Embedding an ontology in form fields on the web

Georg Hohmann\textsuperscript{a}, Mark Fichtner\textsuperscript{b}
\textsuperscript{a}Germanisches Nationalmuseum Department of Museum Informatics, Kartäusergasse 1, 90402 Nuremberg, Germany, g.hohmann@gnm.de
\textsuperscript{b}Zoologisches Forschungsmuseum Alexander Koenig, Department of Biodiversity Informatics, Adenauerallee 160, 53113 Bonn, Germany, m.fichtner@wiss-ki.eu

Abstract. Data on the Semantic Web is usually drawn from traditional database systems. Form fields are the typical input method for data gathering. Native data gathering methods for semantic data are rare. One of the main problems is the entry barrier when it comes to data generation. This paper describes how form fields can be used to gather semantically enriched data. Furthermore it describes how the reversed approach can be used to semi-automatically create XHTML web forms from OWL DL ontologies in a knowledge system based on semantic web technologies. The described method is illustrated by an extensive example taken from field of documentation in the cultural heritage domain.

Keywords: Ontology, Digital Cultural Heritage, User Interface, Mapping, Semantic Web, Museum, Forms

1. Introduction

Museums among others are the richest institution in data acquisition. Many objects in museums are subject of research projects which generate tons of data. In these data complex dependencies between historical people, objects, locations etc. are built. The semantic web would be a perfect environment for the data. However this data hardly is brought to the semantic web but published in catalogues and other paper based data carriers. The problem is not that the researchers in the museums are not used to digital data retrieval. They often use databases to gather the required data for the catalogues as data storages, but the databases are hardly published as they do not comply with the demands on quality of the researchers or of the museum. Furthermore even if the database would be suitable for publication, additional effort must be taken to publish the data complying with the according formats for the semantic web.

Another reason is that it's hard for people to think in networks composed of properties, classes and instances as they are used to gather data by forms for a long time. Researchers in scientific institutions who are used to gather data on their research topic are often not willing to learn new ways to do it. In order to lower the entry barrier for these groups of people the gap between web 1.0 technologies and the semantic web approach must be closed. There are approaches to compensate people's discomfort in hiding those networks behind modern web forms but this often requires to simplify the underlying semantics [1] or to manually add additional information to the ontology [2].

This paper introduces an approach that embeds ontology-based patterns for the generation of semantic enriched data. As a second step these patterns are used for the semi-automated creation of web forms from a complex OWL DL [3] ontology without the need to simplify or prune the semantic model. It is illustrated by an example taken from the field of documentation in the cultural heritage domain. In this domain the common way to gather data is to enter data manually in more or less elaborated data entry forms based on traditional database systems. Furthermore this domain provides a well-defined and detailed ontology called CIDOC Conceptual Reference Model (CRM) [4]. The approach is exemplified by creating web forms based on existing data entry forms of established
documentation systems which are semantic enriched using an OWL-DL implementation of the CRM.

To make this paper self-contained the used ontology language and the ontology are briefly described first. The following example of a form currently used for data acquisition in the cultural heritage domain serves as a template for analysis of the inherent semantics. Afterwards, these semantics are mapped to the given ontology. Using these mappings a web form is created and semantically enriched while the web form elements are controlled by the ontology's constraints and restrictions.

2. Prerequisites

2.1. Web Ontology Language

Ontologies in computer science are based on formal logic with well-founded semantics. The Web Ontology Language (OWL) [3] developed by the World Wide Web Consortium (W3C) is the most common knowledge representation language on the Semantic Web. Version 1 became a W3C recommendation in 2004 and was extended in 2007 to become a recommendation as OWL 2 [5] in 2009. The approach described in this paper deals with OWL 1 but could be easily adopted to OWL 2.

