Produce and Consume Linked Data with Drupal!*

Stéphane Corlosquet[1,2], Renaud Delbru[1], Tim Clark[2,3],
Axel Polleres[1], and Stefan Decker[1]

[1] Digital Enterprise Research Institute, National University of Ireland, Galway, Ireland
[2] Massachusetts General Hospital, Department of Neurology, Boston, MA, USA
[3] Harvard Medical School, Department of Neurology, Boston, MA, USA

Abstract. Currently a large number of Web sites are driven by Content Management Systems (CMS) which manage textual and multimedia content but also - inherently - carry valuable information about a site's structure and content model. Exposing this structured information to the Web of Data has so far required considerable expertise in RDF and OWL modelling and additional programming effort. In this paper we tackle one of the most popular CMS: Drupal. We enable site administrators to export their site content model and data to the Web of Data without requiring extensive knowledge on Semantic Web technologies. Our modules create RDFa annotations and – optionally – a SPARQL endpoint for any Drupal site out of the box. Likewise, we add the means to map the site data to existing ontologies on the Web with a search interface to find commonly used ontology terms. We also allow a Drupal site administrator to include existing RDF data from remote SPARQL endpoints on the Web in the site. When brought together, these features allow networked RDF Drupal sites that reuse and enrich Linked Data. We finally discuss the adoption of our modules and report on a use case in the biomedical field and the current status of its deployment.

1 Introduction

Since the late 90ies and early 2000s a paradigm shift has taken place in Web publishing towards a separation of data (content) and structure (mainly layout). The first ideas to have the data represented in a way which allows its reuse in various ways and not only HTML, emerged in parallel in various systems such as Typo3 (1997), Plone (1999), WebML (2000) [6]. These open source systems and their commercial counterparts are typically subsumed under the term *Content Management System*s (CMS).

While it is worthwhile to mention that the first of these systems appeared at around the same time as Semantic Web technologies emerged, with RDF [14] being standardized in 1999, the development of CMSs and Semantic Web technologies have gone largely separate paths. Semantic Web technologies have matured to the point where they are increasingly being deployed on the Web. But the HTML Web still dwarfs this emerging Web of Data and – boosted by technologies such as CMSs – is still growing at much faster pace than the Semantic Web.

* This work has been supported by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-2). Preliminary results of this work were presented at the SFSW2009 workshop. We thank the reviewers and workshop audience for their valuable feedback.

A. Bernstein et al. (Eds.): ISWC 2009, LNCS 5823, pp. 763–778, 2009.

This is a pity, since actually both worlds could significantly benefit from each other. Particularly, large amounts of RDF data can now be accessed over the Web as *Linked Data* and built into Web applications [11]. Yet, widely available support for exporting and consuming Linked Data in CMSs – especially for non-experts – is still lacking, despite the fact that the need for dynamic inclusion and export of structured data in Web sites is widely recognized: at present, RSS feeds are the only agreed way to share and syndicate data across Web sites. However, the RSS format is quite limited when it comes to semantics. It was designed to carry news information, but today it is employed for carrying other types of information likely due to unawareness of compelling alternatives such as Linked (RDF) Data. We believe RDF is a good candidate for improving interoperability between sites because RSS can only carry a flat list of news item, while RDF can express any structured data and – by RDFS and OWL – even describe its own structure. Moreover, it offers a structured query language and protocol in order to extract and retrieve data matching a set of criteria, whereas with RSS, one first needs to fetch an entire feed and then process it locally to extract the desired information.

It is remarkable therefore, that although most common CMSs support RSS, RDF is still being largely ignored as a much richer potential syndication format. This is especially true since CMSs are typically built on a structured model of the domain of the site, which is reflected in both the underlying database, but – also and often more accurately – in the content types defined in the CMS itself by the site administrator. It is reasonable to assume that such structured models map naturally to RDFS and OWL. Additionally, the recently finished RDFa [1] standard could support the exposure of structured data on CMSs by allowing RDF to be embedded directly into the HTML pages, as opposed to a separate document (like in RSS). Hence RDFS, OWL and RDFa seem to be more adequate means to expose machine-readable Linked Data on Web sites. Likewise, the consumption and aggregation of Linked Data on a CMS site offer new possibilities further described in this paper. Approaching site administrators of widely used CMSs with easy-to-use tools to enhance their site with Linked Data will not only be to their benefit, but also significantly boost the Web of Data.

To this end, we extend Drupal – a state-of-the art CMS which has been gaining popularity recently by its rapidly growing user community, openness and modular design – with a number of features:

Firstly, we make it possible to expose the content types and fields typically defined by a site administrator using Drupal's Content Construction Kit (CCK) automatically in terms of classes and properties of a canonical OWL ontology, the *site vocabulary*. With our RDF CCK module, the site data itself become available as RDFa embedded in the live Web site following Linked Data principles. While this off-the-shelf solution already makes any Drupal site which installs our module amenable to Semantic Web tools such as RDF crawlers, search engines, and SPARQL engines to search and query the site, this is only a first step.

Secondly, for better linkage to the Web of Data, we enable mappings of the site vocabulary to existing ontologies. To this end, we extend the RDF CCK module to allow importing existing RDF vocabularies and map the site model to their terms.

Thirdly, to keep the learning curve low for site administrators, we support them in finding existing ontologies to reuse by a search facility. Unintended modifications of existing ontologies as well as the introduction of potential inconsistencies on the Semantic Web are avoided by our user interface as far as possible.

Fourthly, upon installation of an additional module, the site data can – on top of the RDFa data embedded in the HTML pages – be accessible via a standard SPARQL query interface, following the SPARQL protocol.

Finally, we allow administrators to dynamically integrate data from other RDF enhanced Drupal sites or Linked Data producers.

In the rest of this paper we briefly outline how our approach differs from earlier work in Section 2. Then we move on to describing specific details on Drupal in Section 3 and outline the goals our implementation should achieve in Section 4, before we look closer into each of our modules in Section 5. Adoption and deployment of our work are discussed in Section 6, conclusions are drawn in Section 7.

