ED 432 229 IR 019 599

AUTHOR	Kammerdiener, Troy; Smith, Lon
TITLE	Supporting a Web-Based Curriculum with a Diverse Mix of Authoring Competency.
PUB DATE	1999-03-00
NOTE	7p.; In: SITE 99: Society for Information Technology & Teacher Education International Conference (10th, San Antonio, TX, February 28-March 4, 1999); see IR 019 584. Figures may not reproduce clearly.
PUB TYPE	Reports - Descriptive (141) -- Speeches/Meeting Papers (150)
EDRS PRICE	MF01/PC01 Plus Postage.
DESCRIPTORS	*Authoring Aids (Programming); Computer Software; Design Preferences; *Hypermedia; Instructional Design; *Material Development; Screen Design (Computers); *World Wide Web
IDENTIFIERS	*HTML; *Web Pages; Web Sites

ABSTRACT

This paper presents an academic World Wide Web authoring tool called AutoHTML. This tool, a work in progress, is designed to allow the novice user the ability to construct high level Web pages with a consistent look and feel. The system works with a standard Web browser without the addition of component plug-ins. Additional HTML meta-tags are introduced that allow the duplication, deletion, and modification of elements of a Web page. The paper gives an overview of what AutoHTML does and discusses the usability of the system by both a novice and experienced Web author. The need for consistency in course Web pages is also discussed. (Author/AEF)

Supporting a Web-based Curriculum with a Diverse Mix of Authoring Competency

Troy Kammerdiener
Dept. of Computer Science
Northeast Louisiana University, U.S.A
tkammerd@cs.nlu.edu

Lon Smith
Dept. of Computer Science
Northeast Louisiana University, U.S.A.
smith@cs.nlu.edu

Abstract: This paper presents an academic web-authoring tool called AutoHTML. This tool, a work in progress, is designed to allow the novice user the ability to construct high level web pages with a consistent look and feel. The system works with a standard web browser without the addition of component plug-ins. Additional HTML meta-tags are introduced that allow the duplication, deletion, and modification of elements of a web page. The paper also discusses the need for consistency in course web pages.

Introduction

Currently, there are many authoring tools on the market that ease the development of web pages. Yet, these commercial authoring tools are inadequate for the incrementally developed web pages seen in academics. Academic web authoring tools must have several key features: 1) they must work from a common framework and 2) they must allow for easy modification of the common framework. These two points are the motivating ideas behind our development of the AutoHTML system. It is important that students see consistent structure and format of class web pages throughout the course. The reason for this is that students should be able to acquire the needed information without repeated interpretation of its presentation from the instructor. This has been true in the hard-paper classroom for generations. All of the tests have a similar format to all of the other tests in the class, and all the assignments have a similar format to all of the other assignments. This is also true for quizzes, the syllabus and all other repeating documents. The only difference in form, between documents of the same type, will generally be the addition or removal of some fields and the repetition of other fields. For example, the first quiz may have eight questions and the second quiz may have ten; both documents have the students place their names at the top of the quiz and the format of the questions is the same for both quizzes. It is for such reasons that it is necessary to work from a common framework. A common look and feel for web-based documents can be realized with AutoHTML. Another theme of AutoHTML is the desire to make it useful for web authors of all capabilities. The following section will give an overview of what AutoHTML does, then subsequent sections discuss the usability of the system by both a novice and experienced web author.

Executive System Overview

The simplest AutoHTML system consists of a template and a set of scripts which transform this template according to the design of the template and interaction with the user. Templates are ordinary web pages written in HTML which also include meta-data in the form of Merge Fields and Sections. Both Merge Fields and Sections resemble enclosure-style markup tags, but they are embedded in standard HTLM comments, so that they do not appear when the page is viewed in a web browser. Merge Fields are used to identify parts of the page that may need to be modified. Sections are used to identify parts of the page that may need to be duplicated, destructively deleted, non-destructively deleted, or restored (from a non-destructive deletion). Merge Fields can be nested within Sections, but no other nesting of Merge Fields or Sections is allowed. Processing an AutoHTML template consists of 4 primary steps:

1. Create a web-based form from the template, which allows the user to indicate which action to take for each Section in the template and submit this form back to the system.
2. Use the actions submitted in step 1 to modify the template by duplicating, destructively deleting, non-destructively deleting, or restoring sections of the template.
3. Create a web-based form from the template, which allows the user to make changes to the contents of Merge Fields in the template and submit this form back to the system.
4. Use the new Merge Field values created in step 3 to modify those Merge Fields in the template.

