The Talia library platform

Rapidly building a digital library on Rails

Michele Nucci[1], Daniel Hahn[2], and Michele Barbera[2]

[1] Semedia Group - 3mediaLabs
Università Politecnica delle Marche
mik.nucci@gmail.com
[2] NET7 - Pisa
hahn@netseven.it

Abstract. Talia is a web-based distributed digital library and publishing system, designed for scholarly research in philosophy. Talia is based on semantic web technology; it's being developed with the Ruby on Rails web framework.
Rails is a relatively new environment, which allows developers to easily create well-structured web applications. By combining its power with semantic web technology and leveraging on existing solutions like ActiveRDF, it is possible to quickly create a full-featured semantic library platform, from scratch, in a short time.
Talia does not aim at creating new semantic web technology as such, but at providing practical solutions for embedding the existing technology in modern web applications.
This paper focuses on a few select features of Talia to show the possibilities.

1 Project

Discovery[3] is a European project to create an extensive online library for philosophers. The four content partners of the project will provide tens of thousands digitised and annotated pages, so that the project will start with a large body of material.

Philosophers usually work in a traditional, print-and-paper based way. However, finding all relevant publications on a topic is difficult and acquiring copies is yet another matter. This is especially true for original manuscripts, which can be notoriously hard to acquire [1].

By creating a comprehensive resource with original manuscripts and secondary writings, we can provide an invaluable tool to the scholarly research community.

Discovery will be based on *Philosource*, a federation of interlinked online libraries. The libraries currently use the *Hyper* platform. *Hyper* was created for the preceding *HyperNietzsche* project (now *NietzscheSource*[4]).

[3] http://www.discovery-project.eu/
[4] http://www.nietzschesource.org/

The *Hyper* platform was developed specifically for a Nietzsche library. Particular assumptions about Nietzsche are hardcoded into the application, and the codebase is not flexible enough to be a viable long-term solution for the project. It will be superseded by the *Talia* platform described in this paper.

In the humanities, the text itself is the subject of study. Talia aims to provide an integrated library system that offers tools to *work with original content*. This is in marked contrast to citation systems like CiteSeer[5], CiteULike[6] or even online archives like arXiv[7]. These focus on bibliographical metadata, the actual content is not more than an opaque, downloadable document.

Talia provides a tight integration between a semantic backend store and a powerful interface toolkit, making it unlike existing "infrastructure" library systems, such as BRICKS [2] or Fedora[8]. These provide a backend on which an interface has to be built from scratch. The JeromeDL system [3] is somewhat similar to Talia, but aimed at a different audience.

2 Requirements and Technology

Within the Discovery project, Talia does not exist in a vacuum – it's a means to an end. There are a number of features and components that *must* be implemented in Talia to make the software useful within the project:

- Users expect that the **User interface** contains all features that are already present on the Hyper platform.
- **Metadata and ontology support** is essential. Each group of scholars will create their own *domain ontology* to order their material.
- A **Remote Federation API** has to be implemented to allow automatic bidirectional references between documents in different libraries.
- **Publication and Workflow**. Scholars will be able to publish new results online. Talia must offer a number of workflow models for peer-reviewed publication.

A *speedy development* is also essential, since the existing content has to be migrated in the second year of the project and Talia must be running for the project to succeed.

The rapid development of Talia would not be possible without leveraging existing technology, both from commercial web development and semantic web research.

- **Ruby on Rails**[9] is a web development framework that has been picking up a lot of pace recently. It uses the Model-View-Controller [4] pattern and allows Developers to create full-featured web applications with a minimal amount of code. It's available under the MIT license [5].

[5] `http://citeseer.ist.psu.edu/`
[6] `http://www.citeulike.org/`
[7] `http://arxiv.org/`
[8] `http://www.fedora-commons.org/`
[9] `http://www.rubyonrails.com/`

- **ActiveRDF**[10] [6] is a Ruby toolkit that provides an object-RDF mapping for a number of existing RDF triple stores.
- An **RDF store** is central to the application. During development the Redland RDF[11] engine is used, but Talia can work with any storage that is supported by ActiveRDF. The development team will also provide a setup for the Sesame[12], store which supports inferencing.
- Of course Talia will also need "normal" web application features, like user sign-on and permission management.