2 Related Works

Let us explain how our approach differs from previous attempts to link CMSs with the Semantic Web and why we believe that it is more adequate.

Although some earlier approaches proposed ontology based CMSs running natively on ontology management systems with RDF stores as back-ends [22,13], current CMSs run on very traditional Web application servers with relational databases back-ends. We do not aim to replace established infrastructures, but to build on top of them, with minimal intrusion and maximal reuse of the existing CMS infrastructure. We aim to extract and link ontologies from the content models and within the tools that site administrators are familiar with nowadays. We believe that taking users from where they are will enable more rapid adoption of Semantic Web technologies and lower entry barriers.

Somewhat closer to our approach is the idea to map the relational database schema underlying a CMS to RDF/RDFS [23]. Triplify[2] also follows this path, providing a generic mapping tool from relational databases to Linked Data (e.g. providing some predefined mappings to wrap Drupal sites' back-end databases into RDF). Our approach is again significantly different, as we want to capture the site content model and its constraints rather than the underlying database structure. Actually, a good part of the success of CMSs is grounded precisely in the fact that site administrators do not need to delve into the details of the underlying database system or schema – which may vary between different versions of Drupal and be affected by changes of the content model in non-obvious ways. Our approach works on a more abstract level (Drupal's CCK) directly in the API and user interface the site administrator is used to. We consider this model of the information structure more adequate than the underlying database schema.

Finally, none of the above approaches provide an all-in-one solution for exporting, aggregating and mapping Linked Data from within a commonly used CMS. This is what distinguishes our work, which is tailored for easy of use.

As for pre-existing work specifically in Drupal, a recent paper [8] describes the related SCF (Science Collaboration Framework) Node Proxy architecture. This module, developed specifically for biomedical applications, enables RDF from defined SPARQL queries to be mapped to specific Drupal content types. These mappings must be generated individually per content type - Node Proxy is not a general RDF-to-Drupal mapping facility, it does not support CCK, and it is read-only (into Drupal from Linked Data). Nonetheless, it was a significant first step along the path we develop further and more generally here. The Node Proxy architecture is currently used to retrieve Gene information from a common RDF store for StemBook[1], an open access review of stem cell biology. It will be superseded in the SCF Drupal distribution by the much more general and fully-featured modules we describe here. We will provide more details on the SCF use case in Section 6.

3 Drupal: A Popular CMS

Drupal (`http://drupal.org/`) is among the top three open-source CMS products in terms of market share [21] and accounts for more than 175 000 installations on the Web[2]. The system facilitates the creation of Web sites by handling many aspects of site maintenance, such as data workflow, access control, user accounts, and storage of data in the database.

As is typical for CMSs, a *site administrator* initially sets up a site by installing the core Drupal Web application and choosing from a large collection of *modules*. Site administrators do not write code; this is done by module developers instead. After the site has been set up, Drupal allows non-technical users to add content and handle routine maintenance tasks.

Each item of content in Drupal is called a *node*. Nodes usually correspond more or less directly to the pages of a site. Nodes can be created, edited and deleted by content authors. Some modules extend the nodes, for example a taxonomy module allows assignment of nodes to categories.

The *Content Construction Kit (CCK)* is one of the most popular and powerful modules used on Drupal sites. It allows site administrators to define types of nodes, called *content types*, and to define *fields* for each content type. Fields can be plain text fields, dates, file uploads, or references to other nodes, etc. Additional field types can be added via modules. When defining content types and fields, the site administrator has to provide *ID*, *label*, and *description* for content types and fields. Additionally, CCK allows to specify the following constraints on fields: (i) *Cardinality:* fields can be optional or required, and may have a maximum cardinality (ii) *Domain:* fields can be shared among one or more content types; (iii) *Range:* fields can be of type text, integer, decimal, float, date, file attachment, or node reference; node reference fields can be restricted to nodes of specific content types; text fields can be restricted to a fixed list of text values.

3.1 Motivating Example: The Project Blogs Site

Throughout this paper we will use a running example to illustrate our approach: a project blogs Web site[3] contains various information about the researchers at DERI and their collaborators, including their publications, blog posts and projects they work for. Our goal is to expose the site data and structure in a machine-readable form as well

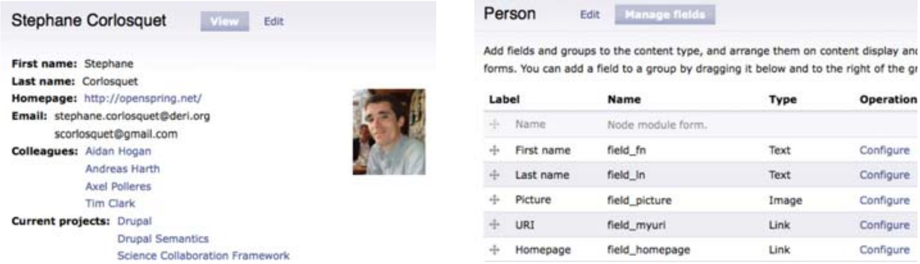
[1] http://www.stembook.org/

[2] `http://drupal.org/project/usage/drupal`

[3] Demo-site available at `http://drupal.deri.ie/projectblogs/`

as pull in data available from the Linked Data cloud or other Drupal sites in order to enrich the information displayed on the site.

Fig. 1 shows the typical look and feel of a Drupal page and administrative interface for the *Person* content type, without our extensions installed. This content type offers fields such as *name*, *homepage*, *email*, *colleagues*, *blog url*, *current project*, *past projects*, *publications*. Particularly, we will illustrate in the further sections how to extend the *publications* field to automatically display a list of publications pulled from various data endpoints.



