A Logic-Based RDF to HTML Generator
for Spinning the Semantic Web

Eero Hyvönen and **Markus Holi** and **Kim Viljanen** and **Arttu Valo** [1]

Abstract.

The Resource Description Framework RDF is used to describe content, such as HTML pages and other documents, for the machines to interpret on the Semantic Web. In contrast, we consider the problem of rendering RDF content for the human user by transforming RDF descriptions into a repository of semantically indexed and linked HTML pages. This paper presents a two level method for such a transformation: On the HTML level, the layout of the pages can be described by an HTML designer by using templates and tags. On the RDF level, the semantics of the tags are specified by a system programmer in terms of logical rules based on the RDF(S) repository. The idea is to apply logic to defining the semantic linkage structure and the indices of the page repository. This approach provides a simple method for publishing a human readable version of RDF content on the web. The idea has been implemented as a tool called *SWeHG* for generating a static, semantically linked site of HTML pages from an RDF repository. As case applications, web exhibitions generated from museum collection metadata are presented.

1 PUBLISHING PROBLEMS ON THE SEMANTIC WEB

A key idea of the Semantic Web[2][1, 3] is to enrich the web with metadata describing resources, such as web pages[3], documents, photos, and real world objects in a machine understandable form. For representing the metadata and ontologies, the resource description framework RDF[4] is commonly used. Semantic web languages are intended for the machine to interpret but the ultimate goal is usually to present the semantic content in a human readable form also to the user.

Semantic portals[5] are widely used for publishing dynamic semantic web content for the humans. This approach has, however, created publication obstacles from the viewpoint of content providers: 1) Only content of certain type conforming to the portal's application ontologies can usually be published. 2) The publication process is dependent on the organization maintaining the portal application. 2) Ordinary Internet Service Providers (ISP) do not usually allow ordinary users to run servers or provide semantic portal services for their customers. 3) Content publishers do not necessarily have the required

technical skills to maintain semantic portals. 4) The portal may provide some of the content only in the "hidden web" which hinders Internet search engines such as Google from indexing the content. 5) Using the semantic portal may not scale up to large amounts of user's if lot's of dynamic data processing is needed. 6) Special care is needed to ensure security of the portal.

The notion of semantic portal is in these respects in contrast with the very idea of the current web, where everybody can publish content easily and independently by just maintaining HTML files in a public directory. To address these real life publishing problems, we have developed a method and a tool "Semantic Web HTML Generator *SWeHG*" [9] for transforming RDF(S) repositories into repositories of semantically indexed and mutually linked static HTML pages.

In the following, we first show an example of a semantically indexed and linked HTML repository generated by *SWeHG*. The specifications needed for the RDF to HTML transformation are then discussed. After this, the transformation process and its implementation are presented, and application cases of *SWeHG* are discussed. In conclusion, experiences of our research and experimentation are summarized, related work is described, and directions for further research are outlined.
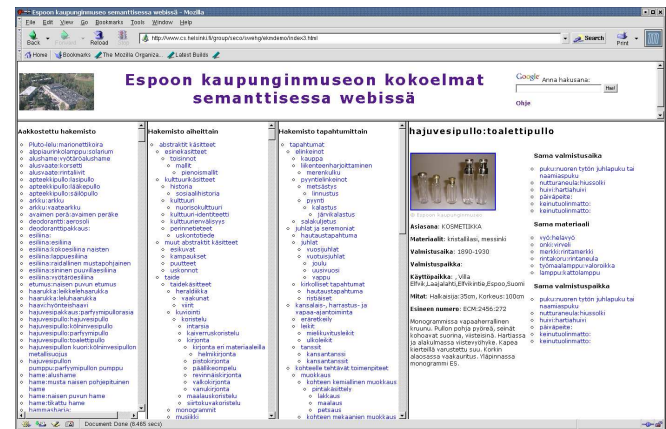
2 THE USER'S PERSPECTIVE

Figure 1. A photo exhibition generated with *SWeHG*.

