Ganesha*

RDF/XML editor

Introduction

As more and more information is published on the Internet (especially on the World Wide Web) it is becoming increasingly important to be able to find specific and relevant data reliably and effectively. Furthermore it would be desirable to have tools that not only aid us in finding this information but also help with analysing or evaluating that data.

To achieve this it must be possible for computers to interpret information on the Internet so that they can help us work with it. Current search engines already do a fairly good job of 'understanding' web pages written in HTML or similar formats in order to index them, but there is much room for improvement and when faced with non-textual data such as images, sounds or video the search engines are generally at a loss.

Fortunately the World Wide Web Consortium (W3C)[i] has created the Resource Description Framework (RDF)[ii], which provides a standard way of publishing descriptive meta data (i.e.: data about data) for virtually anything. RDF is a key technology in the W3C's efforts to establish the so-called Semantic Web - "an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation."[iii].

Although RDF has existed for several years now it is not yet widely used. This is partly due to the fact that RDF (especially when encoded in XML) can appear quite complex and daunting. Average computer users do not want to be concerned with the gritty details of RDF and XML. Professional web designers and companies running large sites can afford to create custom tools to generate RDF data or to teach their staff the necessary know-how, but there appears to be no easy to use solution for individuals or maintainers of small to medium sized websites.

The intention of this project is to fill this gap with a simple, graphical user interface (GUI) based RDF editor. Primarily it will be aimed at individuals creating or maintaining small to medium-sized websites who are seeking a way to give more comprehensive meta information about their work than (X)HTML meta-tags are capable of. The hope is that just as simple WYSIWYG website tools made web-design accessible to the masses, so the tool resulting from this project will make creating RDF meta data simple enough for the average user.

---

*Since meta-data is all about letting computers 'know' more about data so that they can make people's lives easier by letting them find and analyse data more reliably the application that the project will produce is named after the Hindu god of knowledge and destroyer of obstacles - "Gane sha".

Project objectives

The final goal of this project is to produce an application capable of creating and editing RDF files. The project has been split into a number of objectives that will lead to this goal:

Obj. 1 Analyse the uses and capabilities of RDF

Obj. 2 Decide which features will be relevant to the target user

- Obj. 2.1 Determine user expectations from RDF
- Obj. 2.2 Determine potential difficulties of RDF for users
- Obj. 2.3 Decide which features of RDF will be most relevant to users

Obj. 3 Design the Ganesha application

- Obj. 3.1 Choose appropriate technologies for the application
- Obj. 3.2 Design structure of application
  - Obj. 3.2.1 Internals
  - Obj. 3.2.2 Externals (user interface)

Obj. 4 Create the Ganesha application

- Obj. 4.1 Code individual components according to designs from Obj. 3.2.1
- Obj. 4.2 Produce documentation of code
- Obj. 4.3 Debug individual components
- Obj. 4.4 Produce user manual for Ganesha

Obj. 5 Test the application

- Obj. 5.1 Test and debug application as a whole
- Obj. 5.2 Perform usability testing on application and user manual

Obj. 6 Present finished Ganesha application

Obj. 7 Produce final project report

Objective 1 – Analysis of RDF

"The RDF is a language for representing information about resources in the World Wide Web" - *W3C RDF Primer*[iv]. The identifier of each resource being described (usually its Uniform Resource Locator (URI)) is known as the *subject*. Each subject can have properties, referred to as *predicates*, which have associated values – the *objects*. This RDF model can be depicted as a graph:

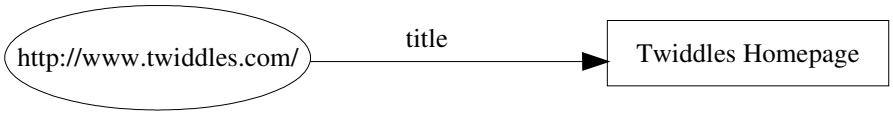

fig. 1 – Example of RDF model

In practise the predicates are taken from standardised meta-data vocabularies such as Dublin Core[v]. Many vocabularies exist for different areas of interest.

For use in computer systems RDF models can be serialised as N-Triples or XML. The RDF/XML for the above model could look like this:

```
<rdf:Description rdf:about="http://www.twiddles.com/">
        <title>Twiddles Homepage</title>
</rdf>
```

RDF/XML itself has several formats: normal and abbreviated. Both are equivalent and the two can be mixed as well.

Apart from literal values as in the example above the objects of an RDF resource can themselves be new resources or containers. Containers are a mechanism within RDF to have a sequence, a bag or a list of alternative resources as an object. A person being described in a RDF model may have multiple email addresses so in that case one could use the *alt* container to group them and indicate that they form a list of equivalent alternatives.

