OntoViews – A Tool for Creating Semantic Web Portals

Eetu Mäkelä, Eero Hyvönen, Samppa Saarela, and Kim Viljanen

Helsinki Institute for Information Technology (HIIT), University of Helsinki
P.O. Box 26, 00014 UNIV. OF HELSINKI, FINLAND
{Firstname.Lastname}@cs.Helsinki.FI
http://www.cs.helsinki.fi/group/seco/

Abstract. This paper presents a semantic web portal tool OntoViews for publishing RDF content on the web. OntoViews provides the portal designer with a content-based search engine server, Ontogator, and a link recommendation system server, Ontodella. The user interface is created by combining these servers with the Apache Cocoon framework. From the end-user's viewpoint, the key idea of OntoViews is to combine the multi-facet search paradigm, developed within the information retrieval research community, with semantic web RDFS ontologies, and extend the search service with a semantic browsing facility based on ontological reasoning. OntoViews is presented from the viewpoints of the end-user, architecture, and implementation. The implementation described is modular, easily modified and extended, and provides a good practical basis for creating semantic portals on the web. As a proof of concept, application of OntoViews to a deployed semantic web portal is discussed.

Keywords: Semantic web, information retrieval, multi-facet search, view-based search, recommendation system.

1 Introduction

Much of the semantic web content will be published using semantic portals[1] [1]. Such portals typically provide the end-user with two basic services: 1) a search engine based on the semantics of the content [2] and 2) dynamic linking between pages based on the semantic relations in the underlying knowledge base [3].

This paper presents OntoViews — a tool for creating semantic portals — that facilitates these two services combined with user interface and Web Services functionality inside the Apache Cocoon framework[2]. The search service of OntoViews is based on the idea of combining multi-facet search [4,5] with RDFS ontologies. The dynamic linking service is based on logical rules that define associations of interest between RDF(S) resources. Such associations are rendered on the user interface as links with labels that explain the nature the semantic linkage to the end-user.

These ideas were initially developed and tested as a stand-alone Java application [6] and as a Prolog-based HTML generator [7]. OntoViews combines and extends these

[1] See, e.g., http://www.ontoweb.org/.
[2] http://cocoon.apache.org/

two systems by separating the search and link recommendation services into independent servers (Ontogator and Ontodella) to be used on the Semantic Web, and by providing a clear logic-based interface by which ontology and annotation schema specific RDF(S) structures can be hidden from the servers.

In the following, OntoViews is first presented from the viewpoint of the end-user. We present an application developed using the tool: MuseumFinland[3] [8]. This system is a deployed semantic portal for publishing heterogeneous museum collections on the Semantic Web. After this the architecture and the implementation of the framework are discussed mostly from the portal designer's viewpoint. In conclusion, benefits and limitations of the system are summarized, related work is discussed, and directions for further research are outlined.

2 OntoViews from the End-User's Perspective

OntoViews provides the end-user with a semantic view-based search engine and a recommendation system. In MuseumFinland, these services are provided to the end-user via two different user interfaces: one for desktop computers and one for mobile devices. In below the desktop computer web interface is first presented.

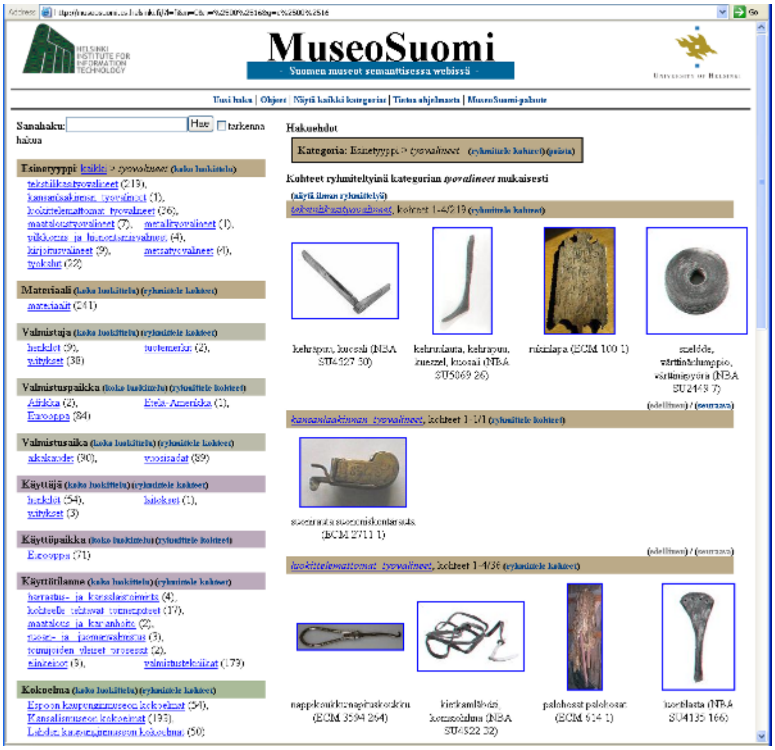
2.1 A Multi-facet Search Engine

The search engine of OntoViews is based on the multi-facet search paradigm [4,5]. Here the concepts used for indexing are called *categories* and are organized systematically into a set of hierarchical, orthogonal taxonomies. The taxonomies are called subject *facets* or *views*. In multi-facet search the views are exposed to the end-user in order to provide her/him with the right query vocabulary and for presenting the repository contents and search results along different views.

In MuseumFinland, the content consists of collections of cultural artifacts and historical sites in RDF format consolidated from several heterogeneous Finnish museum databases [9]. The RDF content is annotated using a set of seven ontologies. From the seven ontologies, nine view-facets are created. The ontologies underlying the application consist of some 10,000 RDF(S) classes and individuals, half of which are in use in the current version on the web. There are some 7,600 categories in the views and 4,500 metadata records of collection artifacts and old cultural sites in Finland.

