Colonising Web Sites of Wiki Pages with Ultra Lightweight Web Applications

Dr Michael Rees [HREF1], School of Information Technology [HREF2], Faculty of Business, Technology and Sustainable Development, Bond University [HREF3], Qld 4229, Australia.
mrees@bond.edu.au

Abstract

As an ultra lightweight web application DotWikIE (Rees, 2006) showed that a single web page, loaded directly from the local machine's filestore, could support a wiki application running within a web browser. This allows the web page to carry data content which can be used and manipulated by a browser on any machine without requiring an Internet connection. The single web page contains both the application logic and data repository for the wiki, is highly portable, and can easily copied for backup and deployment.

While DotWikIE is useful a web-based wiki with the same functionality has the advantage of being accessible on any Internet-connected machine. DotWikIEWeb is the evolution of DotWikIE as an ultra lightweight web application that works either from the local filestore or from a web site. This paper presents the technological problems and discusses an implementation of DotWikIEWeb and its ability to become the single page seed of a colony of associated wiki pages. DotWikIEWeb retains the benefits of single page web applications while gaining the capability to operate on a web site.

Because of their flexibility wikis in general tend to become unstructured quickly as the user grasps the freedom to populate and format each wiki component in an ad hoc way. This is seen as one of the main advantages of a wiki. The paper concludes by discussing some approaches to how wikis could retain a more regular structure for their content.

Keywords

Ajax, XML, JavaScript, single page application, ultra-lightweight web application, wiki, wiki interchange format

Introduction

As a mark of recognition of the concept of browser-based software packages there is now a Wikipedia entry [HREF4] for single page applications. The definition used is 'a web application that runs entirely in the client web browser'. A single page application consists of individual web pages that can interact with the user and dynamically update their content in situ. This is achieved using the Ajax technologies of Javascript code and background, asynchronous HTTP requests to save and load additional page content.

Rees (2006) discusses ultra lightweight web applications that are a sub-category of single page applications. An ultra lightweight web application refers to a highly-restrained single page application that is completely self-contained. This means that all XHTML and XML content, CSS style sheet rules and Javascript code is held within a single web page file. As has been described (Rees, 2006), ultra lightweight web applications offer significant benefits:

1. Data content updated by user interaction overwrites the original content forming a permanent data store.
2. The whole application plus content is highly portable, and can
    - be carried with the user on a USB drive, for example, and
    - works on any machine with the supported browser installed.
3. Only a single file needs to be managed which is also easily duplicated and backed up.

Figure 1 shows the components of the DotWikIE ultra lightweight application described by Rees (2006) drawn approximately to scale. The CSS embedded style sheet, JavaScript code, XHTML

markup and the XSLT style sheet are of fixed length and occupy 47K in each DotWikIE page. DotWikIE version 2.1 from which the sizes were taken also contains about 7K of standard I-grains, the wiki sections that form the content.

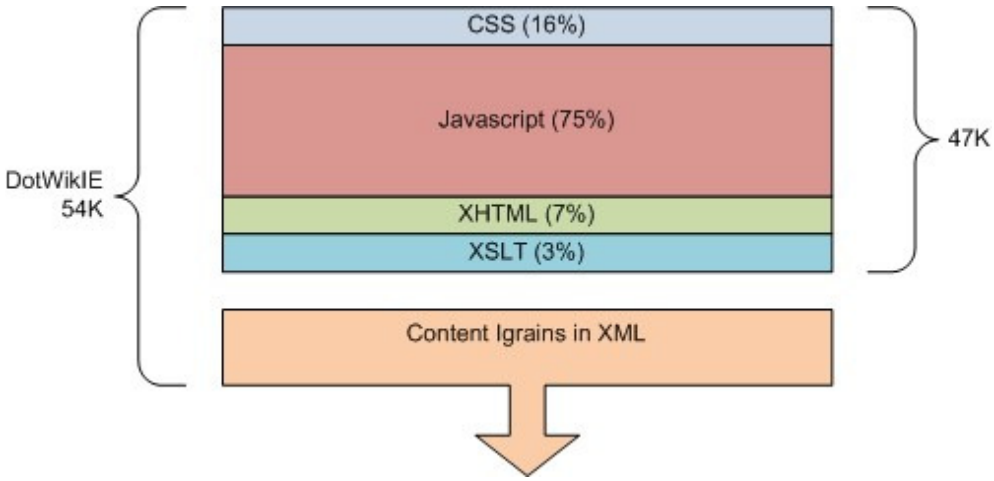

Figure 1. DotWikIE Components.

Of course the user edits the I-grains during normal use so that this content will typically grow in size. The simple nature of DotWikIE in practice limits the number and content of the I-grains. The author uses DotWikIE to collect and store web page fragments and text, images and links copied from other documents and email. In this scenario the I-grain contents rarely exceed about 50K and average 25K over about 20 individual DotWikIE pages.

