

Building Environmental Semantic Web Applications with Drupal

Denis Havlik

▶ **To cite this version:**

Denis Havlik. Building Environmental Semantic Web Applications with Drupal. 9th International Symposium on Environmental Software Systems (ISESS), Jun 2011, Brno, Czech Republic. pp.385-397, 10.1007/978-3-642-22285-6_42 . hal-01569188

HAL Id: hal-01569188
https://inria.hal.science/hal-01569188

Submitted on 26 Jul 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution| 4.0 International License

Building environmental semantic web applications with Drupal

Denis Havlik

Austrian Institute of Technology - AIT,
Donau-City-Straße 1, A-1220 Vienna, Austria
denis.havlik@ait.ac.at
http://www.ait.ac.at

Abstract. Efforts required for publishing information as Linked Data often appears too high compared to obvious and immediate benefits. Consequently, only a tiny fraction of the web can be easily used as a semantic "database" today. Drupal 7 changes the rules of the game by integrating the functionality required for structuring and semantically annotating arbitrary content types in the Drupal "core", as well as encouraging the module authors to use this functionality in their Drupal extensions. This paper presents the authors recent experiences with strengths and shortcomings of the Drupal 6 and Drupal 7 semantic web extensions, and discusses feasibility of the future semantic web environmental applications based on a Drupal platform. The intention of the paper is (1) to analyse the state of the art in semantic web support, as well as the potentials for further development in Drupal today; (2) to prove the feasibility of Drupal based semantic web applications for environmental usage area; and (3) to introduce the idea of Drupal as a rapid prototyping development environment.

Keywords: Drupal; Semantic Web; Linked Data; web services; rapid prototyping; environmental usage area

1 Introduction

A huge and growing number of environmental observations are collected on a daily bases, and a large part of these observations are published on the web today. Many of these observations are unfortunately published in a form unsuitable for further processing, for instance as a human readable text or as color-coded maps. Finding and binding the available observations, even if they have been published e.g. as structured XML [5] and annotated by appropriate ontology terms, for the value added application one wants to develop can often be a challenge of its own, since major search engines often ignore the existence of such data.

Linked Data [12] provides an alternative approach for publishing of arbitrary data in a form that is both suitable for automated processing, and easily discovered on the web: rather, or in addition to publication of the data through some form of a web service, the actual human-readable information published

on the web site is annotated by ontology terms embedded in eXtensible HyperText Markup Language (XHTML, [3]). The resulting combination is described in "Resource Description Framework in attributes (RDFa)in XHTML" W3C Recommendation [6]. Unfortunately, the effort required for publishing the Linked Data can often be quite high compared with the effort required to publish a human readable content on the Web. The seventh release of the Drupal Content Management System (CMS) promises to drastically lower this barrier: Drupal 7, which has been realised in January 2011 is the first major CMS platform with full integration of the key semantic web technologies [10]. Consequently, the additional effort and knowledge required to semantically enable a web site build upon a Drupal platform is expected to drastically lower within the next year or two. This may in turn positon Drupal as a primary platform for all stakeholders interested in painless transition from classical CMS to semantic web enabled applications.

This paper takes a critical look at the support for "Resource Description Framework" (RDF, [4]), SPARQL Query Language for RDF [8], and RDFa [7] in Drupal, from the point of view of a stakeholder interested in developing a lightweight semantic-web enabled applications, while at the same time keeping all advantages of modern CMS-es.

It starts by introducing the Drupal platform and the state of the art of semantic web support in Drupal 6 and Drupal 7 releases in Q1 2011 (section 2). This section is mainly intended as overview for the readers considering the use of semantic web Drupal extensions in their own projects.

Section 3 "Proof of Concept Application" presents a simple application which takes advantage of the Drupal's semantic web extensions. Due to Drupal's popularity in the biodiversity community, the use cases are pertinent to biodiversity usage area. This information is provided as a base for discussion, and in the hope it may be of some use for readers with little or no experience in semantic web and interested in building similar applications for other usage areas.