Figure 1 shows the search interface of MuseumFinland. The nine facet hierarchies, such as Artifact ("Esinetyyppi") and Material ("Materiaali"), are shown (in Finnish) on the left column. For each facet hierarchy, the next level of sub-categories is shown as links. A query is formulated by selecting a category by clicking on its name. When the user selects a category c in a facet f, the system constrains the search by leaving in the result set only such objects that are annotated in facet f with some sub-category of c or c itself. The result set is shown on the right grouped by the sub-categories of the last selection. In this case the user has selected "Tools", whose sub-categories include Textile making tools ("tekstiilityövälineet") and Tools of folk medicine ("kansanlääkinnän työvälineet").

[3] http://museosuomi.cs.helsinki.fi/


HELSINKI INSTITUTE FOR INFORMATION TECHNOLOGY

MuseoSuomi

- Suomen museot semanttisessa webissä -

University of Helsinki

Uusi haku | Ohjeet | Näytä kaikki kategoriat | Tietoa ohjelmasta | MuseoSuomi-palaute

Sanahaku: Hae ☐ tarkenna hakua

Esinetyyppi kaikki > *työvälineet* (koko luokittelu)
tekstiilkäsityövälineet (219), kansanlääkinnän työvälineet (1), luokittelemattomat työvälineet (26), maatalostyövälineet (7), metallityövälineet (1), viikkoinis ja hienotasmisvälineet (4), kirjoitusvälineet (9), metsätyövälineet (4), työkalut (22)

Materiaali (koko luokittelu) (ryhmittele kohteet)
materiaalit (241)

Valmistaja (koko luokittelu) (ryhmittele kohteet)
henkilöt (9), tuotemerkit (2), yritykset (38)

Valmistuspaikka (koko luokittelu) (ryhmittele kohteet)
Afrikka (2), Etelä-Amerikka (1), Eurooppa (84)

Valmistusaika (koko luokittelu) (ryhmittele kohteet)
aikakaudet (90), vuosisadat (89)

Käyttäjä (koko luokittelu) (ryhmittele kohteet)
henkilöt (54), laitokset (1), yritykset (3)

Käyttöpaikka (koko luokittelu) (ryhmittele kohteet)
Eurooppa (71)

Käyttötilanne (koko luokittelu) (ryhmittele kohteet)
harrastus- ja kansalaistoiminta (4), kohteelle tehtävät toimenpiteet (17), maatalous ja karjanhoito (2), riista- ja jaoomerauksus (3), toimijoiden yleiset prosessit (2), elinkeinot (9), valmistustekniikat (179)

Kokoelma (koko luokittelu) (ryhmittele kohteet)
Espoon kaupunginmuseon kokoelmat (54), Kansallismuseon kokoelmat (193), Lahden kaupunginmuseon kokoelmat (50)

Hakuehdot

Kategoria: Esinetyyppi > *työvälineet* (ryhmittele kohteet) (poista)

Kohteet ryhmiteltyinä kategorian *työvälineet* mukaisesti

(näytä ilman ryhmittelyä)

tekstiilkäsityövälineet, kohteet 1-4/219 (ryhmittele kohteet)

kehräpuu, kuosali (NBA SU4327 30) | kehruulauta, kehräpuu, kuezzel, kuosai (NBA SU5069 26) | niisilapa (ECM 100 1) | [illegible], värttinänkumppio, värttinäpyörä (NBA SU2443 7)

(edellinen) / (seuraava)

kansanlääkinnän työvälineet, kohteet 1-1/1 (ryhmittele kohteet)

[illegible] (ECM 2711 1)

(edellinen) / (seuraava)

luokittelemattomat työvälineet, kohteet 1-4/36 (ryhmittele kohteet)

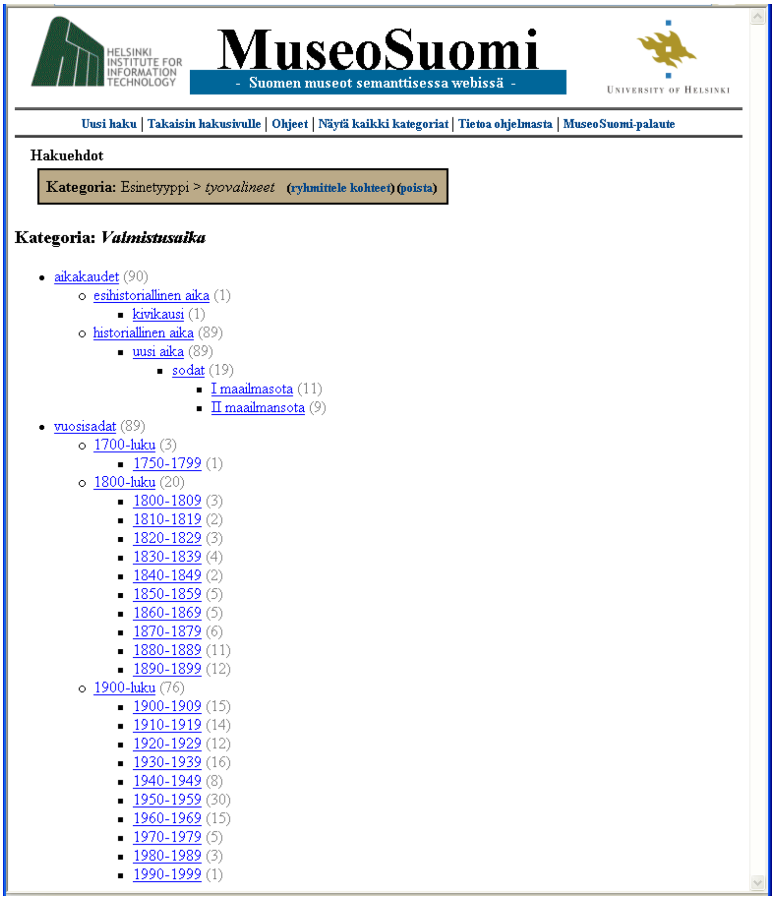
[illegible] (ECM 3594 264) | [illegible], [illegible] (NBA SU4522 32) | [illegible] (ECM 614 1) | [illegible] (NBA SU4135 166)

Fig. 1. MuseumFinland search interface after selecting the category link Tools ("työvälineet").

