nähdä kaikki kuvat aluksesta.

Surffatessaan netissä Teppo on eksynyt wwwmuseon aloitussivulle yliopiston sivustolla olleen linkin kautta. Hänellä on vielä 15 minuuttia aikaa ennen seuraavaa luentoa ja hän haluaisi katsella vanhoja kuvia silläväliä.

Tepon kaveri on toimittajana ylioppilas lehdessä. Hän on pyytänyt tepolta sopivia kuvia artikkeliin, jossa käsitellään yliopiston rehtoreiden tekemisiä wanhalla ylioppilastalolla.

Samaiseen artikkeliin tarvittaisiin kuvia, joissa näkyy sekä rehtori, että promovoitava.

Anita on 55 vuotias filosofian maisteri. Hän luennoi maantieteen laitoksella.

käyttötapauksia

Jukka on tullut luokkansa kanssa tutustumaan Helsingin yliopiston museon kuvakokoelmiin. Jukkaa kiinnostaa, että minkälaisia bileitä Wanhalla on järjestetty viime vuosisadalla.

Jukka on 17 vuotias lukiolainen. Hän on suunnitellut jatkavansa opintoja Helsingin yliopistossa.

käyttötapauksia

Anita on lyönyt kolleegansa kanssa vetoa siitä, että liittyikö vuonna 50 promootiotilaisuuteen risteily vai ei. Hän toivoo löytävänsä vastauksen museon kuvakokoelmasta.

Anitaa kiinnostaa se, että miten naisten pukeutuminen on muuttunuut 50 luvulla pidetyissä juhlatilaisuuksissa.

4 Tärkeimmät vaatimukset järjestelmälle (RH)

Ohjelmisto toteuttaa HiBrowse-selailumallin kuvien selailuliittymänä. Järjestelmän tekoälyosa ehdottaa käyttäjälle linkkejä muihin mielenkiintoisiin kuviin. Tekoälyosa, tietokantaosa, kuvien esittäminen ja mielenkiintoisten linkkien esittäminen toteutetaan palvelinpuolella.

4.1 Alustustiedostot

Ohjelmisto pystyy tekemään vaadittavat asetukset valmiiden, palvelimella olevien ontologioiden ja tallennetun statistiikan perusteella. Ulkoisesta asetustiedosta tarvitaan ainoastaan paikallisten hakemistojen nimet ja palvelun www-osoitteet.

4.2 Laitteistovaatimukset

Ohjelmiston tulisi toimia kohtuullisen hyvin nykyaikaisessa selaimessa, tietokoneella jonka työpöydän resoluutio on vähintään 800*600, mieluiten 1024*768. Selaimessa tulee olla käytettävissään riittävä määrä värejä kuvien tyydyttävään näyttämiseen (väh. 16-bittiset värit).

Suorituskyyltään ohjelmiston tulee pystyä palvelinpuolella käsittelemään n. 30000 kuvan kokoelmaa nykymittapuun mukaan kohtalaisen nopeissa (686-tasoisissa tai vastaavissa) työasemissa ja palvelimilla.

4.3 Laitteistoliittymät

Ohjelmisto on laitteistoriippumaton. Toteutettavat ohjelmistot toimivat javan ajonaikaisen ympäristön versiossa 1.3.1 (Java Runtime Environment, JRE). Palvelimella toimiva kuvia näyttävä kokonaisuus vaatii että palvelimelle on asennettu JRE, Tomcat ajoympäristö sekä Jena-ohjelmakirjasto. Järjestelmän on toimittava yleisimmissä selaimissa. Käyttäjältä vaadita minkäänlaisia erikoistaitoja, vaan selaimen ja graafisten käyttöliittymien peruskäyttötaidot riittävät.

5 Suunnittelurajoitukset (EM)

Seuraavassa luetellaan lisämominaisuuksia, joita ei toteuteta, mutta joiden myöhempään toteuttamiseen varaudutaan:

5.1 Demografiatiedot

Ohjelmistoon tulee olla mahdollista lisätä myöhemmin demografiatietoja käyttäjästä ja niiden avulla tehtävä kuvien ja älykkäiden linkkien rajaus. Kuvien ja linkkien rajaamis/arvottamisfunktio suunnitellaan modulaariseksi ja laajennettavaksi.

5.2 Laitteistoprofiili ja monikielisyys

Laitteistoprofiilien ja monikielisyyden tueksi käyttöliittymä ja muu koodi erotetaan toisistaan mahdollisimman hyvin. JSP -sivuissa käytetään tagikirjastoja.

6 Osajärjestelmät (SS)

Kuvassa 1 on kuvattu WWW-Museon osajärjestelmät UML luokkakaaviotekniikalla. Järjestelmä jakaantuu kuuteen osaan: käyttöliittymä, profiili, suodatin, metatieto, kuvavarasto ja metatietovarasto. Tässä luvussa kuvataan nämä osajärjestelmät yleisellä tasolla ja luvuissa 7, 8, 9 ja 10 yksityiskohtaisesti.

Käyttöliittymä toteuttaa järjestelmän ulkoisen www-liittymän. Käyttöliittymä jakautuu kolmeen osaan: HiBrowse, mielenkiintoisetlinkit eli varsinainen kuvaesitys ja istunto. HiBrowse toteuttaa aineiston HiBrowse-navigointijärjestelmän. Mielenkiintoisilla linkeillä tarkoitetaan varsinaista kuvasivua, jossa esitetään joko yksi tai useampi kuva ja näihin liittyvät tiedot sekä jatkolinkkejä käyttäjää todennäköisesti kiinnostaviin toisiin kuviin tai kuvasarjoihin. Istunto määrittää käyttäjäkohtaiset tiedot, sekä profiilin että käyttöliittymän tarvitsemat tiedot käyttäjästä, istunnon keston ajan.