Finally, the section 4 "Conclusions and Outlook" section summarises the lessons learned in preparing this paper, and presents a vision of Drupal as a rapid prototyping development environment for building web services and semantic web applications.

2 Drupal and semantic web

Drupal is one of the most sucesfull open Source CMS platforms today. By the end of February 2011 there were almost 400k active Drupal installations [15]. Out of these, 25k sites already used the Drupal 7 release, which was published only two months earlier, in January 2011.

Drupal code is highly modular. "Drupal core" provides the functionality used by a great majority of the web sites and the large number of third party modules [9] providing more specialised functionality (e.g. "project management", "visualisation on the map", or "e-commerce"). In early February 2011, 5264 open source modules for Drupal 6, and 1069 Drupal 7 modules were available

for download on the Drupal web site. Two weeks later, there were already 1119 modules for Drupal 7. In the same period, the number of Drupal 6 modules also grew to 5311, indicating that the Drupal 6 is still considered as a main development target by many developers [15].

Drupal 6 already provides a number of possibilities for structuring and implicitly associating of meaning to information, but it does not associate this implicit knowledge with explicit machine readable representation. The development of third party semantic extensions for Drupal 6 resulted in a heterogeneous set of add-on modules that never reached maturity. Therefore only a small fraction of the Drupal 6 sites and of the Drupal 6 add-on modules ever made use of these semantic extensions. Table 1 summarises the development status of the key Drupal modules used in the preparation of this paper, and Table 2 lists some additional modules pertinent to the papers topic.

Core functionality required by semantic web Drupal applications is provided by CCK, Feeds, Views, RDF and RDF CCK modules. CCK allows site administrators to structure the data, Feeds module allows them to import the data from various sources and Views module provides a mechanism to define how the result will be published. RDF module provides RDF data type and an API for Drupal, while the RDF CCK (RDF in Drupal 7) allows us to annotate the node types and fields with ontology terms. It is therefore interesting to note that the Drupal 7 integrates most of the functionality offered by CCK, Views and RDF modules in the "core" Drupal code base.

Remaining modules provide functions for importing, exporting, annotating and manipulating structured data, as well as semantically annotated data, in Drupal. For example:

- SPARQL and VARQL modules provide support for SPARQL queries;
- Calais module illustrates the feasibility of automated semantic annotation of the texts associated with Drupal nodes;
- Semantic Markup Editor does the same for manual annotations;
- Finally, the modules such as Feeds and its plugins (e.g. Feeds XPath Parser, Feeds View Parser, Feeds QueryPath Parser), RDF SPARQL Proxy, VARQL and the Lin Clark's SPARQL_Views provide ways to both integrate external data into Drupal and transform between various Drupal's data representations (e.g. nodes, views, taxonomies).

The information presented in tables 1 and 2 clearly indicate that semantic functionality in Drupal hasn't reached maturity yet, in spite of the excellent support for RDF and RDFa in Drupal 7. Development of Drupal 6 RDF-related modules in most cases stopped at "proof of concept" level, and the Drupal 7 versions of these modules may not be even available yet. Consequently, most of the experiments presented in this paper have nevertheless been conducted using experimental Drupal 6 implementations.

Development of Drupal 7 semantic web modules is an on-going activity and many of the modules are expected to issue stable releases within next 12 months.