Objective 2 - Features

The Ganesha application will be primarily aimed at maintainers of small to medium-sized websites. This group includes individuals with personal web-pages, semi-professional web-designers and people new to web-design. It is fair to assume that many will prefer to use WYSIWYG (What you see is what you get) website tools and either have no or only a passing knowledge of HTML, XML or RDF syntax.

Regarding their knowledge and expectations of meta-data and RDF (Objective 2.1):

- Users' prior experience of meta-data will be limited to descriptions and keywords that they can embed in a web-page.
- They will expect RDF to be a more powerful and advanced way of doing this.

Difficulties preventing them from using RDF (Objective 2.2) are a lack of understanding of XML syntax, RDF models and associated terminology. Even if they have a passing knowledge of HTML the fact that there are several ways to serialise RDF data into XML and that XML is has a much stricter syntax than HTML will overwhelm them.

Of interest to the target users (Objective 2.3) will be the simple creation and editing of RDF data for web-pages and similar resources. Due to the widespread support of XML on the Web and its similarity to HTML only the RDF/XML serialisation will be supported. Users will not want to be bothered with the details of this so there will be no choice between abbreviated or non-abbreviated RDF/XML output and there will be no ability to see or edit the XML code directly.

Although many RDF meta-data vocabularies exist only few will be of interest to the target users. Dublin Core (for describing common attributes of documents) & VCARD (for describing contact details of a person) will be the most used. Therefore built-in templates for these vocabularies shall be incorporated into Ganesha. The import of arbitrary vocabularies will not be featured in Ganesha since this would create too much complexity for the target user (See design discussions of core feature 1.7 in the next section).

Beyond creating and editing RDF Ganesha will have some convenience features for working with HTML. It will be able to extract meta-data from meta tags in HTML pages to use in RDF and it will be able to insert a link into a HTML file to the RDF file containing its description.

For the further discussions the features will be grouped into core and optional features. Core features are the basic ones that must be present in order for the user to perform simple work with RDF files and thus will definitely be implemented by the project's completion. Optional features represent ones that go beyond the basic editing of RDF/XML files. Since these are not essential they will be implemented after the core features have been finished if time permits.

1 Core features
  1.1 Loading existing RDF/XML files
  1.2 Creating new RDF/XML files
  1.3 Navigating an RDF/XML file
  1.4 Editing individual resources
  1.5 Adding individual resources
  1.6 Creating bags, sequences and lists of alternatives within resources
  1.7 Providing templates for common RDF schemas
2 Additional features
  2.1 Linking to RDF file from HTML file
  2.2 Extracting meta-data from HTML files

Objective 3 - Design

The Ganesha application requires the use of technologies (Objective 3.1) that facilitate:

- Implementing a graphical user interface (GUI)
- Reading and writing RDF/XML data into and out of the program
- Manipulating RDF models within the program

Therefore the program shall be written in the Java programming language due to:

- Its widespread and cross-platform support
- GUI functionalities (Java AWT & Swing)
- Its free (as in beer) availability

The RDF functionality will be supplied by the Jena API[vi]. It was chosen due to:

- Its maturity and successful use in other RDF applications like BrownSauce[vii] and IsaViz[viii]
- Its open source license
- The quality of its documentation

Jena can read in RDF/XML data and convert it to an internal memory model that can Java programs can interact with. Using the API, RDF data can be queried, modified, deleted and created. The model can then be written back out to an RDF/XML file.

The internals of Ganesha (Objective 3.2.1) will be grouped into 3 layers. At a high level the application's code will be structured as follows:

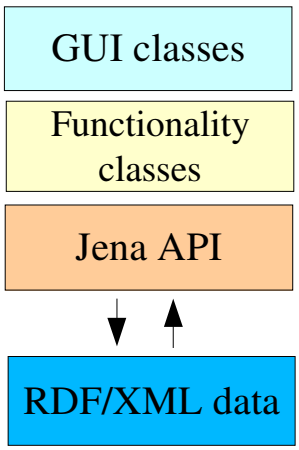

fig. 2 – Overview of Ganesha's structure

Starting from the bottom RDF/XML data can be read or written using Jena. The internal model of any RDF data that has been read into the program is maintained by Jena.

The features the user can access (as listed above) are implemented as methods within the functionality classes. Where those features involve working with RDF data the functionality classes will use Jena.

The GUI classes are responsible for presenting the user interface to the user and reacting to user input. Any events in the GUI will trigger method calls to the functionality classes. Since all features of Ganesha are implemented there no direct access to Jena is necessary.

The core features 1.1 – 1.7 will largely rely on Jena's basic RDF functionality as well as Java's file input and output classes. The ability to import new meta-data vocabularies were considered in Objective 2.3, however Jena does not support this directly. Meta-data vocabularies for RDF can be written as RDF Schemas[ix] and there is a tool called *schemagen*[x] that can turn RDF schemas into Java source-code for use with Jena. However, unless the user knows how to compile the source-code and install it properly it will remain useless. As a) compiling Java source-code is not an expected skill of the target user and b) we cannot assume that the full Java development kit is present on the user's machine so the process cannot be automated, the vocabulary import feature will not be present in Ganesha.

