

User Interfaces for Search Information Management Web Sites

Dr Michael Rees [[HREF1](#)], Faculty of Information Technology [[HREF2](#)] , Bond University [[HREF3](#)], Old 4229, Australia. mrees@bond.edu.au

Abstract

Now and into the future software packages are increasingly delivered and utilized via web browsers. This type of software, known generically as a web application, is accessed and executed by following a standard hyperlink within the browser, and guarantees the user makes use of the latest version.

Such access to web applications is simple, but the constraints of user interaction in the browser execution environment can be relatively severe. Some modern browsers now incorporate enhanced interaction such as rich text (HTML) editing. Harnessing these built-in tools allows the web application developer to approach the sophistication of the high-response interaction expected of conventional, free-standing applications.

This paper describes the design and implementation of a class of editor web applications that allows the user to construct whole web sites. While the top-level web site structure is fixed by templates, the contents of each page can be varied in an extremely flexible way to allow a wide range of page content. Designing and authoring web pages in this fashion raises specific user interface design issues which are discussed further.

DotTegular is the web application built to allow shared editing of complete web sites. It has proved very capable of supporting a new class of software now being called search information management, the collecting, storing and restructuring of fragments of useful information, fraglets. The exact nature of the repository and data representation of *fraglets* is presented. Other possible uses and applications of DotTegular are discussed at the conclusion of this paper.

Keywords

In-page editing, universal canvas, user interface design, web page representation, search information management.

Introduction

This The topic of this paper is inspired by making web pages act like a universal canvas. According to Udell [[HREF4](#)] it was Microsoft that coined the term 'universal canvas' in this context when introducing their .NET initiative. Quoting Udell [[HREF4](#)], 'The term connotes a single viewing and editing surface on which heterogeneous data types fluidly interact.'

For this discussion, the universal canvas is taken to be a web page displayed in a web browser. Other terms for this specific type of universal canvas are '2-way web', 'interactive web' and the 'blank web'. The intention is to allow web pages that may be blank or contain standard HTML content to be edited directly by the end user. Following such in-page editing, subsequent accesses to the page should reflect the new contents.

Publicly accessible web pages can be viewed by any user so that a page acting as a universal canvas can become a collaborative sheet of information. In essence, a universal canvas web page is a groupware tool. However, such editable pages can also be used for a very simple and powerful web publishing mechanism that does away with the need to purchase and become familiar with other page publishing packages. This latter usage forms the focus for the web application described in this paper.

Cross-referencing web information produced by others is, of course, one of the major reasons that the World Wide Web has become so popular. Hyperlinks allow one page to reference related information present in other pages. However, even though it is possible to link to a particular position within another page, the reference is always at the page level. Often, only a small part or fragment of a web page needs be cross-referenced. In addition, when collecting useful information, it is beneficial to quote or reproduce these web page fragments.

A secondary focus of the web page editing tool is to allow a single page to be built from identifiable fragments referred to for brevity as fraglets. Furthermore, if these fraglets are stored in a repository accessible to all authors using the tool, then reuse of fraglets becomes possible. The whole web page can be represented by a list of fraglet identifiers.

Sections in this paper begin by reviewing other in-page editing tools that are currently available. These editing tools are compared with similar efforts by the author with a view to eliciting a set of user interface requirements. The author's early tool for building pages from fraglets, DotNotelets, is described together with the new style of in-page user interface that makes fraglets visible and allows them to be manipulated by the user. Next, the new tool, DotTegular, for building complete web sites from fraglets is described and its use for search information management is discussed. Sections at the end of the paper address the user interface needs of the tool and its implementation.

In-page Editing Tools

By the mid 1990s it became clear that web browsers had settled into a pattern of being display-only applications, capable of viewing web pages but not updating them. Forms and CGI scripts, of course, offered a limited two-way interaction for relatively small amounts of data. Web application developers interested in a two-way web page publishing mechanism have been active since that time in an attempt to provide a clean user interface for page editing in standard browsers. The author has described tools provided by a number of other groups and himself in Rees (2000a, 2000b, 2002).

For this paper, the new tool DotTegular is worthy of comparison to the spectrum of Wiki tools first created by Ward Cunningham, see Cunningham (1999). Once a user has access to a Wiki web page it is possible to display the page in the normal way (the default), or, by clicking on an ?edit? link, to modify any and all the content of that page.

A sample Wiki page is displayed in Figure 1. The header of the page is automatically generated and consists of the Wiki logo and the unique title of that page consisting of at least two words with initial capitals run together known as a WikiWord. A space is added between each word to form a readable title. The hyperlink in the header leads to a page which lists all other pages in the Wiki that refer to this one.



Michael Rees

I am a computing science professor at Bond University on the Gold Coast, Australia.

Research

- web applications development
- collaborative web-based tools
- in-page editing tools in particular
- topic maps

Some of my tools are available at <http://comet.it.bond.edu.au/dot>.

Find out more about me at <http://michaeljrees.com>.

CategoryHomePage

[EditText](#) of this page (last edited [September 11, 2002](#))

[FindPage](#) by searching (or browse [LikePages](#) or take a [VisualTour](#))

Figure 1. A Sample Wiki Page.

The shared editing, search and similarity features are included in the automatically generated footer of the page. The ?FindPage? link allows searching by page title and page contents within the Wiki. Pages with similar titles are located with the ?LikePages? link, and the ?VisualTour? link shows pages linked from this one.

Wikis use a very simple but powerful mechanism for page creation. Entering a WikiWord into a page will generate a link to the page with that title. If the title does not exist a hyperlinked ??? is generated instead, and an empty page with title is created. Clicking on the ??? takes the user to the new page and the opportunity to edit the content.

More pertinent for this paper is the ?EditText? link of Figure 1 which allows the page contents to be edited. When this link is followed a new page is displayed with the page content appearing in an editable form as shown in Figure 2. Note that two round trips to the Wiki server is needed to generate this. One round trip is needed to see the confirmation page that also notes suspicious word spellings, and one to display the changed page.

The user is explicitly barred from using HTML. This preserves the integrity of the web page and allows the contents to be edited by users who don't know HTML.

MichaelRees

```

I am a computing science professor at Bond
University on the Gold Coast, Australia.
----
''Research''
    * web applications development
    * collaborative web-based tools
    * in-page editing tools in particular
    * topic maps

Some of my tools are available at
http://comet.it.bond.edu.au/dot.

    Find out more about me at http://michaeljrees.com.
----
CategoryHomePage

```

I can't type tabs. Please [ConvertSpacesToTabs](#) for me when I save.

I'm just doing minor edits. Please divert the usual logging to [RecentEdits](#) instead.

[GoodStyle](#) tips for editing.
[EditPage](#) using a smaller text area.
[EditCopy](#) from previous author.

Figure 2. Editing a Wiki Page.

A number of conventions are applied to the raw text to give control over simple HTML formatting. This approach has come to be referred to as structured text. Some examples from Figure 2 give a flavour for structured text formatting:

- Non-indented text becomes normal HTML paragraphs
- '----' maps to a horizontal line
- Repeated single quotes lead to emphasis, italics in this example
- A tab (or 8 spaces) followed by an asterisk maps to an unnumbered list
- Indented paragraphs are indented and set in monospaced font
- Words beginning ?http? are turned into hyperlinks

There is also support for numbered and definition lists, other forms of emphasis, images and ISBN links to monographs. Note also that some special WikiWords show links to starting points in the Wiki. ? CategoryHomePage? for example links to person pages.