Hits in different categories are separated by horizontal bars and can be paged through independently in each category.

When answering the query, the result sets resulting from the selection of each category seen on the screen are recomputed, and a number (n) is shown to the user after the category name. It tells that if the category is selected next, then there will be n hits in the result set. For example, in figure 1, the number 193 in the Collection facet ("Kokoelma") on the bottom tells that there are 193 tools in the collections of the National Museum ("Kansallismuseon kokoelmat"). A selection leading to an empty result set ($n = 0$) is removed from its facet (or alternatively disabled and shown grayed out, depending on the user's preference). In this way, the user is hindered from making a selection leading to an empty result set, and is guided toward selections that are likely to constrain the search appropriately. The query can be relaxed by making a new selection on a higher level in the facet or by dismissing the facet totally from the query.

Above, the category selection was made among the direct sub-categories listed in the facets. An alternative way is to click on the link Whole facet ("koko luokittelu") on a facet. The system then shows all possible selections in the whole facet hierarchy with hit counts. For example, in figure 2 the user selected in the situation of figure 1 the link Whole facet of the facet Time of Creation ("Valmistusaika"). The system shows how the


HELSINKI INSTITUTE FOR INFORMATION TECHNOLOGY
MuseoSuomi
- Suomen museot semanttisessa webissä -
University of Helsinki
Uusi haku | Takaisin hakusivulle | Ohjeet | Näytä kaikki kategoriat | Tietoa ohjelmasta | MuseoSuomi-palaute

Hakuehdot

Kategoria: Esinetyyppi > *tyovalineet* (ryhmittele kohteet) (poista)

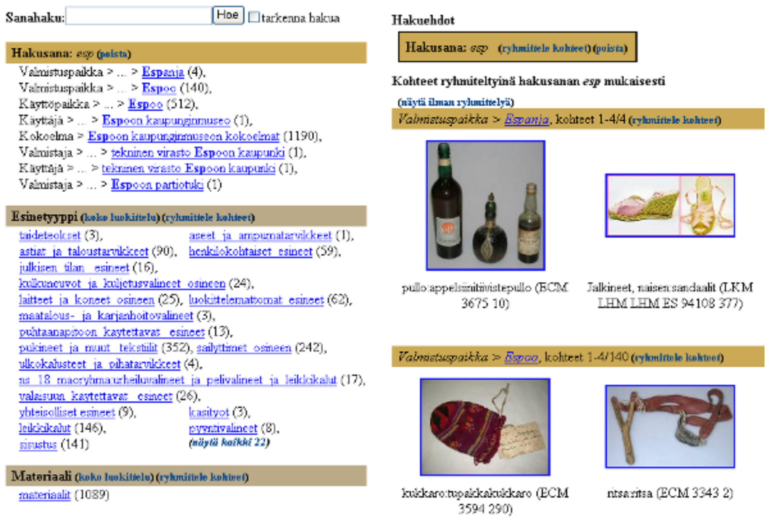
Kategoria: *Valmistusaika*

- aikakaudet (90)
  - esihistoriallinen aika (1)
    - kivikausi (1)
  - historiallinen aika (89)
    - uusi aika (89)
      - sodat (19)
        - I maailmasota (11)
        - II maailmansota (9)
- vuosisadat (89)
  - 1700-luku (3)
    - 1750-1799 (1)
  - 1800-luku (20)
    - 1800-1809 (3)
    - 1810-1819 (2)
    - 1820-1829 (3)
    - 1830-1839 (4)
    - 1840-1849 (2)
    - 1850-1859 (5)
    - 1860-1869 (5)
    - 1870-1879 (6)
    - 1880-1889 (11)
    - 1890-1899 (12)
  - 1900-luku (76)
    - 1900-1909 (15)
    - 1910-1919 (14)
    - 1920-1929 (12)
    - 1930-1939 (16)
    - 1940-1949 (8)
    - 1950-1959 (30)
    - 1960-1969 (15)
    - 1970-1979 (5)
    - 1980-1989 (3)
    - 1990-1999 (1)

Fig. 2. The Time facet hierarchy classifying the result set of tools in figure 1.

tools in the current result set are classified according to the selected facet. This gives the user a good overview of the distribution of items over a desired dimension. With the option of graying out categories with no hits, it is also immediately obvious where the collections are lacking artifacts.

When the user is capable of expressing her information need straightforward in terms of keywords, then a Google-like keyword search interface is usually preferred. ONTO-VIEWS seamlessly integrates this functionality in the following way: First, the search keywords from the search form are matched against category names in the facets. A new dynamic facet is created in the user interface, containing all facet categories matching the keyword shown with the corresponding facet name. Second, a result set of object hits is shown. This result set contains all objects contained in any of the categories matched in addition to all objects whose metadata directly contains the keyword, grouped by the categories found. This way, the keyword search also solves the search problem of finding relevant categories in facets that may contain thousands of categories.