OWL 1 is a family of the three languages OWL Lite, OWL DL and OWL Full. The later covers all language features, whereas OWL Lite is just a small subset. The DL in OWL DL stands for Description Logics, so it has some special characteristics that make it appropriate for knowledge engineering and reasoning. OWL DL 1 is a syntactic equivalent to the DL SHOIN(D) which allows decidable inferences, but high expressivity. Ontologies in OWL DL have a lot of advantages: The syntax is readable by computers and humans and the formally defined semantics give a clear meaning to expressions. The computation of semantics based on OWL DL supports modelling and therefore the creation of these ontologies. There are already implemented tools like editors, reasoners and other semantic web software.

2.2. CIDOC Conceptual Reference Model

The CIDOC Conceptual Reference Model [4] is an ontology that has been developed for more than ten years by knowledge experts from museums, archives and libraries in conjunction with philosophers and computer scientists. This group is a working group of the International Committee for Documentation (CIDOC) as part of the International Council of Museums (ICOM). The CRM claims to be a formal ontology intended to facilitate the integration, mediation and interchange of heterogeneous cultural heritage information. In 2006 it has been accepted as ISO-standard 21127. Nowadays, it is wide spread and often taken into account for developing information management systems in the cultural heritage domain.

The current version 5.0.2 defines 86 so-called entities (i.e. classes or concepts) and 137 properties (i.e. relations or roles), each of them provided with constraints, explained by a scope note and illustrated with examples. A special feature of the CRM is its event-centric approach [6]. Every aspect of a real world item is connected to an event. The events serve as “mediation”-classes which connect actors with things and can be related to specific time-spans and places.

2.3. Erlangen-CRM

The CRM is an abstract model that is only provided as a paper document. As the presented approach is based on OWL DL, the Erlangen CRM/OWL\(^1\) (ECRM) was chosen as an OWL DL implementation of the CRM. The ECRM incorporates all classes, properties, scope notes and examples of the CRM and claims to be as close to the definition text as possible. This also implies the definition of cardinalities and constraints that are not explicitly defined in the CRM but mentioned in the scope notes [7]. The ECRM is actively maintained, updated to the latest version of the CRM and the only complete OWL DL implementation of the CRM, which makes it suitable for the approach described below. The ECRM is open source and free for use.

\(^1\)http://erlangen-crm.org
3. Digital documentation in museums

One key task of museums is to inventory and document the objects of its collections. The purpose of an inventory is to support the identification and the retrieval of objects. The documentation of an object is the primary place to keep information about an object. It gathers as much data as possible on every aspect of the object and its history. Not only the object itself, how it looks like or how it was made is documented, but also the original and historical context of an object is described: persons, places, events, other objects that are in either way connected to the object. The traditional way to organise a museum inventory is the use of file card sorted by a specific aspect, which can be the inventory number, the artist name or the material it is made of. File cards hold primary information about an object, such as title, creator or dimensions, which allows identifying a specific object. The documentation of an object is often kept in separate folders. In that case the file cards reference to the folder where the scientific documentation of an object can be found.

Today the inventory and the documentation rely on digital systems. There are plenty of modern museum management systems available which have the capability to hold all the information and documentation of an object all in one [CHIN 2003]. One of the most applied systems world-wide is KESoftware’s Electronic Museum (KE Emu), followed by GallerySystems’ The Museum System (TMS), PastPerfect and Adlib Museum. In Europe and especially in Germany additionally Startext’s HiDA, Zetcom’s MuseumPlus and imdas pro are widely used [Withthaut 2004, p. 41].

---

http://www.kesoftware.com/emu-home.html
http://www.gallerysystems.com/tms
http://www.museumsoftware.com/pastperfect5.shtml
http://www.adlibsoft.com/products/museum-software
http://www.startext.de/produkte/hida/hida
http://www.zetcom.com/museumplus
http://www.imdas.at/index.php?id=297
Although these software systems are quite different in functionality and layout they all rely on relational database as backend. Furthermore their primary interface for data entry mimics the look and feel of file cards.

