Brainlets: "instant" Semantic Web applications

Giovanni Tummarello, Christian Morbidoni, Michele Nucci, Onofrio Panzarino

SeMedia group, Dipartimento di Elettronica, Intelligenza Artificiale e Telecomunicazioni, Università Politecnica delle Marche, via Breccie Bianche, 60100 Ancona
g.tummrello@gmail.com, c.morbidoni@deit.univpm.it, mik.nucci@gmail.com

Abstract
In this paper we present the "Brainlet" paradigm, a way to create rich Semantic Web user interfaces and interaction environments. Brainlets are half way between configuration files and light scripts and are "executed" by the DBin rich Semantic Web Platform. The main motivation behind Brainlets is enabling domain experts, rather than programmers, to create rich Semantic Web environments and communities. Brainlets can in fact be created simply by XML configuration files and XML based scripts along with the proper ontologies. Advanced Brainlets can be created based on the internal DBin API and/or the Eclipse Rich Client platform API. Brainlets are distributed as plug-ins and enable user communities, providing a common 'vision' of the domain and tools, to collectively create and exploit rich Semantic Web datasets.

1. Introduction

A fundamental design goal of the DBin rich Semantic Web platform [1] has been the ability to enable domain experts, rather than Semantic Web programmers and hackers, to independently start user communities about any topic of choice. We obtain this result with the definition of a scripting/configuration framework which enables the creation of rich Semantic Web interaction environments using tools such as XML and OWL editors. Such environments, which an end user might even perceive as a full featured domain specific applications, are named Brainlet. This paper illustrates their creation process and features.

Brainlet are distributed to the users as plug-ins (e.g. posted on a web page), thus enabling communities to share a common vision of the domain and a set of tools to collectively create and exploit rich Semantic Web datasets. In order to understand the need for the features offered by the Brainlet paradigm, lets first address the question: *in which way one can create a 'community' of 'Semantic Web users' ?*

Traditional Internet user aggregation channels such as IRC, newsgroups and web forums make it rather straightforward to create user communities but interactions among users are limited to textual messages. Finding information on such channels is, in the best scenario, only possible trough textual search, structured data is not tractable.

On the other hand, the goal of Semantic Web based user communities - supported by the DBin platform - is to enable the cooperative creation and exploitation of semantically structured knowledge bases. It is clear how a standard message board is a very simple subset of the possibilities enabled by such Semantic Web communities.

In this paper we assume that issues such as metadata exchange algorithm and trust based filtering are, among others, solved by the facilities which the DBin platform includes. These are the RDFGrowth [2] P2P algorithm, providing topic based *metadata exchange channels*, and the RDF Digital Signature methodology [3].

In this condition, Semantic Web communities startup basically boils down to create domain specific user interfaces which shields the users of the communities from complexities and at the same time provides domain specific browsing, editing and querying tools.

Providing a generic semantic web GUI is all but a trivial task; a lot of approaches have been proposed recently, among which [4], [5], [6], [7], [8]. While pro and cons can be argued for each specific solution, it is clear that user interface issues are complex ones and that it will be very difficult to think about a single solution that can be as usable as ad hoc ones for each specific domain. Aware of this, we propose an approach based on the aggregation of simple, generic components coordinated in a domain specific way by a mix of XML/OWL and scripting technologies.

2. DBin and background scenario

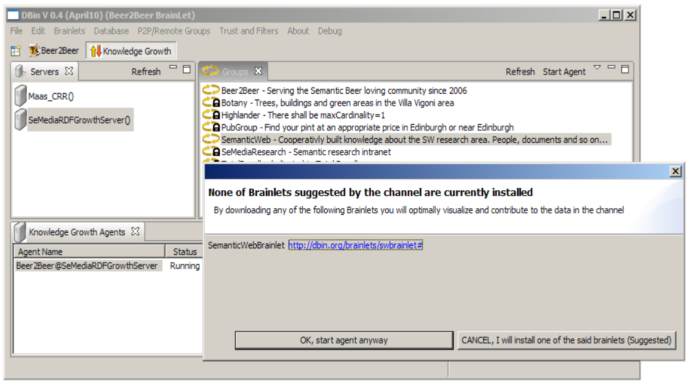
A typical use of DBin might be similar to that of popular file sharing programs, the purpose however being completely different. While usual P2P applications “grow” the local availability of data, DBin grows RDF knowledge. Once a user enters a metadata exchange channel, RDF annotations just start flowing in and out “piece by piece”.