Module name	**Summary**	**D6 status**	**D7 status**	**No. sites**
Views	Smart query builder that, given enough information, can build the proper query, execute it, and display the results in various ways.	stable	core, alpha	>280k
Content Construction Kit (CCK)	Web interface allowing the administrator to add custom fields to nodes using a web browser.	stable	core, dev	>260k
Feeds	Import or aggregate data as nodes, users, taxonomy terms or simple database records.	beta	alpha	>12k
Resource Description Framework (RDF)	Provides comprehensive RDF functionality and interoperability for the Drupal 6.x platform.	alpha, dev	core, dev	>10k
Feeds XPath Parser	Feeds plugin for parsing XML and HTML documents. It enables site builders to leverage the power of Feeds to easily import data from complex, external data sources.	stable	beta	1k
SPARQL	enables the use of SPARQL queries with the RDF API for Drupal 6.x.	alpha	dev	0,3k
RDF external vocabulary importer (Evoc)	Evoc caches any external RDF vocabulary in Drupal, and exposes its classes and properties to other modules. RDF CCK relies on Evoc to offer classes and properties to be mapped to CCK fields, node title and body.	alpha	RDF	0,2k
RDF CCK	Allows site administrators to map each content type, node title, node body and CCK field to an RDF term (class or property).	dev	RDF	0,2k
Feeds View Parser	This module enables Feeds to take in data from a view output (i.e. from Views module). Combined with pluggable query backend in Views 3.x, this will enable Drupal to import data from practically anywhere from the web	dev	-	20
SPARQL_Views	Allows to query RDF data in SPARQL endpoints and RDFa on Web pages, and bring the data into Views.	alpha, GIT	GIT	2

Table 1. Key Drupal modules used in this paper and their development status (GIT= only available through Github; dev=development snapshot, alpha, beta, or stable; core=integrated in Drupal core; RDF=provided by Drupal 7 RDF module)

Module name	Summary	D6 status	D7 status	No. sites
Calais	Integrates Drupal with Thomson Reuters Calais service (http://www.opencalais.com/). Calais analyses the text, and finds the entities, facts and events related with the text.	stable	-	>5k
Taxonomy import/export	Import and export vocabularies and taxonomy terms via XML, Comma Separated Values, RDF and other formats.	stable	-	>2,5k
QueryPath	Integrates the QueryPath library (PHP equivalent of jQuery) for searching and manipulating HTML and XML documents into Drupal	stable	stable	>1k
Feeds QueryPath Parser	A plugin for the Feeds module that allows to run CSS queries against an XML, or HTML document (requires QueryPath)	beta	beta	0,1k
RDF SPARQL Proxy	Allows to instantiate RDF resources on demand (lazy loading) via SPARQL CONSTRUCT queries.	dev	-	12
VARQL	Provides SPARQL Query backend for Views in Drupal 7	-	dev	9
Semantic Markup Editor	Allows users to add semantic (RDFa) annotations tags to arbitrary texts.	dev	-	6

Table 2. Additional Drupal modules related to semantic web and their development status (GIT= only available through Github; dev=development snapshot, alpha, beta, or stable; core=integrated in Drupal core; RDF=provided by Drupal 7 RDF module)

3 Proof of Concept Application

In order to develop a proof of concept for the abstract use case of building a semantic web application capable of discovering, binding, processing and visualisation of the environmental observations, I first had to choose a concrete instance related to one of the environmental domains. For this paper, the choice fell to "biodiversity", in anticipation of the biodiversity-related developments that are planned in the ENVIROFI project [17].

The idea was to find some data sets with geo-temporal context which can be visualised on a map and on the time scale and then to mash this data with additional information from another source and present the results in several ways. Obvious candidates for such initial data are for instance "species occurrence", "species sightings", or "habitat extent". Good candidates for additional data

could be for instance "species/habitat descriptions", illustrations and translations to various languages.

In order to narrow down the problem at hand even further, I decided to concentrate only on the cats and their relatives (family: Felidae; genus: *).

3.1 Retrieving biodiversity data from GBIF and DBpedia

All the data used in this application originates either from DBpedia (http://dbpedia.org), or from the the experimental "GBIF data" site (http://data.gbif.net). DBpedia data can be directly retrieved through DPpedia sparql endpoint at http://dbpedia.org/sparql, while the GBIF site only allows querying by humans using the web interface and downloading of the resulting data sets[1]. Searching for a "cat" on either of the sites does not lead to satisfactory results, but the search for "Felidae" on GBIF data site quickly leads us to the web page featuring the map shown in the figure 1.

