

The DBin Semantic Web platform: an overview

Giovanni Tummarello, Christian Morbidoni, Paolo Puliti, Francesco Piazza

Università Politecnica delle Marche (Italy)

<http://semanticweb.deit.univpm.it>

Abstract

DBin is a tool to build “Semantic Web P2P communities”. It provides a general platform on top of which it is possible to create and run P2P Semantic Web applications where users contribute by annotating topics of common interest. DBin introduces a Semantic Computing scenario where users “slowly”, but in a sustainable way, build a local DB and can then enjoy a variety of semantic based activities such as browsing or intelligent interaction with the local media and files. DBin accommodates a number of experimental modules to deal with specific kind of metadata (audio metadata extraction, textual analysis, desktop integration) as well as a domain oriented user interface. DBin includes an RDF subgraph digital signature facility enabling personalized trust policies to provide filtering out unwanted information. Maximum extensibility is guaranteed by the use of the Eclipse Rich Client platform and by the Open Source model.

1. Introduction

In this paper we introduce DBin, a platform integrating Semantic Web technologies with specific task modules and a rich user interface. Most DBin operations (browsing and querying) revolve around a local, personal, knowledge store. This database is built by a number of DBin modules, notably one integrating a novel P2P Semantic Web algorithm as well as modules which reflect the content of the local machine (desktop integration).

All the knowledge stored in DBin is expressed using the languages defined in the Semantic Web initiative (RDF, RDFS [1]) but the user doesn't necessarily have to be aware of this.

Use 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 has selected the topics of interest and has connected to a semantic web P2P group, RDF annotations just start flowing in and out “piece by piece” in a scalable fashion.

For example, a user who expresses interest in a particular artist and his/her music could review new pieces of relevant “information” in the morning that the system was able to retrieve during the night. As RDF metadata points to actual URLs on the web, the user would then see real data such as a picture made by a fan, a review of a live concert or an MP3 of someone attempting to sing a song from one of his/her albums. He would at this point possibly be able to vote, comment or relate any of these to other content he might have knowledge of.

At the database level, all this information is coherently stored as RDF. At the user level however, the common operations and views are grouped in domain specific user interfaces (“Brainlets”, see section 6) which can be created using XML files by power users. Power users can create new P2P communities in this way around specific domains of interest.

Due to the open nature of the P2P model, DBin also implements an RDF digital signature infrastructure that can be used by end users to perform custom trust based information filtering as well as signing annotations to be inserted in the system. (Section 4).

The database is also accessed and updated by a number of DBin modules integrating different sources of metadata. Among these are a local desktop integration metadata extractor, which traverses the local disk resources and extracts metadata using various techniques, and a multimedia integration module based on the MPEG-7 metadata description format.(see section 7)

2.RDFGrowth: a scalable P2P engine based on the “minimum commitment” principle:

The RDFGrowth algorithm powers DBin ability to collect RDF metadata from other peers with common interests. Previous projects, such as [2][3], have explored P2P interactions among groups of trusted and committed peers. By this we mean that peers rely on each other to forward query requests and collecting and returning results. In contrast, RDFGrowth is designed to operate in a real world scenario of peers where cooperation is relatively frail. By this we mean that peers are certainly expected to provide some external service, but commitment should be minimal and in a “best effort” fashion.

To obtain this, DBin's P2P algorithm, called RDFGrowth, follows a peculiar philosophy: minimum external burden.

(minimum signable unit) which are the partitions in unique blank node closures of the triples directly connected to the URI. MSGs properties turn out to be very useful when applying digital signatures.

From a computational point of view, RDFN fulfill the basic “minimum external burden” requirement: they are both very simple to calculate (constant time or linear with the number of blank nodes composing it) and results can be very effectively cached. For details, a better discussion of the RDFN properties and a review of the state of the art in P2P SW see [4]

- *The only required on line facility is a shared Hash Table or a DHT.*

The Hash table will be used to keep the *signature* of the most recent RDFN. A RDFN *signature* is a set of values supporting an heuristic which chooses the best peer to perform a direct request to. A simple MD5 on a canonically serialized RDFN already serves the purpose as it can be trivially proved to be a sufficient condition for the group knowledge to converge. More advanced signatures are expected to consistently speed up the convergence time. The number of entries in the hash table is simply a small factor (2-3) of the entries in the GUED.