For example, a user who expresses interest in a particular topic (say “Beers” as in DBin demo group) will keep a DBin open, possibly minimized, connected with a related P2P knowledge exchange channel where relevant information bits will be collected from and exchanged with other participants. Such information bits might be pure metadata annotations (e.g. “the alcoholic content of beer X is Y”) but also advanced annotations containing pointers to rich media (e.g. a picture of the glass, a PDF document, a Wikipedia page etc..).

With the support of the proper Brainlet, the user could then browse, reply or further annotate such information bits either for personal use or to contribute to the group knowledge. If such replies include attachment data (e.g. a picture), DBin automatically takes care of the needed web publishing using services such as that offered at *http://public.dbin.org* . At database level, the information is stored in DBin as RDF; At the user level, however, the common operations and views are grouped in domain specific user interfaces, the Brainlets. The installation of a specific Brainlet (e.g. targeted to the Beer domain), is not necessary to connect to the group and receive information but it enables users to visualize and edit such domain specific annotations, which, without proper domain knowledge and settings, are only 'row' triples in the DB.

3. Brainlets: creation, configuration and scripting

Brainlets are plug-ins (technically Eclipse Rich Client Platform [9] compliant plug-ins) that can be installed into the DBin platform simply copying a file in a proper directory. When a user selects a metadata exchange channel the remote server might suggest the use of a specific Brainlet to best experience the information contained in the group and participate in the annotation. Such suggestion is made by whoever created the group, very likely but not necessarily, the same person who created the Brainlet. Figure 1 shows an actual runtime screen-shot.



Beer2Beer - Serving the Semantic Beer loving community since 2006
Botany - Trees, buildings and green areas in the Villa Vigoni area
Highlander - There shall be maxCardinality=1
PubGroup - Find your pint at an appropriate price in Edinburgh or near Edinburgh
SemanticWeb - Cooperativly built knowledge about the SW research area. People, documents and so on...
SeMediaResearch - Semantic research intranet

Agent Name	Status
Beer2Beer@SeMediaRDFGrowthServer	Running

None of Brainlets suggested by the channel are currently installed

By downloading any of the following Brainlets you will optimally visualize and contribute to the data in the channel

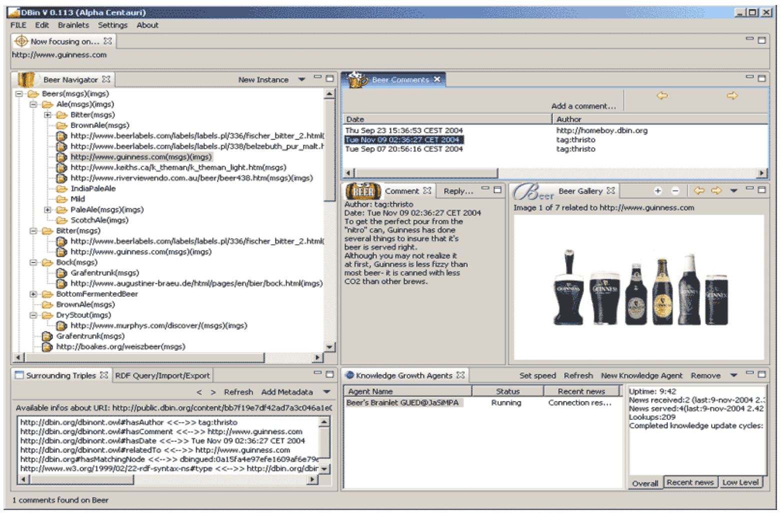
SemanticWebBrainlet http://dbin.org/brainlets/swbrainlet#

Figure 1: Joining a metadata exchange channel (right window) offered by a RDF-Growth server (left window) in the DBin platform. In this case the server suggests the user to download a specific Brainlet to better interact with the knowledge exchanged.