Sanahaku: Hoe tarkenna hakua

Hakusana: *esp* (poista)

Valmistuspaikka > ... > Espanja (4),
Valmistuspaikka > ... > Espoo (140),
Käyttöpaikka > ... > Espoo (512),
Käyttäjä > ... > Espoon kaupunginmuseo (1),
Kokoelma > Espoon kaupunginmuseon kokoelmat (1190),
Valmistaja > ... > tekninen virasto Espoon kaupunki (1),
Käyttäjä > ... > tekninen virasto Espoon kaupunki (1),
Valmistaja > ... > Espoon partiotuki (1)

Esinetyyppi (koko luokittelu) (ryhmittele kohteet)

taideteokset (3), aseet ja ampumatarvikkeet (1),
astiat ja taloustarvikkeet (90), henkilokohtaiset esineet (59),
julkisen tilan esineet (16),
kulkuneuvot ja kuljetusvalineet osineen (24),
laitteet ja koneet osineen (25), luokittelemattomat esineet (62),
maatalous- ja karjanhoitovalineet (3),
puhtaanapitoon kaytettavat esineet (13),
pukineet ja muut tekstiilit (352), sailyttimet osineen (242),
ulkokalusteet ja pihatarvikkeet (4),
ns 18 maoryhma:urheiluvalineet ja pelivalineet ja leikkikalut (17),
valaisuun kaytettavat esineet (26),
yhteisolliset esineet (9), kasityot (3),
leikkikalut (146), pyyntivalineet (8),
sisustus (141) *(näytä kaikki 22)*

Materiaali (koko luokittelu) (ryhmittele kohteet)

materiaalit (1089)

Hakuehdot

Hakusana: *esp* (ryhmittele kohteet) (poista)

Kohteet ryhmiteltyinä hakusanan *esp* mukaisesti

(näytä ilman ryhmittelyä)

Valmistuspaikka > *Espanja*, kohteet 1-4/4 (ryhmittele kohteet)

pullo:appelsiinitiivistepullo (ECM 3675 10)

Jalkineet, naisen:sandaalit (LKM LHM LHM ES 94108 377)

Valmistuspaikka > *Espoo*, kohteet 1-4/140 (ryhmittele kohteet)

kukkaro:tupakkakukkaro (ECM 3594 290)

ritsa:ritsa (ECM 3343 2)

Fig. 3. Using the keyword search for finding categories.

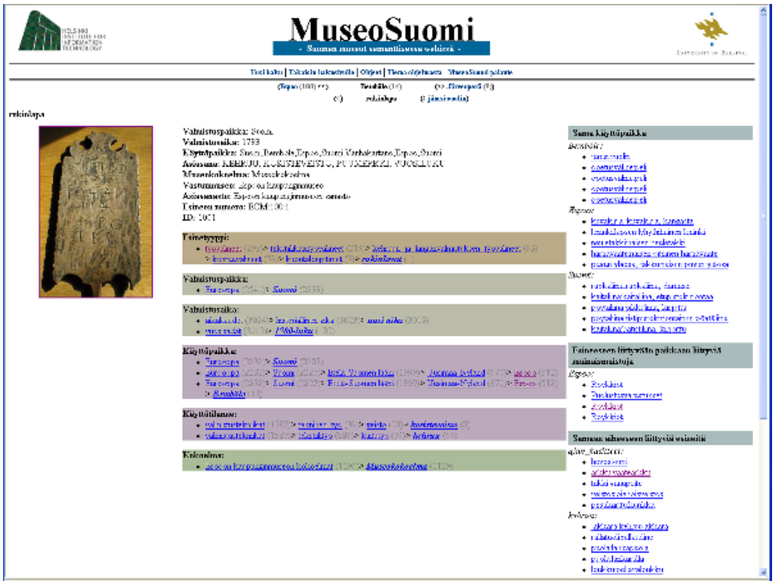
A sample keyword search is shown in figure 3. Here, a search for "esp" has matched, for example, the categories Spain ("Espanja" in Finnish) and Espoo in the facet Location of Creation and the category Espoo City Museum ("Espoon kaupunginmuseo") in the facet User ("Käyttäjä"). The categories found can be used to constrain the multi-facet search as normal, with the distinction that selections from the dynamic facet replace selections in their corresponding facets and dismiss the dynamic facet.

2.2 The Item View with Semantic Links

At any point during multi-facet search the user can select any hit found by clicking on its image. The corresponding data object is then shown as a web page, such as the one in figure 4. The example depicts a special part, distaff ("rukinlapa" in Finnish) used in a spinning wheel. The page contains the following information and links:

1. On top, there are links to directly navigate in the groups and results of the current query.
2. The image(s) of the object is (are) depicted on the left.
3. The metadata of the object is shown in the middle on top.
4. All facet categories that the object is annotated with are listed in the middle bottom as hierarchical link paths. A new search can be started by selecting any category there.
5. A set of semantic links on the right provided by a semantic recommendation system.

The semantic links on the right reveal to the end-user a most interesting aspect of the collection items: the implicit semantic relations that relate collection data with their context and each other. The links provide a *semantic browsing* facility to the end-user.



Valmistuspaikka: [illegible]
Valmistusaika: 1793
Käyttöpaikka: [illegible]
Asiasana: [illegible]
Museokokoelma: [illegible]
Vastuumuseo: [illegible]
Asiasanastu: [illegible]
Esineen numero: [illegible]
ID: 1001

Esinetyyppi: [illegible]

Valmistuspaikka: Eurooppa [illegible] > Suomi [illegible]

Valmistusaika: [illegible]

Käyttöpaikka: [illegible]

Käyttötilanne: [illegible]

Kokoelma: [illegible]

Sama käyttöpaikka

Esineeseen liittyvään paikkaan liittyviä muinaisesineistoja

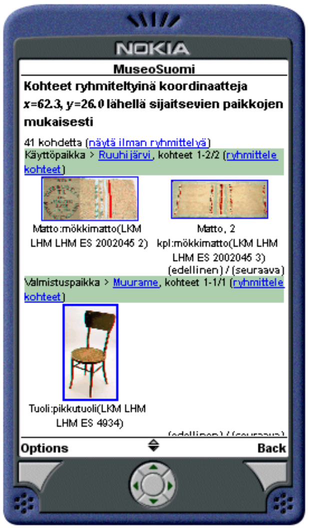
Fig. 4. Web page depicting a collection object, its metadata, facet categories, and semantic recommendation links to other collection object pages.

