Creating and Editing HTML Documents With ez/htmltext

by Robert Kemmetmueller

1995 Andrew Technical Conference

Introduction

HTML ("HyperText Markup Language") is a tag language used for styled text in World Wide Web documents. **ez**, being a multimedia editor, is well-suited for handling them. This paper describes a set of dataobjects designed to allow **ez** to edit HTML documents, as well as converting existing **ez** documents into HTML format.

The ***htmltext*** dataobject, its related dataobjects, and the **ez2html** application, are separate from Nick Williams' ***html*** and related dataobjects. When first started, Nick's early ***html*** effort had some architectural problems (which have since been corrected) so it was decided we should start from the ground up rather than expanding/fixing ***html***. As a result, there is quite a bit of redundancy. As time and resource permits, we hope to combine the best features of each and make the result part of the base ATK distribution.

***htmltext*/ez2html** can be fetched from ftp.andrew.cmu.edu:
/pub/AUIS/misc/html.v941121.tar.Z (the last Class-C version [for ATK 6.x])
/pub/AUIS/misc/html.v950519.tar.Z (latest C++ version [for ATK 7.2])

Internal Architecture

The dataobjects and their function:

- ***attlist*** - a subclass of ***list*** that holds tag attributes in the environments.
- ***ez2htmlapp*** - an application that loads a file into a dataobject, copies the contents into an ***htmltext*** dataobject, and lets that write itself to stdout.
- ***hidden*/*hiddenview*** - designed as a generic text inset that is able to NOT draw itself when appropriate. Used here as a holder for unrecognized HTML markup that needs to be preserved. Doesn't appear unless user has write access to document, or explicitly enables its appearance.
- ***hr*/*hrv*** - a simple inset that draws itself as a horizontal rule across the width of the window.
- ***htmlenv*** - an ***environment*** subclass with the additional capability of holding attributes and their values.
- ***htmltext*** - a subclass of ***text*** that can read and write HTML-format documents as well as ATK datastreams.
- ***htmltextview*** - a subclass of ***textview*** with additional user interfaces for editing tag attributes, creating anchors and lists, etc.

htmltext recognizes most paired HTML tags/endtags and represents them in the dataobject as ***htmlenv***s. Symbols such as 'ta 8n `ñ` become their textual counterpart. HTML line breaks such as "'ta 8n `<P>`" and "'ta 8n `<BR>`", are represented as newlines. "'ta 8n `<HR>`" tags become ***hr*** insets, "'ta 8n `<TABLE>`" tags become ***table*** insets, and "'ta 8n `<IMG>`" tags become ***image*** insets.

In the case of images, the view-type ***htmlenv*** wrapped around the inset holds the ’ta 8n SRCattribute the same way a *style*-type ***htmlenv*** would contain attributes of its corresponding tag. If the ’ta 8n SRCattribute points to a directory-relative URL, that image is fetched and displayed (it doesn’t know how to talk to actual Web servers, so "remote" images are represented as a generic picture of a globe).

When ***htmltext*** encounters markup it can’t represent in a more appropriate way, it places the raw markup into a ***hidden*** inset. The contents of these insets are written as-is, thereby preserving the tags that couldn’t be represented as styles or insets.

htmltext uses the standard ATK datastream format for its cutbuffer interface. This allows text and insets to be seamlessly copied/cut from an HTML document, and pasted into other ATK documents and messages in its WYSIWYG form, and vice-versa. A difficulty here is that the datastream format doesn’t allow for "attributes" that HTML tags may contain. This problem was solved by using a special style name to "hide" the attributes in, and nesting it inside the environment containing those attributes. Pasting that into an ***htmltext*** dataobject will re-parse and apply those attributes to the new ***htmlenv***, while pasting it into a *different* type of ***text*** dataobject will make the attributes appear as text (not very attractive, but often useful).

htmltextview provides a menu to display the document as raw HTML markup, and toggle back to its normal view. This is rarely necessary, since raw HTML tags can be inserted and edited as ***hidden*** insets, and there is a mechanism for editing attributes in the normal view as well.

Difficulties and Design Choices

The HTML format is designed to be content-oriented, not presentation-oriented. Along with the spec come "suggestions" about how a browser should render the tags, but it’s ultimately left to the browser’s discretion. *Browsers* must be very lenient about "properly" handling improper markup, ignoring and omitting tags it doesn’t know about, and doing what it thinks is right when pieces are missing (like end tags or symbol delimiters).

The *creator* of an HTML document, however, is at the other end of the spectrum. It has to follow the HTML spec to the letter, to account for all the different browsers, which may interpret sloppy markup in different ways, or may have bugs that appear when given unexpected input, or don’t support certain tags.

An HTML *editor* is in a unique and unfortunate position. It has to do a decent job of interpreting potentially sloppy and invalid markup, render it in a way that the user can edit it, and then write it back out as tight, valid HTML (and preferably in a way that "looks nice" when viewed in its raw form).