From another viewpoint an ultra lightweight web application can also be regarded as a *self-managed document* which contains its own processing and layout rules as well as the document content. However this paper continues with the web application theme and the docucentric approach will be the subject of another publication.

In its original incarnation DotWikIE was designed as a free-standing web page stored and used directly from the local filestore. Here both Firefox, IE6 and IE7 allow JavaScript to write back to the filestore after prompting the user for permission to do this. When placed on a web site and accessed with the http:// protocol the client-side JavaScript is sandboxed and cannot affect the web server's filestore, so the wiki becomes read only. Nevertheless in this mode DotWikIE files can be used as simple and effective static web pages.

Being read-only when accessed on a web site turns out to be a fortuitous by-product of an ultra lightweight web application. Since the web site owners typically have direct access to DotWikIE web site files they are the only ones able to effect changes to the DotWikIE web pages. Used in this way DotWikIE becomes a very simple and effective web page publishing mechanism.

Moving Ultra Lightweight Web Applications to the Web

Rees (2006) describes an extension to DotWikIE that mimics the read-write behaviour on local filestore when viewed on a web server. This extension is called DotWikIEWeb and uses a simple server-side script resident in an .asp file to perform the page overwriting on the web server. DotWikIEWeb simply checks to see if it being used on the local filestore or on the web and executes the appropriate JavaScript code for the execution environment. In fact this extra functionality is included in the sizes of DotWikIE components shown in Figure 1. DotWikIE and DotWikIEWeb are currently the same web page that adjusts to its environment automatically.

However DotWikIEWeb in this form consists of two files:

1. the DotWikIEWeb.htm page containing the client-side JavaScript (and CSS, XML and XSLT data)
2. The .asp server-side script file that performs the file handling on the web server

The .asp file used by DotWikIEWeb of course means it is no longer a genuine ultra lightweight web application but comes with a satellite server-side script. In the case of DotWikIEWeb the satellite file is 3.2K, 78 lines of server-side JavaScript. This limits use to Microsoft's IIS web server. However rewriting the satellite in a 'P' language such as PHP for Apache is a very straightforward task.

The challenges reported in this paper are to:

- Reduce the two files to a single web page to conform to the ultra lightweight web application definition once more
- Allow DotWikIEWeb to spawn copies of itself on the web server to create a colony of wiki pages that can act in a simple way as a single composite wiki

The remaining sections in this paper outline the design objectives, some related work in this area, consideration of implementation options and the eventual implementation of the enhanced DotWikIEWeb. A final section before the conclusion discusses some approaches to how a simple but effective wiki like DotWikIEWeb might begin to impose a structure on the ad hoc nature of wiki content and thus exploit the XML wiki content already present in each DotWikIEWeb file.

Previous Work

As was acknowledged by Rees (2006) the ultra lightweight web application in a single page concept used by DotWikIE was inspired by the TiddlyWiki [HREF6] application from Jeremy Ruston. Although lacking several of the tagging, searching and plugin features of TiddlyWiki, a major advantage of DotWikIEWeb is the substantial improvement in WYSIWYG editing for wiki content (see below).

Independently Ruston's work on TiddlyWiki has inspired Simon and Daniel Baird of BidiX in Australia to create a web-based version. BidiX has worked on TiddlyWiki extensions [HREF9] and now has an impressive web application called TiddlySpot [HREF10] available for public use at no cost. Sign up is quick and simple and in just a few seconds a user has access to a single wiki page with the extensive features of TiddlyWiki on the web. Figure 2 shows a part of the wiki page that is accessible via a URL like http://mrees.tiddlyspot.com.