Such simple-to-learn conventions lead to tidy web pages with quite powerful support for linking Wiki pages together in a natural way. It is not surprising that the use of Wikis is spreading rapidly. They are

being used to create web page collections for projects, discussions, document production and the like. A good example of Wiki software is the freely available ZWiki product [HREF5] for the open source Zope web application server [HREF6]. With a one-minute setup time and some adjustments to user access permissions a full-featured Wiki can be up and running.

From a user interface perspective, the only downside of Wiki web page editing is the dual-mode nature of the editing interaction. The user only sees the results of the text modification when the form contents are submitted. Even if response is good, the constant switching of mode is annoying although tolerable.

An in-page editing tool implemented by the author, DotEdit1, also uses a textarea tag in an HTML form to allow the user to edit nominated fragments of a web page. The design and operation of this tool is described in Rees (2002b) and is available for download from On-The-Dot Software [HREF7]. The mode switching in the user interface is still present but is substantially alleviated by manipulating the in-memory HTML Document Object Model under client script control. This allows immediate mode switching without the need for a round trip to the web server to generate a new web page. In addition, DotEdit1 saves the changes to the web page in the background, again without the need to refresh the web page by a round trip to the server.

Another type of editable web-based application gaining in popularity is the web log. Here again users must be able to update the main contents of the web log pages, although, like the Wiki, many of the useful links are generated automatically. There are many web log applications and servers.

The Radio Userland [HREF8] web log by Dave Winer is chosen for discussion because it has partially solved the dual-mode editing problem. When the web log owner comes to alter the page content using Internet Explorer a rich text edit box is presented as shown in Figure 3.

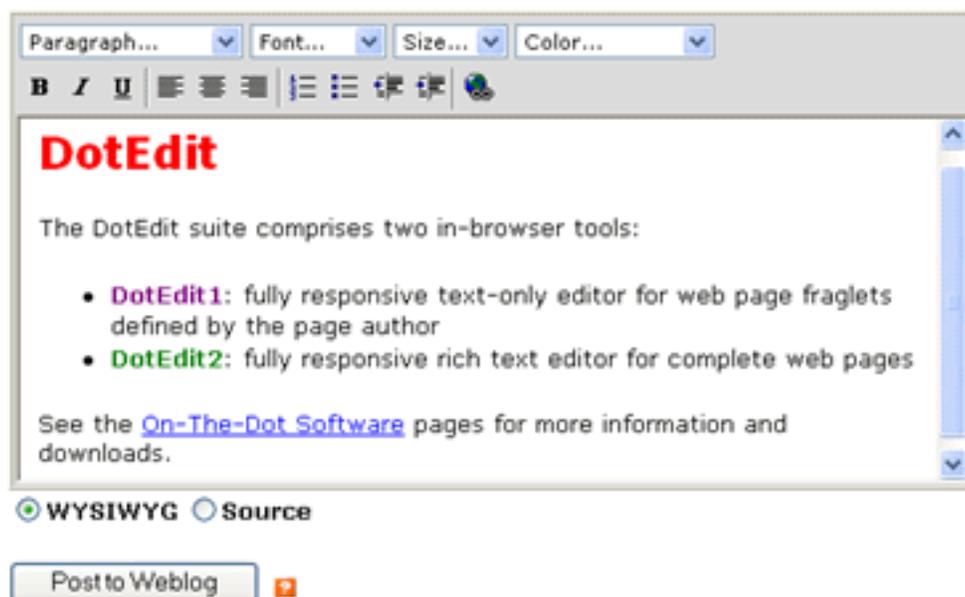


Figure 3. Radio Userland Web Log Editing.

Like the author's tools described in later sections, the Radio Userland web log application makes use of the built-in rich text editing control present in Internet Explorer.

The user interface of Figure 3 presents a fairly conventional set of toolbars for rich text editing as might be found in a word processor, but this is just an area of a web page. The user can edit text and then format it for paragraph style, font face and size, and foreground and background colour. Font emphasis by way of bold, italics and underlining, paragraph alignment, indentation and lists are also provided. In addition, hyperlinks can be created and edited. For those users familiar with HTML, it is possible to click on the ?Source? button to view and edit the raw HTML for the entry. The Radio Userland development team has chosen to stop there, but there are many other sophisticated features available with this editing control.

The main point is that the user sees the exact formatting as editing progresses without the need to switch modes. At the same time a user is faced with a familiar editing user interface and need not know any HTML.

When posted to the web log, the exact same text format becomes another entry in the web log page as shown in Figure 4. Of course, this application does a great deal more than just edit web log entries. It distributes the entries to a central web log server and allows other bloggers to subscribe and take feeds from this shared material.

Michael's Intercom : Michael Rees Radio Blog

USERLAND

Radio 8.0.8: [Home](#) | [News](#) | [Stories](#) | [Shortcuts](#) | [Folder](#) | [Events](#) | [Themes](#) | [Tools](#) | [Prefs](#) | [Help](#)

Paragraph... Font... Size... Color...

B *I* U [List] [Align] [Indent] [Outdent] [Undo] [Redo]

WYSIWYG Source

Post to Weblog

September 2002

Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					

Aug Oct

Cloud Links:

[Home](#) is the public page for your weblog. Send this link to your friends and colleagues.

[Referers](#) shows you which sites are pointing to yours and how many hits they're delivering.

[Ranking by Page Reads](#) lists the 100 most popular Radio-managed weblogs, today and all-time.

[Updates](#) shows you which weblogs in your community have updated recently.

Status Center:

12/09/2002; 3:42:59 PM

[Cloud status](#): 100% of 40.0MB is free.

[News aggregator](#): Last scan complete at 3:30 PM on 12/09/2002; 0 new stories. You are [subscribed](#) to 12 news sources.

[Hotlist](#): The list of Most-Subscribed-To

Previous 10 posts...

Post Title	Time	Action
<input type="checkbox"/> DotEdit	3:42:57 PM	EDIT

The DotEdit suite comprises two in-browser tools:

- DotEdit1**: fully responsive text-only editor for web page fraglets defined by the page author
- DotEdit2**: fully responsive rich text editor for complete web pages

See the [On-The-Dot Software](#) pages for more information and downloads.

Figure 4. Rich Text Entry Showing in the Web Log.

Figure 4 shows how the newly edited entry is displayed together with the standard links and calendar in the page header and down the right side. Other users viewing this page will, of course, see all the content except the rich text edit box.

With the experience of using his own tools and those just mentioned, the author has developed a new version of the in-page editing tool now known as DotEdit2. This tool is distributed via the On-The-Dot Software site mentioned previously in a footnote. DotEdit2 employs the same rich text editor component seen in Radio Userland. Figure 5 gives a flavour of the highly responsive user interface.

When the page is first displayed in the browser, a small reminder banner is placed at the top of the page to indicate to the user that the page contents can be edited as required. Figure 5 shows the case when the user has right-clicked on the paragraph after the ?Whole Page Editing? header. This caused the editing toolbar (with grey background) to be inserted in the page below that paragraph. At the same time, all page contents, not just the right-clicked paragraph, become editable. The editing tools operate upon whatever part of the page is selected by the user. A larger range of available editing features is exposed compared to the Radio Userland web log application described above.

A page editable with d•tEdit version 2.2.2 © 2001-2002 M J Rees

DotEdit 2 Editing via the Browser

Whole Page Editing

With the addition of a single <script> line, the whole HTML page is available for in-page editing. The user simply right clicks near the text position to be edited. This instantly switches the whole page contents into edit mode and inserts an editing toolbar near the site of editing.



The editing toolbar is not included in the page when the new contents are saved back to the server.

No round trips back to the web server are involved. Editing happens in-place with immediate response. Saving the contents to the server happens in the background.

DotEdit 2 Setup

A very simple setup is required. For each page that needs to use the shared, in-place editing, the following steps are needed:

1. a single <script> tag is added
2. the page permissions are changed by an administrator to allow overwriting

A safety interlock system prevents inconsistent simultaneous editing occurring.