Profiili toteuttaa käyttäjäkohtaiset preferenssit, joiden avulla pyritään selvittämään juuri tätä kyseistä käyttäjää kiinnostavat tiedot. Käyttäjän profiili muodostuu ensisijaisesti HiBrowsessa valituista kiinnostuksen kohteista sekä historiatiedoista. Profiiliin voidaan tallettaa käyttäjästä myös muita tietoja, esimerkiksi sellaisia, joiden pohjalta pystytään valitsemaan käyttäjään parhaiten sopiva demografia-kuvaus.

Suodatin päättelee annetun käyttäjäkohtaisen profiilin, ontologian sekä mahdollisten muiden kiinnostavuutta mittaavien tekijöiden avulla juuri tätä käyttäjää todennäköisimmin kiinnostavat linkit.

Metatietomoduuli toteuttaa rajapinnan, jonka avulla metatietoa eli ontologiaa ja sen avulla kuvattuja dokumentteja (kuvia) pystyy käsittelemään verkkona. Lisäksi tarjotaan palvelu tietyt hakuehdot täyttävien luokkien, instanssien ja kuvien hakemiseksi. Tietorakenne-osa ei ota kantaa siihen, mitä tarkoitusta varten tietoja tarvitaan eli esimerkiksi HiBrowse ja suodatin osille tarjotaan täysin sama näkymä metatietoverkosta, vaikka HiBrowse ei huomioikaan mahdollisia muita ominaisuuksia kuin perimisen.

Kuvatietovaraston tehtävä on kuvien tallentaminen niin, että saman kuvan eri versioihin on mahdollista viitata URL:lla WWW:stä. Kuvan eri versiot ja näiden todelliset URL:t tulee olla pääteltävissä kuvan tiedoista.

Metatietovaraston tehtävä on metatiedon tallennus sellaisella tavalla, että sitä pystyisi hyödyntämään myös muut, mahdollisesti muilla ohjelmointikielillä toteutetut sovellukset.

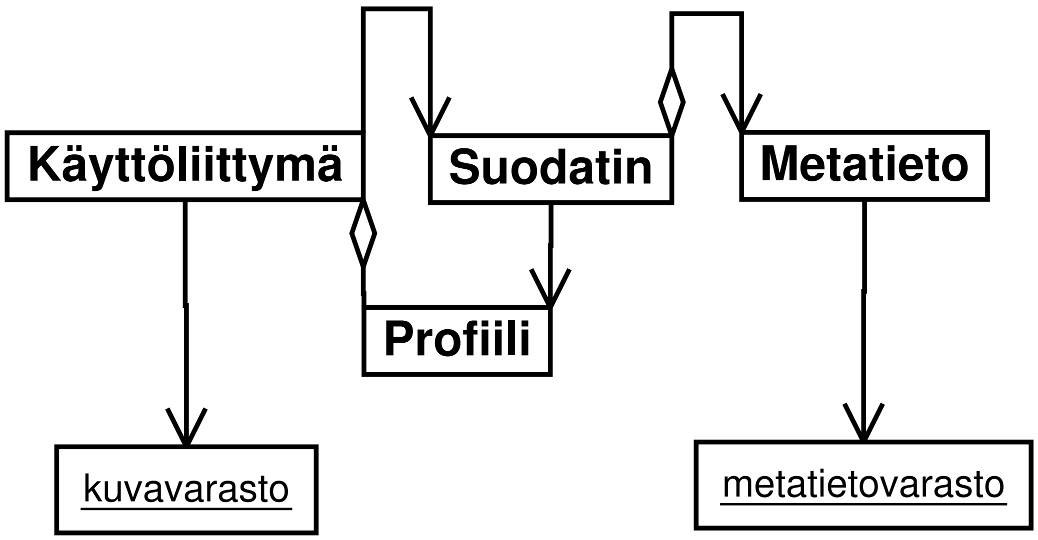
Java-pakkaukset Sovelluksen Java-koodi jaetaan pakkauksiin

1. `fi.helsinki.w3m.filter` suodattimen Java-luokat,
2. `fi.helsinki.w3m.metadata` metatietomodulin Java-luokat ja
3. `fi.helsinki.w3m.profile` profiili.

Käyttöliittymän Java-luokat sijoitetaan Tomcat Servlet-palvelimen vaatimusten mukaisesti erilleen muista luokista. Kuvatietovarastoa ja metatietovarastoa ei toteuteta Javalla.

7 Metatietomoduuli (RH, KJ)

Metatietomoduuli luo ja ylläpitää oliomallin käytetyistä ontologioista, ontologialuokkien ilmentymistä ja kuvakorteista. Moduuli tarjoaa korkeamman tason palveluja ohjelmiston muille osille. Ontologioiden hallintajärjestelmänä käytetään



Kuva 1: WWW-Museon komponentit.

Jena-kirjastoa. Kuvakortit, ontologia ja ontologian luokkien ilmentymät ovat palvelimen levyllä RDF-muodossa. Itse kuvatiedostot ovat erillisessä hakemistossa palvelimen levyllä. Näitten välinen suhde on visualisoitu kuvassa 2.