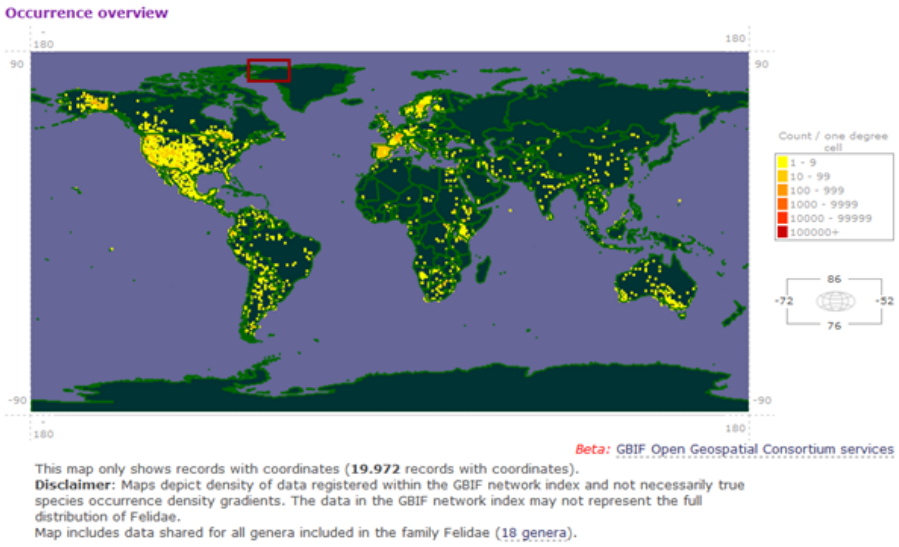


Fig. 1. GBIF occurency overview for felidea family (detail from data.gbif.org web site)

The "Actions" box (not shown in figure 1 provides a link to "Occurences", which in fact offers a comfortable query builder. The final query consisted of: (1) "Classification includes Family: Felidae"; (2) "Coordinate status is Includes coordinates"; (3) "Coordinate issues is No issues detected"; (4) "Basis of record is Observation"; (5) "Year range is between 1900 and 2011"; and (6) Data originating from Argentina, South Africa or Spain. This returned 2.419 occurence records; 1.766 thereof in Spain [16].

[1] In fact, GBIF data site also provides a RESTful service interface, but I did not realise this on time.

GBIF data can be downloaded in several formats, including the semantically annotated "Darwin Core" format. The first occurrence record from the downloaded GBIF file is shown below.

```
<dataProviders>
  <dataProvider gbifKey="326" rdf:about="326">
    <name>Administracion de Parques Nacionales, Argentina</name>
    <dataResource gbifKey="10868" rdf:about="10868">
      <name>VERTEBRADOS DE VALOR ESPECIAL EN REAS PROTEGIDAS DE LA ARGENTINA</name>
      <occurrenceRecords>
      <tax:TaxonOccurrence gbifKey="233877940"
          rdf:about="http://data.gbif.org/ws/rest/occurrence/get?key=233877940">
          tax:catalogNumber>APN-ML-21</tax:catalogNumber>
        <tax:country>AR</tax:country>
        <tax:decimalLatitude>-50.428</tax:decimalLatitude>
        <tax:decimalLongitude>-69.066</tax:decimalLongitude>
        <tax:earliestDateCollected>2006-05-03 00:00:00.0</tax:earliestDateCollected>
        <tax:identifiedTo>
          <tax:Identification>
            <tax:taxon>
              <tax1:TaxonConcept gbifKey="13815711" rdf:about="13815711">
                <tax1:hasName>
                  <tax2:TaxonName>
                    <tax2:nameComplete>Puma concolor</tax2:nameComplete>
                    <tax2:genusPart>Puma</tax2:genusPart>
                    <tax2:specificEpithet>concolor</tax2:specificEpithet>
                    <tax2:scientific>true</tax2:scientific>
              </tax2:TaxonName>
                </tax1:hasName>
              </tax1:TaxonConcept>
            </tax:taxon>
            <tax:taxonName>Puma concolor</tax:taxonName>
          </tax:Identification>
        </tax:identifiedTo>
        <tax:latestDateCollected>2006-05-03 00:00:00.0</tax:latestDateCollected>
      </tax:TaxonOccurrence>
```