Michael Rees 1999 - 2002

Figure 5. DotEdit2 Whole Page Editor.

The changed contents of the page can be saved back to the server by clicking on the conventional ?

Save? icon. In the background, without any round trips being needed, the page contents are written back to the server. Other users viewing the page will then see the new contents. If the user right-clicks on any part of the page the editing toolbar is removed, and the contents of the page marked as non-editable again. This transition between editing and page display is essentially instant, and requires no round-tripping to the server.

From a user interface perspective, DotEdit2 achieves a very responsive whole-of-page editing capability. Apart from the addition of the compact toolbar, the user sees exactly the format of the final page as editing proceeds. Switching between editing and display requires a simple right-click. The normal functioning of the right click within the browser is achieved by holding down the Ctrl key while right-clicking.

The browser-based editing tools described in this section show a gradual improvement in the nature and responsiveness of the user interface. The Wiki pages use conventional HTML forms, CGI scripts and multiple round trips to the server. As a consequence the user is faced with multiple-mode operation with potential lack of responsiveness on slow network links. (In fairness, the Wiki software is lightweight and server response is extremely fast, only the network delays lead to an increase in response time.) DotEdit1 limits the user to editing raw text in fragments of the page, but otherwise eliminates round trips for displaying forms, and saving the page contents occurs in the background.

Radio Userland exploits the built-in rich text editor component to allow users to see the actual page display format as they edit fragments of a page, but still suffers from round-trip delays when saving the updated contents. DotEdit2 allows whole-page editing like Wiki, but eliminates many round trips, and the users see the exact page format as they edit the contents. This DotEdit2 tool comes very close to providing a universal canvas for single web pages containing text, images, simple multimedia and hyperlinks. In the next section user interfaces for constructing complete web sites are considered.

Editing Whole Web Sites

Exploiting the rich text editing component still further, the author experimented with a new style of web user interface. This has led to a simple tool called DotNotelets for organizing ideas in a concept map. Each idea or information fragment (fraglet) is represented in a rectangular area within the page called a notelet. The operation and use of DotNotelets is described in Rees (2002a). A small example concept map is shown in Figure 6.

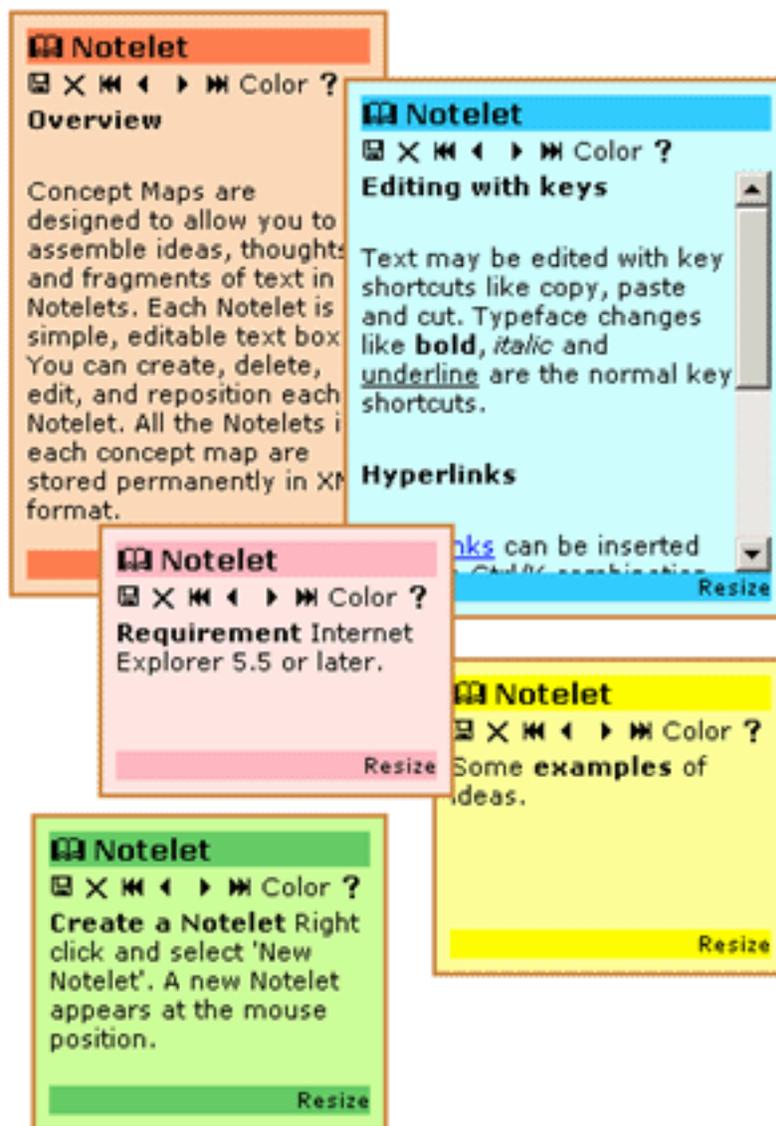


Figure 6. Notelets Forming a Concept Map.

A very simple editing toolbar is included in each notelet which acts like a window within the web page. A number of standard key combinations are available, such as Ctrl-B for controlling font emphasis, and Ctrl-K for editing hyperlinks. Notelet windows can be resized and moved within the page, possibly overlapping other notelets. The front-to-back order is controlled with the arrow icons in the editing toolbar.

The order of arrangement in two dimensions is significant. Right-clicking on the page gives access to a menu for creating new notelets, and for serializing the notelet concepts into a single web page as well as other options. After experimental use the serial display order that appears to be most intuitive is top-to-bottom, then left-to-right. Thus, the content of the web page is built from the concatenation of notelets.

Ideas and experience from the DotNotelets user interface have been carried forward into the main web site editing tool described here called DotTegular. The name is derived from the word for tiled because the web pages are built from tile-like fraglets or teglets as they are called in DotTegular. The focus of DotTegular is to allow users to build a series of complete web sites. Each web site is composed of pages created from collections of teglets.

Note that in this discussion of DotTegular the terms fraglet and teglets are interchangeable. An example teglet is shown in Figure 7. The similarity with a notelet from the DoteNotelets tool will be readily apparent.

A number of extensions have been introduced to the editing user interface:

- The editing toolbar is extended to include most features of the rich text editor, and is displayed in a short and long form.
- The actual HTML of the teglet can be edited as raw text.
- A title for each teglet is compulsory, and can become a heading in the final web page.

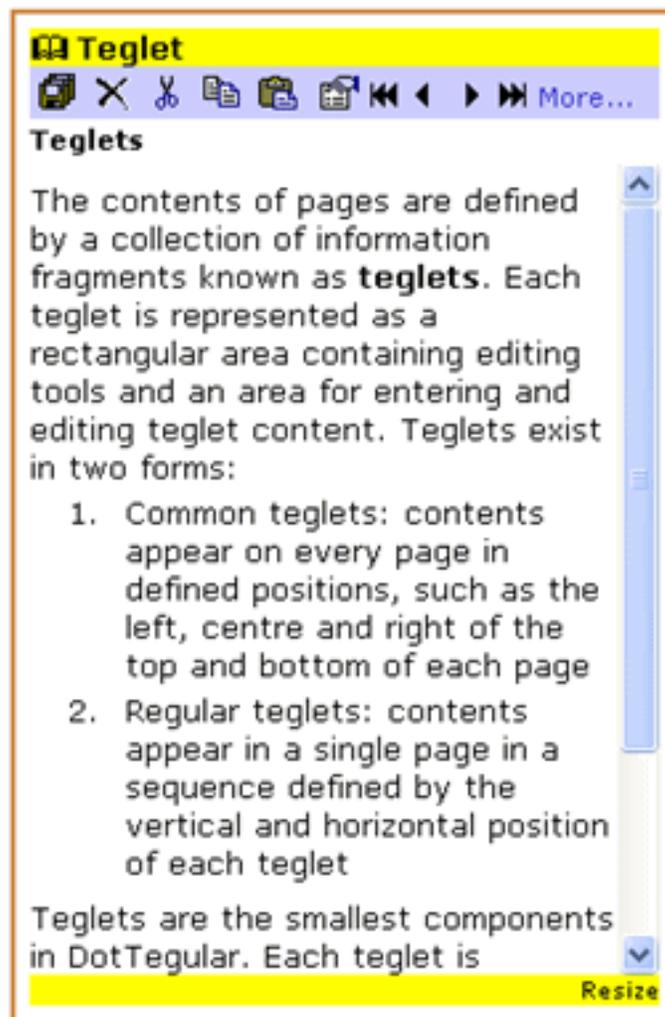


Figure 7. An Example Teglet.