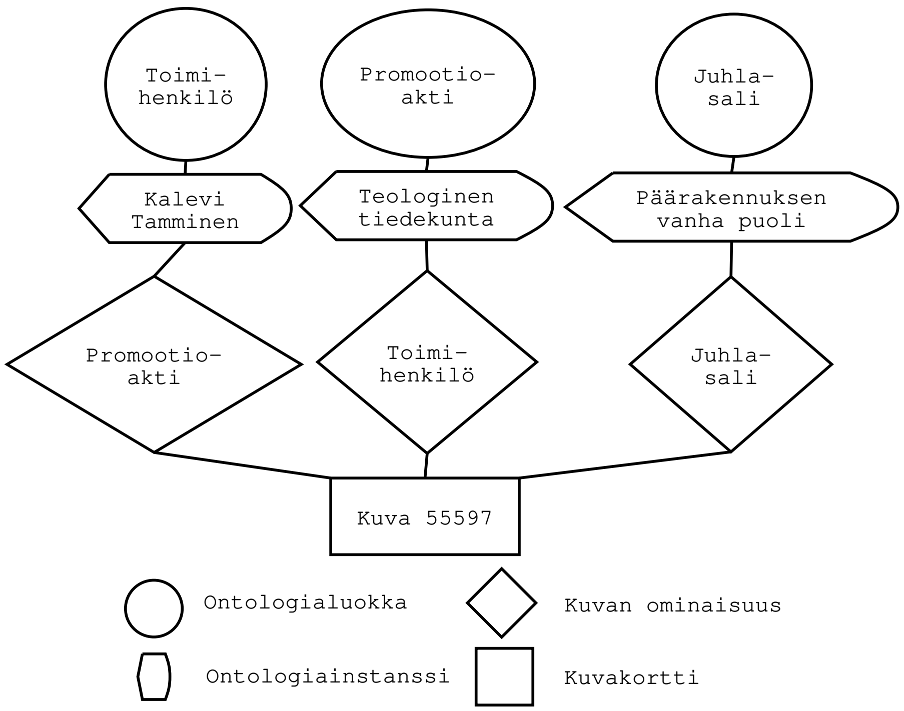
7.1 Luokat

Moduulissa on kahdeksan luokkaa. Moduulin luokkarakenne on kuvattu kuvassa 3.

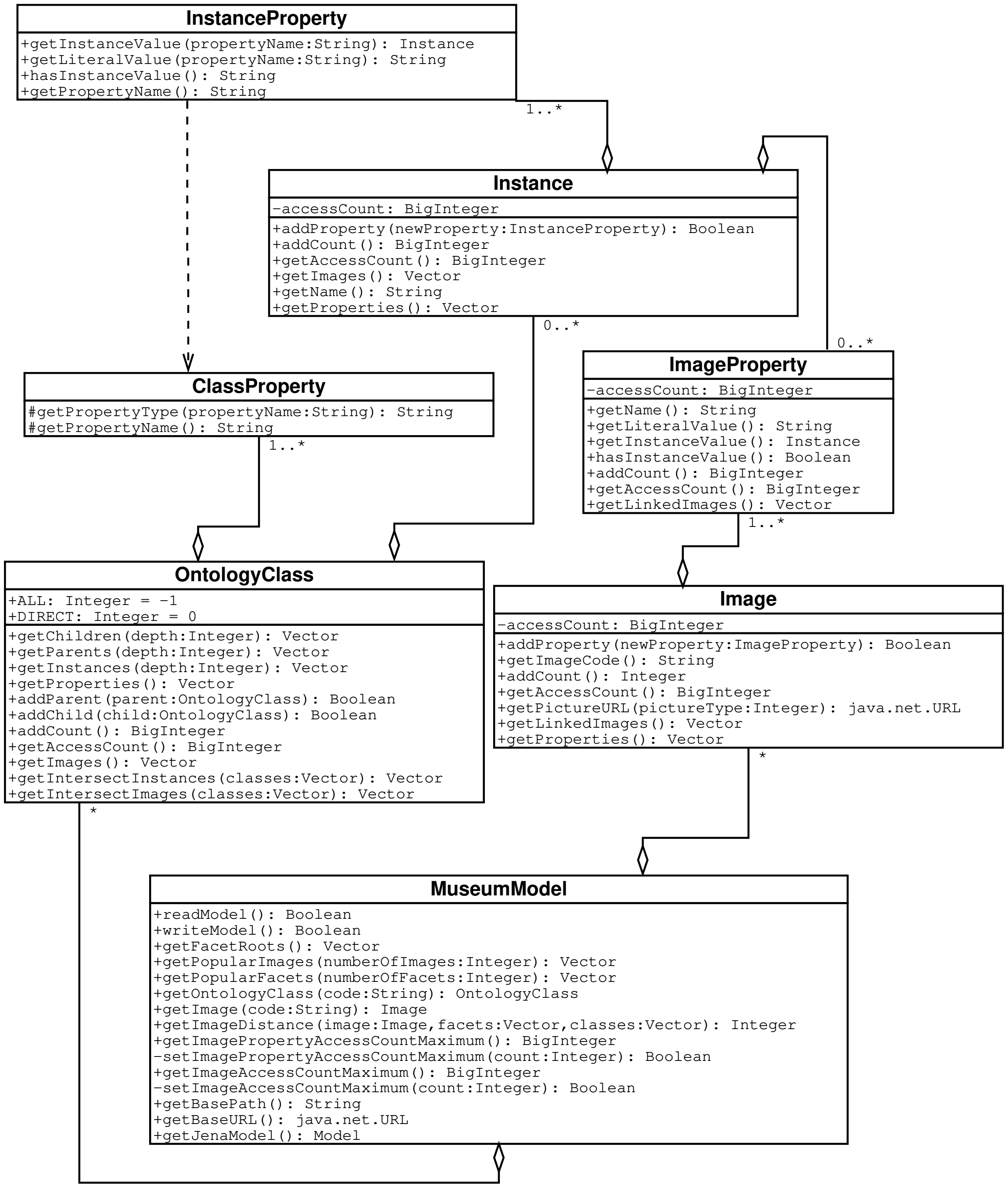
Seuraavaksi on kuvattu metatietomoduulin rajapinnat sekä luokkien keskeiset, rajapinnan toteuttamiseksi tarvittavat tietorakenteet.

