SemperWiki: a semantic personal Wiki

Eyal Oren
eyal.oren@deri.org

Digital Enterprise Research Institute
Galway, Ireland

Abstract. Wikis are collaborative authoring environments, and are very popular. The original concept has recently been extended in two directions: semantic Wikis and personal Wikis. Semantic Wikis focus on better retrieval and querying facilities, by using semantic annotations of pages. Personal Wikis focus on improving usability and on providing an easy-to-use personal information space. We combine these two developments and present a semantic personal Wiki. Our application SemperWiki offers the usability of personal Wikis and the improved retrieval and querying of semantic Wikis. Users can annotate pages with RDF together with their normal text.The system is extremely easy-to-use, provides intelligent navigation based on semantic annotations, and responds instantly to all changes.

1 Introduction

The amount of information we process, maintain, search, and use in our daily work is enormous. All this information is contained in our *desktop*, our personal working space. Efficiently and effectively retrieving the relevant information from our desktop is currently a problem, especially since many related pieces of information are spread over various applications that do not communicate with each other (e.g. some emails, documents, appointments, and websites on the same topic).

Adding semantic annotations to desktop data could alleviate this problem and offer better retrieval possibilities [2, 5]. By annotating data on the desktop in a semantic language such as RDF we get (i) a uniform data format for the separate pieces of information, (ii) integration of information pieces based on URIs, (iii) enhancement of information with background knowledge in ontologies.

Semantic annotations add application-independent meaning and structure to data; they allow better retrieval of our desktop data. To some extent, these annotations can be provided by special application wrappers that know the structure and semantics of some particular application data and can export this in RDF. But the user will also have to make manual annotations, because the semantics of some information chunk is not always available or derivable from the application data.

For example, a wrapper can export meeting appointments from a calendar application into RDF, but if the user wants to annotate the participants, topic, discussion, and outcome of the meeting, he will have to add these annotations manually because these data are not available in the calendaring application. Also, if the user wants to relate the meeting agenda and the meeting notes to the meeting appointment, he will have to add these annotations manually (because again, the application wrapper cannot make these relations automatically).

But adding semantic annotations to information pieces requires effort. Users will only make this effort if they benefit from it. Manually annotating data should therefore be easy and rewarding, since users will otherwise not annotate their desktop data. If we want the user to add annotations, we need to entice him to do so.

We present a desktop application that does exactly this: it entices users to create semantic data. This semantic personal Wiki (SemperWiki) serves as a personal information system. It is extremely easy to use and provides instant gratification for adding semantic annotations.

SemperWiki is a best-of-breed between semantic Wikis and personal Wikis, two orthogonal extensions to the original Wiki concept. Personal Wikis focus on extreme ease of use; semantic Wikis focus on improved retrieval and navigation. Combining them offers the usability of personal Wikis and the gratification of semantic Wikis.

SemperWiki is not limited to the annotation of desktop data; it is a general-purpose tool for creating and using semantically annotated data that addresses a basic prerequisite towards a better desktop: helping and enticing users to add semantic annotations. It can be used as a stand-alone semantic information system or as user interface of a semantic desktop system.

The paper is structured as follows: first we shortly recall the original idea of a Wiki in section 2. Then we review the characteristics of semantic Wikis in section 3 and personal Wikis in section 4, and we analyse the functionality of several Wiki systems in section 5. We introduce the concept of semantic personal Wikis and the SemperWiki application in section 6, and conclude with points for future work.

2 Wikis

Wiki Wiki Webs were first introduced by Leuf and Cunningham [4]. Wikis are interlinked web sites that can be collaboratively edited by anyone. Pages are written in a simple syntax so that even novice users can easily edit pages. The syntax consists of simple tags for creating links to other Wikipages and textual markups such as lists and headings.

The user interface of most Wikis consists of two modes: in reading mode, the user is presented normal webpages that can contain pictures, links, textual markup, etc. In editing mode, the user is presented an editing box displaying the Wiki syntax of the page (containing the text including the markup tags). During editing, the user can request a preview of the page, which is then rendered by the server and returned to the user.

Many Wiki engines exist for anyone who wants to setup a Wiki, most of these engines are open-source. Many sites run a Wiki as a community venue, enabling users to discuss and write on topics. For example, many open-source projects have a documentation Wiki, where users can collaboratively add documentation about the project. The burden of editing is thus shared over the whole community, while still allowing anybody to quickly find relevant documentation (which is harder in e.g. a forum or bulletin board). Popular Wikis such as Wikipedia[1] can grow very fast, since interested visitors can edit and create pages at will.