The next step was to find some complementary information on various Felidae species from DBpedia. A search for "Felideae" using the DBpedia's "faceted search" interface (http://dbpedia.org/fct/) returned over 200 matches and the quick analysis of the results shows that the entries we are interested in, share the "Family is dbpedia:Felidae" relationship. Furthermore, the search results included a wealth of highly structured information consisting of names and descriptions of the Felidae members in various languages, photos, links to external resources and many other properties including the "genus", "species" and "binomial". For Iberian Lynx, the DBpedia genus is "Lynx", species "L. pardinus" and binomial "Lynx pardinus" which perfectly maps to "genus Lynx; species pardinus" in the GBIF data sets.

DBpedia faceted search results provide all information needed to build SPARQL queries and to retrieve the data. For example, the following SPARQL query returns some basic information on Iberian Lynx (Lynx pardinus).

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX db-owl: <http://dbpedia.org/ontology/>
PREFIX db-prop:<http://dbpedia.org/property/>
PREFIX db-res:<http://dbpedia.org/resource/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT DISTINCT ?species ?label ?abstract ?thumbnail ?depiction WHERE {
```

```
?feline db-owl:family db-res:Felidae;
    db-owl:abstract ?abstract;
    rdfs:label ?label;
    db-prop:binomial ?species;
    db-owl:thumbnail ?thumbnail.
    FILTER (
        langMatches(lang(?abstract), 'en') &&
        langMatches(lang(?label), 'en') &&
        regex(str(?species), ".*pardinus")
    ).
    OPTIONAL {?feline foaf:depiction ?depiction}
}
```

3.2 Defining the data model and semantic annotations

The basic unit of data in Drupal is called a "node". In most cases, each node corresponds with a single web page, but for our purpose it is convenient to see a node as Drupal equivalent of an SQL table entry. Each node is structured according to a node type and new node types can be easily defined by Drupal users possessing adequate rights (administrator, web-developer). For this application, I defined two node types: "animal description" and "animal occurrence". Definition of the animal description node type is illustrated in figure 2.

LABEL	NAME	FIELD	WIDGET	OPERATIONS
Title	title	Node module element		
Identification	group_ad_name	Fieldset	**fieldset** collapsible **classes**	delete
English Name	field_ad_name_en	Text	Text field	edit delete
Scientific Name	field_ad_name_sci	Text	Text field	edit delete
Abstract	field_ad_abstract	Long text	Text area (multiple rows)	edit delete
Thumbnail	field_ad_thumbnail	Text	Text field	edit delete
Depiction	field_ad_depiction	Text	Text field	edit delete
Add new field Label	field_ Field name (a-z, 0-9, _)	- Select a field type - Type of data to store.	- Select a widget - Form element to edit the data.	

Fig. 2. Field definitions of the "animal description" node type.

The simplest "animal occurrence" node type needs to contain a Latin name, observation time and georeference. In addition, a node reference field is required in order to mash up occurrence and description data later and an unique ID helps to keep the data synchronized with the original source.

Next, is to decide upon semantic annotations for each of the node types and for each of the fields. In Drupal 7, all data is automatically annotated with a

site ontology, but users can define additional annotations if they want. For this application, I choose to re-use the original annotations used by DBpedia and GBIF data site respectively. Detailed description of the process has been described in "Produce and Consume Linked Data with Drupal" paper by Stphane Corlosquet et. al. [14]. Probably the most appropriate way to proceed here is to (1) decide which ontologies will be used; (2) use Evoc module to upload them; and (3) use the classes and terms from these ontologies to annotate the data.

Once the RDF annotations have been defined, Drupal will automatically add RDFa annotations to all data of this type, as illustrated below.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN"
  "http://www.w3.org/MarkUp/DTD/xhtml-rdfa-1.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en" dir="ltr"
  xmlns:Animal_description="http://www.havlik.org/d6/?q=rdf/schema/Animal_description#"
  xmlns:db-owl="http://dbpedia.org/ontology/"
  xmlns:db-prop="http://dbpedia.org/property/"
  xmlns:db-res="http://dbpedia.org/resource/"

 ...

      <div class="field-label">Scientific Name: </div>
      <div class="field-items">
        <div class="field-item even"
          property="db-prop:binomial" datatype="">Puma concolor</div>
      </div>
 ...
```