Figure 1 presents the home page of the exhibition "Espoo City Museum on the Semantic Web" that SWeHG generated for the mu-

---

[1] Helsinki Institute for Information Technology (HIIT), University of Helsinki, P.O. Box 26, 00014 UNIV. OF HELSINKI, FINLAND, email: FirstName.LastName@cs.Helsinki.FI, home: http://www.cs.helsinki.fi/group/seco/
[2] http://www.w3.org/2001/sw/
[3] See, e.g., http://dmoz.org.
[4] http://www.w3.org/RDF/
[5] See, e.g., http://www.ontoweb.org and http://www.mindswap.org.

seum[6]. It consists of some 1200 *resource web pages* (RPage) describing objects in the museum's collection database, pages indexing the contents along different classifications, and a short user's guide. On the left, three frames containing indices for the underlying content are seen. The alphabetical index ("Aakkostettu hakemisto") contains links to the RPages in alphabetical order. By selecting a link, the RPage is shown on the right. In the figure, the user has selected a link to an RPage depicting perfume bottles. Before making a selection, the user's guide was shown in the same frame. The classified index ("Hakemisto aiheittain") is based on the RDFS taxonomy of the underlying cultural MAO ontology [8] that was used when creating the collection metadata. When selecting a concept, the rightmost frame shows links to its subconcepts together with links to RPages whose objects are directly related to the concept. By selecting a subconcept link there, the taxonomy can be browsed further downward; by selecting a link to an RPage, the corresponding collection object with its metadata can be viewed in the frame. The third index "Hakemisto tapahtumittain" classifies collection object by associating them with the different events, processes or activities is which the objects are used or otherwise related to.

By using the indices, the user can find collection objects of interest. An alternative way is to use a conventional search engine. In the upper right corner of figure 1 a form for using Google to search for the pages in the repository is seen. The hit list will be shown in the rightmost frame.

After finding an PRage of interest, the collection can be browsed by using the semantic links generated between related collection items. For example, in figure 1 links to objects manufactured at the same location, objects of similar material etc. can be clicked. The semantic links are generated based on the underlying ontologies, metadata, and logical recommendation rules defined by a domain expert.

The RDF(S) metadata to the exhibition was originally created for a semantic portal [6]. By using SWeHG the same content could be reused without any modifications in order to create a separate application for the museum's own independent usage. The museum can publish the content by just copying the pages into a public HTML directory. This is of practical importance, since museums typically do not have competent IT personnel, servers, and resources to create and maintain semantic portals of their own.

To sum up, the output of SWeHG is a semantically linked space of HTML pages of the following kind: 1) *Resource pages (RPage)* depict selected resources with their metadata. 2) *Index pages (IPage)* classify RPages along conceptual hierarchical classifications, that will be called *facets* or *views* [10]. By using IPages, RPages can be found along different facets. 3) A *home page (HPage)* defines the entrance page to the HTML repository.

Figure 2 illustrates the RDF to HTML transformation. The RDF graph is on the left. Each R_n corresponds to a resource corresponding to a data entry in the RDF repository. In our example, the data entries are collection objects with their metadata. On the right, the HPage has links to various IPages classifying the underlying RPages that are related with each other by semantic links.

3 DEFINING TAG SEMANTICS IN LOGIC

SWeHG is based on descriptions on two levels: 1) The layout of the HTML pages is described on the *HTML level* by templates using custom tags. 2) The semantics of the tags is defined on the *RDF level* in
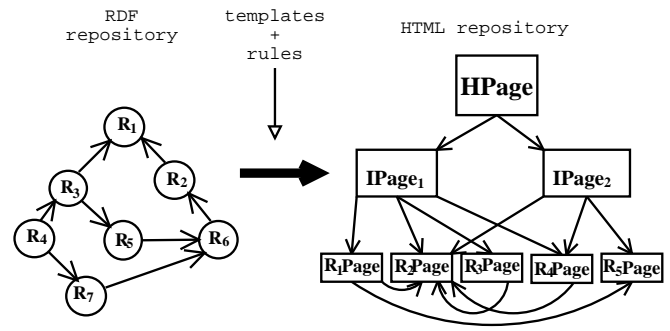
Figure 2.   Transforming an RDF repository into HTML pages.

terms of logical rules based on the input RDF(S) content. The idea is that an HTML designer can design the layout of the page repository to be generated by using tags without knowing details of the underlying RDF structures, RDFS ontologies, and Prolog programming. RDF(S) related knowledge as well as programming capability in Prolog is needed only for the system programmer when defining the tags. The same tags definitions can be re-used in applications conforming to similar ontological schemas.