Wikis are inherently server-based which has the advantage of user platform independence (users only need a browser) but the disadvantage of requiring a round-trip for all changes[2]. For example, after each edit the page is committed back to the server, and the newly rendered page is returned to the user. Although the usability of most Wikis is good compared to other web applications (being simple and fast), the need for server round-trips is a disadvantage compared to desktop applications.

A problem with large Wikis is finding relevant information. Since almost all the information in current Wikis is textual, the only possibility to locate relevant information is a full-text search on some keywords. The only semantics of pages lies in the links between pages. Indeed almost all Wiki engines generate navigational benefits from these links: one can see all pages linking to the current one, and go to these related pages. But this navigation through related pages is quite limited, and does not address the need for more intelligent information retrieval.

3 Semantic Wikis

Several systems have been developed that enable users of Wikis to semantically annotate the information they publish in the Wiki. They enable users to structure and annotate the content of pages; they reward the user for doing so by adding navigational possibilities to these annotated pages.

3.1 Examples

We list some examples of semantic Wikis. We discuss only those systems that are being actively developed or that have been published about.

Platypus Platypus Wiki is a Wiki that is augmented with semantic technologies [7]. Users can annotate information about pages by constructing RDF triples of the form (page, predicate, object). The subject of the triple is always the current page, the predicate and object can be resources.

In Platypus the editing interface consists of three pane: one pane contains the page text in Wiki syntax, the other two contain RDF statements about the

[1] http://www.wikipedia.org

[2] approaches exist to enhance user experience by communicating in the background, but these do still not compare to the usability possibilities of a desktop application.

page. Writing text on a page and writing semantic annotations are therefore distinct activities in the Wiki, and the user has to consciously switch between editing normal text and editing semantic annotations.

Platypus offers a query interface, separated from the normal page view. The query possibilities are rich, and the background knowledge in the ontologies is used in retrieving all answers to the queries. This inferencing capability means that users can retrieve information that was not added explicitly but derived by the system.

Platypus Wiki is an initial attempt to augment Wikis with semantics, all pages in the Wiki can be annotated in RDF, and these annotations can be queried. But Platypus does not directly entice the user to annotate pages: first, there is no direct added benefit from annotating (because one needs to make a separate query to see the results of annotating), and secondly annotating requires much user action: it is a separate activity, different from the normal writing.

Shawn Shawn [1] can be seen as semantic Wiki that addresses Platypus' shortcomings in both writing semantic annotations and in rewarding users for these annotations.

In Shawn authors write the normal text and the semantic annotations at the same time, in the same input field. Semantic annotations are syntactically distinguished from normal text by having the following form: `predicate:object`. All such statements are converted to RDF triples using the page on which they appear as subject. One can for instance have a page called `Shakespeare`, containing some text about the life of Shakespeare and his work. One line in the text may read `authorOf: Hamlet`, which will be interpreted as the triple `(Shakespeare authorOf Hamlet)`. This style of semantic authoring poses very little burden on the writer; users are free to embed semantic annotations at will and can do so while writing the normal text.

Annotating pages leads to instant gratification for users in two ways: navigational links are generated from the annotations and all semantic annotations can be queried.

Shawn generates sidebar links (both forwards and backwards) from the semantic annotations. For example, using the triple about Shakespeare and Hamlet, the page `Shakespeare` would show a sidebar link to `Hamlet (authorOf)`, and the page `Hamlet` would show a sidebar link to `Shakespeare (authorOf)`. If Hamlet was annotated with `rdfs:type Book` then the page would also show links to other books in the system. Users thus get rewarded for adding annotations: pages contain more information than explicitly written.

Shawn also allows users to query for pages containing certain statements using `predicate=object`; one can omit the predicate retrieving all pages that have some predicate with the specified object, and one can specify conjunctive queries by concatenating these statements. For example, `authorOf=Hamlet` would retrieve the page `Shakespeare`. Queries can be embedded in page text, prompting a live query each time the page is viewed. Users can thus create a persistent view of the data.

3.2 Characteristics

To summarise, semantic Wikis have Wiki syntax for page authoring and use RDF for annotation of pages. Users can annotate subtype relationships of pages and predicate:object statements having the page as subject. Based on the semantic annotation intelligent navigation can be offered, such as dynamic sidebar links to pages related via some predicate. Querying of the annotated data is possible through a (simple or powerful) query language.

4 Personal Wikis

An orthogonal extension of the original Wiki idea is the concept of a *personal Wiki*. Personal Wikis focus on providing extreme usability for personal information management. They are desktop applications and do in general not offer collaboration functionality. They serve as very lightweight note-taking programs, allowing users to related notes to each other by Wiki links. Although at first hand it seems self-contradictory with the collaborative nature of a Wiki, several of these applications have quickly become quite popular.