7.2 Rajapinta

Jenaa pyritään hyödyntämään niin paljon kuin mahdollista, jotta uutta koodia jouduttaisiin kirjoittamaan mahdollisimman vähän. Kaikki luokat on tarkoitettu myös oliomallin ulkoisen käyttöön, ja siksi erillistä sisäistä rajapintaa ei ole. Seuraavissa metodeissa ei ole otettu kantaa siihen, toteutetaanko ne suoraan Jenalla



Kuva 2: Kuvakortti joka liittyy kolmeen ontologialuokan, ominaisuuksiensa ja ontologian ilmentymien kautta.



Kuva 3: UML-kaavio tietokantamoduulista.

vai ohjelmoimalla itse. Tästä tehdään lopullinen päätös toteutusvaiheessa, kun Jenaa aletaan hyödyntää ja käyttää täysipainoisesti. Vasta tällöin sen puutteet ja sopivuus projektin kannalta selviävät lopullisesti.

Metatietomoduulin rajapinnat (RH, KJ)

Rajapintakuvaukset noudattavat Javan syntaxia lukuunottamatta tietorakenteita joissa on käytetty Pascal-tyylistä "rakenne of sisältö"-notaatiota.

Luokka MuseumModel

Vastaa yhtä oliomallia.

Museum Model(URL baseURL, String basePath)
Luokan konstruktori jolle annetaan parametrina levyhakemisto jossa löytyvät rdfs- ja rdf-dokumentit rdf-alihakemistossa ja kuvat kuvat/tyyppikoodinimisissä hakemistoissa (esim. kuvat/0).

Boolean readModel()
Luo oliomallin levyllä olevista tiedostoista.

Boolean writeModel()
Päivittää levyllä olevat tiedostot muistissa olevan oliomallin mukaiseksi.

Vector of OntologyClass getFacetRoots()
Antaa ne solmut jotka ovat fasettien juuria.

Vector of OntologyClass getPopularFacets(Integer numberOfFacets)
Antaa halutun määrän suosituimpia fasettijuuria niitten käyttöasteen suhteen järjestettyinä.

Vector of Image getPopularImages(Integer numberOfImages)
Antaa halutun määrän suosituimpia kuvia niitten käyttöasteen suhteen järjestettyinä.

Integer getImageDistance(Image image, Vector facets, Vector classes)
Antaa tietyn kuvan etäisyyden HiBrowsen määrittämään valintaan. Etäisyys 0 = sopii täysin valintaan. Korkeammat numerot sopivat huonommin.

Image getImage(String code)

Antaa kuvakorttia vastaavaan Image-olion annetun kuvakoodin mukaan.

OntologyClass getOntologyClass(String code)

Antaa ontologialuokkaa/fasetin solmua vastaavaan OntologyClass-olion annetun nimen mukaan.

BigInteger getImagePropertyAccessCountMaximum()

Antaa suurimman kuvan ominaisuuden laskuriarvon.

BigInteger setImagePropertyAccessCountMaximum(Integer count)

Asettaa suurimman kuvan laskuriarvon. Jos olemassaoleva arvo on suurempi, asetus ei onnistu ja metodi palauttaa epätoden arvon.

BigInteger getImageAccessCountMaximum()

Antaa suurimman kuvan laskuriarvon.

BigInteger setImageAccessCountMaximum(Integer count)

Asettaa suurimman kuvan laskuriarvon. Jos olemassaoleva arvo on suurempi, asetus ei onnistu ja metodi palauttaa epätoden arvon.

Luokka Image

Vastaa yhtä kuvakorttia.

Image(String imageCode)

Luokan konstruktori jolle annetaan parametriksi kuvan koodi, jonka perusteella se haetaan Jenan oliomallista. Kuvakoodi on annettu kuvakorttia vastaavassa levyllä olevassa RDF-dokumentissa.

Boolean addProperty (ImageProperty newProperty)

Lisää kuvaan annetun ominaisuuden, tarkistuksen jälkeen. Ilmentymä-arvoisille ominaisuuksille on löydyttävä ilmentymä ontologiasta.

Vector getProperties()

Antaa kuvan ominaisuudet.

Vector getLinkedImages()

Antaa kuvan ominaisuuksiin liittyvät muut kuvat.

String getImageCode()
Antaa kuvan kuvakoodin.

java.net.URL getPictureURL(Integer pictureType)
Antaa kuvatiedoston URLin tyypin mukaan.

BigInteger addCount()
Lisää käyttölaskuria yhdellä.

BigInteger getAccessCount()
Antaa käyttölaskurin arvon.

Luokka ImageProperty

Vastaa kuvakortin ominaisuutta. Voi olla arvoltaan String tai Instance.

ImageProperty (String propertyName, String propertyLiteralValue)
Luokan konstruktori jolle annetaan parametriksi ominaisuuden nimi ja literaali arvo.

ImageProperty (String propertyName, Instance propertyInstanceValue)
Luokan konstruktori jolle annetaan parametriksi ominaisuuden nimi ja instanssiarvo.

String getName()
Antaa ominaisuuden nimen.

String getLiteralValue()
Antaa ominaisuuden literaaliarvon jos sellainen on.

Instance getInstanceValue()
Antaa ominaisuuden instanssiarvon jos sellainen on.

Boolean hasInstanceValue()
Palautusarvo tosi jos ominaisuuden arvo on instanssi, muuten epätosi.

Vector getLinkedImages()
Antaa tähän omainaisuuden kohdeinstanssiin kuuluvat kuvat.

BigInteger addCount()
Lisää käyttölaskuria yhdellä.

BigInteger getAccessCount()
Antaa käyttölaskurin arvon.

Luokka Instance

Ontologialuokan instanssi. RDF tehty Protegella.

Instance (String instanceName, OntologyClass parent, MuseumModel parentModel)
Luokan konstruktori jolle annetaan parametriksi se ontologialuokka jonka instanssi olio on.

