Managing Web Content using Linked Data Principles – Combining semantic structure with dynamic content syndication

Norman Heino
Dept. of Computer Science, AKSW
Leipzig University
Leipzig, Germany
heino@informatik.uni-leipzig.de

Sebastian Tramp
Dept. of Computer Science, AKSW
Leipzig University
Leipzig, Germany
tramp@informatik.uni-leipzig.de

Sören Auer
Dept. of Computer Science, AKSW
Leipzig University
Leipzig, Germany
auer@informatik.uni-leipzig.de

***Abstract*—Despite the success of the emerging Linked Data Web, offering content in a machine-processable way and – at the same time – as a traditional website is still not a trivial task. In this paper, we present the *OntoWiki-CMS* – an extension to the collaborative knowledge engineering toolkit OntoWiki for managing semantically enriched Web content. OntoWiki-CMS is based on OntoWiki for the collaborative authoring of semantically enriched Web content, vocabularies and taxonomies for the semantic structuring of the Web content and the OntoWiki Site Extension, a template and dynamic syndication system for representing the semantically enriched content as a Web site and the dynamic integration of supplementary content. OntoWiki-CMS facilitates and integrates existing content-specific content management strategies (such as blogs, bibliographic repositories or social networks). OntoWiki-CMS helps to balance between the creation of rich, stable semantic structures and the participatory involvement of a potentially large editor and contributor community. As a result semantic structuring of the Web content facilitates better search, browsing and exploration as we demonstrate with a use case.**

***Keywords*-Linked Data; Semantic Web; Semantic Wiki; Content Management**

I. Introduction

The rapidly emerging Linked Data Web enables machine-interpretable data to be published and interlinked on the Web. However, offering the same content in a machine-processable way and – at the same time – as a traditional website for consumption by human readers is not a trivial task. In order to solve this problem strategies for the management of semantically enriched Web content have to be developed. In this paper, we present the *OntoWiki-CMS* – an extension to the collaborative knowledge engineering toolkit OntoWiki for managing semantically enriched Web content. OntoWiki-CMS is based on three components:

- *OntoWiki* for the collaborative authoring of semantically enriched Web content.
- *Vocabularies and taxonomies* for the semantic structuring of the Web content.
- *OntoWiki Site Extension* – a template and dynamic syndication system for representing the semantically enriched content as a Web site and the dynamic integration of supplementary content.

The rationale behind OntoWiki-CMS is that a one-size fits all solution can hardly be the most efficient and effective one for all different content types. Hence, the design goal of OntoWiki-CMS was to facilitate and integrate existing content-specific content management strategies. In particular for news, bibliographic as well as personal information we developed syndication strategies, which enable users to manage such content types with specialized and better adapted tools – i. e. *webblogs* or *microblogging systems* for news, *BibSonomy* for bibliographic information and *social networks* for personal information.

By leveraging the comprehensive OntoWiki application framework and its underlying Erfurt API, OntoWiki-CMS comprises a large number of different interfaces for human access (e. g. faceted-browsing, query builder, HTML/RDFa, forms) and machine access (RDFa, Linked Data, SPARQL), a pattern-based knowledge base evolution engine, a comprehensive set of editing widgets for different content types and many other features.

The adaptive knowledge engineering methodology implemented by OntoWiki helps to balance between the creation of rich, stable semantic structures and the participatory involvement of a potentially large editor and contributor community. The semantic structuring of the Web content facilitates better search, browsing and exploration. In addition, the semantic structuring helps to blur the classic and mostly artificial distinction between editorial and data content. OntoWiki-CMS was developed for the Web site of the EU-funded research project "LOD2 – Creating Knowledge out of Interlinked Data" (available at http://lod2.eu) and is evaluated in this use case. The content on the website is managed by approx. 30 people from the ten LOD2 consortium member organizations. The LOD2 website is visited more than 3000 times each month.

Although there are already some prior approaches to manage Web content semantically, to the best of our knowledge OntoWiki-CMS is the first one, which is directly based on the RDF data model and at the same time seamlessly

syndicates with other semantic or other structured content sources.

The paper is structured as follows: We describe the overall architecture and the building blocks for managing semantic content in Section II. We outline the vocabularies and taxonomy structures used to semantically organize Web content in Section III. In Section IV we present the strategy for syndicating and integrating dynamic content such as blog posts, bibliographic data, Twitter feeds etc. We showcase the LOD2 website application scenario in Section V, where our strategy for semantic content management was successfully applied. We review related work in Section VI and conclude with an outlook on future work in Section VII.

II. Architecture and Building Blocks

In this section we describe the basic architecture and the components of our implementation. We chose OntoWiki as a basis since it already implements a number of features required for semantics-based web content management, such as serving Linked Data-enabled content in a resource-centric way. We leverage OntoWiki's extension system to enable the classic separation between frontend and backend usually found in web content management systems. A number of different components that were used, particularly in building the front-end, will be described in the following subsections.

A. Overall OntoWiki-CMS Architecture

The overall architecture of OntoWiki-CMS is depicted in Figure 1. It is built on the semantic data wiki OntoWiki and the underlying Erfurt Framework, which uses either MySQL or Virtuoso as a RDF storage backend (center of Figure 1). Since RDF is a first-class citizen in OntoWiki, standard interfaces such as Linked Data, RDFa and a SPARQL endpoint are supported out of the box (left part of Figure 1). Content is organized according to a site vocabulary and a content taxonomy, which can be both easily managed using OntoWiki (bottom of Figure 1). The OntoWiki Site Extension is used to generate specifically designed Web pages from the semantic content and to integrate additional dynamic content syndicated from third party applications (upper right of Figure 1). In the remainder of this section, we describe the most crucial components of OntoWiki-CMS in more detail.