The algorithm that a peer follows can be sketched as:

- a peer will cycle trough the URIs matched by the GUED
- for each URI will synchronize with the peer that knows the most according to the entry in the DHT providing the most “interesting” RDFN signature. Synchronize implies a direct P2P query asking for the full RDFN (but future versions might have RSync like algorithms).
- Whenever new information is detected locally in the DB (e.g. after joining with the most knowledgeable peer in the group and still having a different signature) if a broadcast channel is available the peer might simply use it to communicate the “news”.

A basic monotonic framework supporting local non monotonic behaviors

Such a “growth only” scenario matches the monotonic nature of the RDF semantics. To obtain more information about a resource can’t in fact “hurt” since, by definition, previously inferred statements will still hold when new data becomes available. It is of course possible, in the real world applications, to rely on “context” information to later apply non monotonic rules on the local database, but it should remain a local peer decision to do so with no consequences on the shared knowledge. Digital signatures are an example of such context information which support several fundamental higher level non monotonic behaviours of the overall system.

4. Supporting authorship authentication

The MSG definition and properties highlighted in the previous section, when combined with a canonicalized serialization as suggested in [5], provide a the framework for authorship assessment, trust based filtering and information revision in DBin.

Given its mathematical properties, it is in fact possible to sign a MSG in an efficient way: the signature information needs only to be attached to a single triple composing it.

Once the authorship of a MSG can be verified, a variety of filtering rules can be applied at will. These, in DBin, are always non-destructive; information that doesn't match certain trust criteria can be hidden away but does not get deleted.

Other than authenticating provenience, this methodology is used for providing capabilities of “information revision” In short, once a MSG has been signed, the hash can be used as an Inverse Functional Property (IFP, see [6]), that is, as a unique way to name to the MSG itself. This in turn can be used in a subsequent MSG to indicate the one that it substitutes. Given that the paternity of this subsequent MSG can be verified to be identical, the client can safely perform the information update, no matter where it received the update patch from.

RDFTrust: support for a centralized certification authority

DBin also includes a preliminary support for a centralized certification authority, dubbed RDFTrust. By obtaining a signed certificate (which involves a time consuming identification procedure), users can enjoy instantly an higher degree of trust. If trust was to be abused (e.g. By spamming or malicious “ontology violations”) the certificate would be revoked thus causing all the previously inserted information to “sink” in each DBin installation. The certification authority web site is currently being implemented as a web application with a matching application API.

5.The URI Bridge component

As the RDFGrowth P2P algorithm only exchanges pieces of RDF graphs, some facility is needed to provide the user with actual content (e.g. Images, text etc). Once a URL is available for a specific annotation, it is retrieved over standard HTML by the URIBridge upload/download facility. While downloading is straightforward, the uploading part requires that each DBin is configured for an upload server, much like an EMail client requires a SMTP server. While the default installation of DBin comes with a simple upload server, this limits the users to small file. For power users, installing a personal upload server is however trivial, just the deploy of a simple PHP script.

6.User interface: “Brainlets”

There has been a lot of work recently on Semantic Web visualization and a number of user interface have been proposed [7], [8], [9], [10], [11].

While pro and cons can be argued for each specific approach, it is clear that user interface issues are a complex issue with no clear single solution.

Rather than providing a single answer to this issue, Dbin provides a general set of “application oriented” generic GUI tools by which power users can build applications which are specific to the targeted domain of interest.

Dbin domain specific applications, are called “Brainlet”, and can in a sense be directly related to the concept of “GUED”, the operator which decides how to select the knowledge to be exchanged in the p2p group, from the RDFGrowth algorithm.