Boolean addProperty(InstanceProperty newProperty)
Lisää instanssiin annetun ominaisuuden, tarkistuksen jälkeen. Sallitut ominaisuudet ovat isäontologialuokan ominaisuuksia.

BigInteger addCount()
Lisää käyttölaskuria yhdellä. Tallennusongelma?

BigInteger getAccessCount()
Antaa käyttölaskurin arvon. Tallennusongelma?

Vector of Image getImages()
Antaa tähän instanssiin liittyvien kuvien Image-oliot.

String getName()
Antaa instanssin nimen.

Vector of InstanceProperty getProperties()
Palauttaa instanssin ominaisuudet.

Luokka InstanceProperty

Vastaa instanssin ominaisuutta. Vastaava pitää olla määritelty isäsolmun ominaisuutena.

InstanceProperty (String propertyName, String propertyLiteralValue, MuseumModel parentModel)
Luokan konstruktori jolle annetaan parametriksi ominaisuuden nimi ja ominaisuuden literaali arvo.

InstanceProperty (String propertyName, Instance propertyInstanceValue, MuseumModel parenModel)
Luokan konstruktori jolle annetaan parametriksi ominaisuuden nimi ja ominaisuuden instanssiarvo.

Instance getInstanceValue (String propertyName, MuseumModel parentModel)
Antaa ominaisuuden instanssiarvon jos sellainen on.

String getLiteralValue (String propertyName)
Antaa ominaisuuden literaaliarvon jos sellainen on.

Boolean hasInstanceValue()
Palautusarvo tosi jos ominaisuuden arvo on instanssi, muuten epätosi.

String getPropertyName()
Antaa ominaisuuden nimen.

Luokka OntologyClass

Solmu ontologiaverkossa. Syvyyttä varten on luokassa määritelty vakiot DIRECT=0 ja ALL=-1, josta ALL antaa koko alipuun instanssit.

OntologyClass (String name, MuseumModel parentModel)
Luokan kostruktori.

Vector of OntologyClass getChildren(Integer depth)
Antaa ontologialuokan ja sen depth-syvyisen alipuun lapset.

Vector of OntologyClass getParents(Integer depth)
Antaa ontologialuokan ja sen depth-syvyisen alipuun vanhemmat.

Vector of Instance getInstances(Integer depth)
Antaa ontologialuokan ja sen depth-syvyisen alipuun instanssit.

Vector of Instance getProperties()
Antaa ontologialuokan ominaisuudet.

Boolean addParent(OntologyClass parent)
Antaa ontologialuokan ominaisuudet. Tarpeellinen?

Boolean addChild(OntologyClass child(OntologyClass child)
Antaa ontologialuokan ominaisuudet. Tarpeellinen?

BigInteger addCount()
Lisää käyttölaskuria yhdellä. Tallennusongelma?

BigInteger getAccessCount()
Antaa käyttölaskurin arvon. Tallennusongelma?

Vector of Image getImages()
Antaa tähän ontologialuokkaan liittyvien kuvien Image-oliot.

Vector of Instance getIntersectInstances(Vector of OntologyClass classes)
Antaa tähän ontologialuokkaan ja parametreina annettuihin ontologialuokkiin liittyvien instanssien Instance-oliot.

Vector of Instance getIntersectImages(Vector of OntologyClass classes)
Antaa tähän ontologialuokkaan ja parametreina annettuihin ontologialuokkiin liittyvien kuvien Image-oliot.

String getName()
Palauttaa luokan nimen.

Luokka ClassProperty

Solmun ominaisuus. Malliksi instansseille.

ClassProperty (String propertyName, String propertyType, MuseumModel parentmodel)
Luokan konstruktori jolle annetaan parametriksi ominaisuuden nimi ja tyyppi.

String getPropertyType (String propertyName)
Antaa annetun nimisen ominaisuuden tyypin.

String getPropertyName ()
Palauttaa ominaisuuden nimen.

7.3 Tiedonhakupolut metatietomoduulissa

Tässä luvussa selitetään yleisellä tasolla miten tietyn tiedon haku etenee metatietomoduulissa. Poluilla on hyötyä sekä moduulin käyttöohjeena, tiedon siirtymisen kartoittajina sekä moduulin luokkien rajapintojen puutteiden havaitsijoina.

Kuvan hakeminen koodin perusteella

Tarvitaan MuseumModel-olio.

Kutsutaan MuseumModelin metodia getImage parametrinaan koodi. Metodi palauttaa Image-olion halutusta kuvakortista.

Image-oliolta voidaan pyytää fyysisen kuvansa URL metodilla getPictureURL, joka vaatii parametrikseen kuvan tyyppikoodin. Tyyppikoodi 0 vastaa aina alkuperäistä kuvamuotoa. Kuvan URL palautetaan java.net.URL-luokan ilmentymänä.

Ontologialuokan hakeminen nimen perusteella

Tarvitaan MuseumModel-olio.

Kutsutaan MuseumModelin metodia getOntologyClass parametrinaan URI, johon sisältyy namespace-tieto. Metodi palauttaa OntologyClass-olion halutusta ontologialuokasta.

Kuvaan liittyvien linkkien hakeminen

Tarvitaan halutun kuvan Image-olio.

Haetaan kuvaan liittyvät ominaisuudet päämoduulin getLinksForImage -metodilla.

Metodin palauttaman vektorin alkioina on taulukko jonka ensimmäisessa alkiossa on ominaisuus johon linkki liittyy ja toisessa alkiossa kohdekuvan Image-olio.

Ominaisuuden käyttölaskurin saa kutsumalla ImageProperty-olion getAccessCount-metodia.

HiBrowsen muuttuneen tilan päivitys

Tarvitaan MuseumModel-olio sekä fasettijuurien ja faseteissa valittujen solmujen nimet.

Kutsutaan MuseumModelin getOntologyClass-metodia kuten yllä saadaksemme fasettijuurten sekä valittujen solmujen ontologialuokat.