B. OntoWiki

OntoWiki [2] is an RDF-based data wiki that enables resource and set-based semantic browsing and authoring scenarios. It makes no assumptions on the data model and can thus be used with any RDF knowledge base. For managing Web content, we developed a core ontology (see Section III), a `skos`-based taxonomy for navigation and populated both with instance data representing the Web site content and metadata. OntoWiki provides a configureable navigation hierarchy that can be used to display `skos` hierarchies (see Figure 4). In addition, it has search capabilities and leverages resource interlinks in order to provide different paths for browsing knowledge bases. Being a wiki, it also fosters versioning of changes to a resource, discussion and editing, which are described below.

Semantic Authoring with RDFauthor: The OntoWiki backend automatically creates pages annotated with RDFa. For editing content, these builtin semantic annotations are leveraged to automatically create an editing form (depicted in Figure 4). The system in use here has been made available separately as *RDFauthor* [17]. To this end, it incorporates several technologies:

- semantics-aware widgets that support the user while editing content by automatically suggesting resources to link to based on queries to the local knowledge store and Sindice[1].
- Updates are sent back the the RDF store via SPARQL/Update – an update extension to the current specification currently in standardization.

The described editing system was also incorporated into the frontend by simply adding RDFa to the templates. Thus, a user with editing privileges on the LOD2 instance graph can make changes to the website right from the frontend.

Extensibility: OntoWiki started as an RDF-based data wiki with emphasis on collaboration but has meanwhile evolved into a comprehensive framework for developing Semantic Web applications [7]. This involved not only the development of a sophisticated extension interface allowing for a wide range of customizations but also the addition of several access and consumption interfaces allowing OntoWiki installations to play both a provider and a consumer role in the emerging Web of Data.

Evolution: The loosely typed data model of RDF encourages continuous evolution and refinement of knowledge bases. With *EvoPat*, OntoWiki supports this in a declarative, pattern-based manner [13]. Such basic evolution patterns consist of three components:

- a set of variables,
- a SPARQL select query selecting a number of resources under evolution,
- a SPARQL/Update query template that is executed for each resulting resource of the select query.

In addition, basic patterns can be combined to form compound patterns – suitable for more complex evolution scenarios. In order to facilitate the semi-automatic application of evolution patterns, *bad smells* can be defined that serve as a detection mechanism for ontology design anti-patterns or data modeling problems. If certain conditions are met, this process is even fully automatable.

[1] http://sindice.com/

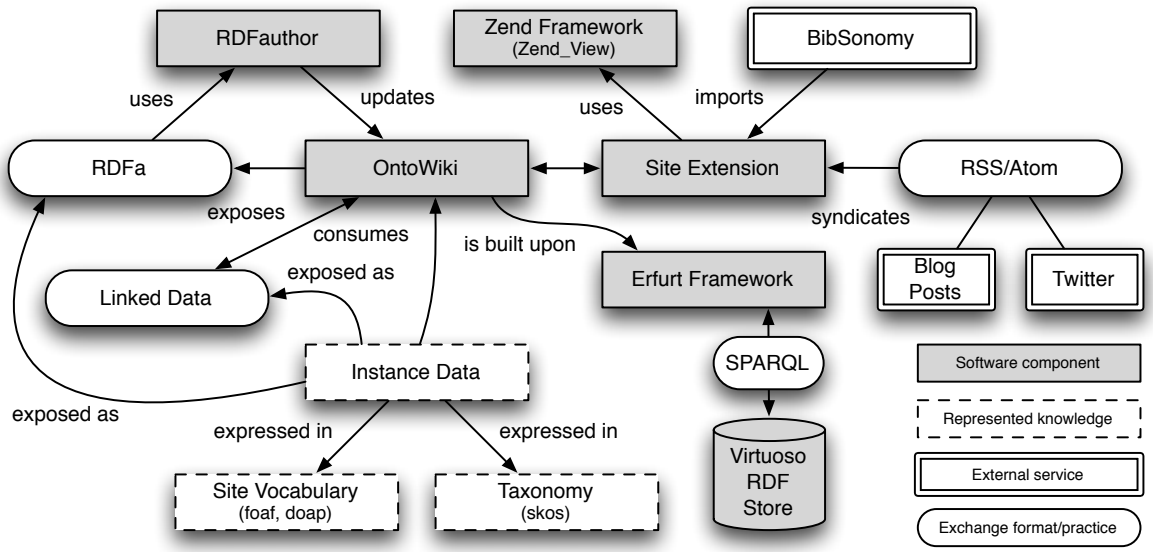


Figure 1: OntoWiki-CMS overall architecture and building blocks.

C. Access Interfaces

In addition to human-targeted graphical user interfaces, OntoWiki supports a number of machine-accessible data interfaces. These are based on established Semantic Web standards like SPARQL or accepted best practices like publication and consumption of Linked Data.

SPARQL Endpoint: The SPARQL recommendation not only defines a query language for RDF but also a protocol for sending queries to and receiving results from remote endpoints[2]. OntoWiki implements this specification, allowing all resources managed in an OntoWiki be queried over the Web. In fact, the aforementioned RDFauthor authoring interface makes use of SPARQL to query for additional schema-related information, treating OntoWiki as a remote endpoint in that case.