Brainlets are composed of XML, OWL ontologies and scripts. As an overview, they contain:

- The ontologies to be used for annotations in the domain (e.g. The beer ontology);
- A general GUI layout: which components to visualize (e.g. A message board, an ontology browser, a “detail” view) and how they are cascaded in terms of selection/reaction;
- Templates for domain specific “annotations”, e.g. a “Movie” Brainlet might have a “review” template that users fill. This allow a “reviews” view to have useful orderings based on the known fields in the review. The GUI for the templates is generated automatically;
- Templates for readily available, “pre-cooked” domain queries, which are structurally complex domain queries with only a few simple free parameters, e.g. “give me the name of the cinema where the best movie of genre X is being shown tonight”;
- A suggested trust model and information filtering rules for the domain. e.g. public keys of well known “founding members” or authorities, preset “browsing levels”;
- A script for guiding the user in creating new URIs for domain resources, e.g. inserting a new "beer" in the DB.
- Scripts connected to Brainlet specific menus or buttons and that implement domain specific functions;
- Support material, customized icons, help files etc..
- Supporting Java code and libraries;
- A basic RDF knowledge package, conforming to the information shared in a specific group.

Almost all of these elements are optional in a Brainlet. Figure 2 shows a typical runtime view of the Beer2Beer Brainlet (*http://www.dbin.org/brainlets/beer2beer*). A Semantic Web Research Brainlet, which demonstrates how the DBin platform can naturally handle completely different domains of interest, is available at *http://www.dbin.org/brainlets/swbrainlet*. The following sections are concerned with how the specific features and aspects listed above can be configured.



http://www.guinness.com

```
Beers(msgs)(imgs)
  Ale(msgs)(imgs)
    Bitter(msgs)
    BrownAle(msgs)
    http://www.beerlabels.com/labels/labels.pl/336/fischer_bitter_2.html(
    http://www.beerlabels.com/labels/labels.pl/338/belzebuth_pur_malt.h
    http://www.guinness.com(msgs)(imgs)
    http://www.keiths.ca/k_theman/k_theman_light.htm(msgs)
    http://www.riverviewendo.com.au/beer/beer438.htm(msgs)(imgs)
    IndiaPaleAle
    Mild
    PaleAle(msgs)(imgs)
    ScotchAle(imgs)
  Bitter(msgs)
    http://www.beerlabels.com/labels/labels.pl/336/fischer_bitter_2.html(
    http://www.guinness.com(msgs)(imgs)
  Bock(msgs)
    Grafentrunk(msgs)
    http://www.augustiner-braeu.de/html/pages/en/bier/bock.html(imgs)
  BottomFermentedBeer
  BrownAle(msgs)
  DryStout(imgs)
    http://www.murphys.com/discover/(msgs)(imgs)
  Grafentrunk(msgs)
  http://boakes.org/weiszbeer(msgs)
```

Date	Author
Thu Sep 23 15:36:53 CEST 2004	http://homeboy.dbin.org
Tue Nov 09 02:36:27 CET 2004	tag:thristo
Tue Sep 07 20:56:16 CEST 2004	tag:thristo

Author: tag:thristo
Date: Tue Nov 09 02:36:27 CET 2004
To get the perfect pour from the "nitro" can, Guinness has done several things to insure that it's beer is served right.
Although you may not realize it at first, Guinness is less fizzy than most beer- it is canned with less CO2 than other brews.

Image 1 of 7 related to http://www.guinness.com

Available infos about URI: http://public.dbin.org/content/bb7f19e7df42ad7a3c046a1e(

```
http://dbin.org/dbinont.owl#hasAuthor <<-->> tag:thristo
http://dbin.org/dbinont.owl#hasComment <<-->> http://www.guinness.com
http://dbin.org/dbinont.owl#hasDate <<-->> Tue Nov 09 02:36:27 CET 2004
http://dbin.org/dbinont.owl#relatedTo <<-->> http://www.guinness.com
http://dbin.org#hasMatchingNode <<-->> dbingued:0a15fa4e97efe1609af6e79e
http://www.w3.org/1999/02/22-rdf-syntax-ns#type <<-->> http://dbin.org/dbir
```

Agent Name	Status	Recent news
Beer's Brainlet GUED@JaSiMPA	Running	Connection res...