SWeHG provides the HTML designer with three major tags: getProperty, getLinks, and getView. The tag <getProperty name=p> is used for rendering a label related to the resource underlying an RPage. For example, the metadata property values of the bottles and the photo in figure 1 are rendered in this way. The relation p can be specified by the system programmer on the RDF level freely by a binary logical predicate.

The tag <getLinks> is used for rendering links between RPages. For example, the tag

```
<swehg:getLinks name="SameLocation"
    listType="ul" listStyle="text-size: 10;"/>
```

could expand into the following HTML code linking photographs taken at the same location:

```
<ul style="text-size: 10;">
  <li><a href="entry.Mediacard_00071.html">
     View from Eiffel-tower</a></li>
  <li><a href="entry.Mediacard_00143.html">
     Cafe Parisienne</a></li> ...
</ul>
```

On the RDF level, the criterion SameLocation for the linkage could defined by the predicate below[7]. It associates the attribute SameLocation with the HTML link label 'Same Place' and the predicate photosWithSameLocation defining the link relation.

```
swehg_relation_rule( 'SameLocation',
 'Same Place', photosWithSameLocation).
photosWithSameLocation(Context, Target) :-
  photo(Context), photo(Target),
  rdf(Context, _:place, Location),
  rdf(Target, _:place, Location),
  not(Context == Target).
```

The tag <getView> renders into a hierarchical index-like view of category resources used in IPages. Each category is associated with a set of subcategories and additional individuals of the categories. A view is defined by specifying 1) the root resource selector, 2) a binary subcategory relation predicate, and 3) a binary relation predicate that maps the hierarchy categories with the individuals used as leaves in the view. For example, the tag

```
<swehg:getView
  roots="buildings" branches="subclass"
  leaves="photoOf" listType="ul" />
```

expands recursively into a hierarchical unordered tree (ul), where the leaves are links to photo record resources related to different building categories. The predicate definitions defining the meaning of the attribute values can be, for example, the following:

```
buildings(URI) :-
  rdf(URI, rdf:type, 'http://some.org#building').
subclass(SubCategory, SuperCategory) :-
  rdf(SubCategory, rdfs:subClassOf, SuperCategory).
photoOf(Class, Record) :-
  rdf(Instance, rdf:type, Class),
  rdf(Record, dc:subject, Instance).
```

Here buildings selects the class building as the view root, and the hierarchy is expanded along the rdfs:subClassOf property. The photoOf predicate relates each building type c of this tree with a set of photo record resources which are used as the leaf categories of c. These are rendered as HTML links to the corresponding RPages. The view expansion into HTML can be controlled with the help of additional tag attributes for, e.g., ordering the categories.

The following is an example of a complete RPage template. It could be used for rendering the images using the HTML img-tag and links to related RPages:

```
<swehg:template selector="photo">
<html>
<body>
  <h2><swehg:getProperty name="Title_Of_Photo"/></h2>
  <p><img src="<swehg:getProperty
              name="PhotoURL"/>" /></p>
  <h3>Photos from the same place:</h3>
  <swehg:getLinks predicate="sameLocation"
                  listType="ul"/>
</body>
</html>
</swehg:template>
```

The tag attribute selector in the tag <swehg:template> tells the criterion for selecting *context resources* from the RDF repository. Each context resource will have an RPage of their own on the HTML level. The attribute value, here photo, is the name of a unary Prolog predicate called *selector* that should evaluate true for context resource URIs.