4.1 Examples

Tomboy Tomboy[3] is an open-source note-taking Wiki for the Gnome desktop (Linux) and very popular in this community. It is a very simple application: each note can be linked to other notes, or to email addresses and URLs. As in a normal Wiki, these links can be traversed; additionally there is a menu of recently accessed notes. Tomboy offers a full-text search to find relevant notes.

Tomboy is globally available in the desktop: it sits on the menu bar and one can open it with a single keystroke. Notes are automatically saved on each change, and there is an full undo stack for each note. These usability features explain its popularity: one can very quickly take notes in it, by just pressing the global key, typing some text, and pressing Escape to close it. There is no need for opening an application, opening a note, saving it, and rendering the result; all that happens on-the-fly.

Newton Newton[4] is another open-source personal Wiki for the Gnome desktop. It supports richer Wiki syntax than Tomboy, allowing for example numbered and bulleted lists. On the other hand, it is slower in usage: there is a editing mode and a viewing mode. In editing mode one sees the Wiki syntax, in viewing mode the page is rendered.

[3] http://www.beatniksoftware.com/tomboy/
[4] http://newton.sf.net/

WikidPad WikidPad[5] is a personal Wiki for Windows. It has a two-pane layout, showing a tree of all pages on the left hand side, and the current page on the right hand side. It supports a simple Wiki syntax for links and simple page markup, it has an auto-complete feature for Wikilinks, and supports full-text search. Users can assign categories to pages (to organise them), and annotate pages with arbitrary attributes (such as priority, or status of an item); these attributes can be used for aggregate views (e.g. all pages that contain todo items) or for changing system behaviour (e.g. export only pages marked as public).

VoodooPad VoodooPad[6] is a commercial application for MacOSX. It shows recently changed notes, backlinks from the current note to all that mention it, full-text search, auto-complete for Wikilinks. Additionally one can embed simple sketches into notes, and assigning categories to notes (to retrieve them more easily). VoodooPad also integrates with the MacOSX address-book: actions such as sending emails can be performed on recognised email addresses.

4.2 Characteristics

We summarise the characteristics of these various systems. Personal Wikis focus on quick note-taking like scenarios; they commit to ease-of-use by being simple and lightweight, and offering unlimited undo and real autosave. They are simple and have a limited functionality: users can type text, link notes, and perform full-text searches. Personal Wikis are tightly integrated in the desktop (global keybinding, system tray, application integration) and are thus desktop-specific.

5 Functionality analysis

We compile a list of possible functionality based on this analysis of existing applications and ideas about Wikis and their extensions. This list will serve as a requirements analysis for our semantic personal Wiki.

We can distinguish three main groups of functionality: functionality of authoring, retrieval, and navigation.

Authoring describes the available functionality to write, edit, and remove information; we distinguish Wiki syntax (for text layout), instant save (annihilating the need for explicit saving), full undo, global keybindings (making the application instantly available throughout the desktop), and application integration (allowing to reuse data available in other applications).

Retrieval describes the available functionality to retrieve available information; we distinguish full-text search, simple, powerful, and embedded queries, and logical inferencing (that returns implicit information).

[5] http://www.jhorman.org/WikidPad/
[6] http://flyingmeat.com/voodoopad

Navigation describes the available functionality to navigate through the system; we distinguish Wikilinks, intelligent navigation (using implicit information), keyboard navigation (enabling fast navigation), and instant update of the navigational entities (annihilating the need for explicit saving or refreshing).

Table 1 list these requirements in the above order, and indicate the level of support in the discussed systems.

	Wiki syntax	instant save	full undo	global keys	app. integration
Wikis	x				
Platypus	x				
Shawn	x				
Tomboy	x	x	x	x	
Newton	x				
WikidPad	x				
VoodooPad	x				x

(a) Authoring.

	full-text	simple query	power query	embedded query	inferencing
Wikis	x				
Platypus	x	x	x		x
Shawn	x	x	x	x	
Tomboy	x				
Newton	x				
WikidPad	x				
VoodooPad	x				

(b) Retrieval

	Wiki links	intelligent navigation	key navigation	instant updates
Wikis	x			
Platypus	x			
Shawn	x	x		
Tomboy	x		x	x
Newton	x			
WikidPad	x			
VoodooPad	x			x

(c) Navigation

Table 1: Capabilities of Wiki systems

6 Semantic Personal Wikis

Both extensions of the original Wiki concept offer distinct advantages. Semantic Wikis offer a better retrieval rate through intelligent navigation and querying, but have mediocre usability. They are slow to start, since one needs to start a browser, surf to the Wiki, and wait for the server. They are slow in usage, since one needs to explicitly save and commit all changes, and wait for the required server round-trip. And they cannot offer a high-level of desktop and application integration since they are user platform independent; this enlarges the group of possible users but decreases the user experience. Personal Wikis address all these usability issues, and are specifically built to offer a very good user experience; on the other hand, they do not offer collaboration functionality.