Uptime: 9:42
News received:2 (last:9-nov-2004 2.:
News served:4(last:9-nov-2004 2.42
Lookups:209
Completed knowledge update cycles:

1 comments found on Beer

Figure 2 A screen shot of the Beer2Beer Brainlet running. The principal "views" are: an ontology (and instances) browsing Navigator, the Knowledge Agents view, showing statistics about the currently running knowledge agents, and a set of "Annotation" views.

3.1. From Eclipse plug-in to Brainlet

To create a Brainlet, it is necessary to fill a given empty template which configures an eclipse plug-in. To append a new "Brainlet" to the list of those known by DBin, one have to add the following XML in the standard eclipse *plugin.xml* file:

```
<extension point="org.dbin.gui.brainlet.core.xmlbrainlet">
   <brainlet perspectiveID="org.dbin.gui.brainlet.beer" file="brainlet/beer.xml"/>
</extension>
```

Other configuration options in the *plugin.xml* will include the invocation and positioning of any desired DBin predefined basic views such as the Navigator (section 3.2), annotations, gallery etc..

3.2. Brainlet configuration in detail

Let's now see the content of the *beer.xml* file, the main Brainlet configuration which has been specified in the previous section as an attribute of the `org.dbin.gui.brainlet.core.xmlbrainlet` entry point.

A Brainlet has a name, a version, a URI identifying it, usually pointing at the Brainlet download location (to be used in the dialog show in Figure 1), and other basic information, specified as attributes of the XML corresponding element. The following is the declaration for the Beer-to-Beer Brainlet:

```
<Brainlet name="Beer2Beer" author="Onofrio Panzarino" version="1.0"
  uri="http://dbin.org/brainlets/beer2beer#" banner="Targeted to beer's fans.
```

```
        Join to learn and share your knowledge about your favorite beers.">
```

Ontologies and base domain knowledge

First step in creating a new Brainlet is the choice of appropriate ontologies to capture the concepts of the target domain. Once existing ontologies have been identified and, if needed, new ones have been developed, the corresponding OWL files are usually included and shipped in the Brainlet itself, although they could be placed on the Web. Each of them will be declared in the XML file, specifying the location of the OWL file, a unique name for the ontology and it's base namespace:

```
<Ontology file="brainlet/ontologies/beer.owl" name="beer"
        base="http://www.purl.org/net/ontology/beer#"/>
<Ontology file="brainlet/ontologies/BeerAnnotations.owl"
        name="beer_annotations" base="http://dbin.org/beer#"/>
```

A Brainlet might need some background RDF data, a starting knowledge base shared by every client using the same Brainlet. The following line cause an RDF file to be installed into DBin:

```
<Data file="brainlet/rdfdata/beer.rdf" base=""/>
```

Tree based navigation of domain resources: the main Navigator

The way concepts and instances are presented and browsed is crucial to the usability of the interface and the effectiveness in finding relevant information. Graph based visualizers are notably problematic when dealing with a relevant number of resources. For this reason, the solution that the main DBin Navigator provides is based on flexible and dynamic tree structures. Such approach can be seen to scale very well with respect to the number of resources, e.g. in Brainlets such as the SW Research one (*http://www.dbin.org/brainlets/swbrainlet/*).

The peculiarity of the approach is that every Brainlet creator can decide which is the 'relation' between each tree item and its children by the use of scripts or semantic web queries (in DBin these are currently expressed in SeRQL syntax [10], but support for other query languages is likely to be provided in future versions). The basic use of such facility is to provide an ontology based navigation. Creating a tree branch by which a class hierarchy based navigation of Beers is as simple as:

```
<Topic name="Beers by Type" uri="http://www.purl.org/net/ontology/beer#Beer">
   <Child query="SELECT X FROM {X} serql:directSubClassOf {$parent}
                WHERE X != $parent">
     <Child subjectBy="serql:directType" icon="/icons/beer.gif"/>
   </Child>
   <Child subjectBy="serql:directType" icon="/icons/beer.gif"/>
</Topic>
```