Linked Data: Each OntoWiki installation can be part of the emerging Linked Data Web. According to accepted publication principles[3], OntoWiki makes all resources accessible by its URI (provided, the resource's URI is in the same namespace as the OntoWiki instance). Furthermore, for each resource used in OntoWiki additional triples can be fetches if the resource is dereferenceable.

Semantic Pingback: Pingback is an established notification system that gained wide popularity in the blogsphere. With Semantic Pingback [16], OntoWiki adapts this idea to Linked Data providing a *notification mechanism* for resource usage. If a Pingback-enabled resource is mentioned (i. e. linked to) by another party, its pingback server is notified of the usage. Provided, the Semantic Pingback extension is enabled all resources used in OntoWiki are pinged automatically and all resources defined in OntoWiki are Pingback-enabled.

D. Exploration Interfaces

For exploring semantic content, OntoWiki provides several exploration interfaces that range from generic views over search interfaces to sophisticated querying capabilities for more RDF-knowledgable users. The subsequent paragraphs give an overview of each of them.

Knowledge base as an information map: The compromise of, on the one hand, providing a generic user interface for arbitrary RDF knowledge bases and, on the other hand, aiming at being as intuitive as possible is tackled by regarding knowledge bases as *information maps*. Each node at the information map, i. e. RDF resource, is represented as a Web accessible page and interlinked to related digital resources. These Web pages representing nodes in the information map are divided into three parts: a left sidebar, a main content section and a right sidebar. The left sidebar offers the selection of content to display in the main content section. Selection opportunities include the set of available knowledge bases, a hierarchical browser and a full-text search.

Full-text search: The full-text search makes use of special indexes (mapped to proprietary SPARQL syntax) if the underlying knowledge store provides this feature, else, plain SPARQL string matching is used. In both cases, the resulting SPARQL query is stored as an object which can later be modified (e. g. have its filter clauses refined). Thus, full-text search is seamlessly integrated with facet-based browsing (see below).

[2]http://www.w3.org/TR/rdf-sparql-protocol/

[3]http://sites.wiwiss.fu-berlin.de/suhl/bizer/pub/LinkedDataTutorial/

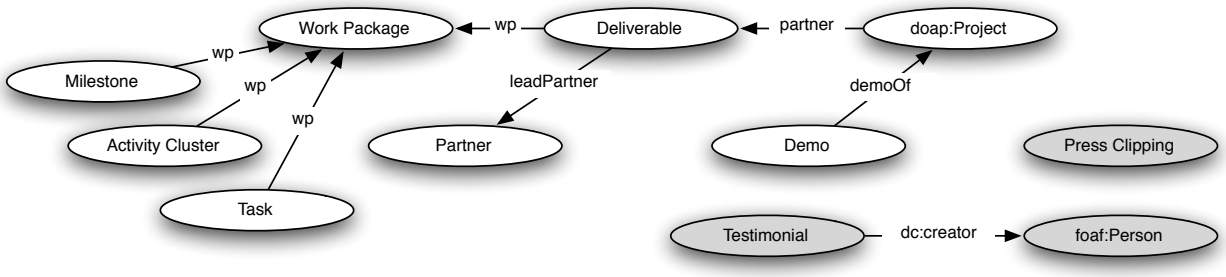


Figure 2: LOD2 website ontology schema; all non-prefixed elements are part of the `lod2` namespace.

Content specific browsing interfaces: For domain-specific use cases, OntoWiki provides an easy-to-use extension interface that enables the integration of custom components. By providing such a custom view, it is even possible to hide completely the fact that an RDF knowledge base is worked on. This permits OntoWiki to be used as a data-entry frontend for users with a less profound knowledge of Semantic Web technologies.

Faceted-browsing: Via its facet-based browsing, OntoWiki allows the construction of complex concept definitions, with a pre-defined class as a starting point by means of property value restrictions. These two views are sufficient for browsing and editing all information contained in a knowledge base in a generic way.

Query builder: OntoWiki serves as a SPARQL endpoint, however, it quickly turned out that formulating SPARQL queries is too tedious for end users. In order to simplify the creation of queries, we developed the *Visual Query Builder*[4] (VQB) as an OntoWiki extension, which is implemented in JavaScript and communicates with the triple store using the SPARQL language and protocol. VQB allows to visually create queries to the stored knowledge base and supports domain experts with an intuitive visual representation of query and data. Developed queries can be stored and added via drag-and-drop to the current query. This enables the reuse of existing queries as building blocks for more complex ones.

E. Site Extension

The OntoWiki Site Extension is the central part of OntoWiki-CMS, since it extends OntoWiki with the functionality to render Web pages based on the semantic representations and additional external content. OntoWiki's extension system allows for different types of extensions depending on the functionality to be implemented [7]. We chose to implement the site extension as a *component*, since this type allows for custom URI schemes to be handled.

Upon a requested URI, the following steps are performed by the Site Extension:

1) Determine whether a resource with the requested URI exists in the underlying RDF knowledge store.
2) Redirect to an information resource according to the *mime type* requested.
3) Load the Concise Bounded Description (CBD, [15]) of the resource to inject it into the main layout template.
4) Depending on the `rdf:type` of the resource, load a content template or use a default content template.
5) If the resource is linked to a SPARQL query, execute that query and inject its result list into the SPARQL query results template. For each item of the query result list, use a template as determined by the query resource.