For example, in figure 4 there are links to objects used at the same location (categorized according to the name of the common location), to objects related to similar events (e.g., objects used in spinning, and objects related to concepts of time, because the distaff in question has a year carved onto it), to objects manufactured at the same time, and so on. Since a decoratively carved distaff used to be a typical wedding gift in Finland, it is also possible to recommend links to other objects related to the wedding event, such as wedding rings. In ONTOVIEWS, such associations can be exposed to the end-user as link groups whose titles and link names explain to the user the reason for the recommendation.

2.3 The Mobile User Interface

Using ONTOVIEWS the same content and services can easily be rendered to the end-users in different ways. To demonstrate this, we created another user interface for MUSEUMFINLAND to be used by WAP 2.0 (XHTML/MP) compatible devices. ONTOVIEWS is a particularly promising tool for designing a mobile user interface due to the following reasons. Firstly, empty results can be eliminated, which is a nice feature in an environment where data transfer latencies and costs are still often high. Secondly, the elimination of infeasible choices makes it possible to use the small screen size more efficiently for displaying relevant information. Thirdly, the semantic browsing functionality is a simple and effective navigation method in a mobile environment.

The mobile interface repeats all functionality of the PC interface, but in a layout more suitable to the limited screen space of mobile devices. In addition, to better facilitate finding interesting starting points for browsing, some mobile-specific search shortcuts were created. The search results are shown first up front noting the current search parameters for easy reference and dismissal, as seen in figure 5. Below this, the actual search facets are shown. In the mobile user interface selectable sub-categories are not shown as explicit links as in the PC interface, but as drop-down lists that replace the whole view when selected. This minimizes screen space usage while browsing the facets, but maximizes usability when selecting sub-categories from them. In-page links are provided for quick navigation between search results and the search form.



MuseoSuomi

Kohteet ryhmiteltyinä koordinaatteja *x*=62.3, *y*=26.0 lähellä sijaitsevien paikkojen mukaisesti

41 kohdetta (näytä ilman ryhmittelyä)

Käyttöpaikka > Ruuhijärvi, kohteet 1-2/2 (ryhmittele kohteet)

Matto:mökkimatto(LKM LHM LHM ES 2002045 2)

Matto, 2 kpl:mökkimatto(LKM LHM LHM ES 2002045 3)

(edellinen) / (seuraava)

Valmistuspaikka > Muurame, kohteet 1-1/1 (ryhmittele kohteet)

Tuoli:pikkutuoli(LKM LHM LHM ES 4934)

(edellinen) / (seuraava)

Fig. 5. Results of a mobile geolocation search initiated somewhere near Ruuhijärvi, Finland.

The item page (corresponding to figure 4) is organized in a similar fashion, showing first the item name, images, metadata, annotations, semantic recommendations, and finally navigation in the search result. There are also in-page links for jumping quickly between the different parts of the page.

The mobile user interface also provides two distinct services aimed specifically for mobile use. Firstly, the interface supports search by the geolocation of the mobile device in the same manner as in the concept-based OntoViews keyword search. Any entries in the Location ontology near the current location of the mobile user are shown in a dynamic facet as well as all data objects made *or* used in any of these locations. In addition, any objects directly annotated with geo-coordinates near the mobile user are shown grouped as normal. This feature gives the user a one-click path to items of likely immediate interest. Secondly, because navigation and search with mobile devices is tedious, any search state can be “bookmarked”, sent by email to a desired address, and inspected later in more detail by using the more convenient PC interface.

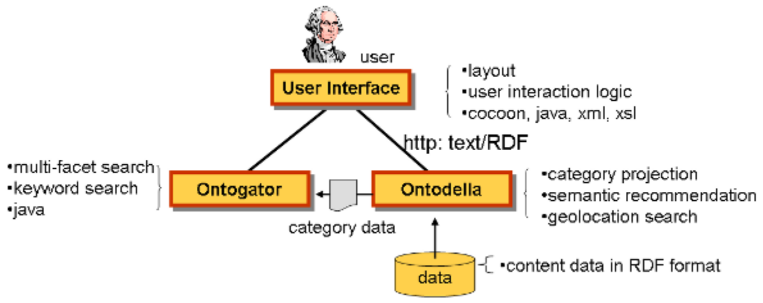


Fig. 6. The components of OntoViews.

3 Architecture and Implementation

OntoViews consists of the three major components shown in figure 6. The logic server of OntoViews, Ontodella, provides the system with reasoning services, such as category generation and semantic recommendations. It is based on the HTTP server version of SWI-Prolog[4]. It has also been extended to provide simple point-of-interest search based on geo-coordinates. It is queried via a HTTP connection.

The multi-facet search engine of OntoViews, Ontogator, is a generic view-based RDF search engine. It defines and implements an RDF-based query interface that is used to separate view-based search logic from the user interface. The interface is defined as an OWL ontology[5]. It can be used to query for category hierarchies and items grouped by and/or constrained by these. Both categories and items can also be queried using keywords. Given a set of category and/or keyword-based constraints, Ontogator filters categories that would lead to an empty result set. Alternatively, filtering the categories, for example by graying out, can be left for the user interface. This is possible since Ontogator (optionally) tags every category with a number of hits. There are also a number of other general options (e.g. accepted language) and restrictions (e.g. max items/categories returned) that can be used, for example, to page the results by categories and/or items. Ontogator replies to queries in RDF/XML that has a fixed structure. Since the search results are used in building the user interface, every resource is tagged with an rdfs:label.