First name: Stephane
Last name: Corlosquet
Homepage: http://openspring.net/
Email: stephane.corlosquet@deri.org
scorlosquet@gmail.com
Colleagues: Aidan Hogan
Andreas Harth
Axel Polleres
Tim Clark
Current projects: Drupal
Drupal Semantics
Science Collaboration Framework

Add fields and groups to the content type, and arrange them on content display an
forms. You can add a field to a group by dragging it below and to the right of the gr

Label	Name	Type	Operation
Name	Node module form.		
First name	field_fn	Text	Configure
Last name	field_ln	Text	Configure
Picture	field_picture	Image	Configure
URI	field_myuri	Link	Configure
Homepage	field_homepage	Link	Configure

Fig. 1. A user profile page (left), editing a content type in Drupal's CCK (right)

4 Publishing and Consuming Linked Data with a CMS

Given a Drupal CCK content model consisting of content types, fields, and nodes that instantiate the content types, what would be a good way of representing it in RDF? We consider the following features desirable for the RDF output which are in line with the Linked Data principles and best practices [3]:

(i) Resolvable HTTP URIs for all resources, to take advantage of existing tools that can consume Linked Data style RDF content. That is, when resolving URIs, one should find machine-readable information describing the URI. On the one hand in Drupal, typically URIs of the running site are simply URLs pointing to Web pages, but on the other hand, each of these pages also represents a node of a certain content type in the CCK content model. Thus, in our model, each node becomes an RDF resource, and the HTML Web page describing the node is enriched with RDFa [1] that reflects the links in the content model. That is, for each node URI

- we add an `rdf:type` triple asserting the membership of the node to a class corresponding to the content type of the node,
- we add a triple for each *field* displayed on the page where the predicate is a property representing the field itself and the field value is either a datatype literal (for text, integer, decimal, float, or date fields) or the URI of the respective node reference.

(ii) Expressing Drupal CCK constraints in OWL. Constraints that are defined on the types and fields (domains, ranges, cardinalities, disjointness) should be automatically published as RDF Schema [5] or OWL [9] expressions. We will enable this by an

auto-generated *site vocabulary* that is linked from the site and which describes all content type and field URIs as classes and properties in an ontology that reflects exactly the constraints expressible in CCK. We will explain this mapping in detail in Section 5.1.
(iii) Re-use of published ontology terms. To support sites talking about arbitrary domains, pre-defined/auto-generated RDF classes and properties are most likely insufficient. In fact, the default site vocabulary only comprises an isolated ontology not related to the rest of the Semantic Web. In order to link content to existing ontologies, we have to provide means to the site administrator to select terms from existing ontologies when setting up the content model. This requires that sites may reuse/import vocabulary terms from common existing ontologies. We will explain this in more detail in Section 5.1.
(iv) Safe vocabulary re-use. Mixing the content model constraints with constraints of a published ontology might have unintended semantic effects, especially since most site administrators will not be familiar with the details of the OWL semantics. The system must prevent such effects as far as possible. Practical examples are given in Section 5.1.
(v) Exposing a query interface. We rely on the SPARQL protocol [19], i.e. the site should expose its data in a SPARQL endpoint associated with the site. It should be easy to set up and should not be a burden for the site administrator.
(vi) Reuse of Linked Data. Where possible, linkage should be defined to other instances of the Linked Data cloud.

5 Implementation

We will now present our extensions of Drupal, which are designed to fulfill the goals outlined in the previous section, in more detail.

5.1 RDF CCK: From Content Models to Site Vocabularies

Administrators use CCK to define a site-specific content model, which is then used by content authors to populate the site. The focus of our work is to expose (i) such a CCK site content model as an OWL ontology that reflects the site structure which the administrator had in mind and (ii) the site content as RDF data using this ontology. We have implemented a Drupal module that enhances Drupal's CCK with the ability to auto-generate RDF classes and properties for all content types and fields. We build a so-called *site vocabulary*, i.e. an RDFS/OWL ontology which describes the content types and fields used in the data model as classes and properties. The field and type names are extracted from field and type *ID*s from CCK, such that – following common conventions – fields are assigned a property name, and content types are assigned a class name. Field and content type labels and descriptions are likewise exported as `rdfs:labels` and `rdfs:comments`. Here goes a typical content type and field definition extracted from CCK into RDFS:

```
site : Person a rdfs : Class; rdfs : label "Person";
  rdfs : comment "Researchers in DERI and their collaborators".
site : fn a rdf : Property; rdfs : label "First name";
  rdfs : comment "First name of a Person";
```

Likewise, field constraints from CCK are reflected in the site vocabulary: *Cardinality* is mapped to cardinality restrictions in OWL, i.e. *required* fields are restricted to `owl:cardinality 1`. whereas fields with a maximum cardinality n are restricted to `owl:maxCardinality` n. For instance, if we assume that each *Person* is required to have a *name* and works in at most 5 *project*s, these constraints in CCK would be exported in OWL as follows.

```
site:Person a rdfs:Class; rdfs:subclassof
  [ a owl:Restriction; owl:onProperty site:name; owl:cardinality 1],
  [ a owl:Restriction; owl:onProperty site:project; owl:maxCardinality 5].
```

Domains are reflected by the `rdfs:domain` constraints. Here, fields used by a single type can be modeled by a simple `rdfs:domain` triple. For instance, assuming that the *colleagues* field for *Person*s is not shared with any other content type in the current content model, we can simply write:

```
site:colleagues rdfs:domain site:Person.
```

CCK fields shared among several types have the union of all types sharing the field as domain. E.g., as *Publication* and *Blog post* share the *title* field, the site vocabulary contains

```
site:title rdfs:domain [owl:unionOf (site:Publication site:Blog_post) ].
```

Ranges of fields are analogously encoded by the `rdfs:range` property. Additionally, we distinguish between fields of range text, float, integer, decimal, or date, and those referring to file attachments or node references. We declare the former as `owl:Datatype-`Property and assign the datatypes supported in Drupal with their respective XSD datatypes, i.e. *text* → xs : string, *float* → xs : float, *integer* → xs : integer, *decimal* → xs : decimal, or *date* → xs : date. For instance, the text field *name* is reflected in the site vocabulary as:

```
site:name rdfs:range xs:string; a owl:DatatypeProperty.
```