An example of a complete IPage template is given below using the view definitions above:

```
<swehg:template>
<html>
<body>
<h1>Building index</h1>
  <swehg:getView
    roots="buildings"
    branches="subclass"
    leaves="photoOf"
    orderby="order_alphabetically"
    listType="ul"/>
</body>
</html>
</swehg:template>
```

4   GENERATION PROCESS

The process for transforming an RDF(S) repository into HTML pages is defined by the algorithms 1 and 2. The input of the procedure is a set of HTML templates, and an RDF(S) repository. The output is an HTML page repository conforming to the templates. The transformation is based on a set of logical rules for selectors, properties, links, and views.

The process is based on generating pages using the HTML templates one after another. If a template is associated with a selector, then it is expanded into a set of RPages corresponding to the selected context resources, else it is expanded once without a reference to a context resource. In the latter case, the HPage and IPages are created. When generating an HTML page, the tags are expanded into HTML in the ways described in the previous section.

Algorithm: RDF2HTML

Data: Templates T, RDF(S) repository R

HTMLPageRepository H = empty;

foreach *Template t in T* **do**

  if *t has a selector rule S* **then**

    foreach *RDF Resource r in R* **do**

      if *S(r) == true* **then**

        h = createHTMLpage(r, t);

        add h to H;

      end

    end

  end

  else

    h = createHTMLpage(T);

    add h to H;

  end

end

Algorithm 1: Main procedure for the RDF to HTML transformation

---

Algorithm: createHTMLpage

Data: Template t, Context Resource r

Result: HTML page

String H = t;

foreach *Tag in H* **do**

  h = executeRule(Tag.rulename, r);

  replace Tag in H with h;

end

return H;

Algorithm 2: Algorithm createHTMLpage for rendering an HTML template. *Tag.rulename* returns the name of the rule, e.g., "getProperty".

Figure 3 depicts the architecture of our implementation. The main program is a Perl script which first builds an XSLT [8] template out of the HTML templates using the module "Template processor". This module also writes out a set of "Processing instructions" into a separate Prolog source code file. These instructions link template tags
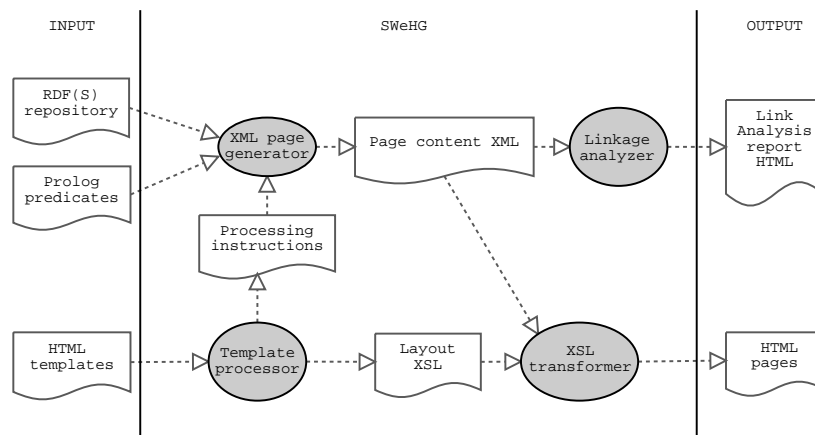
[8] http://w3.org/TR/xslt

Figure 3. Internal architecture of *SWeHG*.

with the Prolog predicates used in them as attribute values. The module "XML page generator" is a Prolog program that applies the predicates used in the HTML tags with respect to the RDF repository according to the Processing instructions. The result is a set of XML files describing the page contents. These XML files are then transformed using Apache Xalan[9] and with the help of the XSLT templates generated earlier into the final HTML pages.

The intermediate XML files in figure 3 are also used as a basis for the "Linkage analyzer" module that tries to identify the following potential problems: *Self loops* (a link that points to the page itself), *Bad links* (link pointing to a non existing page), *Dead ends* (an RPage with no outbound links), *No way in* (an RPage with no inbound links from any RPages or IPages), *Not in index* (an RPage with no inbound links from any IPage), and *Unused rules* (rules that are newer referred to when generating the HTML repository). The analysis results are represented as HTML pages. This helps the designer in debugging the specifications.