Brainlets can be thought of “configuration packages” preparing dbin to operate on a specific domain (e.g. Wine lovers, Italian Opera fans etc..). Given that brainlet include customized user interface, the user might perceive brainlets as full “domain applications” which are run by Dbin. which come with regarded as “integrated packages” which describe and implement “domain applications”.

In short Brainlets are composed of:

- The setup information for the RDFGrowth algorithm and the transport layer to connect with others using the same brainlet (namely, at this point, the name of the rendezvous servers, the channels and the GUED)
- 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”.- Support material, customized icons, help files etc..
- A basic RDF, GUED conforming, knowledge package

Most importantly, Brainlets can be created as much as possible with no programming skills, that is, just by editing XML configuration files; more advanced brainlets can however be made including custom Eclipse plugins as needed.

As in version 0.14 of Dbin, most of the previously mentioned features have been implemented as shown in figure 3, a screen shot of “Beer2Beer”, our first XML based Brainlet.

7. Integrating with local resources

The idea of using RDF and other Semantic Web languages and tools to help management of desktop resources and applications has been recently investigated. In [12] an application, Gwowsis, has been presented which uses different plugins to represent local files in a RDF graph. Users are allowed to add annotations and browse local resources taking advantage of the semantic connections established. A similar tool is described in [13], where also the usefulness of metadata sharing capability is highlighted.

One of the most interesting aspects of Dbin is that metadata coming from local resources can be merged with those coming from external sources (e.g. the RDFGrowth P2P groups).

A few modules are dedicated to the extraction of metadata from local resources. A local file system processor creates a “semantic” representation of the available files inside the triple store. This means that each file will be indicated by its local URI and take part in the RDF model and that different techniques are applied to extract metadata according to the file type.

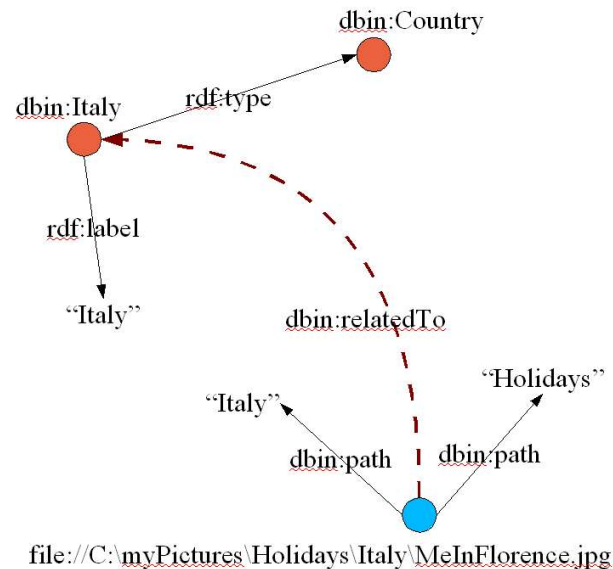


Figure 2 Considering the file “c:\myPictures\Holidays\Italy\MeInFlorence.jpg”, one can extract a simple RDF representation analyzing the file path, identifying two “concepts” probably related to the file itself: “Holidays” and “Italy”. Searching among the ontology classes in the local DB, one can find a term “dbin:Italy” which has a “rdf:literal” value equals to “Italy”. This can be exploited to create an explicit relationship between the file and the ontology term.