Fields that range over texts restricted to a fixed list of text values are assigned an enumerated class of values using `owl:DataRange`s, e.g. *gender* is modeled as

```
site:gender a owl:DatatypeProperty; rdfs:range
  [ a owl:DataRange; owl:oneOf (”male” ”female”) ].
```

Adhering to Linked Data principles. Following the conventions mentioned in the previous section, the site vocabulary is generated and published automatically at the site URL under the default namespace `http://`*`siteurl`*`/ns#`, which we denoted by the namespace prefix `site:` in the examples before. Likewise, any Drupal page on a site will be annotated with RDFa triples that dereference terms of this site vocabulary as classes and properties linking Drupal content nodes as subjects and objects. We are in line with the Linked Data principles and best practices [3] as we provide resolvable HTTP URIs for all resources: Each of the pages also represents a node of a certain content type in the CCK content model. That is, as mentioned before, each node becomes an RDF resource, and the HTML Web page describing the node has describing embedded RDFa [1] using the site vocabulary. By this design, any Drupal site using our module is off-the-shelf amenable to existing tools that can consume Linked Data.

Mapping to Existing Ontologies. While the functionality we have described previously fits Drupal sites well into the Linked Data world, so far, we have created nothing more than an isolated ontology based on the existing site content model. However, the benefits of this exercise remain limited, unless we additionally allow linking the site vocabulary to existing vocabularies and ontologies populating the Semantic Web. For instance, instead of just exposing the *Person* type as a class in the site vocabulary, we might want to reuse a class in an existing ontology, such as `foaf:Person` from the FOAF[4] ontology which some other publishers on the Web already used. Likewise, we may wish to state that a *Publication* is actually a `foaf:Document`, or that a Publications are linked to their Publications by Dublin Core's[5] `dc:creator` property, etc.

To this end, our module adds a new tab "Manage RDF mappings" to the content type administration panel of CCK for managing such mappings cf. Fig. 2. An auto-complete list of suggested terms is shown, based on the keyword entered by the user. The terms are coming from two different sources, which are detailed below.

External vocabulary importer service. The module RDF external vocabulary importer (evoc)[6] has been created to allow the import of vocabularies available on the Web and make the imported terms available in the mapping interface. The site administrator simply needs to fill in a form with the vocabulary URI and the prefix to be used in the system to refer to the vocabulary term when using the CURIE [4] format. A set of SPARQL queries are processed against the vocabulary to extract its classes and properties and some information about them like label, comment, superclass, domain, and range. These are then cached locally to provide a smoother user experience. Given their popularity, the Dublin Core, FOAF and SIOC vocabularies are imported automatically upon installation of the evoc module.

External ontology search service. We have also developed an ontology search service to help users to find ontologies published on the Web of Data. The search engine is entity-centric, i.e. instead of returning a list of relevant ontologies, it returns a list of relevant ontology entities to the user request. The service is currently covering cleaned up Web crawls of DERI's SWSE.org and Sindice.com search engines comprising Web data documents that define properties and classes (+100.000 documents).

Data Pre-Processing. Before being indexed, we perform a sequence of pre-processing tasks on the ontology data. Among them, the most important ones are reasoning and splitting. Reasoning is applied on each ontology in order to infer useful information for a search engine, such as the hierarchy of classes and properties, the domains and ranges of properties, etc. Reasoning over semantically structured documents enable to make explicit what would otherwise be implicit knowledge: it adds value to the information

[4] `http://xmlns.com/foaf/0.1/`

[5] `http://purl.org/dc/elements/1.1/`

[6] `http://drupal.org/project/evoc`

and enables an entity-centric search engine to ultimately be much more competitive in terms of precision and recall [16]. Ontologies are then split into smaller pieces on a per-entity basis. For each authoritative URI[7] found in the ontology, we simply extract all its outgoing links.

Indexing Model. The simplest semi-structured indexing method is to represent an ontology entity as a set of attribute-value pairs using a field-based approach [15]. For example, the ontology class `foaf:Person` will have fields *label*, *comment* and *subClassOf*; index terms are constructed by concatenating the field names with values of this field, for example as *subClassOf:Agent*.

Objects of type *literals* and *URI* are normalised (tokenised) before being concatenated with their field name. It is thus possible to use full-text search not only on literals, but also on URIs identifying ontology terms. For example one could search for ”`Agent`” to match `foaf:Agent`, ignoring the namespace.

We allow search for plain keywords, combinations of keywords, or structured queries (e.g. `student AND subClassOf:Person` or `name AND domain:Person`), search examples are shown in Fig. 2; find more details on the user interface below.

Mapping process. The terms suggested by both of the import service and the ontology search service can be mapped to each content type and their fields. For mapping content types, one can choose among the classes of the imported ontologies and for fields, one can choose among the properties. The local terms will be linked with `rdfs:subClassOf` and `rdfs:subPropertyOf` statements, e.g. `site:Person rdfs:subClassOf foaf:Person` to the mapped terms in the site vocabulary; wherever a mapping is defined, extra triples using the mapped terms are exposed in the RDFa of the page.

The use of subclasses and subproperties for mapping to existing external ontologies – instead of reusing the imported terms directly in the definitions of the site vocabulary – is a simple way of minimizing unintended conflicts between the semantics of local vocabulary and public terms. This way we ensure that, e.g. constraints on local terms such as the cardinality restrictions which we derive from CCK do not propagate to the imported ontology. This ensures *safe vocabulary re-use*, i.e. avoids what is sometimes referred to as “Ontology Hijacking” [12].

Intuitively, safe reuse means that a vocabulary importing another one does not modify the meaning of the imported vocabulary or “hijack” instance data of the imported vocabulary.