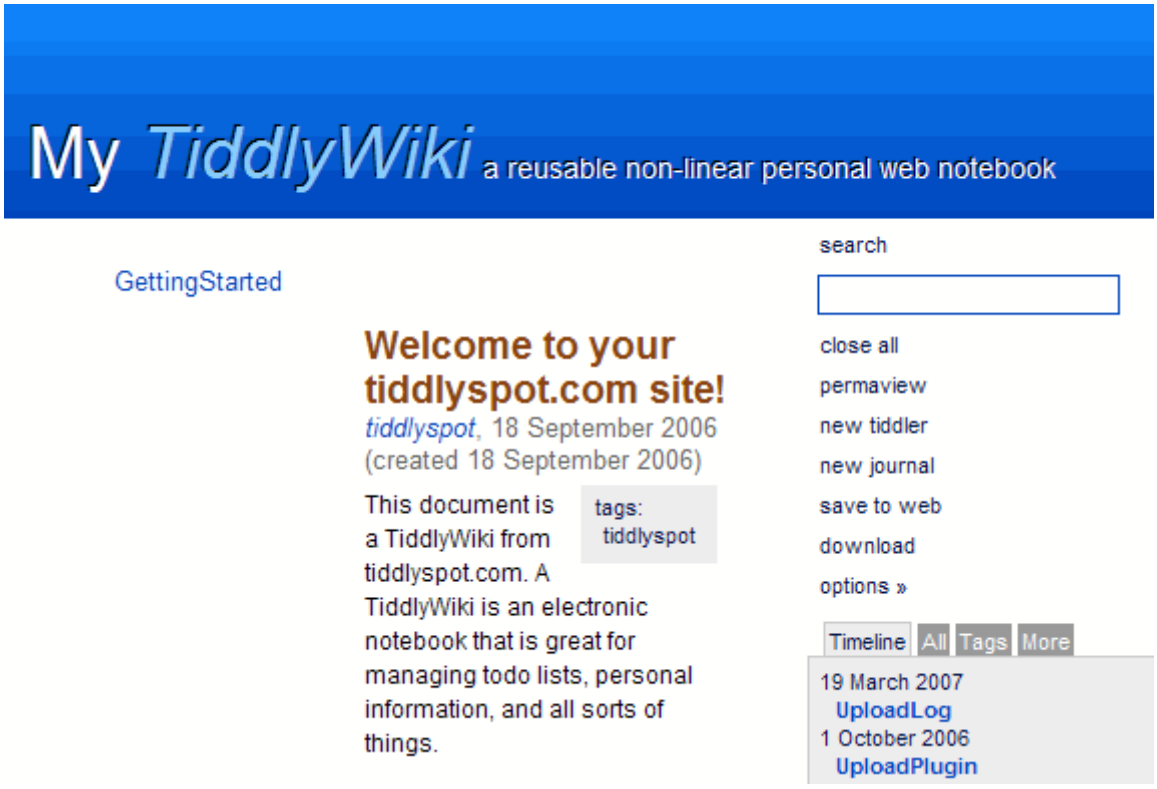

Figure 2. Default Tiddlyspot Wiki Page.

In order to support user names and passwords the Bairds have of necessity broken away from the ultra lightweight web application on the web server. They leverage the standard HTTP authentication modules built in to the Apache web server, so additional files are needed to support Tiddlyspot. Nevertheless the download link in Figure 2 allows the capture of the single page (file) on the local filestore where it becomes a standard TiddlyWiki page. As usual this allows editing and saving locally. With IE6/7 on Windows XP the wiki page needs to prompt the user for permission to run Javascript and use the ActiveX control for manipulating the local filestore just as DotWikIE and DotWikIEWeb do.

Many other web-based wiki software packages are available some open source and some commercial. They adopt the traditional multiple-page paradigm even though the URL displayed in the browser address field may appear to be a single page. Thus they do not fall into the ultra lightweight web application category.

Developing Web-based Wikis with DotWikIEWeb

For the original, non-ultra lightweight web application version of DotWikIEWeb the installation on a web site is exceedingly simple:

1. Copy a DotWikIEWeb page *and* its satellite file to a folder controlled by a web server
2. Allow the satellite server-side script to write to that folder

DotWikIEWeb then allows additional wiki pages to created, edited and deleted as needed. The author has used DotWikIEWeb in this way over several problem-free months. The only unintended outcome was the discovery that when copying and pasting web page content some hyperlinks happen to contain wikiwords. Of course DotWikIEWeb converted these to intra-wiki links within a link! A simple change to exclude this context for the wikiword detection code was able to solve this problem.

Over this trial period the author has used DotWikIEWeb at least daily on a hosting service (Seekdotnet [HREF5]) which offers shared web site hosting on Windows machines running IIS. Indeed the author's browser home page is a DotWikIEWeb page containing frequently-used hyperlinks. It takes a matter of a few seconds to edit an I-grain to add, edit or delete links, and then click the Save All button to make the change permanent. It is equivalent to having permanent access to the organise favourites/bookmarks feature of most browsers when the browser starts up or when the Home button is clicked. Figure 3 shows an example of this master links page and the type of format that is possible in DotWikIEWeb.