Käytetään jotakin puun läpikäymisalgoritmia ja OntologyClassin getChildren-metodia löytääkseen valittujen solmujen alipuuhun kuuluvat ontologialuokat.

Kutsutaan jokaisen alipuun solmun OntologyClass-olion metodia getInstances saadaksemme alipuun ontologialuokkien ilmentymät.

Kutsutaan MuseumModel-olion getImagesForInstances-metodia parametrinaan Vector-olio jossa on alipuun ontologialuokkien ilmentymät. Metodi palauttaa Vector-olion jossa on siihen alipuuhun kuuluvien kuvien Image-oliot. Tämä tehdään kerran jokaiselle fasetille.

Muodostetaan näitten Image-olioiden leikkaus fasettien suhteen.

Lopuksi jokainen kuva ripustetaan jokaiseen fasettiin etsimällä syntyneestä leikkauksesta jokaisen kuvan jokaista ilmentymä-arvoista ominaisuutta vastaava ilmentymä ja paikallistamalla tämä ilmentymän sijainti fasetissa.

Instansseihin liittyvien kuvien hakeminen

Tarvitaan MuseumModel-olio ja yksi tai useampi Instance olio. Lisätään ne Vector-olioon.

Kutsutaan MuseumModel-olion getImagesForInstances-metodia parametrinaan tämä Vector-olio. Metodi palauttaa Vectorin jossa on annettuihin instansseihin liittyvät Image-oliot.

Kuvan etäisyysarvon hakeminen

Tarvitaan MuseumModel-olio, Image-olio kuvasta jolle etäisyys pitää hakea ja HiBrowsen tila, johon sisältyy fasettijuurien ja valittujen solmujen OntologyClass-oliot kahdessa Vector-oliossa.

Kutsutaan MuseumModel-olion getImageDistance-metodia parametrinaan kuvan Image-olio, ja HiBrowsen tilan määrittävät Vector-oliot.

Metodi palauttaa Integer-olion jonka arvo on se määrä ylöspäisiirtoja joita HiBrowsessa tarvitaan jotta kuva sisältyisi HiBrowsen suodatukseen.

Laskureiden käyttö

Image-, ImageProperty-, OntologyClass- ja Instance-olioiden laskurin arvio lisätään olion addCount-metodilla.

Olioiden laskurin arvot luetaan kutsumalla olioden getAccessCount-metodia.

8 Käyttöliittymämoduuli (PL)

8.1 Rajapintojen kuvaus

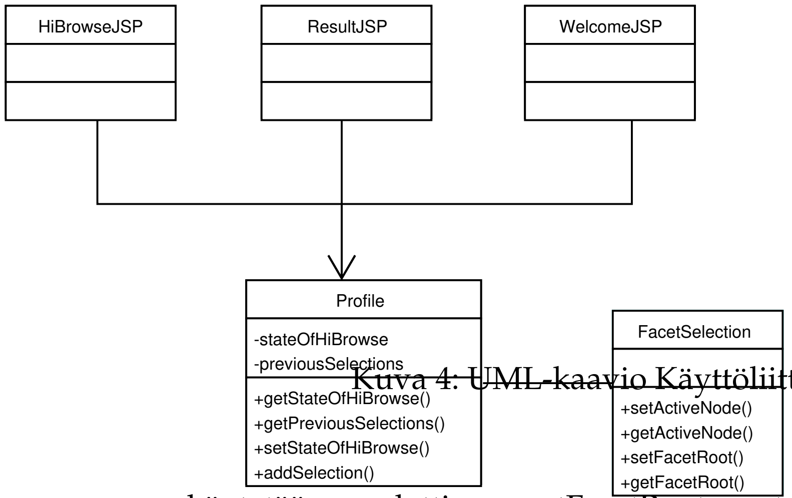
Käyttöliittymä käyttää suodattimen tarjoamia palveluita muodostaessaan tulossivua ja HiBrowse selaamiseen. Suodattimelle käyttöliittymä antaa pyyntöjen yhteydessä tiedeon kulloisenkin käyttäjän HiBrowsen tilasta ja edellisistä kuvavalinnoista.

8.2 Käyttöliittymäkuvaus

Käyttöliittymä jakautuu kahteen osaan. Vasen osa pitää sisällään HiBrowse-komponentin ja oikea osa kuvasivun. HiBrowse-komponentilla käyttäjä voi etsiä haluamiaan kuvia museon kokoelmasta. Käyttöliittymässä tarjotaan käyttäjälle muutama järjestelmän valitsema kuva, jotta käyttäjää ei pakoteta käyttämään HiBrowse-komponenttia. Jokaiseen kuvasivuun liittyy mielenkiintoisia linkejä, jota seuraamalla käyttäjä voi tutustua helposti ja vaivattomasti museon kokoelmiin.

8.3 Luokkakuvaus

HiBrowseJSP Tämä JSP toteuttaa HiBrowse-komponentin. Komponentin puurakenne luodaan käyttäen HTML-taulukoita ja kuvia. Puurakenteen alustamiseen



Kuva 4: UML-kaavio Käyttöliittymä- ja profiilimodulista.

käytetään suodattimen getFacetRoots-metodia. Kun käyttäjä selailee HiBrowse-komponenttia, JSP kutsuu itseään ja URL-koodaa muuttuneen fasetin tilan. Koodaus tapa: "HiBrowse.jsp?path=<fasetinPolku>&action=<toiminta>"fasetinPolku muotoa "juuri.luokka.aliluokka"action parametrejä show, close, open, select, removeFacet, addFacet Kun JSP alkaa tämänjälkeen kokoamaan näkymää, ensin se päivittää Profile-olioon oikean HiBrowsen tilan. Tämänjälkeen kutsutaan suodattimen getFacetChildren-metodia ja getFacetIntersectImages-metodia Profile-parametrilla, jokaista fasettia kohden. Viimeinen aste on vain rakentaa palautusarvoja vastaavat näkymät ruudulle. Mikäli käyttäjä valitsee yksittäisen kuvan, kutsutaan oikean kehykseen ResultJSP:tä, jolle välitetään URL-parametrina valitun kuvan ID. Tässä tapauksessa HiBrowse-komponenttia ei päivitetä.