Each Topic has an associated URI (in this case that of the base class "Beer"), that is every tree conceptually starts from an RDF resource. Topic children are then recursively defined by the results of queries involving the parent resource. In the case of a first level Child, the parent resource will be the resource associated to the entailing Topic. There can be multiple topic branches configured in the Navigator. For example a "Beers by Brewery" branch is configured as follows:

```
<Topic name="Beers by Brewery" uri="http://www.purl.org/.../beer#Region">
   <Child query="SELECT X FROM {X} rdf:type {$parent} WHERE X != $parent">
      <Child subjectBy="http://www.purl.org/net/ontology/beer#brewedBy"/>
   </Child>
</Topic>
```

Note that, of course, different conceptual paths can lead to the very same resource. Figure 3a shows a run time screen-shot of a Brainlet configured with the Navigator Topics that have been just illustrated.

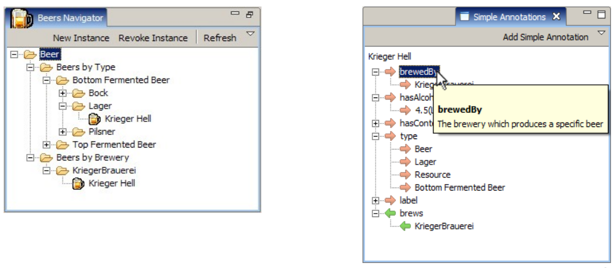


Figure 3: a) The simple annotation view shows the properties of the selected resource (left figure). b) A navigator view generated by the configuration shown above. The instance of a beer can be reached following different paths, by type or by brewery(right figure).

Selection flow across Brainlets parts

At user interface level, as shown in Figure 2, a Brainlet is composed by a set of 'view parts' (as they're called in Eclipse parlor). Usually, each part takes a resource as a main "focus" and shows distinct aspects of the same RDF knowledge 'around' this resource. The Simple Annotations view, for example (Figure 3b), shows the outgoing and incoming RDF statements surrounding the selected RDF node.

As mentioned, the Brainlet creator decides the configurations of the standard view so to have a personalized title, icon, position and so on. Selection flows are also scripted at this point; it is possible to establish the precise cause effect chain by which selecting an icon on a view will cause other views to change. This is a powerful way to customize the flow of information within the user interface. Let's consider the following XML code:

```
<View id="GUEDNavigator" title="Beers Navigator"
        icon="icons/nav.gif" selectorFor="main" listenTo="main" />
<View id="Annotations" listenTo="main" selectorFor="annotation" />
<View id="Content" title="Details"
        listenTo="annotation" selectorFor="annotationContent,main" />
<View id="SimpleAnnotationsView"
        listenTo="main" selectorFor="annotationContent" />
```

Using the attributes 'listenTo' and 'selectorFor', views act both as listeners or as notifiers of "Selection Managers". A view is listener of a selection manager when it will change its content when the manager notifies the change of URI and is a "selector" for a manager when some user action within the view itself will cause the manager to change its selected URI. Many "selection manager" can be created to support the information flow logic within the Brainlet.

Assisted querying tools

Within a specific domain there are often some queries that are frequently used to fulfill relevant use cases. Continuing our "Beer" example, such a query could be “find beers [stronger|lighter] than X degrees”. The "precooked queries" facility gives the Brainlet creators the ability to provide such "fill in the blanks" queries to end users.