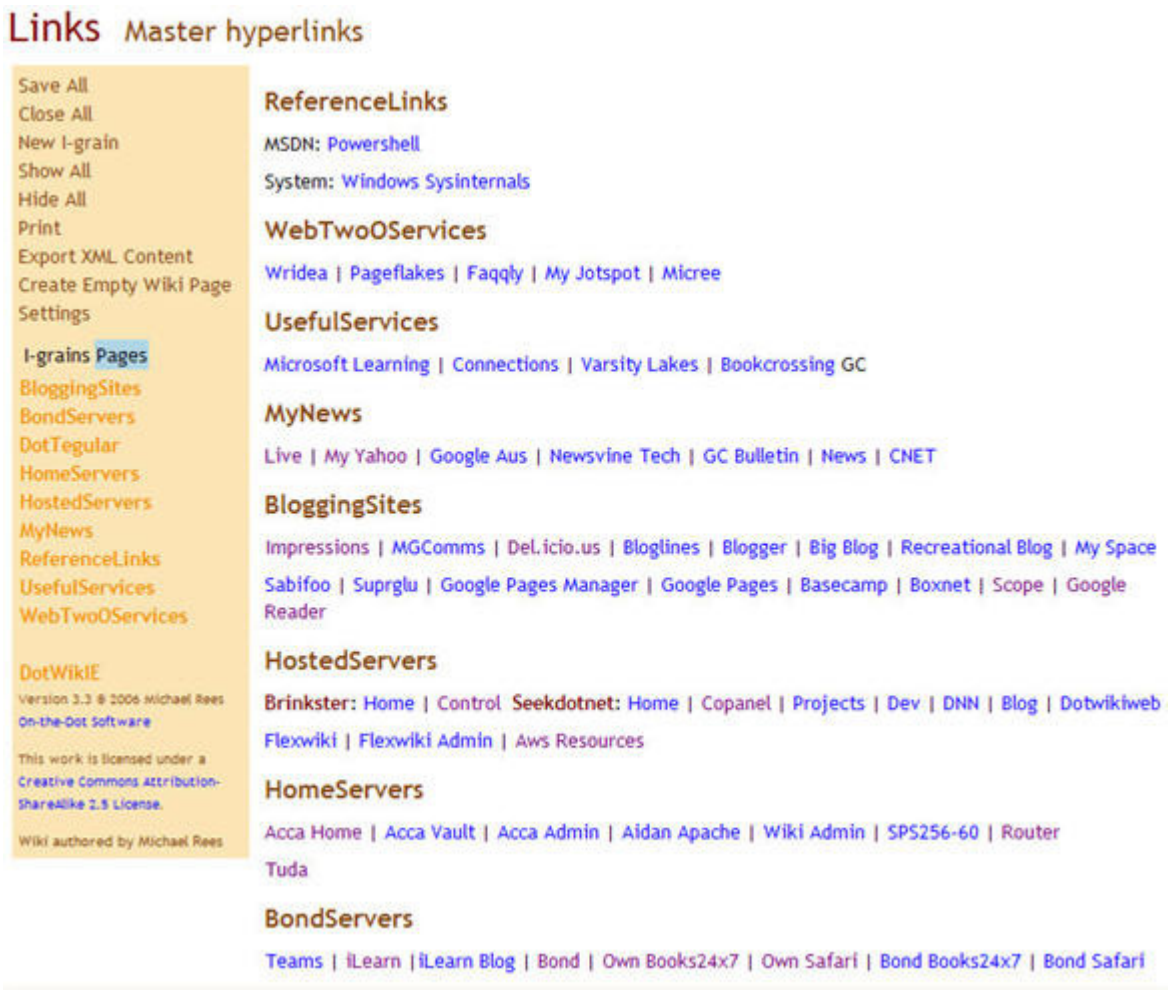

Figure 3. DotWikIEWeb Master Links Page.

The hosted DotWikIEWeb pages are accessible from anywhere on the Internet so the wiki page contents are instantly accessible from any Internet-connected machine running Internet Explorer. There has been a downside to this very simple and effective mechanism. As well as the author anyone with a knowledge of the URL can both view and edit the pages. Over several months this did not prove a problem but obviously an authentication mechanism is needed. Other improvements were needed to make DotWikIEWeb a viable multiple-page wiki served from the web.

The first extension to DotWikIEWeb attempted was to reduce the web-based operation to a single file to reinstate the ultra lightweight web application status. This meant that the DotWikIEWeb page file must be capable of generating the satellite file containing server-side code itself on first activation. To achieve this every DotWikIEWeb page must carry with it the satellite generation mechanism.

Once the DotWikIEWeb.htm page is placed in a writable folder on a web server, and displayed in the browser using it local file path, it checks for the existence of the satellite file. If it is not present then the satellite file is created. This begins the colonisation of the cluster of wiki web pages, ie the *wiki page colony*. DotWikIEWeb need only be accessed once in this way. Of course this leads to a slight increase in the size of the management Javascript code with additional components as shown in Figure 4.