The data entry forms shown in figure 1 and 2 are examples of forms provided by systems that support the documentation in museums.

The form is composed of several labels and associated textfields. The inherent semantics of this form is encoded in the meaning of the labels, the grouping of fields using fieldsets and the order of fields [8]. Additional semantics are provided in the manual.

In order to map these semantics to an ontology the semantics must be expressed explicitly as instances of classes connected by properties of the ontology. Without loss of generality, the CRM will be used as reference ontology for the following mappings. Based on this example, a general approach for embedded mappings in form fields and ontology driven web forms is suggested.

4. From form fields to semantic data

The overall semantics of the form in figure 2 is rather complex. The form is used to gather information on a museum object. The first rather common, but implicit assumption humans make is that all data of a form is about the same museum object. The further semantics is not quite clear if only the label of the field is provided. For example the first field labelled “Institution” may describe the legal owner, the current possessor or even the current location. The semantics of the field “Distinguishing
features” even only is clear to special scientists of the domain. However there are easy fields to start with, too.

The second field “Object Number” clearly is used for input of an identifier of the object. The “Object Number” in museum documentation is a usually unique inventory number that is used to reliably identify an object. In the CIDOC CRM there is a class for all identifiers called “E42 Identifier”. In order to construct the semantics of the field in the terms of the CIDOC CRM, whenever the field is filled, an instance of the class E42 Identifier must be generated and linked to the instance of the museum object via the appropriate property. Moreover the string value filled in the field can be stored by linking it to the instance of E42 Identifier by a datatype property.

Fig. 3. Instances of the museum object, its identifier and the value

This is a very single graph of CRM instances linked by properties. Due to the disunion of the identity of the object and its name, the CRM creates rather complex networks of classes, properties and primitive data.

More complex structures are needed if the form is subdivided in sub-forms. Consider the fieldset “Production” from figure 2. All fields of this field set are not only about the same museum object, but the data in these fields is related to each other by the production. As told before, the CIDOC CRM is an event-centric ontology. Therefore every action taken with a museum object in our example has to be modelled as an event. A production is a special case of a creation of an object. The production is related to the person who produces the object, the time and place when and where the production took place and of course to the object itself. As all information in the fieldset is about the production, a filled fieldset equals the following graph of instances that can be seen in figure 4.

5. Patterns for instance creation: Paths through the ontology

As we have seen from the example in chapter 4, data in common semantic web frameworks is stored in an appropriate database backend by creating instances of certain classes of the chosen ontology and linking them with other instances or primitive data by properties of the ontology. In contrast to this, form fields usually store their data in more common and widely spread database backends.

In order to bring these fundamental different technologies together, a connection between the form field and the storing backend must be established. Moreover there must be mapping instructions for the semantics of every form field. These mapping instructions must be process able for a machine. Whenever a form field is filled with a value and the form field is saved, the mapping must be applied to the form field and the value of the form field. The result of the mapping is a set of instances of classes, linked by properties. This set can then be stored in a database backend suitable for the semantic web.

To develop a deeper understanding of this idea, the example from figure 4 must be extended by the mapping, which must be applied whenever the form is saved. For the sake of simplicity, let’s assume the form only consists of the field “Object Number”. This field with its semantics fully expressed in classes of the CRM and the according instances linked by properties of the CRM can be seen in figure 5.
Whenever this form is saved, a mapping must be carried out. The mapping consists of very basic information on the instances that have to be generated. In our example the mapping is a conceptual pattern, meaning a composition of classes, properties and primitive data types of the ontology that can be seen in the middle of figure 5. For every class of this pattern an instance has to be generated. After this the instances are linked by the given properties that connect the classes of the conceptual pattern. The data value of the form field is mapped to the primitive data value at the end of the conceptual pattern. Such a conceptual pattern is called a “path” as it connects the concept of the source object in mind (the museum object) with the primitive data humans assert (i.e. the string holding the primary identifier). Thus a path is a set of classes of an ontology mentioning concrete properties for linking the classes including primitive data types as endpoints. When data is stored according to this path, instances of every class in the path are generated and linked by the given properties. A set of instances based on the classes of the path, with the properties of the path connecting the instances is called a concrete representation of the path.