A semantic personal Wiki is a combination between the best aspects of semantic Wikis and personal Wikis. It offers the usability of a personal Wiki, allowing users to quickly and unobtrusively write and semantically annotate pages. The semantic annotations are intermixed within the normal text, alleviating the need for a conscious change of mode. The Wiki entices users to add annotations by providing direct rewards in terms of intelligent navigation.

Such a semantic personal Wiki can be seen as a personal information system or as a lightweight ontology editor. It is not aimed towards collaborative work, but on providing usability and gratification: attracting people to use it and add semantics to their information.

6.1 SemperWiki

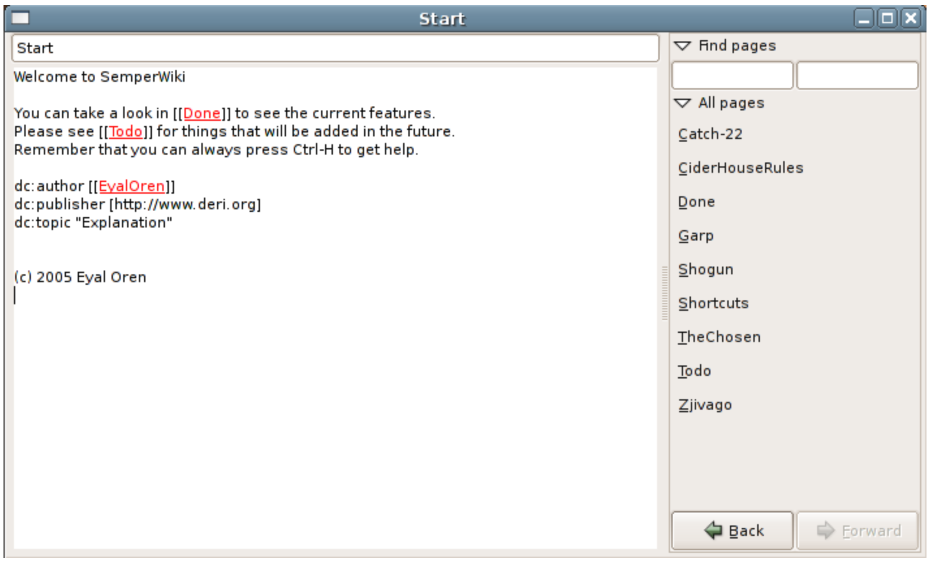
SemperWiki[7] is our implementation of a semantic personal Wiki; it is an open-source application developed for the Gnome desktop. We will give an overview of the system and explain its features in detail.

Overview The user interface of SemperWiki is shown in figure 1. On the left hand side the user can edit pages, on the right hand side the user can navigate; there is no separation between authoring mode and navigation mode. A page can consist of normal text, links to other pages or websites, and semantic annotations.

On the right hand side of the figure we see the navigation bar. This bar contains various ways to navigate through the system. First we see a “find” section, that allows users to query the system for pages containing certain statements. Below the “find” section links to various relevant pages are displayed. In this screenshot it only show the section “All pages”, containing all pages currently in the system, other screenshots show more links as more relevant information is found by the system. Finally we see a history navigation section, that allows users to go back and forth in their navigation history; this feature is very useful if we “jump” into a link to edit some information, and then want to jump back again to where we were before.

Each page represents one resource and annotations state a property about that resource. SemperWiki stores some semantic information about the resources that are described on its pages, using the ontology shown in figure 2. Each resource is represented as a page, each page can contain some text and can have several outgoing links. Other information about a resource is added by the user (the standard graph is enlarged with the semantic annotations).

[7] see http://semperwiki.org/.