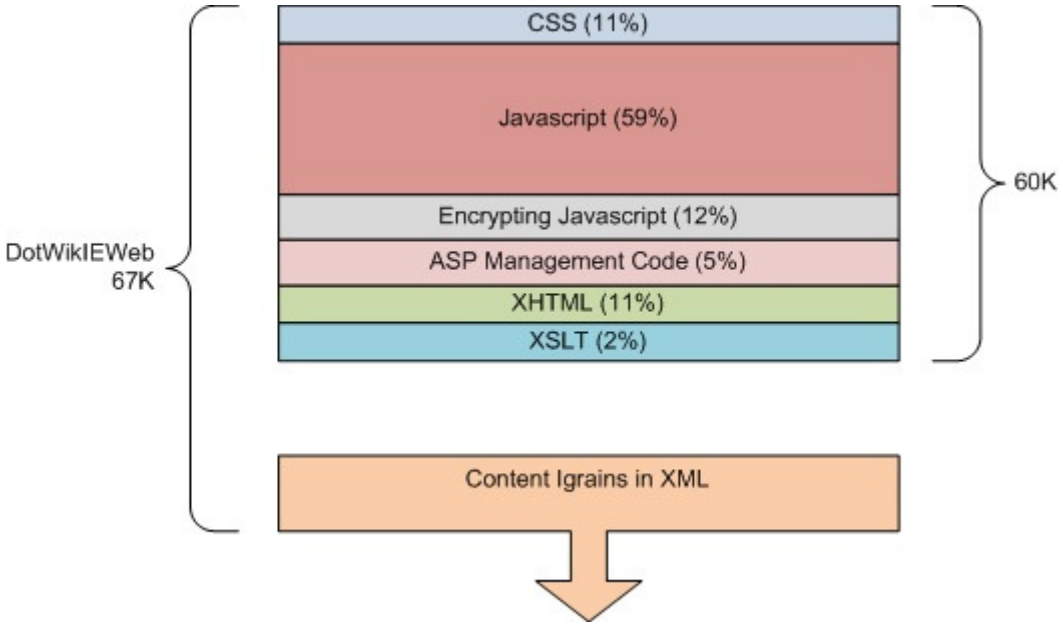

Figure 4. DotWikIEWeb Components.

There are three major changes compared to the makeup of DotWikIE in Figure 2:

1. The ASP code that will form the satellite server-side code file is contained in an XML data island within the page.
2. Additional XHTML is needed to incorporate the extra features of DotWikIEWeb in the web environment.
3. A significant addition to the JavaScript code to implement authentication features described in further detail below.

It will be seen that overall size of DotWikIEWeb as distributed with a standard set of I-grains has grown only from 54K to 67K, a modest 24% increase for the ability to use the same wiki pages in a web environment.

A very simple and straightforward set of features is needed to manage the wiki page colony, the collection of DotWikIEWeb pages in a single folder on a web site. Having spawned the satellite server-side code file, additional copies of the first DotWikIEWeb page are created with the Create Empty Wiki Page button shown in Figure 3. As with all the web-based features the JavaScript code determines the environment, web or local filestore. If it is the web context then the new empty wiki page with the name entered by the user uses an XMLHTTPRequest call, the heart of Ajax, to create the new wiki page. Figure 5 shows the simple dialog for creating a wiki page.

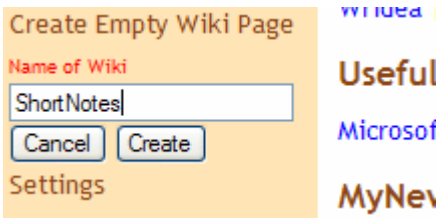

Figure 5. Create New Empty Wiki Page.

Once a colony of wiki pages begins to grow there must a mechanism for quickly moving from one page to another. By default each DotWikIEWeb wiki page lists the I-grain sections in that wiki page. As shown in Figure 3 listing the pages in the wiki colony is available via the Pages button adjacent to the I-grains button. Again this is populated using an XMLHTTPRequest call to the satellite management code to list all the wiki pages in the same folder on the web. Figure 6 shows the simple list of hyperlinks to the wiki page colony.

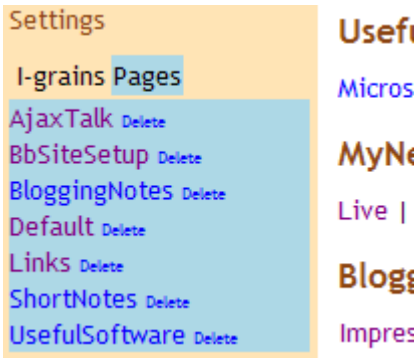

Figure 6. List of Pages in Wiki Colony.

Alongside each page link in the list appears a subsidiary button to delete that page after being confirmed by the user. Each wiki page can be visited and edited as normal via this list. As with DotWikIE there is versioning available in DotWikIEWeb and these versions also appear in the list if they are present. A DotWikIEVersions I-grain used to be the mechanism by which the versions were made visible. This required awkward JavaScript code to maintain and has been deleted in favour of the simple page list features in DotWikIEWeb. This has saved probably 100 lines of JavaScript code amounting to about 2 K in file size. The I-grain list is restored again by clicking the I-grains button.

The requirement to authenticate the user to limit the ability to edit and delete pages in the wiki colony has not been so straightforward to implement. This is hardly surprising as a normal implementation mechanism requires a centralised storage location for user names and passwords. Such a concept violates the ultra lightweight web application principle because it requires additional files.