When users require access to more sophisticated editing they click on the 'More...' link to expose more editing tools as shown in Figure 8. At the end of the extended toolbar appears the 'HTML' link which displays the raw HTML text in a separate editable window. If sufficiently knowledgeable, the user can then edit the HTML text. Obviously this is a security risk, and will be made optional when copies of the tool are released.

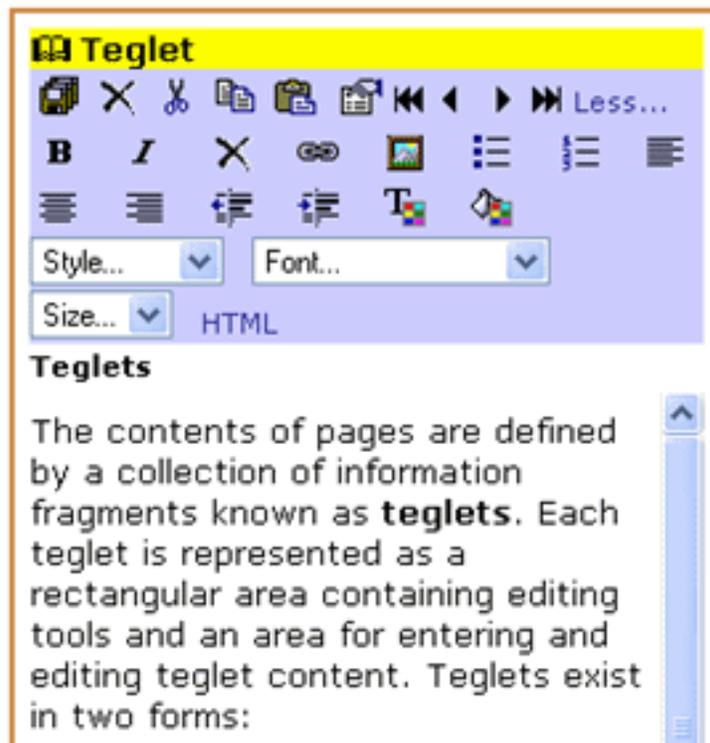


Figure 8. Teglet showing the Extended Toolbar.

In the first version of DotTegular there are three different types of teglets:

1. Regular teglets: the contents are serialized with other regular teglets and become the main contents of a single web page.
2. Common teglets: there are six specific common teglets that can define the content of the top and bottom of each page in the same web site. The top and bottom contents are split into left, centre and right, hence the six subtypes of common teglet.
3. Basket teglets: these form a special scratchpad collection of teglets that can be accessed when editing any web page in any web site. They typically contain frequently needed web page content and can act as templates for HTML layout such as tables.

Like DotNotelets the teglets in DotTegular can be created via the right-click menu options as shown in Figure 9. The user chooses between the three teglet types, and the chosen teglet type is created near the position of the cursor.

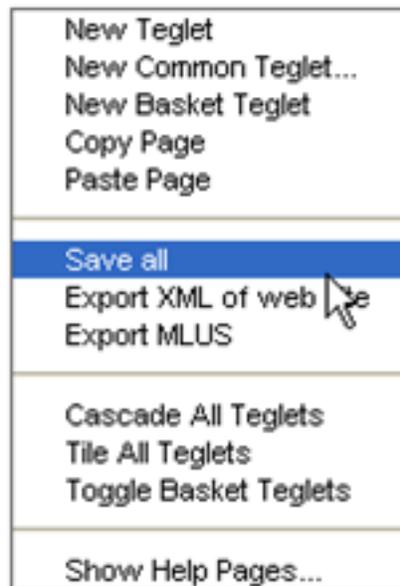


Figure 9. DotTegular Menu Options.

When users first enter DotTegular they are presented with a list of existing web sites available for editing. At this stage they may manage whole web sites and have the capability of creating and deleting whole sites as shown in Figure 10. The user interface design at this level has not received a great deal of effort, but simply serves to make visible the list of web sites and some operations upon them.

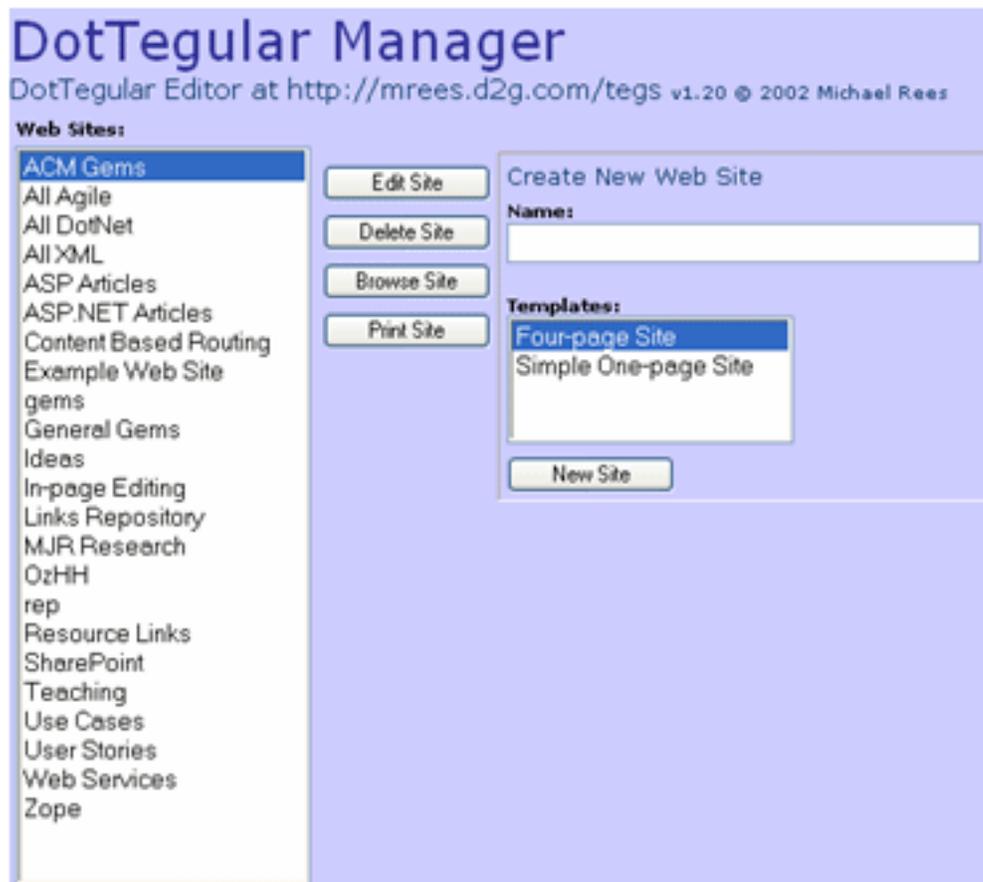


Figure 10. DotTegular Web Site Management.