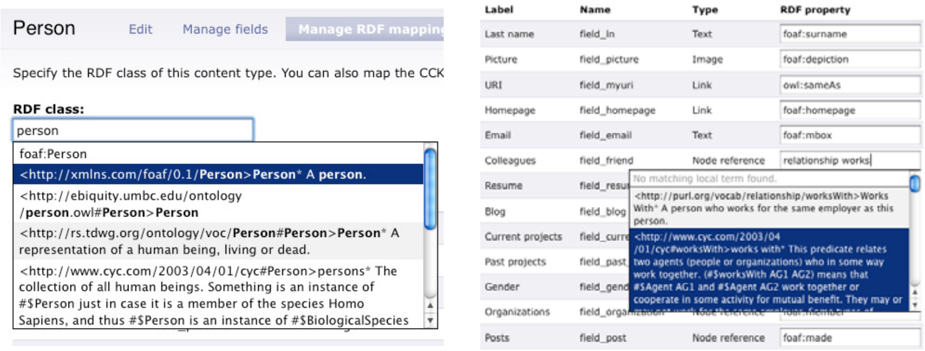
We remark that this measure alone is not sufficient to guarantee consistency of the site vocabulary. Contradicting cardinalities in the site vocabulary and imported properties could make site instance data inconsistent or imply unwanted equalities. Our mapping tool makes several restrictions in this respect such as disallowing properties for mapping with cardinality restrictions that do not comply with those specified in CCK for the respective field.

[7] The Linked Data principles suggest that URIs for named entities should be dereferenceable and should link directly to the data describing the entity itself. Following this recommendation, we define as an *authoritative URI*, a URI that is dereferenceable and is linked to the ontology.

We emphasize that we cannot detect all inconsistencies possibly introduced by ontology reuse in our system. We cannot afford full reasoning services as we want our tool to fit in the Drupal ecosystem as a normal module that is installable on a site without the need to install separate external software components such as a DL reasoner. Our current approach is a best-effort approach to avoid "misuse" of imported ontologies as far as possible while deliberately keeping the interface simple. We refer the interested reader to [7] for further details.

We emphasize that the whole mapping step is optional, and the main benefit of the Web of Data – exposing site data for re-use by third parties – is essentially realized by the default mapping already.

User experience. The first example illustrated on the left of Fig. 2 is the mapping of the Person content type to an RDF class. When typing `person` in the text field, a list of suggestions is pulled from both the local set of terms and from the external search service. The local terms appear first: `foaf:Person` is a CURIE and reuses the prefix definition of the site which can be defined during the import step. Then the list is completed with the terms from the external service, which includes a much greater variety of terms. This might help the user in discovering new terms or ontologies which she may not previously have encountered. Note that the external terms are displayed as full URI as we want to avoid imposing any prefix on them. We are currently evaluating the best way to display these.



Specify the RDF class of this content type. You can also map the CCK

RDF class:

person

foaf:Person

<http://xmlns.com/foaf/0.1/**Person**>**Person*** A **person.**

<http://ebiquity.umbc.edu/ontology/**person**.owl#**Person**>**Person**

<http://rs.tdwg.org/ontology/voc/**Person#Person**>**Person*** A representation of a human being, living or dead.

<http://www.cyc.com/2003/04/01/cyc#Person>persons* The collection of all human beings. Something is an instance of #$Person just in case it is a member of the species Homo Sapiens, and thus #$Person is an instance of #$BiologicalSpecies

Label	Name	Type	RDF property
Last name	field_ln	Text	foaf:surname
Picture	field_picture	Image	foaf:depiction
URI	field_myuri	Link	owl:sameAs
Homepage	field_homepage	Link	foaf:homepage
Email	field_email	Text	foaf:mbox
Colleagues	field_friend	Node reference	relationship works
Resume	field_resu		
Blog	field_blog		
Current projects	field_curre		
Past projects	field_past		
Gender	field_gend		
Organizations	field_orga		
Posts	field_post	Node reference	foaf:made

No matching local term found.

<http://purl.org/vocab/relationship/worksWith>Works With* A person who works for the same employer as this person.

<http://www.cyc.com/2003/04/01/cyc#worksWith>works with* This predicate relates two agents (people or organizations) who in some way work together. (#$worksWith AG1 AG2) means that #$Agent AG1 and #$Agent AG2 work together or cooperate in some activity for mutual benefit. They may or

Fig. 2. RDF mappings management through the Drupal interface: RDF class mapping (left) and RDF property mapping (right)

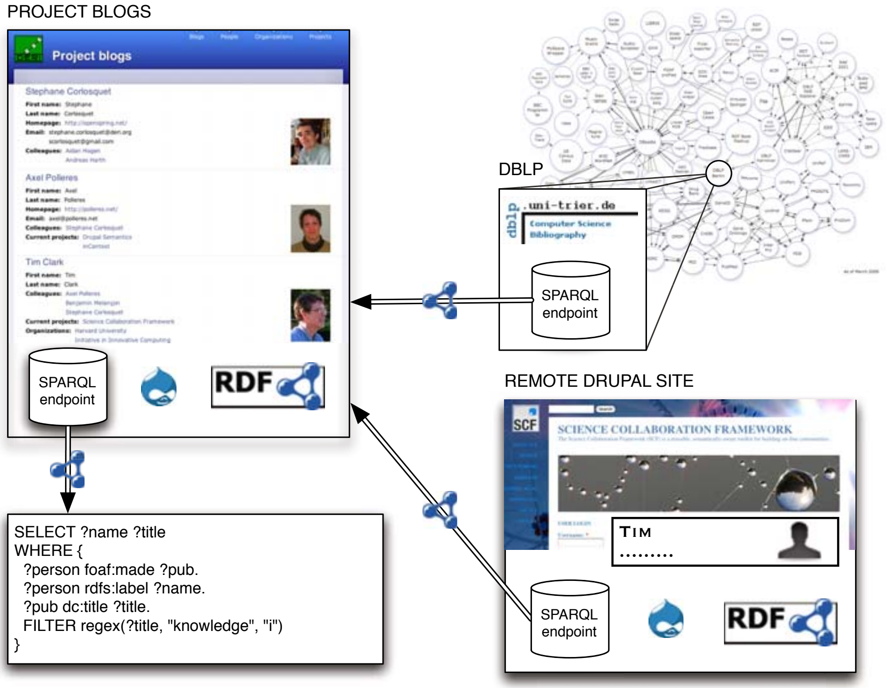
The second example on the right of Fig. 2 illustrates the case where the user wants to reuse the Relationship Ontology[8] to express relationships between colleagues who work with each other. Despite the fact that the Relationship Ontology was not imported locally, the external ontology search Web service (5.1) was able to suggest the right term URI.