Since each DotWikIEWeb page contains a data store the names/passwords could be stored there. However it would be very confusing for users to be faced with a different set of names and passwords in each wiki page. In any case a wiki page colony on the web is likely to be centred on a single human user or at best a small group of users sharing the same wiki colony. With this usage scenario in mind it was decided to manage only a single password which grants creation, update and deletion rights to each wiki page colony. While only one password is needed this must be shared amongst all pages. Since the satellite ASP file must be separately spawned it was decided to add a second password file to hold the encrypted password.

The file DotWikIEWebPassword.pass file is a simple XML file holding the password encrypted with the SHA1 algorithm (see the Wikipedia entry [HREF7]). Even though SHA1 has been compromised it is still sufficiently strong to protect a DotWikIEWeb wiki page colony. DotWikIEWeb adopts the widely used JavaScript implementation of SHA1 from Paul Johnston [HREF8] which occupies only 200 lines of code. Indeed DotWikIEWeb only uses one of several JavaScript functions for SHA1 in the Johnston code so with careful inspection the code size could be substantially reduced.

The single password file is shared by all pages in the wiki colony. A few lines of ASP in the satellite file checks the user's password encrypted with SHA1 against the value in the password file, allowing changes only on a match. A downside is that each DotWikIEWeb wiki page must prompt for the password at least once where changes are made to that page which can be tedious when changes are needed to several pages in succession. If the user allows it this perpetual prompt for a password can be alleviated by using a cookie with an expiration of a typical session such as 20 to 30 minutes.

DotWikIEWeb, as the web version of DotWikIE, was designed to free the user of the shackles of having to learn special wiki formatting syntax. Most wikis use plain text editing and special escape sequences for headings, bullet and number lists, italic and bold and other layouts. DotWikIEWeb employs the WYSIWYG expected from applications like Microsoft Word. Figure 7 shows the edit mode user interface exposed when the user clicks on the Edit button of an I-grain.

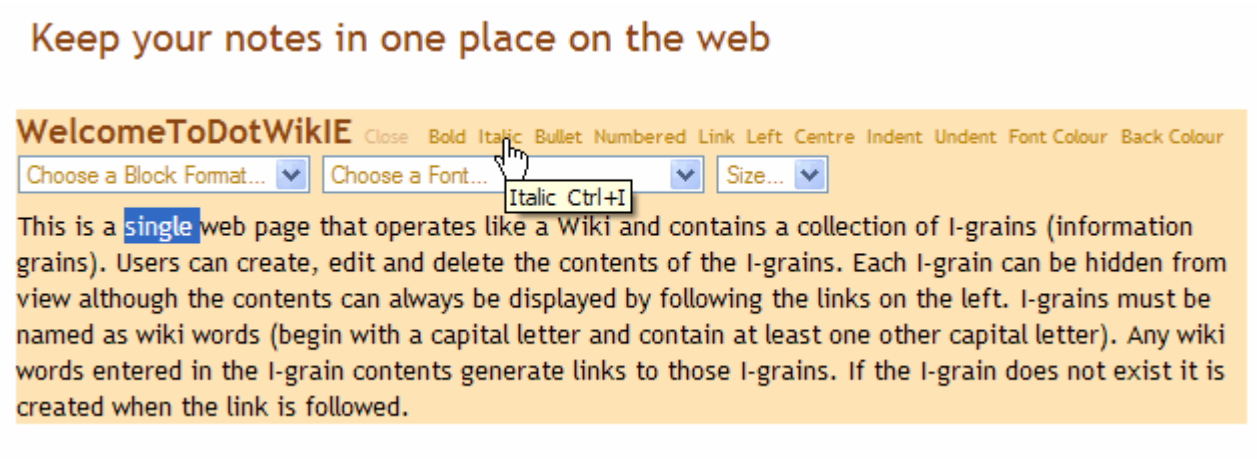

Figure 7. DotWikIEWeb WYSIWYG Editing.

The usual GUI editing process is used: select text or set the insertion point then choose an action. This is made possible by the built-in HTML editor of Internet Explorer which had been available from version 5 onwards. Another Internet Explorer-only feature is the use of a special dialog helper <object> that allows the dropdown lists for block formats and fonts in Figure 7to be dynamically generated.

Usually it is possible to display a row of toolbar icons instead of the text buttons to control the editing functions. The single-file restriction of an ultra lightweight web application prevents this as all images, no matter how small, must be stored in separate image files. Carefully chosen names for the text buttons, while taking up a little more screen real estate, can actually be an improvement over toolbar icons that often need tooltips to explain their operation. As can be seen from Figure 7 tooltips can be used in DotWikIEWeb to elaborate on the button names as well.