Double-clicking on the chosen web site moves the user interface to the web pages within the site as depicted in Figure 11. Here again, a simple collection of lists, buttons and single buttons form the page-level management. The main focus of the interface design has concentrated on the teglets that control the actual page contents, and how these can conveniently be manipulated.

Again borrowing from DotNotelets, the teglet collection is arranged in two dimensions with a front-to-back ordering. A three-color scheme is used in DotTegular to represent the different types of teglets.

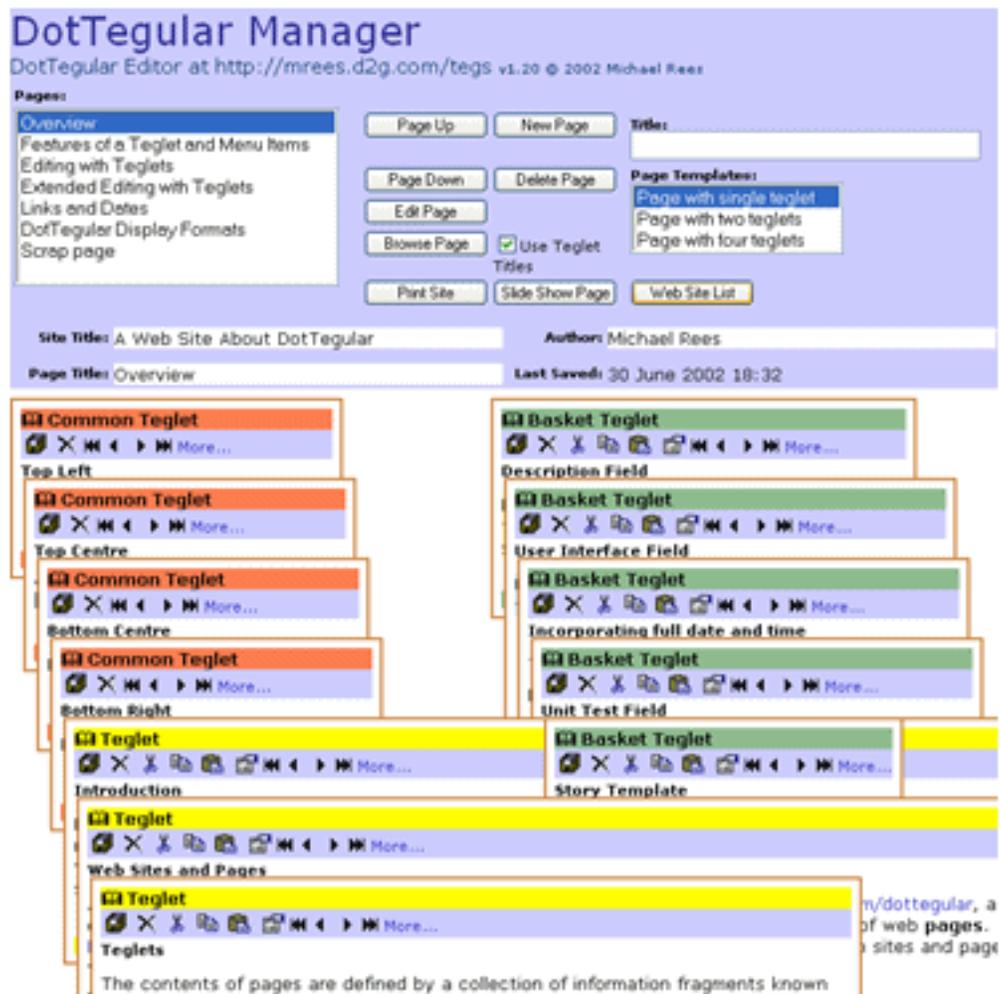


Figure 11. Page Content Management.

In Figure 11 the red common teglets representing the top left, top centre, bottom centre and bottom right, respectively, can be seen in the centre left. The green basket teglets appear in the centre right. These can be toggled on and off using the '?Toggle Basket Teglets?' menu option shown in Figure 9.

The yellow regular teglets of Figure 11 appear at the bottom of the figure. Of course, during the course of editing the page the teglets usually appear in a much more haphazard arrangement with the user constantly resizing, moving and changing the front-to-back order of the teglets. The apparent tidiness of Figure 11 is achieved by selecting the '?Cascade All Teglets?' option from the menu. This tidying process retains the strict ordering defined by the user placing the teglets in the chosen top-to-bottom and left-to-right positions, but changes the front-to-back order for neatness. Using the arrow icons in

the toolbar the user may subsequently readjust the relative front-to-back positions. There is a second ?tidy? option called ?Tile All Teglets?. As its name implies this arranges all teglets vertically on the page with no overlapping. With a large number of teglets on a page this obviously makes for a long page necessitating a great deal of scrolling. On the other hand all the teglet contents are visible without the need to reorder the front-to-back order.

The standard set of page management features shown at the top of Figure 11 allow pages to be created from a series of templates, titled, deleted and moved up and down in the page list. In its first version, DotTegular uses a very simple web page structure for each web site—a linear sequence of pages with the ability to navigate to each page directly. Future extensions envisage a number of different page layout templates that link the pages in number of different ways. A sample page when browsed is shown in Figure 12.



A Web Site About DotTegular

A Sample Teglet Showing Some Edit Features

H3 Heading Style

Normal paragraph text with *emphasised* words. Any words can be [hyperlinked](#) using absolute URLs.

This paragraph is contains coloured **text** and **background** in any **combination**. Font changes can be made as usual.

- bullet point 1
- bullet point 2

There are numbered lists:

1. First point
2. Second point
3. Third point

Some centred text (right adjusted also available).

Text can be set in a **large size**.

Use `formatted text to keep spaces between words`

and `newlines`.

Editable Teglet Title

Normal paragraph of **editable** text.

Untitled

Normal paragraph.

Last saved 30 June 2002 Michael Rees

Figure 12. Web Page Structure in DotTegular.

In Figure 12 the top and bottom sections of the page are generated from the contents of the common teglets. The ?Top Left? common teglet in this example shows an image. The ?Bottom Centre? common teglet shows the current date generated by the special HTML:

```
<span class=dtDate></span>
```

When rendering the page in HTML, DotTegular inserts the date on which the teglet was last saved. There are a number of other classes for long and short forms of the date and time, and the page and web titles. The main top title is that chosen by the author when creating the web site. The hyperlinks on the left of Figure 12 are generated from the page titles defined by the author, and allow the browser to go directly to any of the pages. Within the body of the page the second level headings are the titles of the individual teglets?this feature can be toggled on and off. All other content is derived from the content of each teglet.

Regular teglets can be moved and copied between pages, and copies of any of the global basket teglets can be inserted into any number of pages. Within a particular collection of web sites, the basket teglets bring a degree of conformity.

Representing Web Sites and Pages

In this first version, DotTegular web sites are not stored on the web server as HTML web pages but as XML files. The content of each teglet, taken directly from the rich text editor component, is wrapped in simple XML as shown in Figure 13. Note that the XML records positional and front-to-back order information as well as the teglet title and date and time of last modification.