Figure 4. An analysis page created by *SWeHG*.

Figure 4 depicts a portion of the result from the analyzer. On this page the number of in-coming and out-going links can be seen for each RPage together with a status explanation. The analyzer has found out that the page with label "Aikaisempien yleisten ..." is not connected with any other page or index. Furthermore, the page

"Airueet" has one incoming and two outgoing links but was not included in any index. This kind of connectivity information is vital when debugging the logical rules that produce the HTML pages.

5 CASE APPLICATIONS

We applied *SWeHG* to a number of cultural RDF(S) repositories in order to test and evaluate its usability.

First, a virtual exhibition of a photo archive in the Helsinki University Museum[10] was generated. The archive contained 629 photographs about the promotion ceremonies of the University of Helsinki. The content of the archive was transformed into RDF(S) format in an other application project [7] and was used as it is by *SWeHG*. The domain knowledge consists of six ontologies with 329 promotion-related concept classes, such as "Person" and "Building", 125 properties, and 2890 instances, such as "Linus Torvalds" and the "Entrance of Cathedral of Helsinki".

In the photo annotation schema, the subject of a photograph is represented by a collection of ontology classes and individuals that appear on the image[11]. For example, if Linus Torvalds appears in a photo on a particular street, then the photo record is related *directly* with the corresponding person and street resources with a property corresponding to dc:subject. However, the relation between photos and subjects can be *indirect*, as well, involving traversal through several RDF arcs in the underlying knowledge base. For example, Linus Torvalds is present in a photograph as a Honorary Doctor. Then only an instance of such a role is associated with the image. The person instance in not directly linked with the image, but indirectly through the role instance. *SWeHG* predicate definition facility is very handy in hiding such annotation schema specific details from the HTML designer: the persons can be associated with images either directly or indirectly through roles. The criterion for association can be defined freely and conveniently by a declarative predicate.

Using *SWeHG* to publish the archive provides the end-users with two services. First, the photos can be found along the different orthogonal views based on the ontologies. Second, the hidden semantic

---

[9] xml.apache.org/xalan-j/

[10] http://www.helsinki.fi /museo/

[11] The annotations also include other metadata, such as the photographer, free text descriptions, some technical information of the images, etc.

associations between the photos can be made explicit to the user and can be used as the basis for browsing the photos, as in figure 1.

Second, we applied *SWeHG* to the RDF(S) repositories created for the semantic portal MuseumFinland [6]. Here RPages depict collection items from museum databases. Seven RDF(S) ontologies are used with some 10,000 classes and individuals and the metadata is described in terms of 38 properties. In this work, we could re-use the semantic recommendation predicates and the inference rule base developed for the original system, and an exhibition could be generated in a day or two.

6 DISCUSSION

6.1 Benefits and Limitations

Our initial experiences indicate that the presented RDF to HTML transformation method is feasible. HTML templates can be created fairly easily and can be adapted to different RDF repositories. The idea of using logic and Prolog for defining the semantics of the tags seems powerful. Complicated semantic link relations and views can be defined and modified easily thanks to the declarative nature of logic programming. In contrast to view-based search systems, such as [10, 5], the views are projected from the RDF(S) ontologies. The main benefit is that arbitrary mappings between view categories and data resources can be flexibly defined. The system infers the mapping between views and resources which gives it an "intelligent" flavor. Furthermore, the HTML pages are linked semantically with each other according to the ontologies, metadata, and rule base used. To the end-user, the underlying hidden associations between collection objects is a most interesting aspect of cultural collections. The nature of the associations can be explained to the user by the labels of the links.

SWeHG generates static pages in a batch process before publishing them on the web. This approach has the following benefits when compared with dynamic semantic portals: The page repository can be published easily by just copying it into a public HTML directory. SWeHG can be adapted to different contents conforming to different ontologies. The publication process is independent from semantic portal providers—no special server software is needed. The pages need no special maintenance. The static pages are indexed and searched for by general search engines. The pages can be viewed efficiently. Data security problems are minimal. The properties of the resulting HTML page set can be analyzed efficiently.