[8] http://purl.org/vocab/relationship/

5.2 Exposing and Consuming Linked Data in Drupal with SPARQL

We extend our running use case to the Linked Data cloud environment where we can demonstrate the interoperability of multiple Web sites, as illustrated on Fig. 3. Our goal is to use our project blogs Web site as a hub containing information federated from various remote locations:

- DBLP is a public SPARQL endpoint containing metadata on scientific publications. It is part of the Linking Open Data cloud and runs on a D2R server[9].
- The Science Collaboration Framework Web site which contains information about the SCF team and their scientific publications. It runs Drupal and the modules described in this paper.



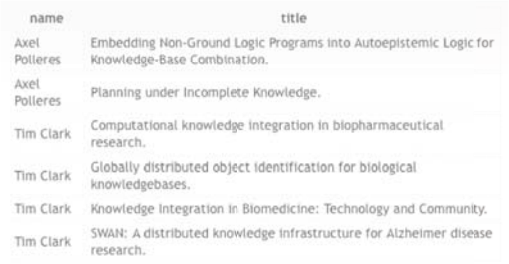
```
SELECT ?name ?title
WHERE {
  ?person foaf:made ?pub.
  ?person rdfs:label ?name.
  ?pub dc:title ?title.
  FILTER regex(?title, "knowledge", "i")
}
```

Fig. 3. Extended example in a typical Linked Data eco-system

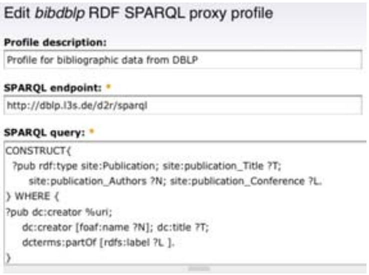
Exposing RDF data with a SPARQL endpoint. The first step to ensure interoperability on the Web of Data is to provide an endpoint which exposes RDF data. The *RDF SPARQL endpoint* module uses the PHP ARC2 library[10]. Upon installation, the module will create a local RDF repository which will host all the RDF data generated by the RDF CCK module (see Section 5.1). The site can then be indexed with a simple click.

[9] `http://www4.wiwiss.fu-berlin.de/bizer/d2r-server/`

[10] `http://arc.semsol.org/`



name	title
Axel Polleres	Embedding Non-Ground Logic Programs into Autoepistemic Logic for Knowledge-Base Combination.
Axel Polleres	Planning under Incomplete Knowledge.
Tim Clark	Computational knowledge integration in biopharmaceutical research.
Tim Clark	Globally distributed object identification for biological knowledgebases.
Tim Clark	Knowledge Integration in Biomedicine: Technology and Community.
Tim Clark	SWAN: A distributed knowledge infrastructure for Alzheimer disease research.



Edit *bibdblp* RDF SPARQL proxy profile

Profile description:
Profile for bibliographic data from DBLP

SPARQL endpoint: *
http://dblp.l3s.de/d2r/sparql

SPARQL query: *

```
CONSTRUCT{
  ?pub rdf:type site:Publication; site:publication_Title ?T;
      site:publication_Authors ?N; site:publication_Conference ?L.
} WHERE {
?pub dc:creator %uri;
    dc:creator [foaf:name ?N]; dc:title ?T;
    dcterms:partOf [rdfs:label ?L ].
}
```

Fig. 4. A list of SPARQL results (left) and an RDF SPARQL Proxy profile form (right)

The RDF data of each node is stored in a graph which can be kept up to date easily when the node is updated or deleted. Fig. 4 (left) depicts a list of publications whose title contains the keyword "knowledge".

Consuming Linked Data by lazy loading of distant RDF resources. With an ever growing amount of data available on the Semantic Web, one site cannot technically afford to host all the data available on the Web, even if the scope was restricted to a specific domain. Instead, each piece of data can be retrieved only when needed. This design pattern is known as lazy loading [10]. Information on the Web of Data is made available in endpoints which can be queried and from where information can be retrieved according to the specific needs of an application. The SPARQL query language [18] allows complex WHERE patterns able to extract the pieces of information desired, but another feature of SPARQL is the ability to specify a schema in which the RDF data should be returned. These CONSTRUCT queries are useful in the case where the information retrieved should retain a particular structure which would not fit in flat array of results such as a SELECT SPARQL query would provide.

Building atop the existing RDF schema provided by the RDF CCK module presented in Section 5.1, we developed *RDF SPARQL Proxy*, a module which allows to import RDF instances on demand, via a CONSTRUCT query in which the WHERE clause corresponds to the schema of the distant data on the SPARQL endpoint, and the CONSTRUCT clause corresponds to the local site schema defined by RDF CCK. As depicted on Fig. 4 (right), site administrators can define profiles, which specify the rules for creating or updating the local RDF instances based on the schema of the distant RDF data. In order to keep these profiles generic, we allow input parameters such as URIs. In this example, we map the publications by an author represented by her URI `%uri` along with the information about each publication (title, name of authors, conference) to our local schema. The value of the `%uri` parameter will be replaced for the value given as input, either in the address bar of the browser or by the API calling the RDF SPARQL Proxy module. For our use case, we have setup two such profiles: one for bridging the DBLP SPARQL endpoint to the project blogs Web site, and a second for bridging the Science Collaboration Framework Web site. When visiting Tim's profile page, the relevant publication information will be fetched from both DBLP and SCF Web sites, and either new nodes will be created on the site or older ones will be updated if necessary.