Zend View Template System: In order to support modularized rendering of resources triples into HTML pages, the Zend Framework's template system[5] is used. The template to be used for rendering a certain resource is determined based on it's `rdf:type` property. Additional properties (e. g. `lod2:query`) can be linked with templates and are included in the resource page.

Serving Linked Data Efficiently: The proposed way of handling a request of a non-information resource (i. e. a resource used as an *identifiers for a concept* as is common on the Semantic Web) is to determine the format the user wants based on the `Accept` HTTP header and then redirect to an information resources that represents that format [4]. This results in each resource being served by at least two requests and not only results in unnecessarily high server load but also increased perceived response times. We therefore decided on a slightly different model were we assume that the requested format does not change during a browsing session (i. e. the requested format is `text/html`). We evaluate the `Accept` header in the first request and redirect to an appropriate information resource. The URI of this information resource is determined by appending a faux file extension to the original resource URI. All internal links on the created HTML page, however, directly point to information resources for HTML. Consider for example a request to `http://lod2.eu/Welcome`

[4]http://aksw.org/Projects/OntoWiki/Extension/VQB

[5]http://zendframework.com/manual/en/zend.view.html

which – carried out in a Web browser – results in `http://lod2.eu/Welcome.html` being served. All links from that page will directly lead to `.html` versions of the resource with no redirects in between. The explicit representation of the format is also more transparent to the user. If a different format is desired, she can request it by just changing the faux file extension in her browser's address bar.

III. Stable, semantic structure for Web content

In this section, we describe the OntoWiki-CMS ontology, which comprises a domain-specific vocabulary for representing the Web content and a taxonomy for rendering the Web site's navigation. To make accidental changes to the ontology less likely, we separate schema from instance triples by storing them in two different graphs. The schema graph is then included via an `owl:imports` property that is automatically evaluated by OntoWiki. In the sequel we describe the individual elements of the CMS ontology in some more detail.

A. Website vocabulary

The rationale of OntoWiki-CMS is to use a usage scenario (i. e. Web site) specific ontology for representing the content. The ontology consists of classes whose instances will also be represented as articles on the generated Web site. Object properties connect different instances in this ontology and they are rendered as HTML links in the respective Web pages. Datatype properties describe the instances semantically and are employed to adapt the presentation of the article.

An example of the website ontology for the LOD2 project website is depicted in Figure 2. It consists of classes and properties imported from `foaf`[6] and `doap`[7] vocabularies as well as resources created specifically for LOD2. The modeled domain is the space of research project descriptions with website-specific enrichments for representing demos, feeds, etc. The schema vocabulary is available from the OntoWiki Google Code repository[8] as well as via Linked Data..

B. Navigation Taxonomy

An excerpt from the navigation taxonomy and description vocabulary is shown in Figure 3. It models the navigation hierarchy as an instance of a `skos:ConceptScheme` with the top level menu entries being a `skos:topConceptOf` of that scheme. Lower-level menu entries are linked to the upper-level entry via the `skos:broader` property.

In addition to these SKOS-based hierarchical structures, we wanted to support a more dense interlinking of our resources by adding auxiliary *next item* and *previous item* links between resources[9]. This type of navigation is well known from weblogs and users like to click through all items of a certain type, e. g. through all partners or workpackages. Since SKOS does not support such links, we added these properties to our domain-ontology (but believe that these property resources should be collected in more generic schema).

Some menu navigation hierarchy concepts are instance of the `sioc:WikiArticle` class, which means landing pages for concepts can contain a rich-text description of the particular concept – often an instance (e. g. "Project") or a list thereof (e. g. "Deliverables"). To allow these rich-text descriptions for all LOD2 concepts, we added the `lod2:content` datatype property to our schema model. This was needed since the SIOC namespaces does not consist of a property which allow content with markup[10] `lod2:content` is a sub property of the RSS 1.0 content module property `encoded`, which allows any kind of markup in the literal string. Based on this datatype property, our RDFauthor WYSIWYG editor widget was configured to accept these statement for editing (see Figure 4).

C. Instance Data

The LOD2 instance data is populated with the description of the formal project structure (5 activity cluster, 12 workpackages with 121 deliverables and 25 milestones) as well as descriptions for persons, software, demonstrations, press clippings and user testimonials. All in all instance data of the LOD2 website currently comprises 324 resources with an average count of 5 statements per resource. Some of these resources are static content which will not or only rarely change during the project. Some resources are changed frequently every month (e. g. mostly articles about the LOD2 software stack) and some classes will be populated with more and more instances during the project (e. g. press clippings).

For the latter category, we utilize an OntoWiki plugin which adjusts the resource URI of a newly created instance according to a specified URI generation scheme. This is essential since most users do not care about a proper URI design. The *resource URI creation* plugin watches for newly created resources and uses the content of the created new triples to decide which URI should be used for identifying this resource. To do this properly, the plugin needs to gather more data from the knowledge base (e. g. to ask for a naming property of a class or the make sure that a URI candidate is not already in use).

IV. Flexible, dynamic content syndication

Relevant content for LOD2 is not entirely kept within the system. Either because channels existed prior to creation

[6] http://xmlns.com/foaf/spec/

[7] http://usefulinc.com/ns/doap

[8] http://ontowiki.googlecode.com/hg/extensions2/site/sites/lod2/model/lod2-schema.n3

[9] Identified as lod2:next and lod2:previous.

[10] The usage of `sioc:content_encoded` is deprecated.