Defining the data model and semantic annotations is straightforward and works "out of the box" in Drupal 7. This is not the case for Drupal 6, where RDFa support can only be introduced by installing the CCK, RDF, RDF CKK, Evoc and Evoc reference modules, and requires patching of the CCK module.

3.3 Binding data into Drupal

In section 3.1 and 3.2, I introduced two biodiversity-related data sources and explained how to prepare Drupal data models. Next critical step is to assure that the Drupal site can actually use the GBIF and DBpedia data. In order to avoid latencies, I opted for pre-fetching of the data to the Drupal site. Feeds module provides an elegant way to pre-fetch the data, cache it in a suitable node type and synchronize it as needed. Feeds parsers allow "feeding" of data from various sources, including for example local and remote HTML, XML, CSV and Excel files, Atom and JSON feeds, as well as the services such as Youtube, Flickr and Slideshare. XPath Parser provides a way to import data from XML files by specifying the XPath expressions[1] for each of the fields. Part of the XPath Parser configuration for GBIF occurrence data is shown in figure 3.

No SPARQL Parser for Feeds exists, but Lin Clark's SPARQL_Views module can be used to fetch the data from a SPARQL server and temporary store it in a Drupal feed. Furthermore, Lin explained how to store the data from a view into nodes using the Feeds View Parser in her blog [13]. The proposed solution is highly experimental and breaks XPath Parser. Similar functionality could be probably achieved in other ways, for instance by exporting the view as XML and



Fig. 3. First part of the XPath Parser configuration for GBIF occurrence data

parsing it with XPath parser. However, the direct feeding of the content from Drupal views to nodes is likely to be more efficient.

3.4 Mashups and visualization

The views module provides a rich set of functions for mashing data. However, the Drupal 6 version of the Views module relies on the foreign key relationship between two nodes for this task. Drupal Node Reference CCK module provides a way to establish this relationship, but these references can not be automatically computed within the Drupal framework. Inability to generate node references may be the biggest shortcoming of the Feeds module today, but fortunately quite easy to work around.

Extending the Drupal functionality with ad-hoc PHP code snipsets is quite easy, and the Computed Field CCK extension allows the administrators to define computed values in custom content types. As a workaround, I therefore defined a custom node field that calculates the reference at node generation time. Unfortunately, I did not succeed in persuading the system to use this field as a node reference. So, eventually I decided to fill-in the node references at SQL database level, which worked without any advert effects. Once the node reference was in place, mashing of two datasets with Views was straightforward, resulting in the table shown in figure 4.