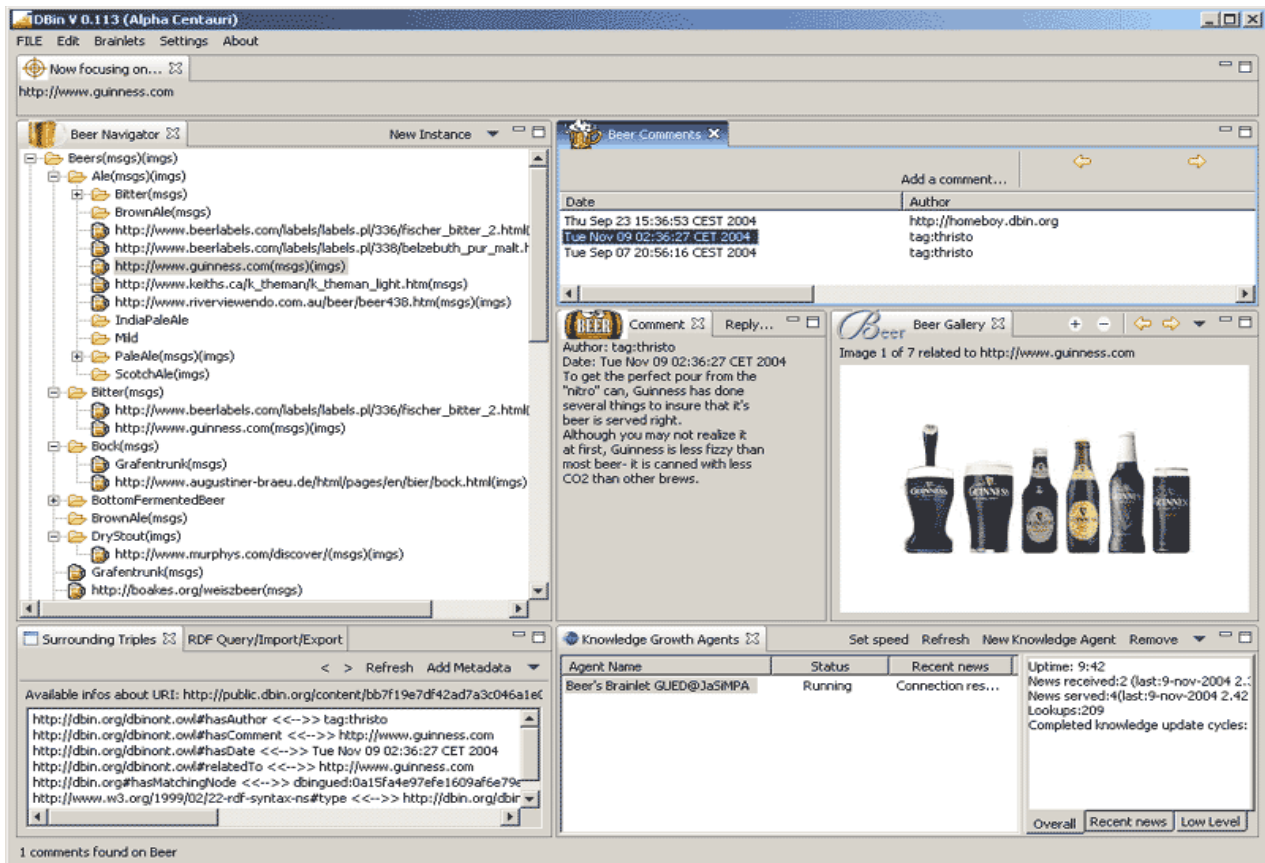


Figure 3 A screen shot of the Beer2Beer Brainlet running. The principal “views” are: an ontology (and instances) browsing Navigator, the Knowledge Agents view, showing statistics about the currently running knowledge agents, and a set of “Annotation” views. Among these a comment view, a picture gallery and an “annotation listing” view.

These include full text analysis and indexing, file type specific operation (e.g. ID3 tags extraction) and file name and path heuristics.

Once this RDF representation is created, a number of semantic links both among files and between files and existing nodes can be suggested. As a first approach Dbin extracts these relationships comparing literal values attached to file representations and pre-existing resources, as exemplified in Figure 2.

These relationships are then shown when browsing the nodes of local database.

8. Integrating with local and remote audio

Dbin includes an experimental support for audio metadata based on the Mpeg7AudioDB [14] library. This component acts as an automatic RDF metadata extractor from low level Mpeg-7 streams, typically describing low level feature of the audio signal (e.g. spectrum spread, fundamental frequency, etc...). Once a collection of Mpeg-7 descriptions is available (e.g. Extracted from a local collection or remotely from a web service), annotations take the form of:

- standalone metadata, e.g. genre annotations, which enhances the audio accessibility.
- links between audio clips, highlighting “audio similarities”, which mainly enhance the user browsing experience.