On the other hand, the static approach taken in SWeHG also has, of course, its limitations. First, static pages can not adapt their content dynamically to different user or patterns of usage. Second, dynamic systems can be connected more easily with other services providing additional functionality. Third, if the RDF repository, the rules, or the HTML templates change, the site has to be regenerated usually from scratch. Dynamic systems can adapt better to such changes. Fourth, if the RDF repository is large and many templates are used, then the number and size of generated pages can be large.

Clearly, both the dynamic and static approach have their own virtues and application possibilities.

6.2 Related Work

Logic and dynamic link creation on the semantic web have been discussed, e.g., in [4, 2]. Our approach is different in it's use of HTML templates and Prolog for describing the static HTML output.

In the *RDF Twig* tool[12] the RDF to HTML transformation is based on XSLT. A problem here is that an RDF graph can be serialized in many ways in XML. In *Spectacle*[13] the RDF to HTML transformation is based on APIs. Then the user must write programs that use the API, and also an application server is needed. In contrast, our approach is based on tags, is declarative, and the result is a set of static pages whose linkage structure is inferred by logical linking predicates.

6.3 Directions for Future Work

SWeHG is a research prototype. More work and testing is still needed in order to evaluate and enhance the usability and extendability of system in different applications. More work is also needed in optimizing the efficiency of the code and in providing better development tools for the HTML designer and system programmer using the system.

REFERENCES

[1] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):34–43, May 2001.

[2] P. Dolong, N. Henze, and W. Neijdl. Logic-based open hypermedia for the semantic web. In *Proceedings of the Int. Workshop on Hypermedia and the Semantic Web, Hypertext 2003 Conference, Nottinghan, UK*, 2003.

[3] D. Fensel, J. Hendler, H. Lieberman, and W. Wahlster, editors. *Weaving the Semantic Web*. The MIT Press, 2002.

[4] C. Goble, S. Bechhofer, L. Carr, D. De Roure, and W. Hall. Conceptual open hypermedia = the semantic web? In *Proceedings of the WWW2001, Semantic Web Workshop, Hongkong*, 2001.

[5] M. Hearst, A. Elliott, J. English, R. Sinha, K. Swearingen, and K.-P. Lee. Finding the flow in web site search. *CACM*, 45(9):42–49, 2002.

[6] E. Hyvönen, M. Junnila, S. Kettula, , E. Mäkelä, S. Saarela, M. Salminen, A. Syreeni, A. Valo, and K. Viljanen. MuseumFinland—Finnish Museums on the Semantic Web. User's perspective. In *Proceedings of Museums and the Web 2004 (MW2004), Arlington, Virginia, USA*, 2004. http://www.cs.helsinki.fi /u/eahyvone/publications/ MuseumFinland.pdf.

[7] E. Hyvönen, S. Saarela, and K. Viljanen. Ontogator: combining view- and ontology-based search with semantic browsing. In *Proceedings of the XML Finland 2003 conference. Kuopio, Finland*, 2003. http://www.cs.helsinki.fi /u/eahyvone/publications/ xmlfinland2003/yomXMLFinland2003.pdf.

[8] E. Hyvönen, M. Salminen, S. Kettula, and M. Junnila. A content creation process for the semantic web, 2004. Paper, submitted.

[9] E. Hyvönen, A. Valo, K. Viljanen, and M. Holi. Publishing semantic web content as semantically linked html pages. In *Proceedings of XML Finland 2003, Kuopio, Finland*, 2003. http://www.cs.helsinki.fi /u/eahyvone/publications/ xmlfinland2003/swehg_article_xmlfi 2003.pdf.

[10] A. S. Pollitt. The key role of classification and indexing in view-based searching. Technical report, University of Huddersfield, UK, 1998. http://www.ifla.org/IV/ifla63/63polst.pdf.

---

[12] http:/rdftwig.sourceforge.net/

[13] http://www.aidministrator.nl/spectacle/