Steps 1 and 2 must be done in sequence. Steps 3 and 4 must also be done in sequence, but the operation cycle may involve repetitions of these sub-cycles as needed.

Creating a Template

General Template Creation Process

Creating a template usually starts by modifying an existing web page that represents a class of web pages that are desired for a site. In other words, there may be many web pages like it, using the same style, general appearance, and basic components, but their specific contents will vary.

Step 1. Identify portions of the page which represent a repetition of the same kind of entry. You will usually note recurring pieces of text or format which distinguish such parts. Enclose one of these in a Section meta-tag, and delete the others.
Step 2. Identify portions of the page which are optional. Enclose them in Section meta-tags.
Step 3. Identify portions of the page whose content may vary in each specific instance of the template. Enclose those parts within Merge Field meta-tags. Remember that *anything* within a Merge Field is subject to being changed. To remind yourself of that fact (and to make your template more descriptive to a user), replace the changeable part with text that makes its purpose clear.

Rules for Creation of Section and Merge Field Meta-tags

The Section meta-tag is an enclosure tag (meaning that it actually consists of two paired tags) of the following form:

```
<!-- ##SECTION## identifier --!>
  contents of the section
<!-- ##/SECTION## --!>
```

Neither meta-tag may be split between lines, although the *contents* of the section will usually span many lines. The identifier is any string of alphanumeric characters, including underscore (_), and is used to uniquely identify a given section. Do not end a section identifier with an underscore followed by a sequence of digits, as this is reserved for system use when the section is duplicated.

The Merge Field meta-tag is also an enclosure tag of the following form:

```
<!-- ##MERGEFIELD## identifier --!>
  contents of the section
<!-- ##/MERGEFIELD## --!>
```

As with Section meta-tags, Merge Field meta-tags must be complete on a single physical line. Merge Field identifiers follow the same rules as Section identifiers, and all identifiers, regardless of type, must be unique. Finally, no meta-tag may be placed within another individual HTML tag. For example:

```
<A HREF="<!-- ##MERGEFIELD url --!>http://cs.nlu.edu
<!-- ##/MERGEFIELD## --!>">
```

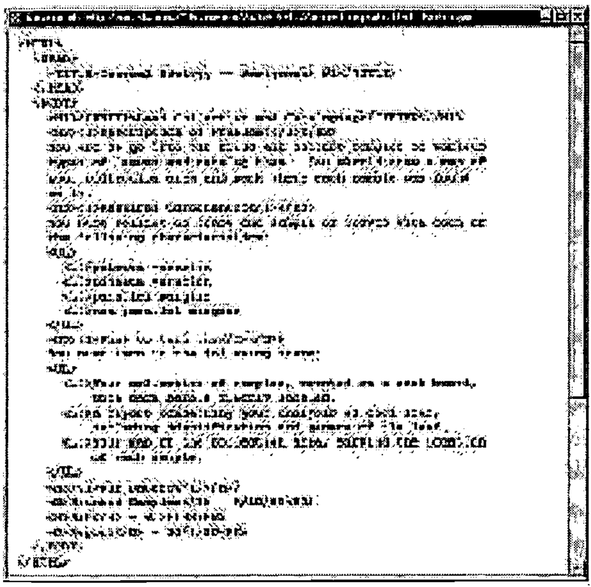
would be an error, but:

```
<A HREF="<!-- ##MERGEFIELD url --!><A HREF="http://cs.nlu.edu">
<!-- ##/MERGEFIELD## --!>
```