5.1. Grouping of paths

However forms usually do not consist of only one form field but of many form fields. For every form field a certain path has to be given to realize the network of the data. The paths of all these form fields usually share a common root - the object the assertions are made about. This common root is a logical connection between the paths. All paths which share such a logical connection construct a “group” of paths. Every path contained in a group shares a common part beginning at the root of the path with all other paths contained in the group. Vice versa a path may be part of a group, if it contains the common part of all other paths. A set of instances matches a group if the paths constructed from the instances all share the same common part of the group.

Figure 6 shows the already noted ontology path extended with CRM classes and properties that express the semantics of the field "Title". Both form fields make assumptions on the same museum object, thus both paths share the CRM class "E84 Information Carrier" as a starting point. If instances are created, it must be guaranteed that only one instance of the class E84 (only one museum object) is created which is assured by grouping these paths.

5.2. Grouping of groups

This technology however still cannot handle the example in figure 3 as on the one hand all paths are about the same museum object and on the other hand all paths of the fieldset “production” are about the same production event. This forces the need of subgraphs in graphs.
In our case a network of classes must be used for the description of the production event, e.g. the actor has a name; the production took place at a certain place with a name and so on. Transferred to the concept of groups and paths, this means that the group for the museum object must have a subgroup describing the production event. This graphical representation of this subgroup can be seen in figure 7. The common part of every path in the group describing the production event is the set of concepts and properties that connects the museum object with the production event. This mechanism ensures that all data is recorded on the same instance of the production event. Accordingly further subgroups could be added to subgroups, too. If the form needs to be extended to collect more information on the actor like the date and place of birth, a subgroup for the actor can be established.

5.3. Using the technology

As described above, the semantics of a form field is usually bound by the label of the form field and every form has implicit semantics e.g. logical groupings like field sets, indentations etc.

For nearly every form a set of paths and groups can be given to map the semantics of the form. First all paths have to be constructed, describing the classes for the instances which have to be created, the properties which must be filled and where the primitive value has to be stored in the resulting graph.

In order to "semantify" a form field, a path has to be embedded.

The next step is to group the according paths that describe different aspects of the same object together. As every filled form field creates a concrete representation of a path and all paths should describe different aspects of the same object, they share a common root. This is equivalent to the definition of a group. Such groups are typically visualized by fieldsets or indentations.

Although the examples only show input fields, this approach can be generalized to all types of form fields. Text areas and password fields can be handled analogously to text fields. Drop-down menus, select boxes and radio buttons need an additional option list though. This option list can be generated by using the path as a search pattern in the existing data. All sets of instances which fulfil the structure of the path are valid options for these types of form fields. For an easy understanding, not the whole path but the xsd:string should be shown in this cases.

5.4. Disambiguation in paths and groups

In the above example the name of the creator of a collection object is recorded. However, the semantics of the form field is not to record the name of the creator of an object, but to establish a connection between a certain person, identified by his/her name and the collection object. If the concerned person does not already exist in the database, it is save to create a new instance. If there already is an instance
for the person, the user usually wants to refer to the instance of the person and not to the instance of the person’s name or even just to the string. Thus in this case a certain class in the path has to be marked. At this class the path is separated in two parts. The first part reaches from the starting class up to the marked class, the second part from the marked class up to the primitive string. The second part is then used for searching in the existing data. If any graph is found that matches this part, the first part is generated and the just found instance is used as filler for the marked class. If several graphs are found, the user has to decide which graph should be used.

In order to make this feature more comprehensible for the user, the options of the autocomplete function of input fields or the choices for radio buttons can be filled with the results of the search. As a result, the user is supported to automatically disambiguate the identity of an instance, and semantically correct graphs are generated.