WelcomeJSP

Tämä luokka generoi oikeean kehykseen aloitussivun, joka sisältää pientä ohjeistusta HiBrowsen käyttöön ja muutaman käyttäjiä yleisesti kiinnostaneen kuvan. Nämä kuvat saadaan suodattimelta Filter-luokan metodilla getDefaultImages.

ResultJSP

Tämä luokka generoi varsinaisen tulossivun oikeaan kehykseen. Tulossivu tarvitsee URL-koodattuna parametrina näytettävän kuvan ID:n. JSP pyytää saamaansa ID:tä vastaavat tiedot suodattimelta getImage- ja getInterestingLinks-metodeilla.

9 Profiili (PL)

9.1 Luokkakuvaus

Profile

Profilessa on tallessa tämänhetkinen HiBrowse-komponentin tila ja lista kuvista, joita käyttäjä on tämän istunnon aikana katsellut. Profiilit ovat varastoituna javax.servlet.http.HttpSession-olioon nimellä profile.

FacetSelection Tässä luokassa on tiedot yhden fasetin valitusta solmusta ja juuresta. Molemmat tiedot tallennetaan String-olioina.

9.2 Liittymäsuunnitelma

Profile-luokan suodattimelle tarjoamat metodit

public FacetSelection[] getStateOfHiBrowse();

Palautettava taulukko sisältää tiedon niiden fasettien tilasta, jotka eivät ole alkutilassa. Kukin taulukon elementti sisältää aktiivisen Node-olion.

public Vector getPreviousSelections();

Palautettava Vector-olio sisältää tämän käyttäjän katsomien kuvien ID:t.

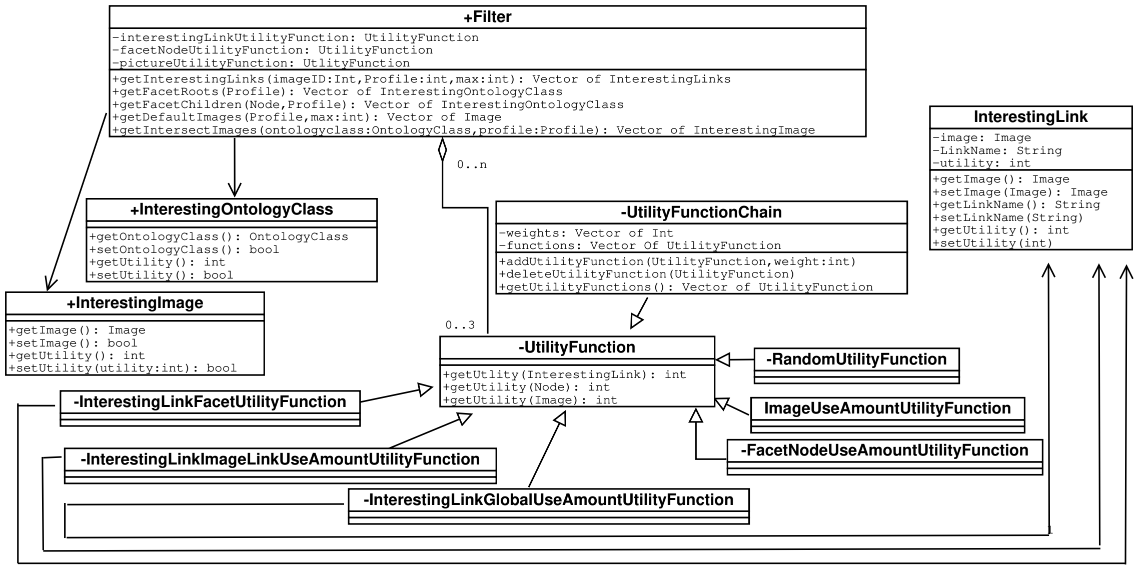
Profile-luokan käyttöliittymämodulille tarjoamat metodit

public void setStateOfHiBrowse(FacetSelection); Asettaa nykyisen HiBrowsen tilan Profile-olioon. Jos tämän fasetintila on jo varastoituna Profile-olioon, muutetaan sen tila vastaamaan uutta tilaa. Jos tämän fasetin tilaa ei löydy, lisätään se Profile-olioon.

public void addSelection(String); Lisää valitun kuva historiatietoihin.

FacetSelection-luokan accessorit

public void setActiveNode(String newActive, String parent);
public String getActiveNode();
public void setFacetRoot(String);
public String getFacetRoot();



10 Suodatin (EM, TR)

Suodatinmoduuli toimii välittäjänä käyttöliittymä- ja metatietomoduulien välillä, järjestäen ja suodattaen käyttöliittymän tarvitsemaa tietoa ’älykkäästi’ saatavilla olevan muun tiedon mukaan.

Olioiden järjestely tehdään utiliteettifunktioilla (=hyödyllisyysfunktio/kiinnostavuusfunktio). Jokainen utiliteettifunktio palauttaa käsitteellisesti väliltä 0.0-1.0 olevan arvion parametrina saadun olion kiinnostavuudesta (nopeussyistä arvo palautetaan kokonaislukuna 0-65535).

Kokonaisutiteetin laskemiseksi useita utiliteettifunktioita ketjutetaan yhteen eri painoarvoilla ja niistä laskettu yhteisutiliteetti normalisoidaan taas välillä 0-65535, jolloin ketju muodostaa taas utiliteettifunktion.