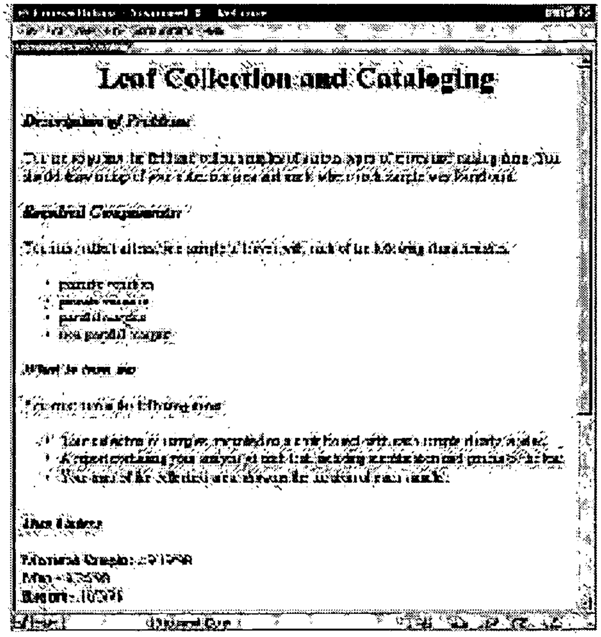
would be fine. Also, at this time, Sections may not be nested within other Sections.

Example of Template Creation

Consider the following web-published class assignment:



```
[illegible]
```



Leaf Collection and Cataloging

Description of Problem

[illegible]

Required Components

[illegible]

- [illegible]
- [illegible]
- [illegible]
- [illegible]

What to turn in

[illegible]

- [illegible]
- [illegible]
- [illegible]

Due Dates

[illegible]

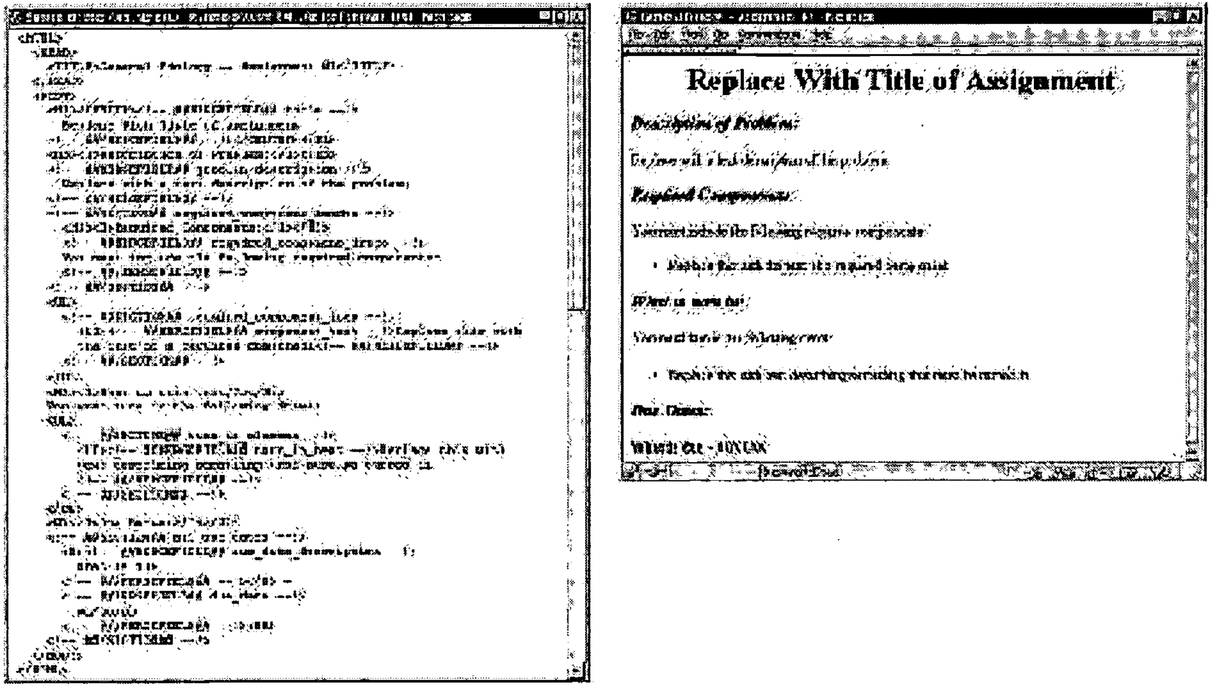
Figure 1: HTML code and browser display of a web page suitable as a basis for a template.