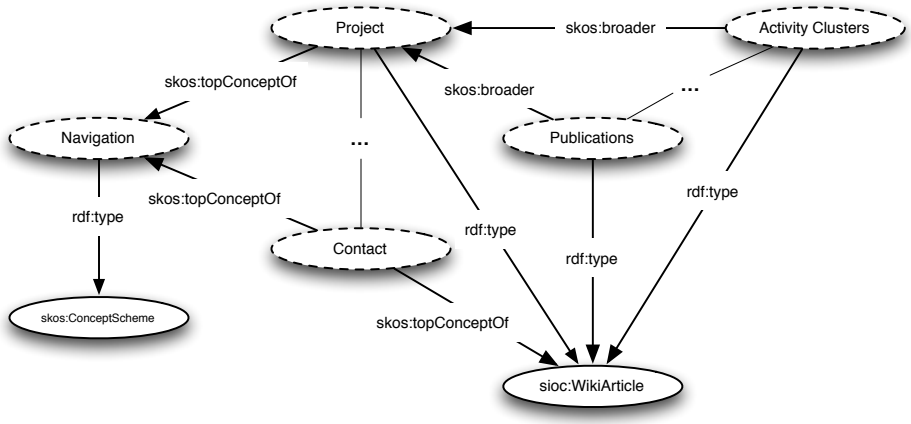


Figure 3: LOD2 website navigation hierarchy.

of the website or because external services provide better integration or established representation standards. Such systems include

- RSS news feeds from weblogs and Twitter,
- Slideshare presentations,
- scientific publication data from BibSonomy[11], and
- information about LOD2 protagonists from their FOAF/WebID resources.

Resources are linked to such sources via special properties that are recognized by the system. Content from these sources is automatically fetched and aggregated on the server upon page request. Thus, it is possible to sort and limit the number of displayed items globally (across all project resources). Since the frontend is aware of these special properties, a content author can decide to include different external sources in the public HTML representation of a resource. In the Linked Data representation, these external sources are only visible as object property links.

A. Integrating news

To integrate articles and short message news from the web into the LOD2 portal we utilize the `sioc:feed` object property from the Semantically Interlinked Online Communities (SIOC, [5]) ontology. This property is designed in an open way, which means in particular, that the domain is not solely restricted to SIOC classes. Therefore, we can relate any LOD2 instance to one or more feed resources and integrate their content in the HTML view of these resources.

After querying for a feed relation, the feed content of all related feeds is fetched and cached by the Zend Framework subsystem. The gathered feed items are merged by using the item URL as an ID and sorted by publication time. After that, the n newest entries are presented in the sidebar of the resource's HTML representation. By using the Zend Framework RSS API, we make sure that feed documents are not fetched more often than required.

[11] http://www.BibSonomy.org/

```
@base <http://lod2.eu/>.
@prefix rdfs: <http://www.w3.org.../rdf-schema#>.
@prefix doap: <http://usefulinc.com/ns/doap#>.
@prefix skos: <http://www.w3.org.../skos/core#>.
@prefix sioc: <http://rdfs.org/sioc/ns#>.
@prefix lod2: <http://lod2.eu/schema/>.
@prefix ping: <http://purl.org/net/pingback/>.

<Project/OntoWiki> a doap:Project;
    rdfs:label "OntoWiki";
    skos:broader <WikiArticle/TechnologyStack>;
    sioc:feed <http://blog.aksw.org/feed/?cat=5>;
    doap:homepage <http://ontowiki.net>;
    lod2:abstract "...";
    lod2:content "...";
    lod2:partner <Partner/ulei>;
    ping:to <pingback/ping/>.
```

Listing 1: Representation of an example `doap:Project` resource

B. Integrating bibliographic information

Scientific publications are one of the major results of the work in an large-scale research project such as LOD2. The presentation of these publications is of particular importance for the project and its partners. This is particularly challenging in projects with many partners from different research groups.

Since most researchers like to be independent in terms of their publications and are mostly active in more than one project, we decided to shoulder the publication management task with the help of a specialized service. BibSonomy [3] is a system for collaborative authoring and sharing of bibliographic information. Registered BibSonomy users can be group members and share items with these groups.

We created such a group and invited all LOD2 project

members to this group and share their own and possible related publications. We agreed on a tagging policy where a special tag (`lod2page`) is attached to a publication if it is a result of the project and should be accordingly presented on the website. Thus, relevant publications can be queried and fetched from BibSonomy via its JSON REST service. The gathered JSON data is fed into the Exhibit [9] application, which was integrated into the LOD2 website. This is realized by creating an LOD2 object property which relates a resource to an exhibit JSON feed. The frontend searches for such a relation and integrates the client-side Exhibit application accordingly.

C. Integrating personal information

Since we all suffer from the fact that we have to copy & paste our personal information to every social site where we want to be active, we decided to integrate personal information from the FOAF/WebID based social network wherever possible. Since OntoWiki is able to gather RDF resources via Linked Data, this integration was easy to achieve. Instead of creating URIs in the LOD2 namespace for every person, we used the WebID resources of our colleagues, where applicable. The data gathering framework of OntoWiki fetches all statements from the WebID and adds them to the LOD2 knowledge base. This can be repeated as needed thus effectively establishing a synchronization between the federated LOD2 knowledge base and individual WebIDs.