Of greater importance to the long term utility of building web sites with DotTegular is the ?ID? attribute within the teglet XML. Generating a GUID for each teglet endows it with a unique global identifier, and, provided other users have network access to it, allows each teglet to be shared by the global community.

```
<notelet ID="{56C7BD15-412D-4014-A455-A9F9D67D252A}"
Type="content" Title="Web Sites and Pages"
LastSaved="200209131233" Top="610" Left="60" Width="770"
Height="210" Zindex="6">
- <![CDATA[
    <P>At a particular web address where DotTegular is
    installed, such as <A href="http://mrees.d2g.com/dc
    <P>Web sites are shown and managed via a simple lis
    <P>Double-clicking on one of the web sites in the l
    <P>&nbsp;</P>
]]>
</notelet>
```

Figure 13. Teglet XML Example.

Each DotTegular web page is composed of a sequence of teglets. In turn each web site consists of a sequence of pages. The XML for a complete web site in outline is shown in Figure 14.

```

<dotWebSite Title="A Web Site About DotTegular" Author="Michael Rees"
  LastSaved="200209131233">
+ <Common>
- <Pages>
  - <Page Title="Overview" LastSaved="200209131233">
    + <notelet ID="{1836D1D9-87CB-4367-92F8-E67F3B21C8B0}"
      Type="content" Title="Introduction" LastSaved="200209131233"
      Top="550" Left="50" Width="750" Height="100" Zindex="5">
    + <notelet ID="{56C7BD15-412D-4014-A455-A9F9D67D252A}"
      Type="content" Title="Web Sites and Pages"
      LastSaved="200209131233" Top="610" Left="60" Width="770"
      Height="210" Zindex="6">
    + <notelet ID="{E6E0987B-42E0-4927-A097-7722B9CFAFB8}"
      Type="content" Title="Teglets" LastSaved="200209131233"
      Top="670" Left="70" Width="580" Height="240" Zindex="7">
  </Page>
  + <Page Title="Features of a Teglet and Menu Items"
    LastSaved="200206211821">
  + <Page Title="Editing with Teglets" LastSaved="200206241131">
  + <Page Title="Extended Editing with Teglets"
    LastSaved="200206211707">
  + <Page Title="Links and Dates" LastSaved="200206241246">
  + <Page Title="DotTegular Display Formats" LastSaved="200206211732">
  + <Page Title="Scrap page" LastSaved="200206211212">
  </Pages>
</dotWebSite>

```

Figure 14. DotTegular Web Site XML.

For ease of implementation, the current version of DotTegular stores the web site XML in a single document on the web server. To browse the web site a simple ASP page is used to render the XML as HTML.

The URL for a DotTegular web site is of the form:

`http://DotTegular/Browse.asp?site=SiteName`

The page title may also be appended to this URL in the form:

`&page=PageTitle`

Hyperlinks in this form are used in the navigation bar to the left of each page as shown in Figure 12.

Once information is captured as an XML representation it becomes possible to render and restructure the information content in any number of ways. This powerful feature is exploited in DotTegular to provide two other useful formats for each web site:

1. Slide show format: each page is rendered as a single `?slide?` using a style sheet employing

suitable enlarged font sizes, and framed with simple navigation controls for moving to each slide or page. See Figure 15 for an example slide.

2. Print format: every page in the site is appended to a single HTML page with a table of contents generated at the head. This form is suitable for printing the contents of a complete web site from within the browser.

There is obviously some scope for adopting other alternative formats for specific applications.

Resource Links

ASP.NET Website Programming Problem - Design - Solution



Author: Marco Bellinaso and Kevin Hoffman

Publisher: Wrox Press

Copyright: 2002

Format: Paper, 540 pp

ISBN: 1861006934

Status: Published March 2002

Retail Price: \$59.99 US

Michael Rees Last saved 13 September 2002

Slide 2

2: Books



Figure 15. DotTegular Slide Show Format.

Adopting such a simple, but generic, format for browsing DotTegular web sites has meant that the tool can be adopted for a wide variety of tasks. In experimental use over a number of months, the author has found DotTegular useful for:

- Simple information sites
- Help pages
- Personal web logs
- Shared document preparation
- Idea generation and brainstorming
- Collections of fragments from other web pages, fraglets

Once released to a user community, it is expected a much wider range of web site uses will emerge. Currently, the user interface has remained substantially unchanged over several months of ? production? use by the author. It is sufficiently stable that the author uses DotTegular for his personal web log.

When considering the range of tools available for constructing user story cards for the Agile Programming paradigm, the author found DotTegular to be the equal of other tools. DotTegular was extended with a few features of use for this particular task set and became DotStories. The features of this tool in comparison with others have been published in the proceedings of Asia-Pacific Software Engineering Conference 2002 (Rees, 2002b).

Collecting Information Fragments

Discussion now returns to one of the most compelling uses found so far for the DotTegular tool; the task of collecting and organizing fragments from existing web pages. In terms of this paper the task involves the creation, storage and structuring of fraglets. Increasingly the name given to this type of application is search information management.

Some work has been undertaken in the past in this area. At one end of the scale, all browsers contain bookmark lists to allow users to store links to whole pages of information. For small numbers of bookmarks the in-browser lists work well, but drop of in usefulness when bookmark collections number in the hundreds (Abrams et al, 1998). Nevertheless bookmarks lists find wide use.

At the other end of the scale, collecting copies of whole pages is the seminal work by Card, Robertson and York (1996). These researchers built an application to allow the collection and management of large collections of web pages. The management metaphors were the book and the bookshelf with an intermediate space around the current book being read or assembled where other recently used books are displayed. Very sophisticated user interfaces allowed the manipulation of the intermediate space and flipping through individual books. One of the downsides noted by the authors is that the collecting of information via the Web Forager is performed at the whole web page level, when often the user needs only a portion of most web pages.

The ability to collect fragments of other web pages is at the heart of the Hunter Gatherer tool developed by schraefel, Zhu, Modjeska, Wigdor & Zhao (2002a). In a related reference (schraefel et al, 2002b) these same authors undertook a task analysis and discovered that fragment collecting was not common because of poor ease of use with traditional word processing tools. Thus the Hunter Gatherer tool was built.

Hunter Gatherer is built upon the Mozilla open source browser and allows users to select fragments of any web page displayed by the browser and to concatenate the fragments in lists. The lists each become a consolidated web page, and the user is able to reorder the fragments in the page. Studies show that on the whole, the Hunter Gatherer tool performed better than using a word processor, Microsoft Word, for doing the same task.

In order to keep track of the exact location of each web page fragment Hunter Gatherer constructs an Aggregated URL (AURL) for each fragment. An AURL consists of the web page URL together with position information derived from the document object model created by translating the web page into XHTML. AURL decay can still happen as the content of the original page changes, but the problem was not a major factor for test users.

While obviously effective Hunter Gatherer is a free-standing application that must be downloaded and installed in a separate operation. Users must switch from their standard browser in order to use it; two disadvantages that a web application eliminates.

It is interesting to observe that Microsoft is taking a similar approach with its new application OneNote [[HREF9](#)] that is now part of Microsoft Office 2003. A condensed screenshot of OneNote is shown in Figure 16. The main note collection surface employs a two-dimensional note pane akin to DotTegular but does not allow any overlap to take place. Moving a note adjacent to another note causes the two notes to be merged. A powerful feature of note capture is the appending of the URL automatically at the foot of the note.

In terms of note structure, OneNote employs a folder organization with pages and subpages within each folder. The pages in Figure 16 can be seen as vertical tabs at the right. Folders appear as horizontal tabs across the top. Clicking on a folder tab reveals a subfolder hierarchy if one is present.