In HTML-format files, certain features are considered "unimportant". Consecutive whitespaces and newlines, for example, will be ignored by the browser anyway, and two ’ta 8n
tags are pretty much the same as a ’ta 8n <P>tag. ***htmltext*** discards these unimportant elements, which might annoy users who try to make their raw markup more

"readable".
The recommendation for such users is to use ***rawtext*** for their HTML files and avoid all of ***htmltext***'s parsing and re-interpreting.

htmltext was designed with existing **ez** users in mind. Ideally, the user would have to know little or nothing about the HTML format itself (how many people know *datastream* format?). Since it's still a work in progress, this isn't quite the case yet. Attributes still need to be added manually in most cases, and creating hyperlinks is only slightly easier than it would be with an ASCII editor. However, for headings and lists and basic style changes, ***htmltext*** does its job.

User Interface

The *htmltext.help* documentation has a more complete description of these functions; this is just a brief summary.

The keys and menus for the basic styles are set up much like the regular ***text*** object and the *default.tpl* template. The "Font" menu contains the usual Bold, Italic, etc, while the "Emphasis" menu has more function-oriented style names like Argument and Person. "Region" appears as usual, but with different choices on it.

For headings, the "Header" menu has HTML's six numbered header styles. If **ez**'s standard Chapter, Section, etc styles are pasted in, they appear appropriately, and will be converted to the closest corresponding HTML header tags when saved.

Notable Additions

Editing Attributes

The "Region, Edit Attributes" menu (Alt-a key) allows the user to add and edit attributes of the environment the cursor is in. If nested styles cause confusion, Alt-s can be used to select the specific environment to be edited before pressing Alt-a.

Hyperlinking

The "Region, Add Link" menu prompts for the URL that the link should target, puts the Anchor style on the selected region, and sets its 'ta 8n `HREF`attribute to the specified URL. Hyperlinks appear in **ez** with a dotted box around them, but aren't functional as hyperlinks when editing.

Creating and Editing Lists

The "List" menu has items for creating and editing unordered, ordered, and definition lists. They can be used to make lists out of existing text, or to make the list style be the "current insertion style". New items can be added by pressing tab with the cursor at the front of a line (or after the "term" in a definition list).

Page/Display Functions

The "Page" menu has some additional items for inserting ***hr*** insets, raw HTML markup, and hiding/exposing those raw HTML markup insets.
The "File, Show Raw HTML" menu displays the entire document as HTML markup, with no styles or insets. The markup's "layout" may not exactly match what it was originally (if created by hand), but the differences will be irrelevant in terms of the document's final appearance when viewed by a browser.

Insets
The "Media" menu has items for the insets that ***htmltext*** supports; presently only ***table*** and ***image***.

The Convertor

ez2html is a program that converts any document **ez** can read, into HTML format. This program should almost NEVER be used to create actual files. The reason is that this conversion process is intrinsically lossy. The HTML spec simply lacks many of the features that are available in datastream files. (Margin and indenting control is almost nonexistent in HTML).

Many insets wouldn't be converted. In the future, insets may be able to draw themselves as GIF images, and that would allow the convertor to convert "still shots" of the insets automatically. But even then, the the functionality and editability of the inset would be lost if it were converted to HTML as a "permanent" measure.

Instead, **ez2html** is best used *by a Web server*. If there are datastream documents to be made available on the Web, let the server run them through the **ez2html** convertor on demand (caching the result if desired), and send the result. No permanent loss of information, no dual maintenance. As the HTML spec is expanded, and the **ez2html** convertor improves, so will the appearance of the converted documents provided by the Web server.

ez2html is optimized for making existing ATK help files available on a Web server. The 'ta 8n `HELPTOPICANCHORSUBST`environment variable (or preference) can be specified to allow text in the Helptopic style to be turned into a functional HTML anchor tag that, when clicked-on in a Web browser, will tell the Web server to convert the referenced help file to HTML and send *that* to the browser.

HTML vs HTML+

"HTML+" was the original name of the draft spec that added multimedia extensions (such as tables) to the original HTML spec (now called "HTML 2.0"). HTML+ has since been renamed "HTML 3.0". Much of ***htmltext***'s documentation and menu content still refers to the old "HTML+" name.

Future Enhancements

A whole mess of bug fixes. Then... (in no particular order)

Internal redesign of ***dataobject***'s "other format" writing mechanism. Ideally, tables and other insets would know how to write *themselves* in other formats (MIME, 80-column punch-card, plain ASCII stream, HTML, ...). Then the code to write tables (and other insets) in HTML format could be torn out of ***htmltext*** ...

An improved user interface. ***imageview*** should be subclassed to remove that ugly chiseled border, and provide additional menus when it has input focus that would allow its 'ta 8n `SRC`attribute to be changed (Presently, the text cursor must immediately precede the image inset, to view/edit its attributes). The attribute-editing interface in general needs a major overhaul (Presently, it displays the attributes as text in the message line and lets the user edit them as such; requires HTML proficiency).

Automatic hyperlinking. With an interface to the Web server, **ez** could be used to "Visit URL" rather than only being able to "Visit File". With that in place, and additional features implemented in ***htmltextview***, users could click on a header in another buffer and select that as a target for the anchor they're creating in their document (much the way "Autolink" works today for the ***link*** inset).

An interface to Web servers would also allow ***htmltext*** to do real fetching of inlined images referenced by the document being edited. It may be feasible (with some serious ***imageview*** enhancements) to add some sort of "imagemap" support as well, highlighting the active areas of hyperimages.

Once ATK's consistent widget set is complete and available, using it to create a WYSIWYG inset for viewing and creating HTML 'ta 8n `<FORM>`s with text-input fields, radio buttons, etc, will become more practical.