As a general rule, Talia tries to avoid any unnecessary complexity. This is also true for the use of RDF metadata:

- Talia is schema-agnostic. Ontology descriptions may be used by the system; however the software does not rely on their presence. It also doesn't attempt to enforce any kind of schema rules on the RDF data.
- Talia does not attempt to do any inferencing on the RDF data. If this is needed, it will be the responsibility of the RDF store.
- Talia tries not to use specialised RDF store features, in order to be compatible with any RDF endpoint.

3 RDF storage and querying

Talia uses a hybrid RDBMS/RDF store solution in the storage backend. A relational database as a highly reliable, transaction-aware storage for the critical data. It is kept in sync with an RDF datastore for advanced semantic features, which may range from SPARQL queries to inferencing, depending on the type of the store. Using a standard RDF store also provides an easy way to interoperate with semantic web software.

The hybrid design is feasible because a Talia library has relatively static content. Data access will mostly be read-only, the few modifications can be easily synchronised without much overhead for the system.

Talia provides a simple API that hides most of the internal workings of the data store. Each document is represented as an object of type `Source`; the object provides access provides access to all properties of the document, no matter if defined as RDF or not. Listing 1.1 shows an example of this API.

Listing 1.1. Basic Operations on a document

```
document = TaliaCore::Source.new("http://url.com/my_document")
document.workflow_state = 2 # non-rdf property
document.dcns::title << "My first document" # RDF property
author.inverse.dcns::creator # "inverse"
# Replace triple
document.dcns::title.replace("My first document", "New name")
document.dcns::title.remove # remove triples
```

[10] `http://www.activerdf.org/`
[11] `http://librdf.org`
[12] `http://www.openrdf.org/`

The interface borrows heavily from the ActiveRDF interface, and ActiveRDF is used in the backend to connect to the RDF store. However, ActiveRDF was designed mostly as an easy read interface for web applications. The library was substantially refactored to improve the data manipulation capabilities (such as deleting triples). Other modifications were made to make it easier to call the library indirectly as part of a backend, instead of directly as a part of a script.

3.1 Queries

Talia provides an unified query interface for both database and RDF metadata, as shown in listing 1.2. For normal queries, the interface hides most of the complexity and automatically decides wether to use a RDF/SPARQL or an RDBMS/SQL query on the backend.

Listing 1.2. Querying for documents

```
# The following will do a query on the RDF store
TaliaCore::Source.find(:all, N::DCNS.Creator => "Daniel")
# The following will LIMIT a query that uses RDF and DB data
TaliaCore::Source.find(:all, N::DCNS.Creator => "Daniel",
                       :workflow_state => 2, :limit => 5)
```

During development we found that the SPARQL [7] query language is not always best suited for web applications, where large result sets are usually broken down into individual *pages*. This is usually done by using the *LIMIT*, *OFFSET* and *COUNT* operators to retrieve the overall size and a subset of the (possibly huge) result set. This method requires, however, that whole query can be executed as a single statement.

SPARQL does not provide an easy way express *OR* statements in a single query (for example "subtype OR supertype"), and it's *filter* mechanism is highly inefficient for large result sets. A straightforward *COUNT* implementation is also missing from the standard. Another problem is that in different RDF stores the SPARQL specifications is implemented to various extents, making it difficult to provide a store-agnostic solution.

Early versions of Talia also encountered the problem of reconciling "mixed" queries that use both the database and the RDF store. With the new hybrid design it will be possible to answer each query either from the RDF data or the database. This will allow the backend engine to select the query language best suited for the job.

The built-in query mechanism is optimised for compatibility with a number of RDF stores and RDBMS. If this is too limiting the developer has the choice issue queries (either SQL or SPARQL) directly and access store-specific features.

3.2 Ontologies

As mentioned, Talia's core functionality does not rely on an ontology description. Still, if one is needed, it can easily be loaded into the RDF store and queried from Talia.