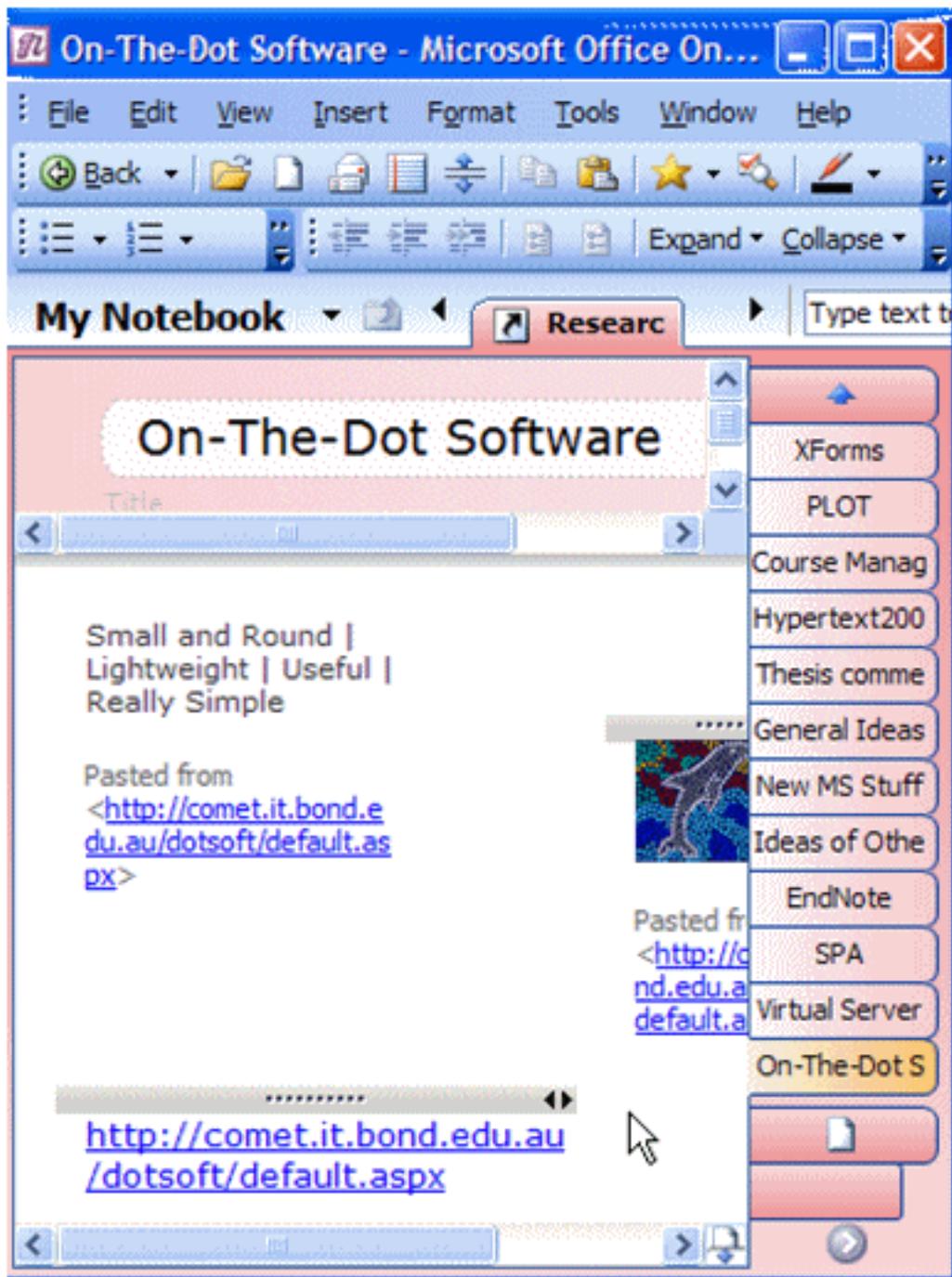


Figure 16. Microsoft OneNote Note-taking Application.

Notes can be dragged and dropped around the page for reordering of the information. Pages and subpages can be reordered as needed. Of course many beneficial features of Microsoft Word are present for formatting and spell-checking and the like. Also notes can be copied and pasted into all other Office applications.

Disappointingly OneNote employs a proprietary format for the note documents but does allow single pages only to be exported as MHTML (multiple web pages) format that can be displayed in Internet Explorer. In addition, being a separate application, OneNote suffers from the same problems as Hunter Gatherer in that note capture involves switching applications and needs to be installed as a free-standing application.

Another recent product for search information management is an Internet Explorer extension called Onfolio [HREF10]. In the words of the software developers Onfolio is "an application for collection, organizing and sharing what you find online." On installation (note this is still not a true web application) it adds a toolbar and management pane to Internet Explorer as shown in the screenshot of Figure 17.

Now we have the improved situation of working within the browser itself. Whole pages, selected parts of pages, links, images and other information fragments can be dragged and dropped in Onfolio collections. These collections share many features with OneNote folders in terms of organizing the information fragments.

Onfolio shows a simple list of fragments with a modicum of their content rather than the full content and layout that OneNote achieves. However, the professional version of Onfolio does include a report generation mechanism that serializes the fragments into complete pages and exports them in the same MHTML format used by OneNote. It must be added that the user interface design and convenience of use of Onfolio is impressive.