GUI designs (Objective 3.2)

The user interface of Genesha is designed to be as simple and intuitive as possible. Considering the target users (discussed in Objective 2.1) the learning curve for using Ganesha must be as flat as possible. Therefore the following basic HCI (Human Computer Interaction) prinicples are central to the design of each part of the GUI:

- Observability (The relevant parts of the GUI should be observable. At any point the user should be able to recognise where or at what stage they are, how they got there and what possible steps they can take)
- Consistency (GUI elements should look and behave consistently)
- Progressive disclosure (The user should have immediate access to basic functions and gradually be able to discover more advanced ones)

RDF by its very nature is quite abstract so there is no obvious metaphor that can be used to present it to the user. The use of RDF graphs would not be appropriate since a) the details of how RDF works are supposed to be hidden from the user and b) RDF graphs can quickly become large und unwieldy and we want to avoid information overload.

Showing a resource as a simple list of properties and values (i.e.: predicates and objects) seems the most simple and intuitive way to present the RDF data to the user. If the value happens to be another resource or an RDF container it can be represented as a link to a separate list of property/value pairs from that resource or container.

This is also in keeping with their expectations from WYSIWYG web-page tools as discussed above. Tools like Microsoft Frontpage and Macromedia Dreamweaver present HTML meta tags as property/value pairs too[xi].

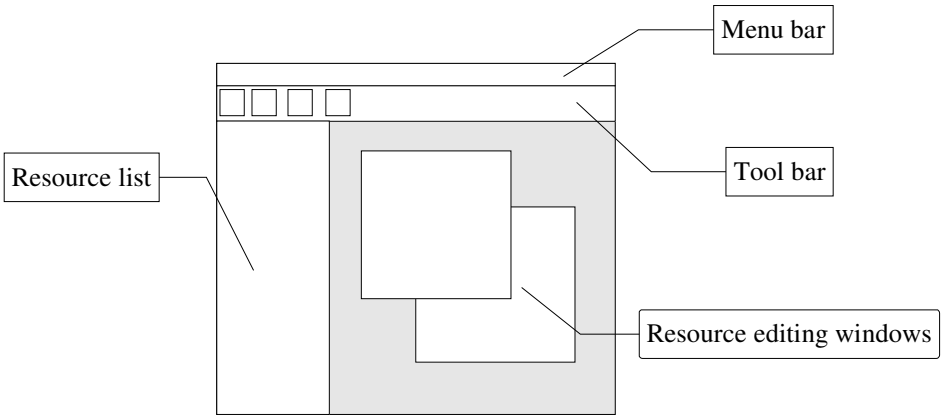

fig. 3 - GUI layout

Upon starting the program the GUI will present itself as in figure 3. Initially the resource list will be blank and there will be no resource editing windows. Any buttons or menu options that cannot be used at this point will be disabled.

The menu bar (see fig. 1) shall contain entries for opening existing or new files (core features: 1.1 & 1.2) and saving files, tidying the internal resource editing windows and other features. The positioning and naming of the menu entries will follow the recommendations in the "Java Look and Feel Design Guidelines"[xii] to provide consistency with other applications. Common commands from the menus shall be mirrored by buttons on the tool bar (see fig. 1) to provide quicker access.

Since they are more advanced, the optional features 2.1 and 2.2 will be present as menu items. That way the user is not confronted with them right away but can find them as the become more competent using Ganesha (progressive disclosure).

When a file is loaded the subjects of all resources in that file shall be listed in the resource list. This may look something like figure 4.

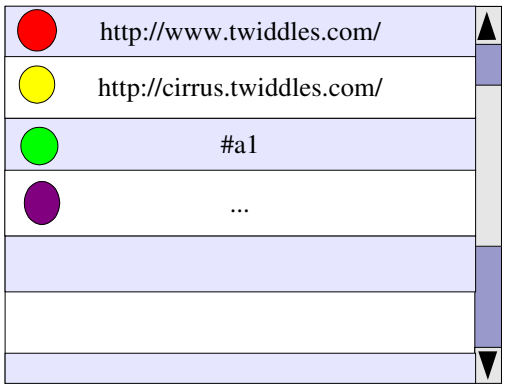

fig. 4 – Resource list with entries

From the resource list users will be able to select individual resources and either edit them (which pops up a resource editing window) or removing them (core feature: 1.4). Attempting to remove resources will prompt the user if he/she is sure they want to do it, thus avoiding frustrating the user by accidentally deleting things. There will also be a button at the bottom of the resource list to add new resources.