In Step 1, the repetitive elements of the web page are identified. Each item in the required components thus makes a section, as does each item in the "What to turn in" part, and each due date. Using Step 2, the heading and introductory text of "Required Components" is identified as optional, but it is decided that the other parts will be present in every assignment. Each Section identified by these steps is enclosed within Section meta-tags.

After identifying the Sections, Step 3 is used to set up the Merge Fields. We want to be able to make changes to the following items in the page:

- Title of the assignment
- Text of the problem description
- Leading text for all required components
- Particular required components
- Items that have to be turned in
- Descriptions of due dates
- Dates of due dates

Note that while most of these Merge Fields will appear *within* a Section, some (such as the assignment title) do not. These are known as "orphan" Merge Fields. For each of these areas, *one* example is enclosed within Merge Field meta-tags, and the others are removed. The text of each is also made more general and explanatory. The resulting template looks like this:



Replace With Title of Assignment

Description of Problem:

Required Components:

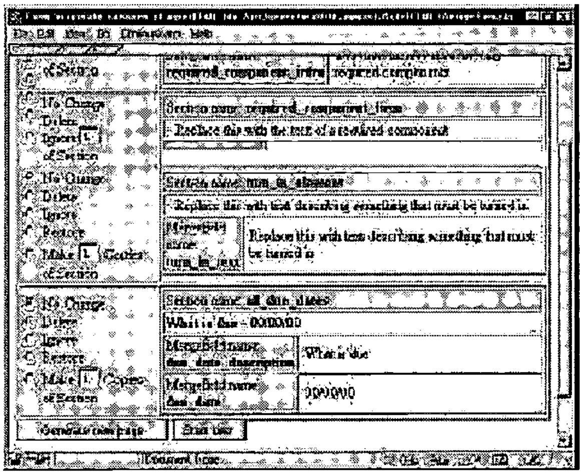
What to turn in:

Due Dates:

Figure 2: HTML code and browser display of the resulting AutoHTML template.

Using a Template

Once a template has been created, a novice user can use it to create an entirely new web page. To do so, enter the AutoHTML system and choose the template you wish to base your web page upon. A form similar to the following is displayed in the browser:



required_component_intro | required components

Section name: required_component_form

Replace this with the text of a required component

Section name: turn_in_element

Replace this with text describing something that must be turned in.

Mergefield name: turn_in_text	Replace this with text describing something that must be turned in

Section name: all_due_dates

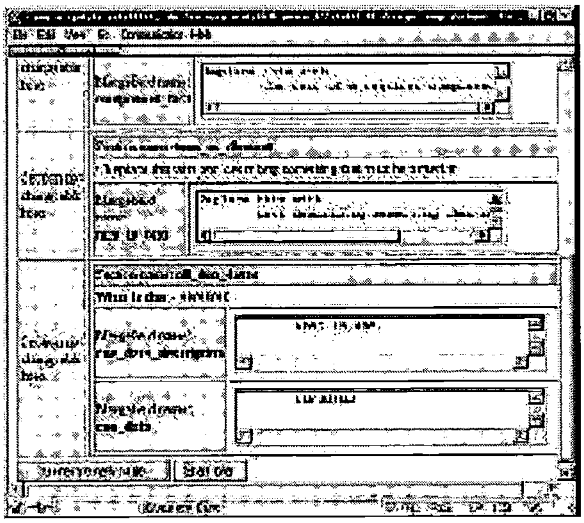
What is due - 00/00/00

Mergefield name: due_date_description	What is due
Mergefield name: due_date	00/00/00

Figure 3: Browser display of a form to update Sections of an AutoHTML template.