V. LOD2 USE CASE

OntoWiki-CMS was developed for the website of the multinational research project LOD2, which is co-financed by the European Union within its 7th Framework Programme. A screenshot of the public LOD2 website which was realized completely on OntoWiki-CMS and which is available at http://lod2.eu is depicted in Figure 4. It shows the Welcome page incorporating three feed items from the LOD2 Wordpress blog, the testimonial section, the top part of the newsfeed section and the beginning of an article. The news feed on that page is aggregated from all news feeds interlinked from partner and software project articles. The menu in the top of the page is generated from website navigation taxonomy represented in SKOS (as depicted in Figure 3). Figure 4 shows the corresponding OntoWiki form for the instance named `Welcome` of the `sioc:WikiArticle` class. The main article content (as shown in the lower part of Figure 4) is represented as an RDF literal attached to `lod2:content` property. The property `lod2:importantFeed` instructs the OntoWiki Site extension to integrate blog posts from the main LOD2 Blog into the page.

Performance Considerations: One of the problems faced when implementing Semantic Web technology is the usually poor performance of RDF data storage relative to relational data storage. This is mainly due to higher retrieval costs for the flexible data model provided by RDF. We therefore deployed an extended version of our caching solution described in [11]. The extension caches complex objects used within the application that may be constructed using several SPARQL queries and is therefore called *Object Cache*. In Table IV we give results obtained using the Apache benchmark tool `ab`[12] by requesting two representative pages: a single resource page (`/Welcome.html`) and a list of all the technology components in the project, generated by executing a SPARQL query (`/TechnologyStack.html`). Column three and four show the average request processing time taken by the server during 10 consecutive requests and the standard

[12]http://httpd.apache.org/docs/2.0/programs/ab.html

Number of pages (i. e. resources)	281
Number of classes	16
Number of properties	48
Number of feeds linked	18
Registered users	22
Edits made	3655

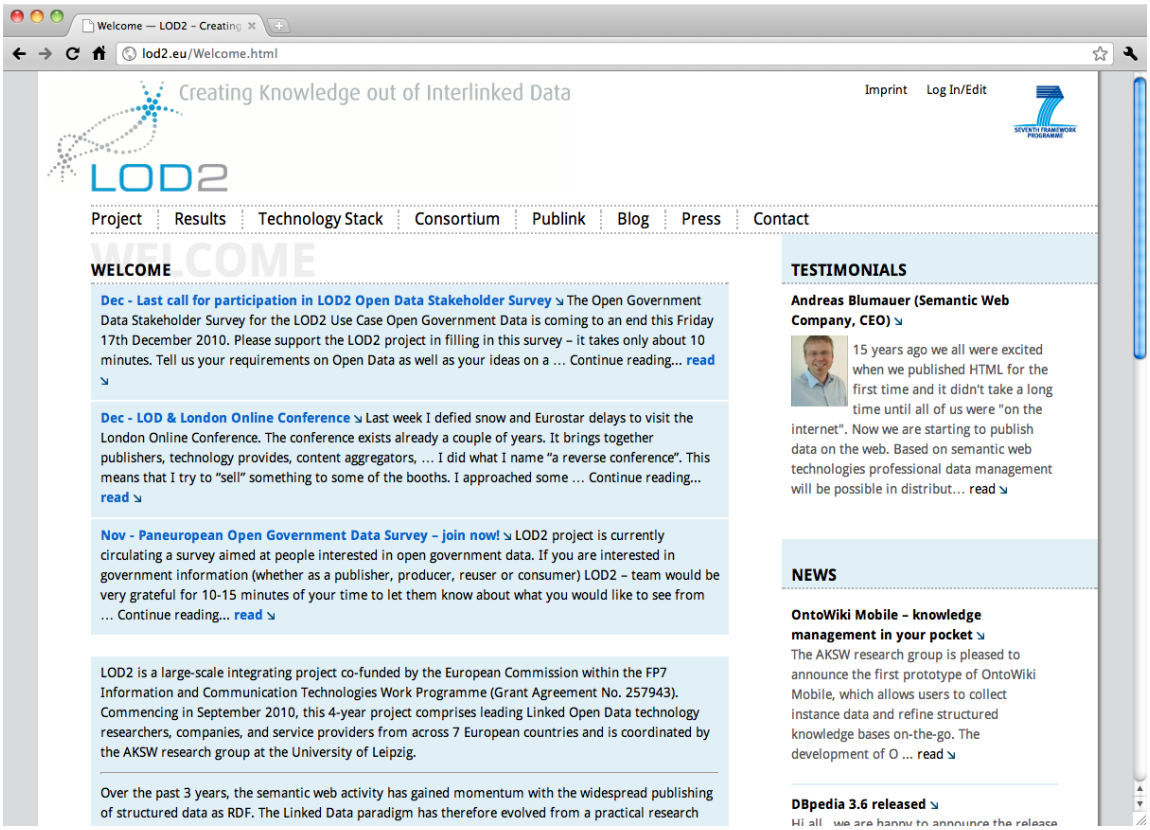
Table I: Some statistics about the LOD2 project website.