Each entry in the resource list will be marked with a coloured icon, chosen randomly by the program (see fig. 5) these coloured icons will appear in the title of the resource editing windows that corresponds to the resource in the list. This will aid the observability of the GUI since the user can associate the editing windows with the resource list entries by their colour.

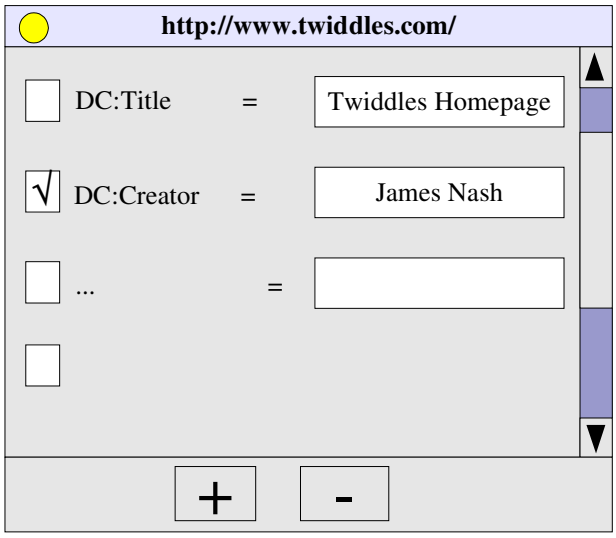

fig. 3 - Resource editing window

Displaying resources (or containers) one at a time in editing windows helps the user to focus on one piece of information at a time thus reducing the cognitive load required to work with Ganesha.

The editing window contains all the properties and values of the given resource in a list form. If the value is a literal then it appears as a simple text box that can be manipulated by the user. If it the value is another resource (and that resource is within the currently opened file) then a button will appear with that resource's subject and colour icon. Clicking it opens the resource editing window for that resource, just as if it had been chosen from the resource list.

So if the user had clicked the first resource (http://www.twiddles.com/) in figure 4. A window like figure 3 would appear. The underlying RDF/XML in this case would be:

```
<rdf:Description rdf:about="http://www.twiddles.com/">
        <DC:Title>Twiddles Homepage</DC:title>
        <DC:Creator>James Nash</DC:Creator>
        ...
</rdf>
<rdf:Description rdf:about="http://cirrus.twiddles.com/">
        ...
</rdf>
...
```

The properties and values of each property/value pair in the resource editing window also has a check-box associated with it. This enables the user to select one or more pairs and remove them by pressing the "-" button at the bottom of the window, just as he would do in the resource list to remove resources.

The "+" button brings up a wizard for adding new values (core feature: 1.5). The user will be prompted to choose the type (like DC:Title or VCARD:Name) and an initial value. Then the new value appears in the editing window.

The RDF containers bag, seq and alt will be available as types in the wizard too (core feature 1.6). After picking one the user will then need to choose a type for the values within the container. This will behave just as adding a single value except that the containers will be omitted as available types this time round. The resulting value in the resource editing window will be a button as if there was a nested resource. Clicking it will bring up a window with the container's contents that is almost identical to the resource editing windows – the only difference will be that the only new values of the same type as the rest in the container may be added.

These behaviours are consistent with what users will be used to from WYSIWYG web-site tools and may other applications.

Future work

The remaining objectives (4 and onwards) will be addressed during the rest of the project' s duration as described here:

Week...

3	4	5	6	7	8	9	10	xmas	11	12	13	14	15	16	17	18	19	20	easter	21	22
Objectives 1 – 3 (completed)																					
							Objectives 4.1 & 4.2														
								Objective 4.3													
											Objective 4.4										
												Objective 5.1									
														Objective 5.2							
																Objective 6					
																			Objective 7		

Progress news will be posted on the Ganesha website at:

http://event-horizon.kicks-ass.net/uni/ganesha/

i World Wide Web Consortium – http://www.w3.org/

ii W3C's RDF pages – http://www.w3.org/RDF/

iii Tim Berners-Lee, James Hendler, Ora Lassila, *The Semantic Web*, published in *Scientific American*, May 2001

iv W3C RDF primer - http://www.w3.org/TR/rdf-primer/

v Dublin Core Meta-data initiative - http://www.dublincore.org/

vi Jena Semantic Web Framework - http://jena.sourceforge.net/

vii BrownSauce RDF browser - http://brownsauce.sourceforge.net/

viii IsaViz Visual RDF authoring tool – http://www.w3.org/2001/11/IsaViz/

ix RDF Vocabulary Description Language - http://www.w3.org/TR/rdf-schema/

x Schemagen HOWTO - http://jena.sourceforge.net/how-to/schemagen.html

xi Adding meta tags in MS Frontpage - http://www.designsbyjoy.net/FrontPageTutorials/Keywords.htm

xii Java Look and Feel guidelines for common menus - http://java.sun.com/products/jlf/ed2/book/HIG.Menus2.html