It is expected that this module will allow testing of a scenario where audio metadata is shared in p2p in order to better browse a local collection of music.

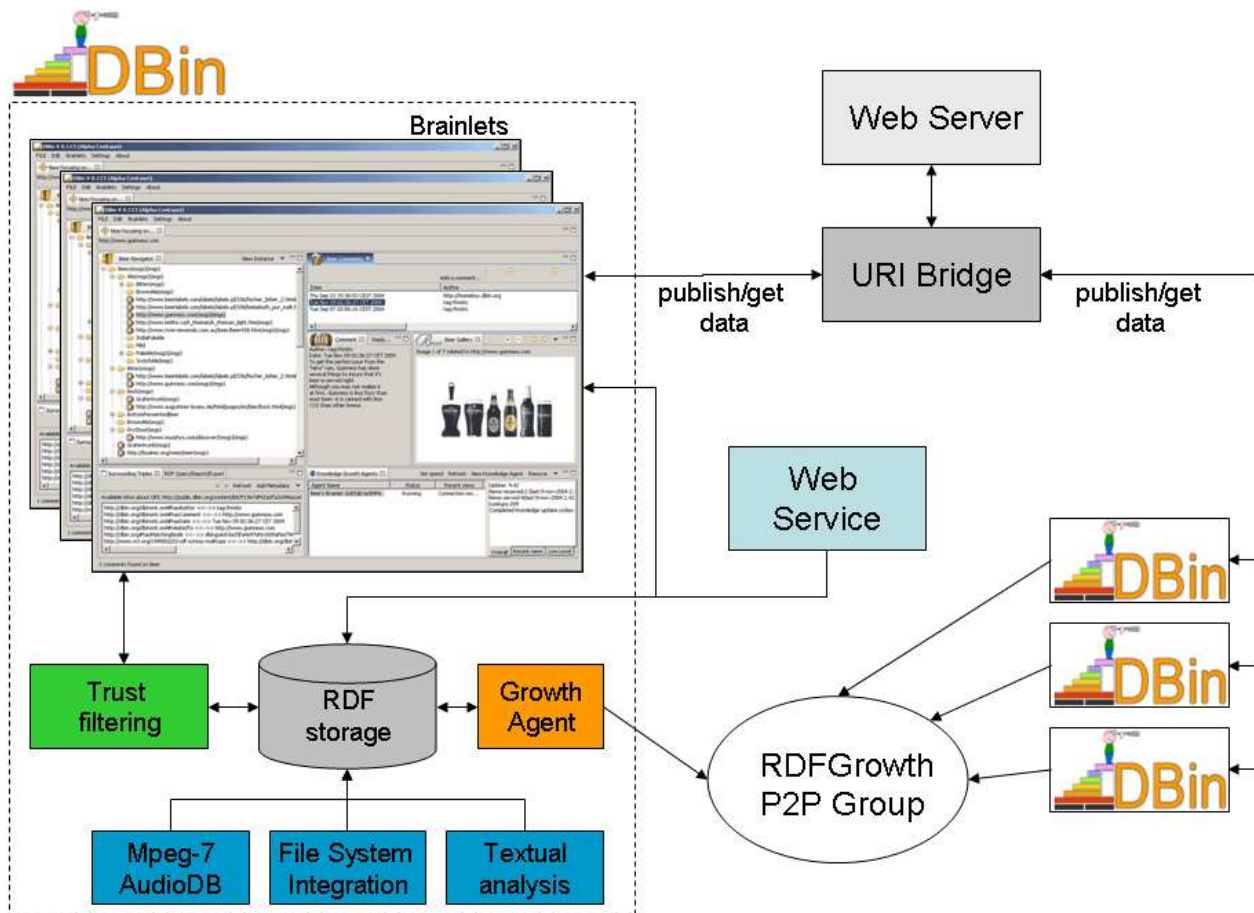


Figure 4 A schema illustrating the overall DBin architecture and the use scenario. Distinct DBin clients participate the same P2P group and exchange metadata with their metadata growth agents implements the RDFGrowth algorithm. Each DBin client, when publishing metadata referring to actual data, also makes sure this data is accessibly by publishing, if needed, in a Web space. Also contributing to the local DB is a set of modules interacting with local and remote resources. Finally, a local trust based filtering is performed on the RDF DB and the resulting content, along with the data retrieved by the URI Bridge, is displayed by domain specific user interfaces (Brainlets).