Count	Class	Count	Class
1	skos:ConceptScheme	3	lod2:Demo
5	lod2:ActivityCluster	9	ow:SparqlQuery
10	lod2:Partner	12	lod2:WorkPackage
14	lod2:Testimonial	15	doap:Project
15	lod2:PressClipping	16	foaf:Person
17	sioc:WikiArticle	18	(burst:Publication)
25	Milestone	121	lod2:Deliverable

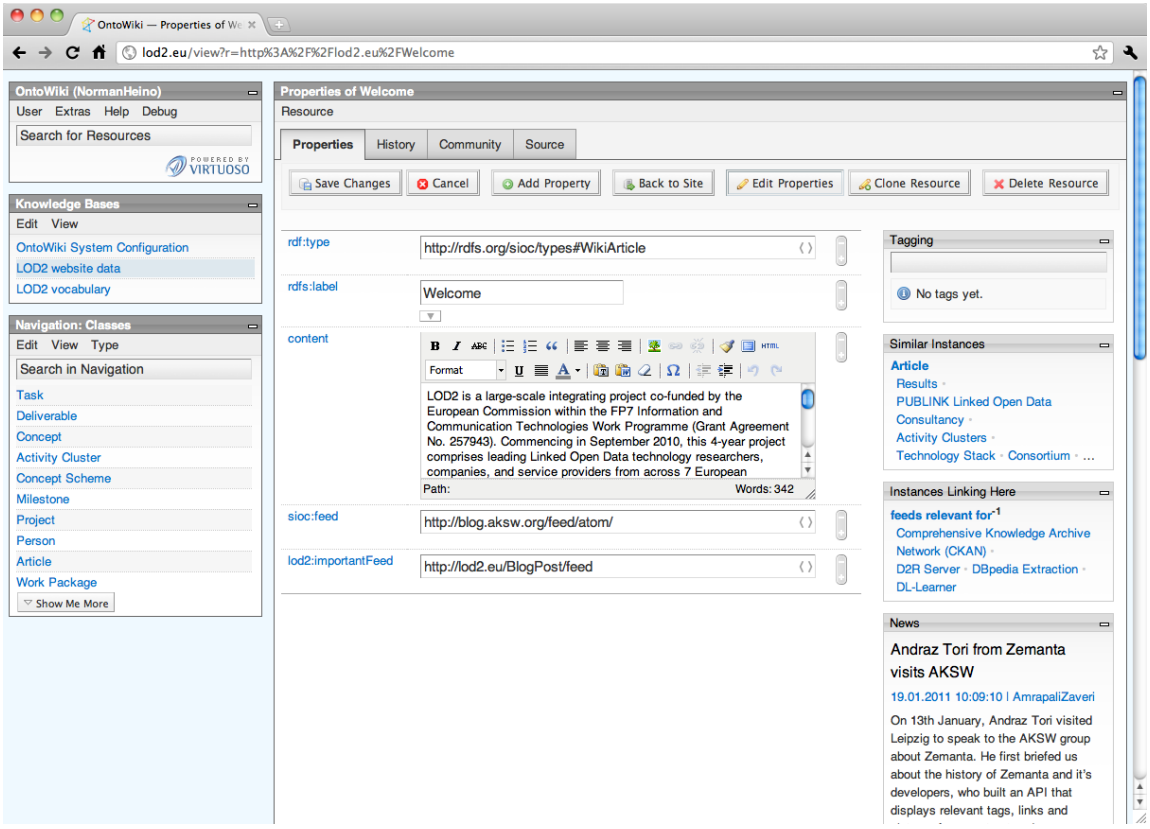
Table II: Counted classes

Count	Property	Count	Property
25	lod2:abstract	94	lod2:content
1	lod2:contentRaw	1	lod2:contentSideTop
15	lod2:date	26	lod2:deadline
121	lod2:deliverableNature	122	lod2:deliveryDate
3	lod2:demoOf	121	lod2:disseminationLevel
12	lod2:endMonth	1	lod2:exhibitData
4	lod2:feedRelevantFor	1	lod2:importantFeed
15	lod2:lang	134	lod2:leadPartner
12	lod2:nameAffix	24	lod2:next
117	lod2:partner	24	lod2:previous
15	lod2:publishedBy	7	lod2:query
12	lod2:startMonth	159	lod2:wp
3	ow:Model	9	ow:generator
35	ow:order	10	ow:sparql_code
8	dc:created	14	dc:creator
10	dc:title	1	sioc:about
1	sioc:content	18	sioc:feed
1	sioc:has_creator	15	doap:homepage
241	rdfs:label	22	rdfs:seeAlso
2	owl:imports	2	skos:altLabel
52	skos:broader	1	skos:note
8	skos:topConceptOf	10	foaf:based_near
28	foaf:depiction	11	foaf:homepage
16	foaf:name	4	foaf:title

Table III: Counted properties, ordered by namespace



(a) frontend screenshot



(b) backend screenshot

Figure 4: LOD2 website screenshots.

deviations, respectively.

Page	Cached	Time (ms)	SD
/Welcome.html	yes	631	99.0
/TechnologyStack.hmtl	yes	583	82.9
/Welcome.html	no	15664	221.8
/TechnologyStack.hmtl	no	1597	290.0

Table IV: Request throughput with and without object caching performed using `ab`.

VI. Related work

The notion of a web portal supported by semantic technologies has already been discussed a few times in literature [14], [10]. In [8], the authors describe an implementation built on *Semantic MediaWiki*. Their solution is based on a conventional wiki engine for the main contents of web pages with added semantic annotations to support more sophisticated search and querying. In arguing that a pre-defined ontology hinders agile content creation the favor a mixture of free text and form-based input. In our approach, we come to the same conclusion but form a different starting point. OntoWiki allows pure form-based input supported by a pre-defined vocabulary. For human consumption we designate one property (`lod:content`) to hold HTML-encoded descriptions which can be edited using a WYSIWYG editor. Linked Data publication was also not of concern for that work.

OntoWiki, which forms the basis of our approach is introduced in [2]. Its architecture as well as its extension system – without which the work addressed in this paper would not have been possible – are are described in [7].