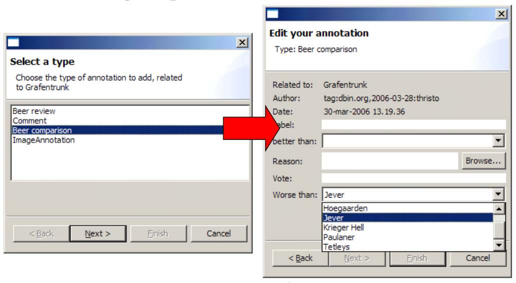
Precooked queries can also be scripts stored in Java classes methods or executed via an interpreter. Here is an example of a Precooked Query:

```
<PrecookedQuery label="Beer Strength"
    message="All beer with alcoholic content greater or lesser than $degree"
    query="SELECT * FROM {X} rdf:type {beer:Beer}; beer:hasAlcoholicStrength {Y}
            WHERE Y $operator $degree" USING NAMESPACES
            beer = http://www.purl.org/net/ontology/beer#
    clipboardQuery="CONSTRUCT {X} rdf:type {dbin:SemClipBoardUri}
                     FROM {X} rdf:type {beer:Beer}; beer:hasAlcoholicStrength {Y}
                     WHERE Y $operator $degree USING NAMESPACS
                            dbin = http://dbin.org/dbinont.owl#,
                            beer = http://www.purl.org/net/ontology/beer#" >
   <Variable name="$operator" legalValues="&lt;,&gt;" defaultValue="&gt;" />
   <Variable name="$degree" />
</PrecookedQuery>
```

In this configuration fragment, a precooked query named "Beer Strength" will show the user a human readable query and allow to chose the operator (< or > , default value >) and a variable value. Such inputs will be used both in a select query, used to find the elements and display them as a table, and in a construct query, used to construct a Semantic Clipboard content.

The Semantic Clipboard is another element that a Brainlet creator might chose to make visible and serves as a temporary graph where the user can put content which can be then passed to other visualizers (e.g. the map viewer as shown below) or external applications by serializing its content to the system clipboard.

Note that a precooked query might link concepts from different domains, and different Brainlets. Suppose to have a Beer Brainlet and a Pub Brainlet. A precooked query as 'Find all the pubs serving beer X which does not contain ingredient Y' is a cross domain query and obviously would only work once sufficient knowledge has been extracted from the two groups.



Select a type

Choose the type of annotation to add, related to Grafentrunk

Beer review
Comment
Beer comparison
ImageAnnotation

Edit your annotation

Type: Beer comparison

Related to: Grafentrunk
Author: tag:dbin.org,2006-03-28:thristo
Date: 30-mar-2006 13.19.36
Label:
better than:
Reason:
Vote:
Worse than: Jever

Hoegaarden
Jever
Krieger Hell
Paulaner
Tetleys

Figure 4: Advanced annotations are defined in OWL and auto generate property visualization and editing interfaces.

Domain specific annotations

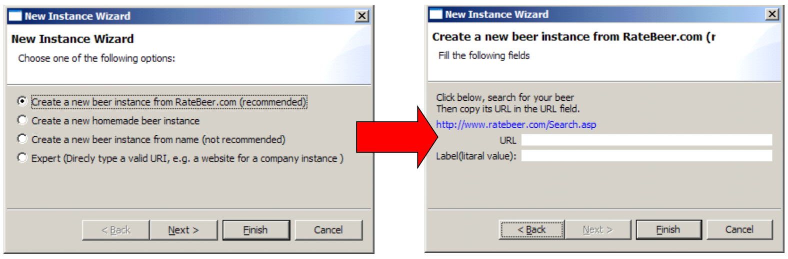
Brainlets assist the users in creating simple annotations, that is in linking resources with properties. Ontology based wizards suggest both the properties that are applicable to the given resource and instances that are currently known to the database and that are of a kind appropriate for a selected property.

A Brainlet creator can however also chose to create "complex annotation types" using a specially defined OWL ontology. An example of such complex annotations is the "Beer Comparison" annotations, which directly compare beers stating which one is better or worse and why. Upon selecting "Add advanced annotation" in DBin the system determines which advanced annotations can be applied to the specified resource and provides a wizard as shown in Figure 4.

Identifier assignment facilities

In our scenario each user is entitled to add new concepts into a knowledge base that is shared within a P2P group. A methodology is needed to make so that different persons, independently adding the same concept (e.g. a particular beer: Peroni Light) into the system, will choose, with a reasonable probability, the same URI to identify it. It is evident that this is necessary for the annotations about Peroni Light to be exchanged among peers, as well as to avoid duplication of concepts. A possible solution might be that of suggesting the user to visit a web site (e.g. RateBeer.com), chosen from the Brainlet creator, where a web page referring to Peroni Light can be found, and to use the relative URL to identify the concept.