6 Adoption and Deployment

Our hypothesis and general rationale is that ease-of-use and a one-click solution to export Linked Data from CMSs will enable many concrete applications of the Semantic Web, and create a bridge between the CMS and Semantic Web technology ecosystems to the benefit of both. Particular use cases and applications of this approach are discussed below.

6.1 Usability

We did a limited-scale user evaluation aimed at showing that linking a site to existing vocabularies with our Drupal module does not impose a significant burden on site administrators. We argue that the benefits of exposing Semantic Web data such as greatly improved searchability, will typically outweigh this extra effort.

Our evaluation was carried out on a group of 10 users, moderately familiar with Drupal and more or less familiar with the Semantic Web. We found that on average, the RDF mapping process took about 50% of the time required to setup the content model. For more detailed results, we refer the reader to [7]. While linking to external vocabularies was subjectively experienced as easy by all users, a significant time was actually spent deciding to which properties and classes to link with the CCK fields. Inspired by this finding we put on our agenda to further investigate how we can better assist non-Semantic-Web-savvy users in finding the “right” classes and properties for their needs.

6.2 Adoption

Besides our closed evaluation, we have also released the RDF CCK module on drupal.org. Since its release Nov. 2008, the RDF CCK module has – steadily increasing – reached a number of 63 deployed installations[11] as shown in Fig. 5. This is encouraging. Our module is currently being tested and will be deployed in the next version of the Science Collaboration Framework (SCF) platform, a special Drupal distribution developed at the Massachusetts General Hospital and Harvard University in collaboration with DERI and other participating institutions [8].

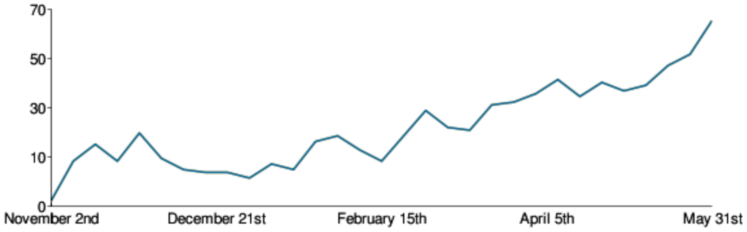


Fig. 5. Evolution of the number of installations of RDF CCK

[11] According to `http://drupal.org/project/usage/rdfcck`

6.3 Motivation and Benefits - The SCF Use Case

Harvard Medical School and DERI are collaborating on a larger use case in the course of which the technologies mentioned in this paper were developed. The Science Collaboration Framework (SCF) [8] is a distributed Drupal installation launched in Beta version at various institutions working in the Biomedical domain.

Biomedical informatics provide one particularly cogent and well-researched set of use-cases for the facility we have built and there are big expectations for the use of Linked Data in this domain, especially in the SCF Project. SCF is building Drupal based sites and tools that will enable scientific collaboration and Semantic search in this area.

Mapping of graph-based metadata embodying controlled terminologies and relationships (ontologies) to CMS-managed content promises to be exceptionally useful in biomedical informatics, and more broadly in scientific communications on the Web. Biomedicine, a highly descriptive, inductive and experimentally based discipline, is rife with complex terminologies. Synonym, subsumption, and other semantic relationships in such terminologies are natural and necessary. But currently we are still limited in the power of text searching across documents and sites if the relationships and properties in the text are not computable across the elements of these terminologies (or ontologies). This requires that certain elements in the text be assigned a semantic context which is computable in the CMS. This is a use case for semantic tagging of documents, which can leverage the well-defined ontologies in this domain.

For example, from scientific papers in this domain we may extract text strings such as “nf-κB”, “nuclear factor kappa B”, or “nf-kappa-B”. By adequate thesauri, or user tagging using CommonTag[12] all of these could actually be matched to the query string “NFKB1”, which the HUGO official gene names[13] and potentially other synonyms all resolve to a common URI represented in the Neurocommons [20] triple store, and the synonymy relationship is represented in RDF available at the Neurocommons SPARQL endpoint. Such extended search facilities, are next on our agenda, once the simple annotation of publications authors like presented in a simplified form in this paper is realised. Here, mapping RDF to an associated CCK generated type in Drupal will import the synonymy relationships and enable term expansion to increase search power.

Existing biomedical ontologies and database records which represent information about genes and other biomedical terms represent structured relationships all of which can be found in RDF and drawn into our site.

This use case becomes particularly compelling when one considers that biomedical research consists of myriad sub-specialities ranging across from basic research to clinical practice, as well as incorporating divisions by biological process, organ, species, cell type, molecule, protein family, technological approach, clinical orientation, disorder, and so forth. Each of these areas can and often does have its own slightly different semantic universe and forms of discourse. The ability to intersect documents and information from and about researchers across these domains of discourse, at scale, with assistance from computers, is dependant upon our ability to leverage formal

[12] `http://commontag.org`

[13] HUGO Gene Nomenclature Committee `http://www.genenames.org/`

terminologies and ontologies by linking them to text in scientific communications. That is precisely the purpose of the module described in this paper. The experts in these domains are hardly IT or Semantic Web experts, though they are able to use easy-configurable tools for aggregating and setting up content like Drupal, setting up the required modules via SCF on their site, and enter relevant data.

At the moment, we deploy RDF CCK in the SCF Beta version and the other modules mentioned in this paper are shortly before deployment and several of them have been co-developed or inspired by existing SCF modules such as SCF Node Proxy module, which we mentioned in the Introduction.