6. Reversing the approach: Generating form field features from the ontology

This technology is able to map semantics to a given form. However the approach can also be reversed. For a given ontology you can construct mappings and therefore build input forms for the ontology. This gives the possibility to directly record semantically enriched data in the familiar environment of forms.

Moreover certain parts of an OWL DL ontology can be used to automatically support special features of the form fields. First, restrictions on properties affect the form fields. In [9] owl:maxCardinality is defined to be used for an upper bound and owl:minCardinality to be used for a lower bound. If the lower bound is larger than 0, the property must be filled. Transferred to form fields, this means that filling the form field is required. If the lower bound is n this means the form field is duplicated n times and the user has to fill every single form field to construct a valid set of data. The upper bound tells how often a property may be used. An upper bound of n means that a form field may be only duplicated n times. Lower and upper bound can be combined.

Second, it describes owl:oneOf for enumerated classes. If owl:oneOf is used, the range of a property is limited to the instances of the given set. A similar behaviour can be seen at special form fields like (single-) select fields and radio buttons. They have a closed set of options and the user can chose one of this options. So whenever owl:oneOf is used in the ontology, the form field usually is a select field or a set of radio buttons.

<table>
<thead>
<tr>
<th>OWL DL constraint</th>
<th>Web form field feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>owl:minCardinality 0</td>
<td>optional, repeatable form field or group</td>
</tr>
<tr>
<td>owl:minCardinality n</td>
<td>n times mandatory, repeatable form field or group</td>
</tr>
<tr>
<td>owl:maxCardinality 1</td>
<td>optional, not repeatable form field or group</td>
</tr>
<tr>
<td>owl:maxCardinality n</td>
<td>optional, up to n times repeatable form field or group</td>
</tr>
<tr>
<td>owl:minCardinality 0 &amp; owl:maxCardinality 1</td>
<td>optional, not repeatable form field or group</td>
</tr>
<tr>
<td>owl:minCardinality 1 &amp; owl:maxCardinality 1</td>
<td>mandatory, not repeatable form field or group</td>
</tr>
<tr>
<td>owl:cardinality 1</td>
<td>mandatory, not repeatable form field or group</td>
</tr>
<tr>
<td>owl:minCardinality n &amp; owl:maxCardinality n</td>
<td>n times mandatory, n times repeatable form field or group</td>
</tr>
</tbody>
</table>

Table 1. OWL DL constraints and their implications for the creation of form fields
Third, OWL DL can support the selection of primitive data types for data storage. The primitive data types are linked to classes by data type properties which have a concrete data type as their range. Usually one class only has few data type properties available. If there is only one data type property available at the end of a path, this data type property and the according primitive data type can be selected automatically.

Moreover the system can implement checks for the validity of input. If a certain path ends with a data type property that restricts the input to the primitive data type xsd:date and a form field is related to this path, only dates are a valid input for this form field.

The following table lists OWL DL constraints and their implications for the creation of form fields

7. Conclusion

The feasibility of the approach has been proved by an implementation of a system that supports the semi-automated generation of web forms based on the concepts, properties and restrictions of OWL-DL ontologies [10]. The system provides an editor interface to create and manage forms that are built of ontology paths and validate data in respect to the restrictions of an ontology.

The semi-automated creation of ontology-driven web forms can sustainably lower the barrier for native data entry on the Semantic Web. Users are not bothered with new forms of knowledge networking and do not have to learn new technologies like OWL, RDF or SPARQL before recording their data. On the other side this approach enables web forms to interact directly with a native triple store without diversion. There is no need to add additional data to an OWL DL ontology or to convert them to another format. These forms do not force underlying ontologies to be simplified and support the full range of OWL DL expressiveness. Overall the ontology-driven web forms described in this paper could be a keystone to bring the Semantic Web closer to the ordinary web user without losing the power of knowledge representation, inference and reasoning.

References