The only other special processing of the I-grain contents is the search for wikiwords. When found these wikiwords must be extended into links to individual I-grains of the same name. This is an expected feature of all wikis. Since the I-grain may be scrolled off screen or currently hidden a JavaScript link is used to make the I-grain visible on the screen when the link is followed. The other important behaviour of a wikiword link is that if the I-grain does not exist it is automatically created when the link is first followed.

Fortunately another excellent by-product of the built-in HTML editor is that a word that appears to be a URL or an email address is automatically turned into the appropriate link. Thus external links are automatically catered for. However that leaves a gap in the supported mechanisms for links to other pages in the same wiki page colony. Thus another mechanism was needed for DotWikIEWeb. In the end a popular markup from plain text wiki editors was adopted. If the user places text within double square brackets it is assumed to be a link and an <a> tag sequence is generated. Thus

    [[ShortNotes]]

inserted into the contents of an I-grain would be transformed into a relative link to the ShortNotes.htm page in the same folder.

Thus the major features of DotWikIEWeb that make it suitable for supporting web-based wikis are:

- built-in wiki page colony creation after the import of the single DotWikIEWeb web page; web folder file management server-side script satellite file and password XML file created
- extension of the empty wiki page creation mechanism to operate on the web
- a simple wiki page list mechanism that incorporates a deletion facility
- a password mechanism for protecting creation/editing/deletion of wiki pages, allowing shared web access by knowing the password
- a simple syntax extension to I-grain content to allow the insertion of link between wiki pages

in the same colony

These code additions, which also incorporated a user interface and code simplication, added 24% to the size of DotWikIEWeb compared to DotWikIE.

Working with the Structure of Wiki Content

Fundamental to the wiki concept is the ability of users to add/edit/delete content as required and with no limit of layout within each page. The free-form and flexible nature of how wiki content grows means that it is difficult for users to maintain an information structure that will stay the test of time. DotWikIEWeb adopts the small wiki fragment concept (I-grains in DotWikIEWeb) from TiddlyWiki that uses tiddlers. These I-grains are listed in alphabetical order on the left as seen in Figure 3. This list acts as an automatic table of contents and is a visual navigation aid within a single wiki page. Individual wiki pages in the colony are listed alphabetically in place of the I-grains when the Pages button is clicked so giving access to a table of contents for the whole colony. Thus DotWikIEWeb provides via the I-grains an additional layer of content structure compared to a normal wiki.

It should be noted here that TiddlyWiki in its latest versions also supports tagging of each tiddler. This adds an additional granularity to the wiki content structure. However the success of tagging is very much dependent upon the effort the individual user puts into an appropriate tag set and the frequency of applying tags at the point of creating and editing wiki content. The names of I-grains which must be wikiwords at least forces the use of meaningful names that can be used to impose some structure on the wiki content.

The importance of extracting wiki content structure is leading increased interest in semantic wikis. The first workshop on this topic, "SemWiki2006 - From Wiki to Semantics", was held at the 3rd Annual European Semantic Web Conference in July 2006. Such semantic wikis attempt to combine the strengths of the semantic web (machine processable, data integration, complex queries) and the wiki (easy to use and contribute, strongly interconnected, collaborative) technologies. The goals of semantic wikis include:

- simple annotations of existing wiki content;
- tools that guide users from informal knowledge contained in texts to more formal structures;
- full-fledged tools for ontology editing where the text is no longer in the focus of the system.

One of the papers from the workshop by Volkel and Oren (2006) addresses the issue of a wiki interchange format that the authors name WIF. Although WIF is a heavyweight format the authors realised that trying to harmonise the myriads of proprietary wiki formats was not useful. Instead they opted to start with a data model shared by all wikis as shown in Figure 8.

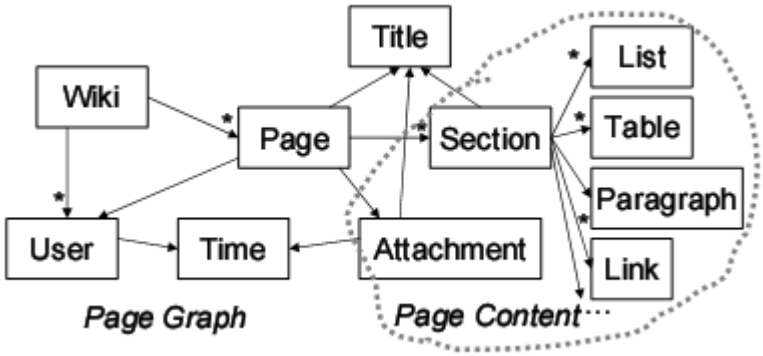

Figure 8. A High-Level View of a Wiki Data Model.

Being a conventional wiki DotWikIEWeb fits the wiki data model of Figure 8 exactly apart from the lack of attachments. Should WIF become a widely used interchange standard it will be straightforward to generate a WIF file from the XML already exported by DotWikIEWeb (see the Export XML Content button in Figure 3).