Welcome to SemperWiki

You can take a look in [[Done]] to see the current features.
Please see [[Todo]] for things that will be added in the future.
Remember that you can always press Ctrl-H to get help.

dc:author [[EyalOren]]
dc:publisher [http://www.deri.org]
dc:topic "Explanation"

(c) 2005 Eyal Oren

Fig. 1: SEMPERWIKI user interface

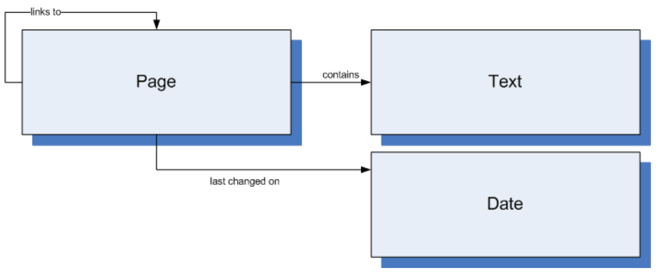


Fig. 2: SEMPERWIKI ontology

SemperWiki stores all information in RDF[8]; we do this for two reasons: RDF is a very flexible representation and allows us to store various information about resources, and secondly RDF statements form the building blocks of the Semantic Web. The collection of triples that SemperWiki stores form a valid RDF model and can directly be exchanged with others.

In the RDF model, each page is identified by a URI, which is formed by prefixing its title with the base URI of the Wiki (defined by the user). Figure 3 shows the RDF triples that are stored about the page `Start`; for readability we have left out several triples.

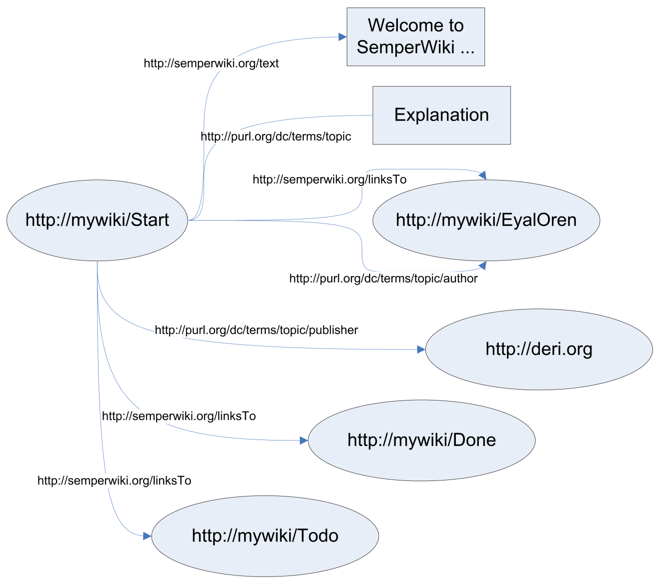

Welcome to SemperWiki ...
http://semperwiki.org/text
Explanation
http://purl.org/dc/terms/topic
http://semperwiki.org/linksTo
http://mywiki/Start
http://mywiki/EyalOren
http://purl.org/dc/terms/topic/author
http://purl.org/dc/terms/topic/publisher
http://deri.org
http://semperwiki.org/linksTo
http://mywiki/Done
http://semperwiki.org/linksTo
http://mywiki/Todo

Fig. 3: RDF statements about the page “Start”

Syntax Figure 1 contains ordinary text, including links to other pages in this Wiki. We also see some semantic markup, stating that the author of this page is a resource in this Wiki (namely the page `EyalOren`), that the publisher is DERI, and that the topic is “Explanation”.

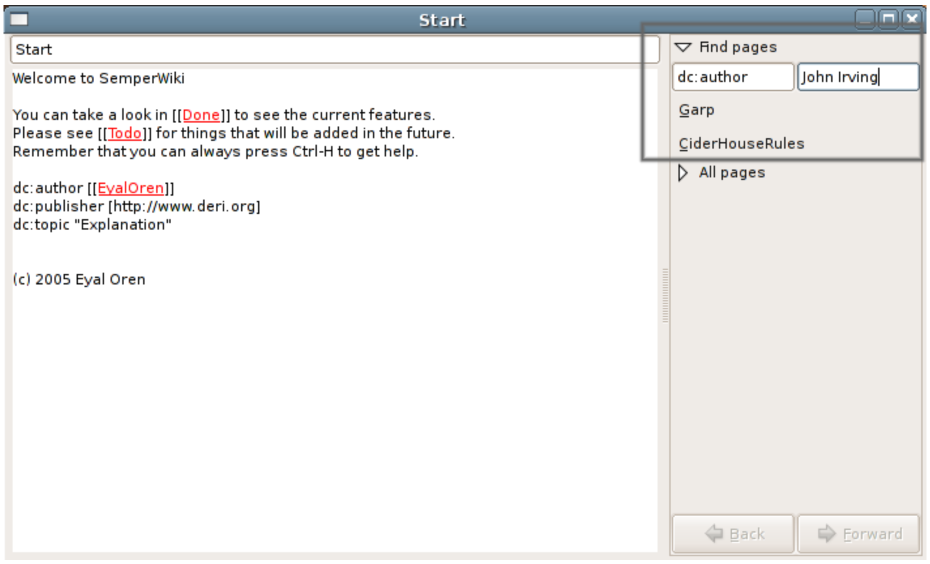
For the semantic annotations we use a simple syntax. A statement is written on a line by itself and consists of a predicate followed by an object. Such a statement is expanded to a triple using the URI of the page as a subject. Predicates are resources, objects are resources or literals. Resources can be written as their full URIs, with prefix notation, or as Wikilinks. Wikilinks are expanded using