10.1 Rajapinnat

Ainoa suodattimen rajapinta on käyttöliittymään, jolle suodatin tarjoaa mahdollisuuden mielenkiintoisten linkkien, näytettävien kuvien ja HiBrowse-käyttöliittymän fasettien järjestämiseen.

Rajapinnan käyttöliittymään muodostaa luokka Filter.

10.2 Rajapintametodit

Kaikki rajapintametodit palauttavat mielenkiintoisuusfunktion arvon (väliltä 65535-0) mukaan laskevaan järjestykseen järjestetyn taulukon kysytyistä objekteista. Tehokkuussyistä joillekin metodeille voidaan antaa parametriksi maksimimäärä max, joka määrää, kuinka monta oliota palautustaulukkossa maksimissaan saa olla. Taulukosta jätetään pois tietysti aina huonoimman utiliteettiarvon saaneet oliot.

public Vector of InterestingLink getInterestingLinks(String ImageId, Profile, int max)

Hakee ja palauttaa johonkin kuvaan liittyvät mielenkiintoiset linkit, järjestäen ne annetun profiilin ja muiden saatavilla olevien tietojen mukaan kiinnostavuusjärjestykseen. Metodi palauttaa maksimissaan max-alkioisen mielenkiintoisuusfunktion arvon (väliltä 65535-0) mukaan laskevaan järjestykseen järjestetyn taulukon.

Alustus:

Image=MuseumModel.getImage(ImageId); InterestingLinks.linkname=Image.getProperties(); InterestingLinks.Image=Image.getProperties().getLinkedImages();

Utiliteettifunktioketju:

InterestingLinkGlobalAccessCountUtilityFunction linkin klikkausten maara/suosituimman linkin klikkausten maara

Property.getAccessCount() / MuseumModel.getImagePropertyAccessCountMaximum();

InterestingLinkImageLinkAccessCountUtilityFunction linkin klikkausten maara/kuvan suosituimman linkin klikkausten maara

Property.getAccessCount() / max(Image.Property.getAccessCount());

InterestingLinkFacetUtilityFunction kuinka hyvin linkki sopii viimeisiin valideihin fasettivalintoihin

1 / (MuseumModel.getImageDistance(Image,Profile.getStateOfHiBrowse())+1);

InterestingLinkImageAccessCountUtilityFunction montako kertaa linkin kohdekuvaa/kohdekuvia on klikattu suhteessa suosituimpaan kuvaan

Image.getAccessCount() / MuseumModel.getPopularImages(1).getAccessCount();

RandomUtilityFunction palauttaa sattumanvaraisen utiliteetin

public Vector of InterestingOntologyClass getFacetRoots(Profile);

Hakee ja palauttaa fasettien juurisolmut mielenkiintoisuuden/käytetyimmyyden mukaan järjestettynä.

MuseumModel.getFacetRoots();

public Image getImage(String imageId);

Palauttaa Image-olion kuvan id:n perusteella.

MuseumModel.getImage(imageId);

public Vector of InterestingImage getIntersectImages(OntologyClass,Profile);

Hakee ja palauttaa HiBrowse-valintoihin sopivat instanssit, mielenkiintoisyuuden/käytetyimmyyden mukaan järjestettynä.

Alustus:

MuseumModel.getImagesforInstances(OntologyClass.getIntersectionInstances(Profile.getStateOfHiBrowse().getActiveNodes()));

Utilititeettifunktioketju:

ImageAccessCountUtilityFunction

montako kertaa kyseistä kuvaa on katseltu suhteessa järjestelmän suosituimpaan kuvaan.

Image.getAccessCount() / MuseumModel.getPopularImages(1).getAccessCount();

RandomUtilityFunction palauttaa sattumanvaraisen utiliteetin

public Vector of InterestingOntologyClass getFacetChildren(OntologyClass, Profile);

Hakee ja palauttaa fasettisolmun lapset mielenkiintoisyuuden/käytetyimmyyden mukaan järjestettynä.

Alustus:

children=Ontologyclass.getChildren();

Utilitieettifunktioketju:

FacetNodeUseAmountUtilityFunction

montako kertaa kyseistä fasettinoodia on käytetty suhteessa suosituimpaan fasettiin

RandomUtilityFunction palauttaa sattumanvaraisen utiliteetin

OntologyClass.getAccessCount() / MuseumModel.getPopularFacets(1);

public Vector of InterestingImage getDefaultImages(Profile, int max);

Hakee ja palauttaa käyttäjälle oletuksena näytettävät kuvat yleisen mielenkiinnon/käytetyimmyyden mukaan järjestettynä.

MuseumModel.getPopularImages(max);

public class InterestingImage

public Image getImage();
public bool setImage();
public int getUtility();
public bool setUtility(int);

public class InterestingOntologyClass

public OntologyClass getOntologyClass();
public bool setOntologyClass();
public int getUtility();
public bool setUtility(int);

public class InterestingLink

public Image getImage();
public bool setImage(Image);

```
public String getLinkName();
public bool setLinkName(String);
public int getUtility();
public bool setUtility(int);
```

Liite 1. Profiilimoduulin sisäinen rajapinta

Profiilin API kuvaus (PL)

```
/*
* Suodatin käyttää tämän luokan tietoja tehdessään mielenkiintoisten kuvien lis-
taa.
*/
public class Profile

private FacetNode[] stateOfHiBrowse;
private Vector previousSelections;

/**
* Taulukossa on fasettien aktiiviset nodet. Taulukossa on vain niiden fasettien tie-
dot,
* jotka eivät ole oletustilassa.
*/
public FacetNode[] getStateOfHiBrowse();

/**
* Vectorissa on String-olioita, joissa on kaikkien edellisten kuvavalintojen ID:t
*/
public Vector getPreviousSelections();
public void setStateOfHibrowse(FasetNode[]);
public void addSelection(String);
```