In the meantime a future version of DotWikIEWeb will concentrate on a more lightweight version of WIF in pure XML format. Referring to Figure 8 DotWikIEWeb already outputs for each wiki page the data values for the Page, Title, Time and Section (I-grain) boxes in the model. Taking a suggestion from the WIF authors it will be a simple matter to extract from each I-grain their proposed XHTML subset of a, dd, dl, dt, em, h1-h6, hr, li, ol, pre, strong, table, td, tr and ul elements. Once the lightweight WIF for each wiki page is generated the XML can be combined into the single Wiki data box from Figure 8.

Another useful approach to introducing more semantic structure during wiki operation is to use light constraints (Di Iorio & Zacchiroli, 2006). At first sight the connecting of "constraint" with "wiki" appears to violate the flexibility and freedom of "the Wiki Way". The authors define light constraints as rules that can be temporarily violated without inhibiting the proper wiki run-time behaviour. Example light constraints can be as simple as spell checkers and the insistence on wikiwords for new I-grains that DotWikIEWeb already implements. However the authors of the light constraint concept use a data model-driven validation mechanism that can be used for all wikis that can insist on the presence of certain meta data before a wiki page (or section) can be marked as validated. Non-validated content is still accepted into the wiki but is marked as such. This gentle persuasion approach is likely to provoke less resistance from users.

Conclusions and Future Work

It has been shown with the DotWikIEWeb implementation that it is possible to build a useful web-based wiki page colony seeded from a single web page. The user can create any number of additional wiki pages in the colony and simple mechanisms are available to access any wiki page and manage the colony collection. While limited to working with Internet Explorer DotWikIEWeb has an extremely simple interface for editing wiki content, much simpler that typical plain text wikis. This makes DotWikIEWeb ideal for copying and pasting web page fragments complete with all fonts, text and background colours, links and media such as image, audio and video. Such fragments retain fidelity within the I-grains and become part of the wiki.

The ultra lightweight web application technique described here for implementing wiki functionality is extensible to many other types of application. It can be used for word processors, spreadsheets, slide shows, simple databases and personal information management tasks. Data for these types of application can be represented in XML format and embedded in the ultra lightweight web application page just as for DotWikIEWeb. Obviously the size of the data introduces natural limits to page size but is likely to work successfully with sizes of up to 0.5 MB on broadband links.

There is still scope for several further improvements to DotWikIEWeb. Using formats such as SVG should allow images to appear within the single page contents and be rendered by browsers or at least with appropriate plugins. While each DotWikIEWeb page may have many I-grains each of these tends to have a small amount of content so that finding information is not usually a problem. However, a search facility both within a page and across the wiki page colony is an obvious extension with a high utility.

A lofty ambition for DotWikIEWeb is to become browser independent. This would require a Javascript-based XHTML editor embedded within a single page that supports all main browsers. There are many open source Javascript XHTML editors that can be loaded as a single, compressed file, but when uncompressed spread over several files, usually including images, in order to function. Perhaps a single page version of such an editor might become available one day. In any case the need for such an editor is already pressing as the built-in HTML editor in IE7 is not available on Windows Vista, so that for DotWikIEWeb to survive a new editor on Vista is required.

References

Di Iorio, A. & Zacchiroli, S. (2006). Constrained Wiki: An Oxymoron?, Proceedings of WikiSym'06, Odense, Denmark, 2006, pp 89-98.

Rees, M. J. (2006). Ultra Lightweight Web Applications: A Single-Page Wiki employing a Partial Ajax Solution, Proceedings of the Twelfth Australasian World Wide Web Conference, Noosaville, 2006.

Volkel, M, & Oren, E. (2006). Towards a Wiki Interchange Format (WIF), Proceedings of the First Workshop on Semantic Wikis: From Wiki to Semantics, Budva, Montenegro, 2006, pp 230-244.

Hypertext References

HREF1 Dr Michael J Rees
http://www.bond.edu.au/it/staff/michael.htm
HREF2 School of IT, Bond University
http://www.sit.bond.edu.au/
HREF3 Bond University
http://www.bond.edu.au/
HREF4 Single page application definition
http://en.wikipedia.org/wiki/Single_page_application
HREF5 SeekDotNet Hosting Service
http://www.seekdotnet.com
HREF6 TiddlyWiki
http://www.tiddlywiki.com
HREF7 Wikipedia Entry for SHA1
http://en.wikipedia.org/wiki/SHA1
HREF8 Paul Johnston SHA1 in Javascript
http://pajhome.org.uk/crypt/md5
HREF9 BidiX TiddlyWiki Extensions
http://tiddlywiki.bidix.info/
HREF10 TiddlySpot
http://www.tiddlyspot.com/

Copyright