[8] http://www.w3.org/RDF/

the Wiki base URI, prefixed URIs are expanded using namespace abbreviations; these abbreviations are configurable through a preference dialogue.

The syntax (for respectively a full URI, a internal Wiki link, a prefix abbreviation, and a literal) is as follows: `[fullURI]`, `[[WikiLinks]]`, `prefix: localname`, `‘‘literal’’`.

Navigation SemperWiki offers various ways to navigate to a page. First, one can click on any Wikilink to jump to a page. Second, if one knows the name of the page, pressing `Ctrl-G` will ask for the name and jump directly to it. Third, the history buttons can be used to go back and forth through visited pages. Fourth, the navigation sidebar will automatically show links to various related pages that can be visited by clicking on them; the next section explains how it works. And finally, one can query for any page containing statements by typing a predicate and/or object in the find section. An example query for all things written by John Irving is shown in figure 4, where the system found two relevant pages.



Start

dc:author | John Irving

Welcome to SemperWiki

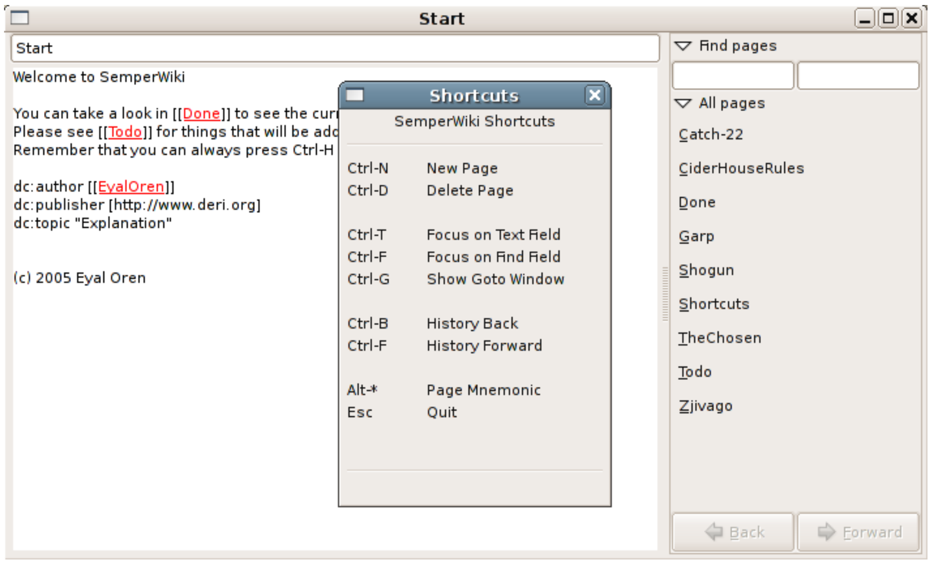
You can take a look in [[Done]] to see the current features.
Please see [[Todo]] for things that will be added in the future.
Remember that you can always press Ctrl-H to get help.

dc:author [[EyalOren]]
dc:publisher [http://www.deri.org]
dc:topic "Explanation"

(c) 2005 Eyal Oren

Fig. 4: Using the “find” section

SemperWiki is designed to help users navigate quickly and efficiently to their information. Therefore all actions in SemperWiki can be triggered by keyboard commands. These shortcuts can be accessed by typing `Ctrl-H`, as shown in figure 5. Also, as can be seen in figure 1, all links are assigned mnemonics (the underlined first letter) which triggers them with a keyboard command; for example `Alt-D` directly jump to the `Done` page.



```
Welcome to SemperWiki

You can take a look in [[Done]] to see the cur
Please see [[Todo]] for things that will be add
Remember that you can always press Ctrl-H

dc:author [[EyalOren]]
dc:publisher [http://www.deri.org]
dc:topic "Explanation"


(c) 2005 Eyal Oren
```

SemperWiki Shortcuts

Ctrl-N	New Page
Ctrl-D	Delete Page
Ctrl-T	Focus on Text Field
Ctrl-F	Focus on Find Field
Ctrl-G	Show Goto Window
Ctrl-B	History Back
Ctrl-F	History Forward
Alt-*	Page Mnemonic
Esc	Quit

Fig. 5: Getting help

Dynamic Sidebar The links on the sidebar are generated automatically based on the available semantic information in the system. We show sets of pages that are related to the current one, and order these relations by the size of the sets of related pages. We show the most specific pages first (the smallest set of related pages), and gracefully decrease the amount of correspondence until we show all pages.