Talia provides a `SourceClass` abstraction to represent RDF classes and to navigate the ontology hierarchy, as shown in listing 1.3. The user may also retrieve metainformation from the ontology, supertypes or subtypes of a class.

Listing 1.3. Navigating the ontology

```
# Get the first rdf type and get sub- and superclasses
first_class = document.rdf_types.first
sup = first_class.supertypes
```

4 Semantic UI templates

`Source` objects can be used in HTML templates to create a web representation a document. Talia uses Rails' standard *rhtml* templates that contain HTML with embedded ruby code. Listing 1.4 show a simple template that renders a HTML snippet with some properties from the RDF store.

Listing 1.4. Sample rendering template

```
<p> The current document is <%= document.rdfs::label.first %>
it's authored by <%= document.dcns::creator.join(",") %> </p>
```

Talia needs a rich user interface for each document type. Unlike many semantic web applications that use an "auto-generated" generic interface for semantic metadata, Talia needs to provide specialised views depending on the RDF type of a resource.

A *automatic semantic template* engine is built into Talia to do just that. Listing 1.5 shows a simple example; the site developer simply passes the `Source` object to the `source_snippet` UI widget, and the semantic template engine does the rest.

Listing 1.5. Rendering a document with a semantic template

```
<% my_document = TaliaCore::Source.new(url,
                                       N::MYONT::the_type) %>
<%= widget(:source_snippet, :source => my_document) %>
```

When the template engine renders a document, it will look at the document's RDF type and attempt to find a template which has a name that matches the namespace and name of one of those types. In the example the document has the type `myont:the_type`; if the template engine finds a template named `_myont_the_type.rhtml` it will use it to render the document. Otherwise it will fall back to a default template.

The template engine allows the UI templates to be easily created by professional web designers who don't know semantic web concepts. In the final web application, template selection happens automatically and it's very easy to add new templates. By providing a sensible default template, the engine is still able to deal with elements of new and unknown types.

5 Conclusions

This paper showed a quick glimpse of Talia's semantic web features and demonstrated how semantic web development can be made a breeze by combining the power of an existing framework with a dynamic RDF store API.

More semantic web features will be included in future version, like semantic links between remote libraries, user-created metadata and integration with the DBin desktop application.

Talia is freely available from its home page[13], the page also contains some instructions and additional documentation for the software. There's also an online demo of the current development version[14].

Acknowledgements

This work has been supported by Discovery, an ECP 2005 CULT 038206 project under the EC eContentplus programme.

The authors wish to thank Eyal Oren and the ActiveRDF team for their work and support.

References

1. D'Iorio, P.: Nietzsche on new paths: The hypernietzsche project and open scholarship on the web. In Fornari, C., Franzese, S., eds.: Friedrich Nietzsche. Edizioni e interpretazioni. Edizioni ETS, Pisa (2007)
2. Risse, T., Knežević, P., Meghini, C., Hecht, R., Basile, F.: The bricks infrastructure - an overview. In: The International Conference EVA, Moscoww (2005)
3. Kruk, S., Woroniecki, T., Gzella, A., Dabrowski, M., McDaniel, B.: Anatomy of a social semantic library. In: European Semantic Web Conference. Volume Sematic Digital Library Tutorial. (2007)
4. Reenskaug, T.: MVC Xerox Parc 1978-79. `http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html` (1979 [accessed March 2008])
5. MIT: MIT License. http://www.opensource.org/licenses/mit-license.php ([accessed March 2008])
6. Oren, E., Debru, R., Gerke, S., Haller, A., Decker, S.: ActiveRDF: Object-Oriented Semantic Web Programming. In: 16th International World Wide Web Conference (WWW2007), Banff, Alberta, Canada. (8-12 May, 2007) 817–823
7. : SPARQL Query Language for RDF. http://www.w3.org/TR/rdf-sparql-query/ (January 2008 [accessed March 2008])

[13] `http://trac.talia.discovery-project.eu/`

[14] `http://demo.talia.discovery-project.eu/` - note that this site may not always be online