9. Software Engineering/License

DBin is programmed in Java and based on the Eclipse Rich Client platform. As such, DBin is naturally multi-platform, features an OS native look and feel and is highly extensible through the well known Eclipse plug-ins and extension points technology.

Both the framework and modules presented here are open source. Licensing terms have not been settled yet, but they're expected to be either LGPL, BSD or CPL.

10. Conclusions and future works

In this paper we presented **DBin**, a cross-platform application which provides the user with a rich interface to a local database of metadata and access to a number of way to retrieve new knowledge. As users individually annotate topics and resources of interest, the main source for new

information is certainly the RDFGrowth P2P infrastructure integrated in DBin. RDFGrowth, is an algorithm targeted at a particular scenario where peers participate in interest groups to grow their internal knowledge about one or more specific topics. RDFGrowth is built to operate in an open Internet scenario; this means that at any time peers operate without any specific commitment and without imposing any considerable burden on others in the group.

Additional metadata sources are given by specific modules dealing with different kinds of data locally available to the client. These include metadata from the files in the local machine and audio metadata thanks to an experimental module based on MPEG-7.

DBin is meant to demonstrate the usefulness and scalability of a model where a large, local, semantic web database is grown and rich user interfaces and filtering is then applied a posteriori. A summary graphic representation of such model is given in figure 4.

Under a usage point of view, while DBin's differ fundamentally from the way the current Web is used, it is not in fact much different from popular P2P file sharing applications.

In the same way as many users have gotten used to wait to obtain data by running a classic P2P file sharing, DBin users will “peacefully” discover new information about topics in which they express interest in.

Content and annotations produced by the user, on the other hand, can reach precisely those who had expressed interest in it and naturally cross the boundary of the P2P group they were posted originally to. Given RDFGrowth design in fact, relevant annotations are intrinsically and automatically bridged by the peers that visit multiple groups or return at later times.

All the work presented here has been implemented in Java as Free Software and is currently available at the respective websites.

References

- 1: RDF, W3C recommendation
- 2: Wolfgang Nejdl, Boris Wolf, “EDUTELLA: A P2P Networking Infrastructure Based on RDF”, 2002
- 3: Paul - Alexandru Chirita, Stratos Idreos, Manolis Koubarakis, and Wolfgang Nejdl , “Publish/Subscribe for RDF-based P2P Networks”, 2004
- 4: Giovanni Tummarello, Christian Morbidoni, Joackin Petersson, Paolo Puliti, Francesco Piazza, “RDFGrowth, a P2P annotation exchange algorithm for scalable Semantic Web applications”, 2004
- 5: Jeremy Carroll, “Signing RDF Graphs”, 2003
- 6: OWL , W3C recommendation
- 7: R. Albertoni, A. Bertone, M. De Martino, “Semantic Web and Information Visualization”, 2004
- 8: “RDF Gravity - RDF Graph Visualization Tool”
- 9: E Pietriga , “Isaviz a visual environment for browsing and authoring rdf models” , 2002
- 10: RDFX, www.rdfx.org
- 11: S. Mazzocchi, P. Ciccarese, ”Welkin, a graph-based RDF visualizer”, 2004, <http://simile.mit.edu/welkin/>
- 12: Leo Sauermann, “Gnowsis Semantic Desktop” ISWC2004 Demo, 2004
- 13: Robert MacGregor, Sameer Maggon, Baoshi Yan, “MetaDesk: A Semantic Web Desktop Manager”, 2004
- 14: G.Tummarello, C.Morbidoni, F.Piazza, MPEG-7 Audio Db, <http://www.sourceforge.net/projects/mpeg7audiodb>