The third component in figure 6, User Interface, binds the services of Ontogator and Ontodella together, and is responsible for the user interfaces and interaction. This component is built on top of the Apache Cocoon framework[6]. Cocoon is a framework based wholly on XML and the concept of pipelines constructed from different types of components, as illustrated in figure 7. A pipeline always begins with a generator, that generates an XML-document. Then follow zero or more transformers that take an XML-document as input and output a document of their own. The pipeline always ends

[4] http://swi-prolog.org/
[5] http://www.cs.helsinki.fi/group/seco/ns/2004/03/ontogator#
[6] http://cocoon.apache.org/

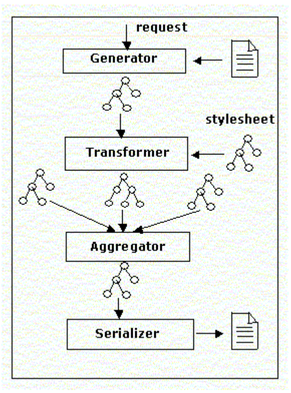


Fig. 7. The components of a Cocoon pipeline.

in a serializer that serializes its input into the final result, such as an HTML-page, a PDF-file, or an image. It is also possible for the output of partial pipelines to be combined via aggregation into a single XML-document for further processing. Execution of these pipelines can be tied to different criteria, e.g, to a combination of the request URI and requesting user-agent.

In OntoViews, all of the intermediate components produce not only XML, but valid RDF/XML. Figure 8 depicts two pipelines of the OntoViews system. The pipe lines look alike, but result in quite different pages, namely in the search result page seen in figure 1 (and another similar page used for depicting results of the keyword search), and in the item page seen in figure 4. This is due to the modular nature of the pipelines, which makes it possible to split a problem into small units and reuse components.

Every pipeline that is tied to user interaction web requests begins with a user state generator that generates an RDF/XML representation of the user's current state. While browsing, the state is encoded wholly in the request URL, which allows for easy bookmarking and also furthers the possibilities of using multiple servers. This user state is then combined with system state information in the form of facet identifiers and query hit counts, and possible user geolocation based information. This information is then transformed into appropriate queries for the Ontogator and Ontodella servers depending on the pipeline.

In the Search Page pipeline on the left, an Ontogator query returning grouped hits and categories is created. In the Item Page pipeline on the right, Ontogator is queried for the properties and annotations of a specific item and its place in the result set, while Ontodella is queried for the semantic links relating to that item. The Ontogator search engine is encapsulated in a Cocoon transformer, while the Ontodella transformer is actually a generic Web Services transformer that creates a HTTP-query from its input, executes it, and creates SAX events from the HTTP-response. The RDF/XML responses from the search engines are then given to user interface transformers depending on the pipeline and the device that originated the request. These transform the results into appropriate XHTML or to any other format, which is then run through an internationalization transformer for language support and serialized. Most of the transformations into queries and XHTML are implemented with simple XSLT-stylesheets. In this way, changes to layout are very simple to implement, as is the creation of new interfaces for different media. The mobile interface to MuseumFinland discussed earlier was created in this way quite quickly.

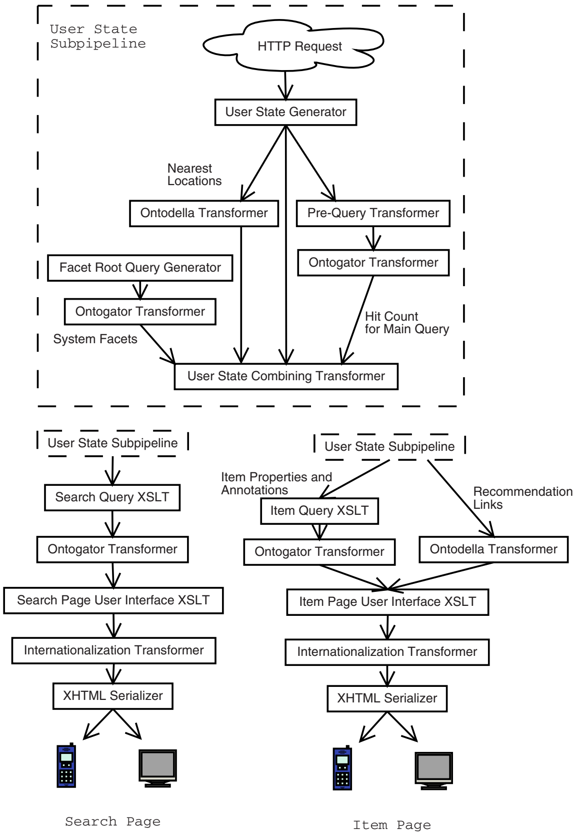


Fig. 8. Two Cocoon pipelines used in OntoViews.

All of the transformer components can also be made available for use in other web applications as Web Services, by creating a pipeline that generates XML from an HTTP-query and returns its output as XML. In this way, other web applications could make use of the actual RDF-data contained in the system, querying the Ontogator and Ontodella servers directly for content data in RDF/XML-format. It also provides a way of distributing the processing in the system to multiple servers. For example, OntoViews instances running Ontogator could be installed on multiple servers, and a main OntoViews handling user interaction could distribute queries among these servers in a round-robin fashion to balance load.

4 Adapting OntoViews to New Data

OntoViews is capable of supporting any RDF-based input data (e.g., OWL). To adapt the system to new data, the following steps must be taken: First, create rules describing how categories are generated and items connected to them for the view based search. Second, create rules describing how links are generated for the recommendations. Third, possibly change the layout templates.

In the following, we describe how the logic rules needed are defined in our system using Prolog. The layout templates are straightforward XSLT and will not be discussed.