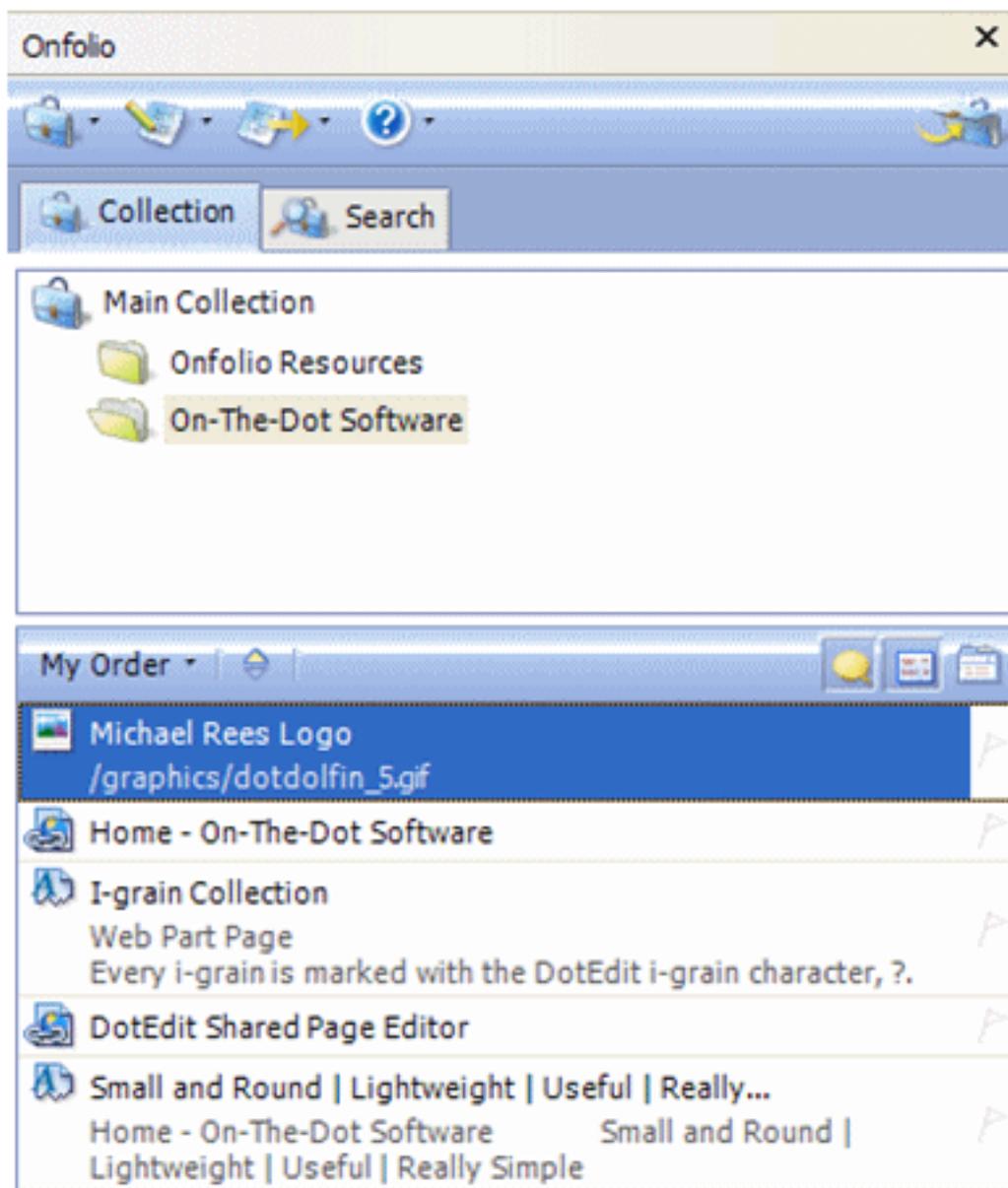


Figure 17. Onfolio Search Information Manager.

In contrast, early usage of DotTegular by the author quite quickly gravitated to collecting fragments from existing web pages, even though this was not the primary focus of the tool. Once DotTegular is installed on a web server of course, no user installation process is required.

The browser is pointed at the DotTegular home page URL that lists the web sites in the various formats (navigable site, slide show and serialized single page) previously described. A link from the home page takes the user to the DotTegular editor page from which the web site is chosen. Opening this site positions the user at an appropriate page containing one of more teglets.

DotTegular information fragment capture starts from a second browser window as the user navigates to an existing web page. Next the user selects the page fragment by dragging with the mouse and copies to the clipboard. Returning to the DotTegular window, the user clicks in the appropriate teglet and pastes the copied fragment. Even more conveniently, the user may drag the selected fragment from one page via the task bar to the DotTegular page and drop the fragment into one of the teglets.

Of great benefit in this simple copy-and-paste process is the automatic translation of relative URLs in the fragment to absolute URLs by the browser. This means that all hyperlinks are copied correctly, and in almost every case the teglet contents look identical to the original page fragment. A good example of this property appears in Figure 15 which shows the outcome of copying a fragment of a publisher's web page for a particular book.

Unlike the other applications discussed in the section though, DotTegular does not copy the original web page URL from which the fragment was copied. The author has found it a very simple matter to copy and paste the URL from the browser address field at the top of the teglet holding the fragment. Nevertheless, this is an additional interaction that causes some user inconvenience, and is due to the inability of JavaScript to manipulate the clipboard contents in sufficient level of detail. Fortunately the copied URL is automatically turned into a hyperlink by the rich text editing component. This allows the user to return to the original web page at any time, and to use the URL for acknowledging the source of the web page content.

The simple web site model adopted by DotTegular treats a site as a sequence of pages. The collection of teglets in each page maps directly to the well known ?pile? metaphor described by Mander, Saloman and Wong (1992). Here information gatherers allocate each information artefact to a pile of roughly similar artefacts. Keeping the number of piles to about ten makes it possible to locate the information artefact or fragment in the future more easily.

Although simple, the pile metaphor of DotTegular does not scale for much larger collections of fragments in terms of locating particular content. Both OneNote and Onfolio contain powerful search capability for convenient location finding. This is a significant deficiency in DotTegular and is currently being addressed in forthcoming versions of the tool. Waiting for XQuery and its implementation is no longer an option!

Like OneNote and Onfolio, DotTegular does have another significant advantage over the Hunter Gatherer in that the user may immediately edit the contents of the copied fragment to make it more acceptable to the collection being built. Indeed, the author finds this necessary for approximately half of all fragments copied. This ability to use DotTegular to assemble collections of existing web page fragments is an unexpected and welcome outcome of the tool development.

Conclusion

The DotTegular tool has explored user interfaces for web page fragments, and applied the resulting design to a tool for building complete web sites using an industry-standard browser. Users need not download any additional software to produce their own complete web sites. Experimental use of the completed tool over several months has shown DotTegular to be effective in an unexpectedly wide range of applications.

Examples of such applications encompass simple information sites like help pages, personal web logs, shared document preparation, idea generation and brainstorming, and building collections of fragments from other web pages. The latter application, now being called search information management, has proved very effective for synopses, educational notes, web logs, research and composites of information gems.

References

- Abrams, D., Baecker, R. and Chignell, M. (1998): Information archiving with bookmarks: personal Web space construction and organization. Proceedings of the SIGCHI conference on Human factors in computing systems, ACM Press/Addison-Wesley Publishing Co., pp 41-48.
- Card, S. K., Robertson, G. G. & York, W. (1996): The WebBook and the Web Forager: an Information Workspace for the World Wide Web. Proceedings Human Factors in Computing Systems, Vancouver, Canada. 111-117.
- Mander, M, Salamon, G & Wong, Y.Y. (1992): A ?Pile? Metaphor for Supporting Casual Organization of Information. Proceedings Computer Human Interaction Conference 1992. 627634.
- Rees, M J. (2000a): Implementing Responsive Lightweight In-page Editing. Proceedings of AusWeb 2000, Cairns, Australia. 134-142.
- Rees, M J. (2000b): Implementing Shared Document Preparation with Lightweight Editing, Proceedings Fifth Australasian Document Computing Symposium (ADCS), December 2000, Twin Waters Resort, Sunshine Coast, Australia. 25-33.
- Rees, M. J. (2002a): Evolving the Browser towards a Standard User Interface Architecture. Proceedings of User Interfaces 2002, Third Australasian User Interface Conference, Australian Computer Science Communications. 24(4): 1-8.
- Rees, M. J. (2002b): A Feasible User Story Tool for Agile Software Development? Proceedings of Asia-Pacific Software Engineering Conference 2002, 4-6 December, Gold Coast, Australia, pp 22-30.
- schraefel, m. c., Zhu, Y, Modjeska, D., Wigdor, D. & Zhao, S. (2002a): Hunter Gatherer: Interaction Support for the Creation and Management of Within-Web-Page Collections. Proceedings World Wide Web 2002, Honolulu, Hawaii, USA. 172-181.
- schraefel, m.c. and Zhu, Y. (2002b): Hunter gatherer: a collection making tool for the web. CHI '02 extended abstracts on Human factors in computing systems, ACM Press, 498-499.

Hypertext References

- href1 Dr Michael J Rees
<http://www.bond.edu.au/it/staff/michael.htm>
- href2 Faculty of IT, Bond University
<http://www.bond.edu.au/it/>
- href3 Bond University
<http://www.bond.edu.au/>
- href4 Udell, J. (2001): Internet Groupware for Scientific Collaboration, Report
<http://udell.roninhouse.com/GroupwareReport.html>
- href5 ZWiki
<http://zwiki.org/FrontPage>
- href6 Zope
<http://www.zope.org>
- href7 On-The-Dot Software
<http://comet.it.bond.edu.au/dotsoft>
- href8 Radio Userland
<http://radio.userland.com>
- href9 Microsoft OneNote
<http://www.microsoft.com/office/onenote/prodinfo/>
- href10 Onfolio
<http://www.onfolio.com>

Copyright

Michael Rees, © 2004. The authors assign to Southern Cross University and other educational and non-profit institutions a non-exclusive licence to use this document for personal use and in courses of instruction provided that the article is used in full and this copyright statement is reproduced. The authors also grant a non-exclusive licence to Southern Cross University to publish this document in full on the World Wide Web and on CD-ROM and in printed form with the conference papers and for the document to be published on mirrors on the World Wide Web.