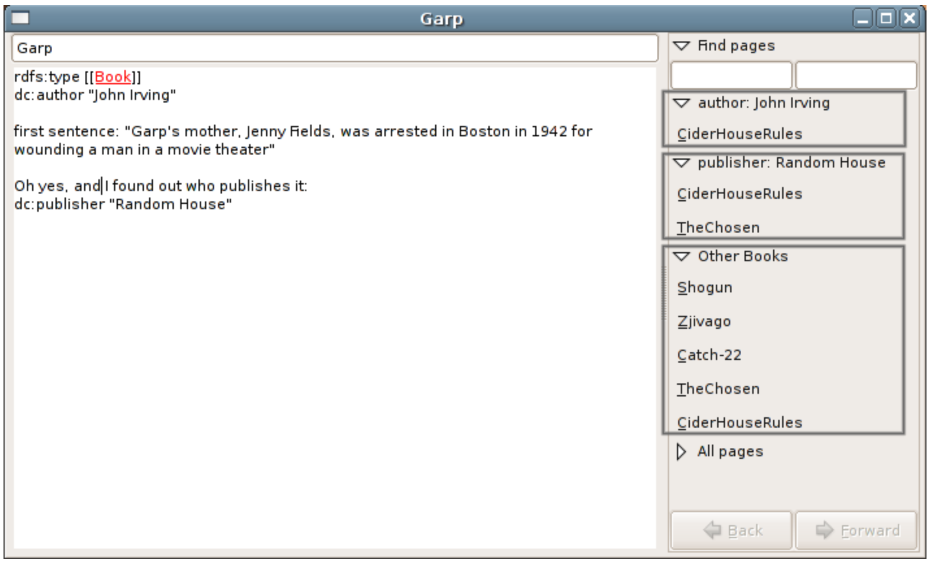
For example, the page `Garp` will show all books that also have John Irving as author (see figure 6). If there are any other predicates:objects in common we also show them, such as other books that are published by Random House.

Then we show for example all pages that are of the same `rdfs:type`; for Garp we show all other books in the system. This set is shown later, because it contains more results than those books that also have the same author as Garp.

We also display pages that have a reversed relation to the current one (but in a different colour). For example, the sidebar of the page `Book` would show links to all books.

Instant Response Everything in SemperWiki behaves instantly. All changes to all pages are saved instantly, without any user interaction. Likewise, the current page and location are saved instantly. Together this means the user can leave the system at any time (by pressing `Esc` and return the next time exactly where he left it.

All links are found and tagged while the user is typing them, without any need for saving or refreshing the page. All sidebar links are updated instantly upon any change in the page. This means that while a user is adding information to


Garp

rdfs:type [[Book]]
dc:author "John Irving"

first sentence: "Garp's mother, Jenny Fields, was arrested in Boston in 1942 for wounding a man in a movie theater"

Oh yes, and I found out who publishes it:
dc:publisher "Random House"

Fig. 6: Pages related to Garp

the page, the sidebar link is continuously up-to-date. If a user adds to some page the statement `dc:author "John Irving"`, the sidebar immediately updates to show other books by John Irving.

Having an instantly responding user interface greatly enhances the user experience. We can provide this responsiveness by caching all querying results and by using efficient regular expressions to parse the typed text. Currently, the system is very fast but we need to evaluate its scalability in terms of the size of the knowledge base.

Implementation The system is targeted at the Gnome desktop. As discussed, targeting a specific desktop is a trade-off between offering more users a lesser experience or offering less users a better experience, in which we have chosen the latter.

SemperWiki is implemented in Ruby[9], using the GTK[10] windowing toolkit for the graphical programming. It uses Redland[11] for RDF storage and retrieval. It currently consists of around 500 lines of code, many of which are related to programming the graphical widgets.

Redland is a mature RDF store, but it is still quite small so that it can be easily embedded in an application. It supports language bindings for many

9 http://ruby-lang.org
10 http://www.gtk.org/
11 http://librdf.org/

different languages, including Ruby. We have also considered YARS [3], but at its current state it does not offer enough functionality.