4.1 Category View Generation

A view is a hierarchical index-like decomposition of category resources where each category is associated with a set of subcategories and data items. A view is defined in Ontodella, the logic server of OntoViews, by specifying a view predicate called `ontodella_view` with the following information: 1) the root resource URI, 2) a binary subcategory relation predicate, and 3) a binary relation predicate that maps the hierarchy categories with the items used as leaves in the view. In addition, each view must have a label.

An example[7] of a view predicate is given below:

```
ontodella_view(
  'http://www.cs.helsinki.fi/seco/ns/2004/03/places#earth',
  place_sub_category,
  place_of_use_leaf_item,
  [fi:'K\"{a}ytt\"{o}paikka', en:'Place of Use'] % the labels
).
```

Here the URI on the second line is the root resource, `place_sub_category` is the name of the subcategory relation predicate and `place_of_use_leaf_item` is the leaf predicate. The label list contains the labels for each supported language. In our case, we support both Finnish (fi) and English (en).

The binary subcategory predicate can be based, e.g., on a containment property in the following way:

```
place_sub_category( ParentCategory, SubCategory ) :-
  SubCategoryProperty =
    'http://www.cs.helsinki.fi/seco/ns/2004/03/places#isContainedBy',
  rdf( SubCategory, SubCategoryProperty, ParentCategory ).
```

The leaf predicate describes when a given resource item is a member of the given category. For example, `place_of_use_leaf_item` in our example above can be described as follows:

```
place_of_use_leaf_item( ResourceURI, CategoryURI ) :-
  Relation = 'http://www.cs.helsinki.fi/seco/ns/2004/03/artifacts#usedIn',
  rdf( ResourceURI, Relation, CategoryURI ).
```

[7] The syntax is slightly simplified due to presentation reasons. We use SWI-Prolog (http://www.swi-prolog.org) as the inference engine and SWI-Prolog syntax in the examples.

Based on these rules, the categories can be generated by iterating through the predicate `ontodella_view`, and by recursively creating the category hierarchies using the subcategory rules starting from the given root category. At every category, all relevant leaf resources are attached to the category based on the leaf rules. When the categories have been generated, they can be navigated using the Ontogator module presented earlier.

4.2 Recommendation Link Generation

Link generation is based on rules that describe when two resources should be linked. Each link rule can be arbitrary complex and is defined by a domain specialist. A linking rule is described by a predicate of the form $p(SubjectURI, TargetURI, Explanation)$ that should succeed with the two resources $SubjectURI$ and $TargetURI$ are to be linked. The variable $Explanation$ is then bound to an explanatory label (string) for the link. In the following, one of the more complex rules — linking items related to a common event — is presented as an example:

```
related_by_event( Subject, Target, Explanation ) :-

ItemTypeProperty =
  'http://www.cs.helsinki.fi/seco/ns/2004/03/artifacts#item_type',
ItemTypeToEventRelatingProperty =
  'http://www.cs.helsinki.fi/seco/ns/2004/03/mapping#related_to_event',

% check that both URIs correspond in fact to artifacts
isArtifact(Subject),
isArtifact(Target),
% and are not the same
Subject \= Target,

% find all the item types the subject item belongs to
rdf(Subject, ItemTypeProperty, SubjectItemType),
rdfs_transitive_subClassOf(SubjectItemType,SubClassOfSubjectItemType),

% find all the events any of those item types are related to
rdf(SubClassOfSubjectItemType, ItemTypeToEventRelatingProperty, Event),
% and events they include or are part of
(
  rdfs_transitive_subClassOf(Even, SubOrSuperClassOfEvent),
  DescResource=TransitiveSubOrSuperClassOfEvent;
  % or
  rdfs_transitive_subClassOf(SubOrSuperClassOfEvent, Event),
  DescResource=Event;
),

% find all item types related to those events
rdf(TargetItemType, ItemTypeToEventRelatingProperty, SubOrSuperClassOfEvent),
% and all their superclasses
rdfs_transitive_subClassOf(SuperClassOfTargetItemType, TargetItemType),

% don't make uninteresting links between items of the same type
SuperClassOfTargetItemType \= SubjectItemType,
not(rdfs_transitive_subClassOf(SuperClassOfTargetItemType, SubjectItemType)),
not(rdfs_transitive_subClassOf(SubjectItemType, SuperClassOfTargetItemType)),

% finally, find all items related to the linked item types
rdf(Target, ItemTypeProperty, SuperClassOfTargetItemType),

list_labels([DescResource], RelLabel),
Explanation=[commonResources(DescResource), label(fi:RelLabel)].
```

The rule goes over several ontologies, first discovering the object types of the objects, then traversing the object type ontology, relating the object types to events, and finally traversing the event ontology looking for common resources. Additional checks are made to ensure that the found target is an artifact and that the subject and target are not the same resources. Finally, information about the relation is collected, such as the URI and the label of the common resource, and the result is returned as the link label.

The links for a specific subject are generated when the OntoViews main module makes an HTTP query to the Ontodella. As a result, the Ontodella returns an RDF/XML message containing the link information (the target URI and the relation description). These links are then shown to the user using the User Interface (cf. figure 6).

5 Discussion

5.1 Benefits and Limitations

The example application MuseumFinland shows that the multi-facet search paradigm combined with ontologies is feasible as a basis for search on the Semantic Web. The paradigm is especially useful when the user is not just searching for a particular piece of information, but is interested in getting a broader view of the contents of the repository, and in browsing of the contents in the large. The addition of keyword-based searching complements multi-facet searching nicely, better addressing the search needs of people with a clear idea of what they want and with means of expressing it. Such a search can be integrated seamlessly into the user interaction logic and visualization of the user interface. It can be used to also solve the search problem of finding appropriate concepts in the ontologies to be used as a basis in multi-facet search. The semantic recommendation system provides further browsing possibilities by linking the items semantically with each other by relations that cannot be expressed with the hierarchical categorizations used in the multi-facet search.