AutoHTML is designed with simple radio buttons to select actions to be taken for each Section in the template. Each Section of the template is displayed with the option to perform "No Change", "Delete", "Ignore", "Restore", and "Make Copies". By choosing "Delete", that Section will be deleted from the template permanently. By choosing "Make Copies", the user can make as many *additional* copies of that section as the user wishes. "Ignore" is a non-destructive delete that allows the user to create a web page that does not display that section in the browser (it will still appear in this form). That section can be added back to the web page by using "Restore" on it at a later time. After all choices are made, the user clicks the "Generate new page" button, and the template is modified on the server.

After the template has been re-generated with appropriate Sections, any Merge Field can be modified to reflect desired textual changes. At this point, the user is presented with the option to view the new web page, or to modify its Merge Fields. When "Edit Merge Fields" is chosen, a form similar to the following is displayed in the browser:



Section name: turn_in_element

Replace this with text describing something that must be turned in

Mergefield name: turn_in_text

Section name: all_due_dates

Mergefield name: due_date_description

Mergefield name: due_date

Figure 4: Browser display of a form to update Merge Fields of an AutoHTML template.

AutoHTML uses a straightforward fill in the blanks format to eliminate the need for the novice to understand HTML tags. Yet, the more experienced user can insert HTML tags as desired, since the text will be replacing actual HTML code. The web page (which is still a template) is then re-generated by clicking the "Generate new page" button. This cycle of adding and removing sections and modifying merge fields can be repeated indefinitely until the desired web page is created.

Benefits of AutoHTML

AutoHTML is a system that gives the novice user a way of developing high-quality and consistent classroom web pages. This increases the web-presence of the classroom without undue workload on the instructor. This added web-presence gives students an accessible resource at all times. This adds immediacy to the instructor's access to the students. In other words, the instructor need not wait until class time to make announcements, answer questions, or to present assignments. This is especially important in the area of distance learning and in classroom environments with infrequent meeting times, such as once a week. Also, since AutoHTML templates are ordinary web pages, they can be modified directly by an expert web author doing more elaborate page development, as long as the simple rules for AutoHTML meta-tags are not invalidated. AutoHTML will simply act as a tool to reduce creation time of similar pages.

AutoHTML also promotes the gradual introduction of HTML to new users. A new user will be gradually exposed to individual HTML elements such as anchors and formatting tags in the content of Merge Fields. As a user gains understanding, new HTML tags can be added to the merge fields. In this way, a new user can use a limited set of tags without having to learn a large set of tags in order to get started.

Consistency in the look and feel of an organization's web site can also be promoted using the AutoHTML system. While tools such as style sheets provide this from a low-level formatting perspective, AutoHTML adds the ability to control specific content and high-level formatting elements as well. This type of consistency makes it clear to a student the difference between the organization's official pages, and external pages that are merely referenced by the official site. It also reduces the learning curve for a student who has taken other courses from the same organization.

Conclusion

AutoHTML is a work in progress. It still requires additional testing and feature additions. It was developed out of the observation that in our own course web sites similar formats were repeatedly used. As competent HTML users, we had the ability to copy old pages, strip out the content without losing structure and add in new structure and content. Yet, other less competent HTML users were not able to do this, beginning each time from scratch. AutoHTML gives less competent users the ability to re-use the structures already in a web page format.

Future directions for the system include the creation of additional components for error detection during template creation. Currently, merge field names and section name fields must be unique; yet the current system allows ambiguous naming to occur. This situation may cause errors in the operation of the scripts. Another need is automation of multi-document structures. For example, it may be desired to link a newly created page into an existing page which contains links to similar documents of the style, such as a class page which references multiple assignment pages, each of which was built from the same template. A multi-document structure should be developed that will change or add references in other pages. Additional progress and updates on this project will eventually be reported from the authors' web sites at http://cs.nlu.edu/~csci/.

Acknowledgements

Our thanks to Elizabeth Smith, who provided valuable comments and proofreading of this paper's various drafts (as well as longsuffering with one of the authors during its creation), and to Kim Taylor who was also of great assistance in proofreading the manuscript.