Drupal[13], which is one of the top open-source content management systems, recently gained out-of-the-box support for semantic web standards [6]. Each content item (*node* in Drupal) can be given a type which in turn determines its properties (*fields* in Drupal). These fields can then be mapped to and RDF vocabulary and are exposed as RDFa. OntoWiki in contrast, does not require a mapping, since an RDF vocabulary is used directly.

The SIOC vocabulary for describing online communities is detailed [12]. To facilitate interlinking, we import some of its properties into our own vocabulary. Triplify [1] is a small script, which is easy to configure and allows to equip relational database backed Web applications with a Linked Data interface reusing existing vocabularies such as SIOC.

VII. Conclusions and future work

With the increasing maturation of semantic technologies the facilitation of semantics-based content management became a crucial requirement. In this article we presented a strategy for managing semantic content based on the semantic wiki paradigm.

With regard to future work we see in particular the following directions:

Integration of automatic linking techniques: Establishing and maintaining links on the Web of Data is still a major challenge. With regard to multi-media data we envision the realization, of a linking dashboard, with pluggable linking services, which particularly facilitate the linking of local multimedia assets based on the extracted meta-data with resources available on the Web of Data.

Fine-grained provenance tracking: Already now basic provenance information such as the editor of a certain translation or multimedia annotation is tracked by OntoWiki. However, we envision a more fine-grained representation, which includes information about the employed tools (such as multimedia metadata extraction, automatic translation etc.) in order to facilitate future revisions (e. g. based on new tool releases etc.)

Facilitation of adaptive previews: A crucial component of multimodal information systems are previews of relevant (parts of the) multimedia assets. We aim at integrating preview facilities, which take the users's context (such as locality, search and exploration history, interests etc.) into account.

[13] http://drupal.org/

Acknowledgments

We would like to thank the LOD2 consortium (http://lod2.eu) for the helpful comments and inspiring discussions during the work described in this article. The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 257943.

References

[1] S. Auer, S. Dietzold, J. Lehmann, S. Hellmann, and D. Aumueller. Triplify: Light-weight linked data publication from relational databases. In *WWW2009*. ACM, 2009.

[2] S. Auer, S. Dietzold, and T. Riechert. OntoWiki – A Tool for Social, Semantic Collaboration. In *ISWC2006*, volume 4273 of *LNCS*. Springer, 2006.

[3] D. Benz, A. Hotho, R. Jäschke, B. Krause, F. Mitzlaff, C. Schmitz, and G. Stumme. The social bookmark and publication management system bibsonomy - A platform for evaluating and demonstrating web 2.0 research. *VLDB J.*, 19(6), 2010.

[4] T. Berners-Lee. Linked Data – Design Issues, 2006. Retrieved 2011-02-03, http://www.w3.org/DesignIssues/LinkedData.html.

[5] J.G. Breslin, S. Decker, A. Harth, and U. Bojars. SIOC: An Approach to Connect Web-Based Communities. *The International Journal of Web-Based Communities*, 2(2), 2006.

[6] S. Corlosquet, R. Delbru, T. Clark, and A. Polleres S. Decker. Produce and Consume Linked Data with Drupal! In *ISWC2009*, volume 5823. Springer, 2009.

[7] Norman Heino, Sebastian Dietzold, Michael Martin, and Sören Auer. Developing Semantic Web Applications with the Ontowiki Framework. In *Networked Knowledge – Networked Media*, volume 221 of *Studies in Computational Intelligence*. Springer, 2009.

[8] D. Herzig and B. Ell. Semantic MediaWiki in Operation: Experiences with Building a Semantic Portal. In *ISWC2010*, volume 6497 of *LNCS*. Springer, 2010.

[9] D. F. Huynh, D. R. Karger, and R. C. Miller. Exhibit: lightweight structured data publishing. In *WWW2007*. ACM, 2007.

[10] A. Maedche, S. Staab, N. Stojanovic, R. Studer, and Y. Sure. SEmantic portAL: The SEAL Approach. In *Spinning the Semantic Web*. MIT Press, 2003.

[11] M. Martin, J. Unbehauen, and S. Auer. Improving the Performance of Semantic Web Applications with SPARQL Query Caching. In *ESWC2010*, volume 6089 of *LNCS*. Springer, 2010.

[12] A. Passant, U. Bojars, J. Breslin, and S. Decker. The SIOC Project: Semantically-Interlinked Online Communities, from Humans to Machines. In *Coordination, Organizations, Institutions and Norms in Agent Systems V*, volume 6069 of *LNCS*. Springer, 2010.

[13] C. Rieß, N. Heino, S. Tramp, and S. Auer. EvoPat – Pattern-Based Evolution and Refactoring of RDF Knowledge Bases. In *ISWC2010*, LNCS. Springer, 2010.

[14] L. Rovan. Realizing semantic web portal using available semantic web technologies and tools. In *ISWC2008 Posters & Demos*, volume 401 of *CEUR Workshop Proceedings*, 2008.

[15] P. Stickler. CBD – Concise Bounded Description. W3c:sub, W3C, 30 2004. http://www.w3.org/Submission/2004/SUBM-CBD-20040930/.

[16] S. Tramp, P. Frischmuth, T. Ermilov, and S. Auer. Weaving a Social Data Web with Semantic Pingback. In *EKAW2010*, volume 6317 of *LNAI*. Springer, 2010.

[17] S. Tramp, N. Heino, S. Auer, and P. Frischmuth. RDFauthor: Employing RDFa for collaborative Knowledge Engineering. In *EKAW2010*, volume 6317 of *LNAI*. Springer, 2010.