7 Conclusions and Outlook

We have presented a number of extensions to Drupal that enable the exposure of site content as Linked Data and likewise allow to aggregate and reuse existing RDF data from the Web in your Drupal site. A list of the modules used throughout this paper is available at `http://drupal.deri.ie/projectblogs/about`. Our most widely deployed module RDF CCK – available at the official Drupal site `http://drupal.org/project/rdfcck`) – allows to link existing and newly deployed Drupal sites to the Web of Data by a few clicks. It auto-exports the content model of a Drupal site to an ontology that is published following common best practices for ontology publication and enables the exposure of Drupal site content as RDFa. With the Evoc module, we link to existing properties and classes from other Semantic Web vocabularies by subclass/subproperty relations and offer a search facility for commonly used vocabulary terms on the Web of Data. A third module – RDF SPARQL Endpoint – exposes upon installation a SPARQL endpoint on the site data without additional configuration steps for site administrators who wish this feature. Finally, a fourth module – RDF SPARQL Proxy – allows to dynamically load data into the site, and displays this data using a lazy loading strategy for minimizing delays in the user experience.

In combination, all four modules offer new possibilities to create networked Web applications and pushing further population of the Web of Data by significantly lowering entry barriers for a large user community – CMS administrators.

Next steps include the extension of RDF/OWL export to other Drupal modules such as the taxonomy module which allows to define tag hierarchies usable in Drupal, which we plan to expose as RDF using SKOS [17], and content type hierarchies, to be reflected by subclass relationships. Moreover, we shall also lower the burden for users of the RDF SPARQL Proxy – which is at the moment only accessible to users knowledgeable in SPARQL – to really reveal the full potential of our approach to a wider audience.

We shall further develop and deploy our approach in the Science Collaboration Framework which we have presented as a promising early adopter use case.

The “infection” of emerging power-user communities such as the rapidly growing Drupal site administrator and developer groups is in our opinion a key in boosting Semantic Web technologies. We shall provide easy-to-use, unobtrusive RDF exposure in a way general enough for a variety of sites, thus potentially contributing significantly to populating the Web with high-quality RDF data.

References

1. Adida, B., Birbeck, M., McCarron, S., Pemberton, S. (eds.): RDFa in XHTML: Syntax and Processing, W3C Rec (2008)
2. Auer, S., Dietzold, S., Lehmann, J., Hellmann, S., Aumüller, D.: Triplify - Lightweight Linked Data Publication from Relational Databases. In: WWW (2009)
3. Berrueta, D., Phipps, J. (eds.): Best Practice Recipes for Publishing RDF Vocabularies, W3C Working Group Note (August 2008)
4. Birbeck, M., McCarron, S. (eds.): CURIE Syntax 1.0: A syntax for expressing Compact URIs, W3C Cand. Rec (January 2009)
5. Brickley, D., Guha, R. (eds.): RDF Vocabulary Description Language 1.0: RDF Schema, W3C Rec (2004)
6. Ceri, S., Fraternali, P., Bongio, A.: Web Modeling Language (WebML): A Modeling Language for Designing Web Sites. In: WWW 2000, Amsterdam, The Netherlands, pp. 137–157 (2000)
7. Corlosquet, S., Cyganiak, R., Decker, S., Polleres, A.: Semantic Web Publishing with Drupal. Tech. Report DERI-TR-2009-04-30, DERI (April 2009)
8. Das, S., Girard, L., Green, T., Weitzman, L., Lewis-Bowen, A., Clark, T.: Building biomedical Web communities using a semantically aware content management system. Briefings in Bioinformatics 10(2), 129–138 (2009)
9. Dean, M., et al. (eds.): OWL Web Ontology Language Reference, W3C Rec (2004)
10. Fowler, M.: Patterns of Enterprise Application Architecture. Addison-Wesley, Reading (2002)
11. Hausenblas, M.: Exploiting Linked Data For Building Web Applications. IEEE Internet Computing 13(4), 80–85 (2009)
12. Hogan, A., Harth, A., Polleres, A.: Scalable Authoritative OWL Reasoning for the Web. Int'l Journal on Semantic Web and Information Systems 5(2) (2009)
13. Jin, Y., Decker, S., Wiederhold, G.: OntoWebber: Model-Driven Ontology-Based Web Site Management. In: Cruz, I.F., Decker, S., Euzenat, J., McGuinness, D.L. (eds.) SWWS (2001)
14. Lassila, O., Swick, R. (eds.): Resource Description Framework (RDF) Model and Syntax Specification, W3C Rec (1999)
15. Luk, R.W., Leong, H.V., Dillon, T.S., Chan, A.T., Croft, W.B., Allan, J.: A survey in indexing and searching XML documents. Journal of the American Society for Information Science and Technology 53(6), 415–437 (2002)
16. Mayfield, J., Finin, T.: Information retrieval on the Semantic Web: Integrating inference and retrieval. In: SIGIR Workshop on the Semantic Web (August 2003)
17. Miles, A., Bechhofer, S. (eds.): SKOS Simple Knowledge Organization System Reference, W3C Cand. Rec (March 2009)
18. Prud'hommeaux, E., Seaborne, A. (eds.): SPARQL query language for RDF, W3C Rec (2008)
19. Clark, K.G., Feigenbaum, L., Torres, E. (eds.): SPARQL protocol for RDF, W3C Rec (2008)
20. Ruttenberg, A., Rees, J., Samwald, M., Marshall, M.S.: Life sciences on the Semantic Web: the Neurocommons and beyond. Briefings in Bioinformatics 10(2), 193–204 (2009)
21. Shreves, R.: Open Source CMS Market Share. White paper, Water & Stone, `http://waterandstone.com/downloads/2008OpenSourceCMSMarketSurvey.pdf`
22. Staab, S., Angele, J., Decker, S., Erdmann, M., Hotho, A., Maedche, A., Schnurr, H.P., Studer, R., Sure, Y.: Semantic community Web portals. Computer Networks 33(1-6), 473–491 (2000)
23. Stojanovic, L., Stojanovic, N., Volz, R.: Migrating data-intensive Web sites into the Semantic Web. In: SAC 2002: Proceedings of the 2002 ACM symposium on Applied computing. ACM, New York (2002)