Depending on the type of instance one is adding (a beer, a book, a movie...), different methodologies could be thought to create (or find) a proper URI. In the case of a well known concept in a particular domain, for example, we can assume that everybody would reasonably refer to it using the same word, and we can build a new URI with this word as fragment identifier. Note that such procedures does not ensure that different users will end up with the same URI, but still can work in a lot of cases.
Within a Brainlet it is possible to define a set of *URIWizards*, basically scripts that implement a given procedure and guide the user in creating a URI, and to assign them to a class. Once instance of this class is being inserted these scripts are activated providing the user with a graphic wizard, as shown in Figure 5 .


New Instance Wizard

New Instance Wizard

Choose one of the following options:

- [✓ Create a new beer instance from RateBeer.com (recommended)]
- Create a new homemade beer instance
- Create a new beer instance from name (not recommended)
- Expert (Diredy type a valid URI, e.g. a website for a company instance)

New Instance Wizard

Create a new beer instance from RateBeer.com (r

Fill the following fields

Click below, search for your beer
Then copy its URL in the URL field.
http://www.ratebeer.com/Search.asp

URL

Label(litaral value):

Figure 5: Choosing to insert a new instance of beer the user is prompted with a wizard and can choose different procedures to create a URI.

An XML example configuration is given:

```
<Class uri="http://www.purl.org/net/ontology/beer#Beer">
   <FromUrlWizard label="Create a new beer from RateBeer.com (recommended)">
      <Message content="Click below, search for your beer &#10;
                         Then copy its URL in the URL field."/>
      <Website>http://www.ratebeer.com/Search.asp</Website>
      <UrlPrefix>http://www.ratebeer.com/Beer/</UrlPrefix>
   </FromUrlWizard>
   <FromPropertiesWizard label="Create a new homemade beer instance">
        <Property label="Label" prefix="$useruri:HomemadeBeer:"
                uri="http://www.w3.org/2000/01/rdf-schema#label"/>
   </FromPropertiesWizard>
</Class>
```

Domain specific script inclusion

In developing DBin we encountered the need to implement some actions which, in a sense, are outside of the scope of the platform itself but yet are needed in order to provide functionalities for the user to experience the power of underlaying semantic knowledge.

For example we wanted the user to be able to organize a pubs tour, choosing a list of pubs (e.g. using precooked queries to find the pubs that serve a particular beer) and automatically calculating the optimal order of pubs to be visited. This is a very specific task and we didn't want to modify the platform architecture in order to solve this kind of problems. On the other hand, the solution imply solving a traveling salesman problem, so a simple graph query it is not enough, a real algorithm is needed. Our 'light' approach is to address these kind of requirements using scripts, the execution of which, thanks to the flexibility of Eclipse RCP, can be easily integrated into the user interface, e.g. pressing a button. DBin accommodate a module supporting BeanShell [11] scripting language. In Figure 6, one sees a button that has been added by a Brainlet specific configuration of the Map View (a view supporting geo-tagging based on the Google-Earth [12] service). Such button is connected to a BeanShell script that performs the evaluation of the optimal pub tour based on an implementation of a (simple) traveling *salesman* heuristic.

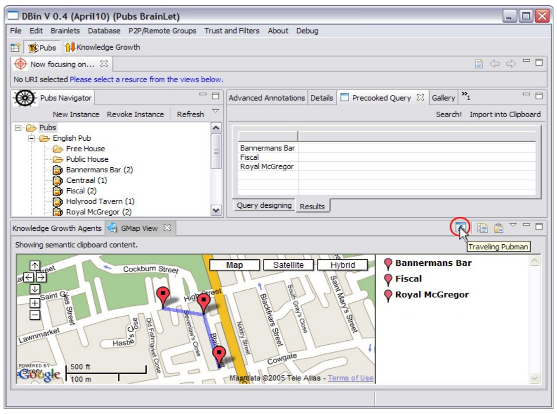


Figure 6: Optimal pub tour around the WWW2006 conference site. A "precooked query" is executed so that certain pubs are selected. The Traveling Pubman script is then invoked to highlight the optimal tour.