[Edit view]

Observation period
From date: 2000
To date: 2002

Items per page: 5 Apply

Animal Name	English Name	Genus	Specific Epithet	GBIF Occurence Key	Observation Latitude	Observation Longitude	Observation Start	Observation End
Leopardus wiedii	Margay	Leopardus	wiedii	233874278	-25.68	-54.17	2000-04-10 (All day)	2000-04-10 (All day)
Puma concolor	Cougar	Puma	concolor	233877622	-25.71	-54.44	2002-05-05 (All day)	2002-05-05 (All day)
Puma concolor	Cougar	Puma	concolor	233877721	-25.68	-54.50	2002-05-15 (All day)	2002-05-15 (All day)
Puma concolor	Cougar	Puma	concolor	233878400	-25.79	-54.04	2002-11-14 (All day)	2002-11-14 (All day)
Puma concolor	Cougar	Puma	concolor	233878402	-25.80	-54.08	2002-12-13 (All day)	2002-12-13 (All day)

1 2 next › last »

Fig. 4. Table illustrating the mash-up of "animal sightings" and "animal description" content types.

Similarly to Feeds, the Views module is itself providing a pluggable environment for add-on modules. One such modules (SPARQL_Views) has already been introduced in this paper. Due to its immense popularity (see table 1), the Drupal site features a separate "Views" section with over 400 Views related modules. On top of this, a huge number of modules exist which can take the result from views and present it in various ways. Figure 5 illustrates the feasibility of presenting the mashed data on a map.

This map was generated using the gmap module which integrates the google maps into Drupal. Far more complex maps can be generated using the Open Layers module, which integrates the OpenLayers javascript library into Drupal [11], but gmap module is easier to set up and provides sufficient functionality for this experiment.

4 Conclusions and Outlook

This paper explores the possibilities of semantically enabled Drupal sites by building a simple application that:

- uses SPARQL query and XPath parsing to bind data from two external sources;
- semantically annotates the imported data and presents is as RDFa;
- produces new data by mashing the two sources; and

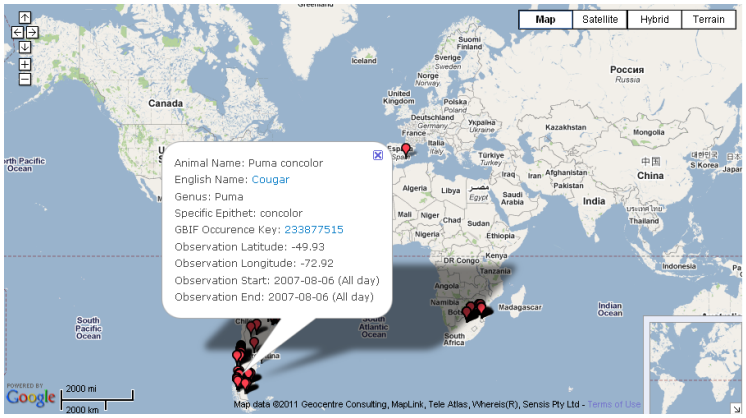


Fig. 5. Representation of the mashed data on a map

– presents the results as a table and on a map.

Drupal 7 release excels at basic semantic web functionality. It provides a possibility to define complex data (node) types, semantically annotate the data and publish the results as RDFa out of the box. Unfortunately, it is currently not possible to import external data to a Drupal 7 site using SPARQL query. Therefore, the final application has been built using Drupal 6.

Following the excellent documentation provided by Lin Clark, I was able to successfully test SPARQL import under Drupal 6, but the proposed solution breaks the XPath Parser module. As a result, the site was able to either import data from a SPARQL source, or from the XML file, but not both at the same time. Other shortcomings of the Drupal 6 solution included unreliable Evoc module and very rudimentary support for semantic annotation and RDFa. Further unsolved issue is establishing of the foreign key relationships between data imported from different sources, which currently has to be done at SQL database level.

Nevertheless, the results obtained so far are very encouraging. Most module developers are likely to issue Drupal 7 updates within next couple of months, so that a working solution (stable, well documented and easy to set up) for applications, similar to one presented in this paper, are likely to become available within next 1-2 years.

