CURIOS

Nguyen, Hai H.; Taylor, Stuart; Webster, Gemma; Jekjantuk, Nophadol; Mellish, Chris; Pan, Jeff Z.

Published in:
CEUR Workshop Proceedings

Publication date:
2014

Document Version
Publisher's PDF, also known as Version of record

CURIOS: Web-based Presentation and Management of Linked Datasets

Hai H. Nguyen[1], Stuart Taylor[1], Gemma Webster[1], Nophadol Jekjantuk[1], Chris Mellish[1], Jeff Z. Pan[1], and Tristan ap Rheinallt[2]

[1] dot.rural Digital Economy Hub, University of Aberdeen, Aberdeen AB24 5UA, UK
[2] Hebridean Connections, Ravenspoint, Kershader, Isle of Lewis HS2 9QA, UK

1 Introduction

A number of systems extend the traditional web and Web 2.0 technologies by providing some form of integration with semantic web data [1,2,3]. These approaches build on tested content management systems (CMSs) for facilitating users in the semantic web. However, instead of directly managing existing linked data, these systems provide a mapping between their own data model to linked datasets using an RDF or OWL vocabulary. This sort of integration can be seen as a read or write only approach, where linked data is either imported into or exported from the system. The next step in this evolution of CMSs is a full integration with linked data: allowing ontology instances, already published as linked data, to be directly managed using widely used web content management platforms. The motivation is to keep data (i.e., linked data repositories) loosely-coupled to the tool used to maintain them (i.e., the CMS).

In this poster we extend [3], a query builder for SPARQL, with an update mechanism to allow users to directly manage their linked data from within the CMS. To make the system sustainable and extensible in future, we choose to use Drupal as the default CMS and develop a module to handle query/update against a triple store. Our system, which we call a *Linked Data Content Management System* (Linked Data CMS) [4], performs similar operations to those of a traditional CMS but whereas a traditional CMS uses a data model of content types stored in some relational database back end, a Linked Data CMS performs CRUD (create, read, update and delete) operations on linked data held in a triple store. Moreover, we show how the system can assist users in producing and consuming linked data in the cultural heritage domain and introduce 2 case studies used for system evaluation.

2 Using CURIOS

We introduce CURIOS, an implementation of a Linked Data CMS.[3] A dataset managed by CURIOS needs to have a structure described by an OWL ontology that imports a small CURIOS "upper ontology". It must relate some of its classes and properties to constructs in that ontology. This has the benefit that they can

[3] Available open-source at `https://github.com/curiosproject/curios`.

be recognised and treated specially by the generated website. For instance, an image can be presented in a special way (see Fig. 1) if it is an instance of the `hc:ImageFile` class and its URL is provided by the `hc:URL` property.

Once the ontology is defined, it is necessary to provide a separate description of which parts of the data (and the level of detail) are to be managed by the website. This description takes the form of an application-dependent *configuration file* which is loaded as part of the Drupal module. This file describes the classes, fields, and relationships to be shown in the website and how these relate to the constructs of the ontology [4]. Although the configuration file could be generated automatically, it is a declarative description and can easily be edited by hand. The configuration file centralises the maintenance of the structure of the CMS with respect to the ontology, e.g., if a new type of page is required, the user can update the configuration and then run the Linked Data CMS mapping to create the required Drupal entities. Additionally our approach can handle some changes to the schema of the ontology. For example if a change in the ontology occurs, such as a domain/range, additional classes or a change of URIs, then the configuration can be reloaded to synchronise Drupal with the ontology schema.

When the CURIOS Drupal module is initialised, it automatically creates a set of Drupal resources and Views based on the configuration file, along with an additional set of pages allowing the linked data to be maintained via CRUD operations. Drupal site administrators can then maintain the website generated by the configuration in the same way as a regular Drupal site.

2.1 Browsing and Update Functionalities

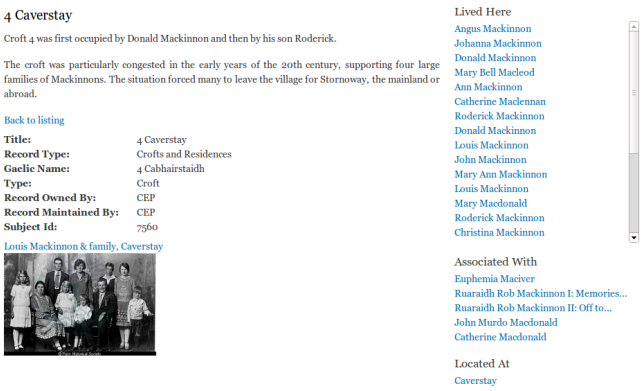
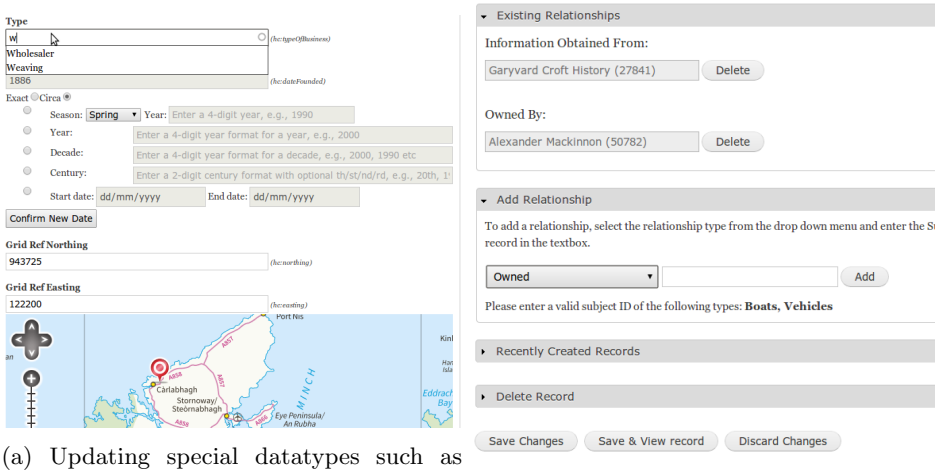