4. Conclusions

In this paper we introduced the concept of domain specific Semantic Web applications created using a combination of ontologies, and XML configuration and scripting. Brainlets are a key concept in enabling the creation of Semantic Web Communities in DBin, a full round semantic web platform. We think however that the Brainlet paradigm could be applied to any other semantic web application based on the availability of an underlying SW Database. This could include, in the future, web based Brainlets players capable of providing similarly rich user interfaces over the Web.

While starting a "Semantic Web community" by creating a Brainlet is certainly much more complex than creating a IRC channel or installing a web forum, in our experience domain experts with knowledge of the OWL language have consistently managed to learn all the basic principles, create and deploy their first Brainlets within hours or less. To the best of our knowledge there are no projects directly related to the ability to create Semantic Web access environments which can be compared to the Brainlet paradigm. The idea of a rich client for the Semantic Web has been developed in the Haystack project [13] resulting in a general purpose, monolithic interface. Differently, Brainlets are intended to be powerful user interfaces which can be scripted and 'shaped' on a particular domain. The Piggy Bank project [14] enables light HTML scraping scripts to be installed with ease, so it has, in some sense, a similar philosophy. Also from the MIT Simile initiative, the Fresnel [15] ontology can be used to create "data views" called "lenses" which facilitate the automatic creation of presentation and editing data forms. As such feature are in fact likely to be beneficial to better allow customizations of data presentation, a Fresnel aware Brainlet engine is among the future works.

Brainlets as fostering of social aggregation around Ontologies and data representation practices

When a Brainlet is plugged into DBin, the ontologies that it requires will be installed and used. This simple mechanism, seems to induce a pragmatic, socially based model for ontological agreement, a notably tough problem in Semantic Web research. In fact, as users gather around popular Brainlets for their topic of choice, the respective suggested ontologies and data representation practices will form an increasingly important reality. If someone decided to create a new Brainlet or, in general, a Semantic Web application targeting the same user group as the said popular Brainlet, it is clear that there would be incentive in using identical or somehow compatible data structures and ontologies.

While this is probably the case for the use of ontologies on the Semantic Web in general, such effect is expected to be easy to spot in the real of Brainlets, given how such paradigm immensely lower the barrier by which a domain expert can turn an idea into a full featured domain specific cooperative Semantic Web application.

References

[1] "DBin project" http://www.dbin.org

[2] G. Tummarello, C. Morbidoni, J. Petersson, P. Puliti, F. Piazza , "RDFGrowth, a P2P annotation exchange algorithm for scalable Semantic Web applications" 2004 First P2PKM Workshop, Boston

[3] G. Tummarello, C. Morbidoni, P. Puliti, F. Piazza , "Signing individual fragments of an RDF graph" 2005 WWW2005, poster track

[4] R. Albertoni, A. Bertone, M. De Martino , "Semantic Web and Information Visualization" 2004 Proceedings of the First Italian Workshop on Semantic Web Applications and Perspectives, Ancona (ITALY)

[5] "RDF Gravity - RDF Graph Visualization Tool" Technical Report: HPL-2004-57

[6] E Pietriga , "Isaviz: a visual environment for browsing and authoring rdf models " 2002 11th International World Wide Web Conference

[7] "RDFX" Technical Report: HPL-2004-57

[8] Welkin, a graph-based RDF visualizer, , 2004, http://simile.mit.edu/welkin/

[9] Eclipse Rich Client Platform, , , http://www.eclipse.org/rcp/

[10] Jeen Broekstra, Arjohn Kampman , "SeRQL: An RDF Query and Transformation Language" 2004 ISWC 2004, Hiroshima, Japan

[11] BeanShel: Lightweight Scripting for Java, , , http://www.beanshell.org/

[12] Google Earth, , , http://earth.google.com/

[13] Quan, Dennis and Karger, David R , "How to Make a Semantic Web Browser" 2004 In Proceedings International WWW Conference, New York, USA

[14] D. Huynh, S. Mazzocchi, D. Karger, "Piggy Bank: Experience the Semantic Web Inside Your Web Browser" 2005 proceedings of the ISWC 2005

[15] "Fresnel - Display Vocabulary for RDF" 2005