The Cocoon-based implementation of the OntoViews is eminently portable, extendable, modifiable, and modular when compared to our previous test implementations. This flexibility is a direct result of designing the application around the Cocoon concepts of transformers and pipelines, in contrast to servlets and layout XSLT. The generality and flexibility of OntoViews has been verified in creating the mobile device interface for MuseumFinland. Furthermore, we have used OntoViews in the creation of a semantic yellow page portal [10], and (using a later version of the tool) a test portal based on the material of the Open Directory Project (ODP)[8]. These demonstrations are based on ontologies and content different from MuseumFinland. With the ODP material, the system was tested to scale up to 2.3 million data items and 275,000 view categories with search times of less then 5 seconds on an ordinary PC server.

The use of XSLT in most of the user interface and query transformations makes it easy to modify the interface appearance and to add new functionality. However, it has also led to some quite complicated XSLT templates in the more involved areas of user interaction logic, e.g., when (sub-)paging and navigating in the search result pages. In using XSLT with RDF/XML there is also the problem that the same RDF triple can

[8] http://www.dmoz.org/

be represented in XML in different ways but an XSLT template can be only tied to a specific representation. In our current system, this problem can be avoided because the RDF/XML serialization formats used by each of the subcomponents of the system are known, but in a general web service environment, this could cause complications. However, the core search engine components of OntoViews would be unaffected even in this case because they handle their input with true RDF semantics.

When applying OntoViews, the category and linking rules are described by a domain specialist. She selects what should be shown in the views and what relations between data items should be presented to the user as semantical links. With the help of the rules, any RDF data can be used as input for OntoViews. The prize of the flexibility is that somebody must create the rules, which can be a difficult task if the input data is not directly suitable for generating the wanted projections and links.

5.2 Related Work

Much of the web user interface and user interaction logic in OntoViews and MuseumFinland pertaining to multi-facet search is based on Flamenco [5]. In OntoViews, however, several extensions to this baseline have been added, such as the tree view of categories (figure 2), the seamless integration of concept-based keyword and geolocation search, extended navigation in the result set, and semantic browsing. The easy addition of these capabilities was made possible by basing the system on RDF. We feel it would have been much more difficult to implement the system by using the traditional relational database model. Our approach also permits OntoViews to load and publish any existing RDF data, once the rules defining the facet projections and semantic links are defined.

5.3 Future Work

OntoViews is a research prototype. The test with the material of the Open Directory Project has proved it quite scalable with regards to the amount of data, but the response-time of the system currently scales linearly with respect to the number of simultaneous queries. Further optimization work would be needed to make the system truly able to handle large amounts of concurrent users. The benefits observed applying the XML-based Cocoon pipeline concept to RDF handling has lead us to the question, whether it would be possible to create a true "Semantic Cocoon" based on RDF semantics and an RDF-transformation language.

References

1. A. Maedche, S. Staab, N. Stojanovic, R. Struder, and Y. Sure, "Semantic portal — the SEAL approach," Institute AIFB, University of Karlsruhe, Germany, Tech. Rep., 2001.
2. S. Decker, M. Erdmann, D. Fensel, and R. Studer, "Ontobroker: Ontology based access to distributed and semi-structured unformation," in *DS-8*, 1999, pp. 351–369, citeseer.nj.nec.com/article/decker98ontobroker.html.
3. C. Goble, S. Bechhofer, L. Carr, D. D. Roure, and W. Hall, "Conceptual open hypermedia = the semantic web?" in *Proceedings of the WWW2001, Semantic Web Workshop, Hongkong*, 2001.

4. A. S. Pollitt, “The key role of classification and indexing in view-based searching,” University of Huddersfield, UK, Tech. Rep., 1998, http://www.ifla.org/IV/ifla63/63polst.pdf.
5. M. Hearst, A. Elliott, J. English, R. Sinha, K. Swearingen, and K.-P. Lee, “Finding the flow in web site search,” *CACM*, vol. 45, no. 9, pp. 42–49, 2002.
6. E. Hyvönen, S. Saarela, and K. Viljanen, “Application of ontology-based techniques to view-based semantic search and browsing,” in *Proceedings of the First European Semantic Web Symposium, May 10-12, Heraklion, Greece*. Springer–Verlag, Berlin, 2004.
7. E. Hyvönen, M. Holi, and K. Viljanen, “Designing and creating a web site based on RDF content,” in *Proceedings of WWW2004 Workshop on Application Design, Development, and Implementation Issues in the Semantic Web, New York, USA*. CEUR Workshop Proceedings, Vol-105, 2004, http://ceur-ws.org.
8. E. Hyvönen, M. Junnila, S. Kettula, E. Mäkelä, S. Saarela, M. Salminen, A. Syreeni, A. Valo, and K. Viljanen, “Finnish Museums on the Semantic Web. User’s perspective on MuseumFinland,” in *Proceedings of Museums and the Web 2004 (MW2004), Seleted Papers, Arlington, Virginia, USA*, 2004, http://www.archimuse.com/mw2004/papers/hyvonen/hyvonen.html.
9. E. Hyvönen, M. Salminen, S. Kettula, and M. Junnila, “A content creation process for the Semantic Web,” in *Proceeding of OntoLex 2004: Ontologies and Lexical Resources in Distributed Environments, May 29, Lisbon, Portugal*, 2004.
10. M. Laukkanen, K. Viljanen, M. Apiola, P. Lindgren, and E. Hyvönen, “Towards ontology-based yellow page services,” in *Proceedings of WWW2004 Workshop on Application Design, Development, and Implementation Issues in the Semantic Web, New York, USA*. CEUR Workshop Proceedings, Vol-105, 2004, http://ceur-ws.org.