Figure 1: Details of a croft

CURIOS allows users to browse and update their linked data in a triple store directly without transforming RDF triples to Drupal content and vice versa. A CURIOS record consists of a set of RDF triples where the subject is a unique URI representing the record identifier. For browsing, depending on

different conditions, CURIOS presents data in different ways. For instance, a list of records or details of a record (see Fig. 1) will be displayed depending on whether the record URI is provided. To navigate between linked individuals, object properties of an RDF individual are presented as hyperlinks to other records instead of the normal text used for datatype properties.

Users are also able to create, update, or delete a record via a user-friendly GUI. Firstly, CURIOS assists users entering data by providing different widgets depending on the datatype the user wants to edit (Fig. 2a). For instance, with geographical coordinates, a map is displayed to allow users to choose a location rather than to type in the coordinates as text. Secondly, to prevent users from entering incorrect values for some special properties such as an occupation or a type of place, an auto-complete widget is provided. Thirdly, it is typical that in the cultural heritage domain, temporal data such as dates are rather vague and not recorded in a consistent format. To facilitate users during data entry process, CURIOS provides a simple treatment to vague dates by introducing the `hc:DateRange` class which consists of two datetime datatype propertes: `hc:dateFrom` and `hc:dateTo`. A user can enter an exact date or a vague date such as a year, a season in a year, a decade, a century, etc, and CURIOS can convert the vague date into an appropriate instance of `hc:DateRange`. Finally, to manage object properties (i.e., links) between individuals, CURIOS allows property add and remove operations as presented in Fig. 2b, which are then mapped onto corresponding SPARQL update queries, e.g., `INSERT` and `DELETE`, to insert and remove appropriate triples.



(a) Updating special datatypes such as dates, geographical coordinates, etc.

(b) Adding/removing object properties

Figure 2: Updating Linked Data in CURIOS

2.2 Use of the Triple Store

Although in principle we could use any SPARQL 1.1 compliant triple store/SPARQL server, in practice we are using Jena Fuseki [5]. The reasoner in Fuseki

creates a complete graph of all the consequences of the stated information when a SPARQL query is presented, and this is kept in a memory cache. Unfortunately, this reasoning has to be repeated after an update has been performed, and especially with complex updates, this can take an unreasonable amount of time that can affect the website's responsiveness. Also, although one ideally wants to show a user all the inferred information in order that they have an accurate model of the system's knowledge, if they are allowed to specify arbitrary updates on this then they may remove a piece of inferred information which is then re-inferred whenever the reasoner is next invoked. For these two reasons, we perform all updates via a Fuseki endpoint where no reasoning takes place. A second endpoint, where reasoning is enabled, is used for normal browsing. With this method, the information shown for browsing gradually becomes out of date as nothing prompts the recomputation of the inference graph. This is overcome by allowing the user to explicitly invoke the recomputation or by having a process outside of the website causing the recomputation at regular time intervals.

3 Case Studies and Future Work

To test the generality of our system, we conduct 2 case studies, one involving historical societies based in the Western Isles of Scotland (Hebridean Connections) and another one with the local historical group at Portsoy, a fishing village located in the North East of Scotland. The dataset used in the Hebridean Connections case study consists of over 45,000 records with about 850,000 RDF triples (before inference), incorporated within a relatively simple OWL ontology. The Portsoy case study uses a similar ontology with 1370 records and 23258 RDF triples (before inference). The Drupal website which we have built with the software is already being used by Hebridean Connections at `http://www.hebrideanconnections.com`.

In future we plan to make the system easier to set up for naïve users as well as to evaluate our system with different SPARQL servers/RDF stores.

Acknowledgements The research described here is supported by the award made by the RCUK Digital Economy programme to the dot.rural Digital Economy Hub; award reference: EP/G066051/1.

References

1. Krötzsch, M., Vrandecic, D., Völkel, M.: Semantic MediaWiki. In: ISWC. (2006)
2. Corlosquet, S., Delbru, R., Clark, T., Polleres, A., Decker, S.: Produce and consume Linked Data with Drupal! In: ISWC. (2009)
3. Clark, L.: SPARQL Views: A Visual SPARQL Query Builder for Drupal. In Polleres, A., Chen, H., eds.: ISWC Posters&Demos. Volume 658 of CEUR Workshop Proceedings., CEUR-WS.org (2010)
4. Taylor, S., Jekjantuk, N., Mellish, C., Pan, J.Z.: Reasoning driven configuration of linked data content management systems. In: JIST 2013. (2013)
5. Seaborne, A.: Fuseki: serving RDF data over HTTP. `http://jena.apache.org/documentation/serving_data/` (2011) Accessed: 2012-10-27.