The choice for the Gnome desktop was a personal one, but it is also the default desktop of Ubuntu, currently the fastest growing Linux distribution. The GTK toolkit is the default toolkit of the Gnome desktop, it is consistent and easy to use. Furthermore, it has excellent Ruby bindings.

The choice for Ruby was made for development speed. As an interpreted language it allows one to program very fast and develop prototypes very easily.

7 Evaluation

We can not evaluate SemperWiki against other semantic personal Wikis, as we are unaware of other systems. Also, we cannot do a user-based analysis of our tool, since the tool does not have an active user base yet. However, we can compare our current functionality against the functionality analysis in section 5.

It is clear that SemperWiki offers very little functionality in querying. That is actually on purpose: since we have no good solution for a simple but powerful query interface, we currently prefer the navigation sidebar: it is much easier to use. Furthermore, we can see that SemperWiki supports all encountered navigational capabilities, and almost all authoring capabilities; the missing "full undo" is actually planned for imminent implementation. Support for desktop application integration might an interesting discussion point for this workshop.

	Wiki syntax	instant save	full undo	global keys	app. integration
SemperWiki	x	x		x	

	full-text	simple query	power query	embedded query	inferencing
SemperWiki		x			

	Wiki links	intelligent navigation	key navigation	instant updates
SemperWiki	x	x	x	x

Table 2: Capabilities of SemperWiki

8 Conclusion

Managing the amount of information in our personal desktops is a challenge. The idea of a Semantic Desktop addresses this challenge by semantically annotating information chunks on our desktop, allowing better retrieval of relevant information. Since these semantic annotations have to be (partially) added by users, one must entice them to do so.

We have presented the idea of *semantic personal Wikis* to allow and encourage users to create semantic data. We have incorporated ideas from both semantic and personal Wikis: semantic Wikis focus on improved retrieval and navigation, personal Wikis focus on usability.

Our application SemperWiki allows users to manage their information: they create pages that contain both normal text and semantic annotations at the same time. They are instantly rewarded for these annotations by improved navigation possibilities. The system is simple, easy to use and responds instantly to all changes.

For future work we aim to address several issues. First, currently pages and resources have a one-to-one correspondence: each resource is represented by exactly one page. This makes annotations easier: statements are made using only a predicate and object, the subject is always the URI of the page. However, it makes editing complex data cumbersome since one needs to edit a new page for each resource, and it poses problems with blank nodes. Furthermore it prohibits making statements about external resources since all subjects have to be Wiki pages. We aim to decouple the relation between pages and resources, thereby allowing annotations of arbitrary resources (internal and external) and more complex statements.

Secondly, we want to improve the query and navigation interface, such that it is still easy to use and yet allows experienced users to make more powerful queries. For example, the list of related pages in the dynamic sidebar could be very long and we currently do not provide means to shorten or search through this list. For this we will look into related work such as the Longwell browser in the Simile project [6].

Finally, we aim to investigate possibilities to share the authored RDF data with others, either by publishing it on a persistent web location, or by sharing it over a peer-to-peer network with other users of SemperWiki. That way, what starts as a contradiction-in-terms for private use (a personal Wiki) returns to the Semantic Web.

Acknowledgements: this material is based upon works supported by the Science Foundation Ireland under Grant No. 02/CE1/I131. We thank Knud Möller and Stefan Decker for valuable discussions, and the anonymous referees for their suggestions of improvement.

References

[1] D. Aumueller. Semantic authoring and retrieval within a wiki. In *Proceedings of the European Semantic Web Conference (ESWC)*. 2005.

[2] S. Decker and M. Frank. The social semantic desktop. Tech. Rep. 2004-05-02, DERI, 2004.

[3] A. Harth and S. Decker. Optimized index structures for querying RDF from the web. In *Proceedings of the 3rd Latin American Web Congress*. 2005.

[4] B. Leuf and W. Cunningham. *The Wiki Way: Collaboration and Sharing on the Internet*. Addison-Wesley, 2001.

[5] L. Sauermann. *The Gnowsis – Using Semantic Web Technologies to build a Semantic Desktop.* Master's thesis, Vienna University of Technology, 2003.

[6] V. Sinha and D. Karger. Magnet: Supporting navigation in semistructured data environments. In *Proceedings of SIGMOD 2005.* 2005.

[7] R. Tazzoli, P. Castagna, and S. E. Campanini. Towards a semantic wiki wiki web. In *Proceedings of the International Semantic Web Conferenc (ISWC).* 2004.