While preparing this paper, I also discovered the Services module, which allows a Drupal site to provide web services via multiple interfaces (including REST [18] and SOAP[2]). I did not test its functionality, but this module is under active development and used by more than 10k Drupal instances. This indicates that the module is of high quality and likely to be ported to Drupal 7 soon. Combination of the Service Module and the semantic web functionality outlined in this paper may transform Drupal from a CMS into rapid prototyping environment for all types of web services and semantic web services.

Acknowledgments

The work leading to the results presented in this paper has been partially funded by European Commission through FP7 *Tagging Tool based on a Semantic Discovery Framework* project (TaToo; EC grant agreement nr. 247893; http://www.tatoo-fp7.eu/tatooweb/).

Furthermore, I would like to thank: (1) Lin Clark and Stphane Corlosquet for their visionary Drupal developments and excellent publications explaining how to use the Drupal's semantic extensions; (2) Fuada Havlik and Gerald Schimak, for encouraging me in this work and proofreading the manuscript; and (3) Linux, Apache, MySQL and Drupal developers, for developing the immensely valuable Open Source software.

References

1. Xml path language (xpath) (11 1999), http://www.w3.org/TR/xpath/
2. Simple object access protocol (soap) 1.1 (05 2000), http://www.w3.org/TR/2000/NOTE-SOAP-20000508/
3. Xhtml 1.0 the extensible hypertext markup language (second edition) a reformulation of html 4 in xml 1.0 (08 2002), http://www.w3.org/TR/xhtml1/
4. Rdf primer (02 2004), http://www.w3.org/TR/rdf-primer/
5. Extensible markup language (xml) 1.0 (fifth edition) (11 2008), http://www.w3.org/TR/xml/
6. Rdfa in xhtml: Syntax and processing a collection of attributes and processing rules for extending xhtml to support rdf (10 2008), http://www.w3.org/TR/rdfa-syntax/
7. Rdfa primer bridging the human and data webs (10 2008), http://www.w3.org/TR/xhtml-rdfa-primer/
8. Sparql query language for rdf (01 2008), http://www.w3.org/TR/rdf-sparql-query/
9. Drupal modules (02 2011), http://drupal.org/project/modules
10. Drupal web site (02 2011), http://drupal.org/
11. Openlayers: Free maps for the web (02 2011), http://openlayers.org/
12. Berners-Lee, T.: Linked data. International Journal on Semantic Web and Information Systems 4 (2006), http://www.w3.org/DesignIssues/LinkedData.html
13. Clark, L.: Importing / syncing content from external sites like wikipedia. Blog entry (11 2010), http://lin-clark.com/blog/importing-syncing-content-external-sites-wikipedia
14. Corlosquet, S., u, R.D., Clark, T., Polleres, A., Decker, S.: Produce and consume linked data with drupal! In: Bernstein, A., Karger, D., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) The Semantic Web - ISWC 2009, Lecture Notes in Computer Science, vol. 5823, pp. 763–778. Springer Berlin / Heidelberg (2009), http://openspring.net/sites/openspring.net/files/corl-etal-2009iswc.pdf
15. Drupal: Drupal module usage statistics (02 2011), http://drupal.org/project/usage

16. GBIF: Biodiversity occurrence data. Provided by: Administracin de Parques Nacionales, Argentina, Borror Laboratory of Bioacoustics, GBIF-Spain, and University of Helsinki, Department of Applied Biology (02 2011), http://data.gbif.org, accessed through GBIF Data Portal
17. Havlik, D., Schade, S., Sabeur, Z.A., Mazzetti, P., Watson, K., Berre, A.J., Mon, J.L.: From sensor to observation web with environmental enablers in the future internet. Sensors 11(3) (2011)
18. Rodriguez, A.: Restful web services: The basics. developerWorks (2008), https://www.ibm.com/developerworks/webservices/library/ws-restful/