Transferring online formatted HTML layouts into Flash and PDF

Nikolaj Cholakov

Abstract: *This paper presents an approach and a realisation based on PHP and JavaScript for online generation of HTML text layouts transferred into Flash movies and PDF documents. This system can be useful for sites using content management systems (CMS) to maintain various information fields presented to the client by Flash movies. Once constructed with the editor, the text layout is automatically transformed to a Flash-compatible form and inserted into the CMS database. Along with that a PDF file representing the same layout in printable form is generated.*

Key words: *HTML formatting, Command identifiers, Flash, PDF.*

INTRODUCTION

The content management system (CMS) is one of the most important aspects of the overall design of a modern web site, no matter if this is a personal web site, a site for a medium sized business, or a common portal. It is especially important when the site will constantly have new products, company news or different articles added to it, which is the case for most of the commercial sites and information portals. The newly added content could be left unformatted, but this is not the preferred solution. So the person from the company who will be adding that content must have much more than beginner's knowledge of HTML. Another possibility is to offer some form of offline template that the operator can edit and upload, but this is neither time effective nor streamlined. A very good issue here would be to provide the CMS operator with a browser based WYSIWYG text editor and formatter not requiring HTML skills from the user.

On the other side, presenting the information fields to the client using Flash becomes more and more popular. Textual components in Flash support some of the HTML and CSS formatting capabilities, so the information field's layout can be transferred intact into Flash. It is also a good idea to offer a printable version for each information field, and the PDF document format is perfect for that purpose.

LAYOUT

The solution of the above-formulated problem can be achieved in three steps: first, to build an online WYSIWYG text editor capable of editing and formatting text using only HTML tags; second, to keep the text layout generated by the editor, and to import this layout into Flash textual components; third, to generate a separate PDF document for each layout, still keeping the formatting intact. These tasks must be done by the CMS system automatically, without an intervention from the operator, who should only provide the text and construct its layout using the online editor.

1. Building the online HTML editor

The online text editing and formatting can be easily done with the capabilities provided by the JavaScript method execCommand() [3]. This method is used to execute a command on a document. In the sense of execCommand(), a command is a pre-defined set of functions that can manipulate the page layout, insert an image, link, list, text box, horizontal rule, etc right into an HTML document in the browser. Each command has its own command identifier, instructing the execCommand() method what to do over a selected part of the document. Almost all text formatting which offers FrontPage for example, can be done in the browser using the command identifiers. At this time the execCommand() method and the command identifiers are not a World Wide Web Consortium (W3C) standard, but they are supported by Internet Explorer and Mozilla Firefox – the most popular browsers for Windows and Linux.

Command identifiers can be applied not only on the whole HTML document, but also on a part of a document, put in an <iframe> tag. This inline frame can be used as a content pane for the online editor, by setting the "designerMode" attribute of the iframe to "on" and putting the text which is to be formatted into it.

The text which is to be formatted comes from the database of the content management system. The CMS operator can activate the editor by clicking on a link called "Format text" and placed nearby each information field. The link activates a PHP function, which generates the editor's layout, pops up a new browser window and renders the editor inside. At the same time the PHP function extracts the text for the corresponding field from the database and loads it into the editor. The text layout can also be built from scratch – by entering the text directly into the editor pane or by copying it from an external file.

Each formatting command, supported by the editor, is presented in the interface by an image or a dropdown box. When the user selects a command, a JavaScript function is called. This function passes the corresponding command identifier to the execCommand() method, which immediately changes the text layout. As an example here is the code of the makeBold() function, which is called to make the selected text bold, by means of a reference to the execCommand() with the "bold" command identifier :

```
function makeBold(){
   document.getElementById('iEditor').contentWindow.document.execCommand(
                                                    'bold', false, null);";
}
```

In this call "iEditor" is the "id" attribute of the editor's <iframe> tag. Functions analogous to makeBold() are provided for each formatting operation, and they all perform a call to ececCommand() with the corresponding command identifier as an attribute.

Figure 1 shows the overall look of the editor with some text formatted using the editor's capabilities:

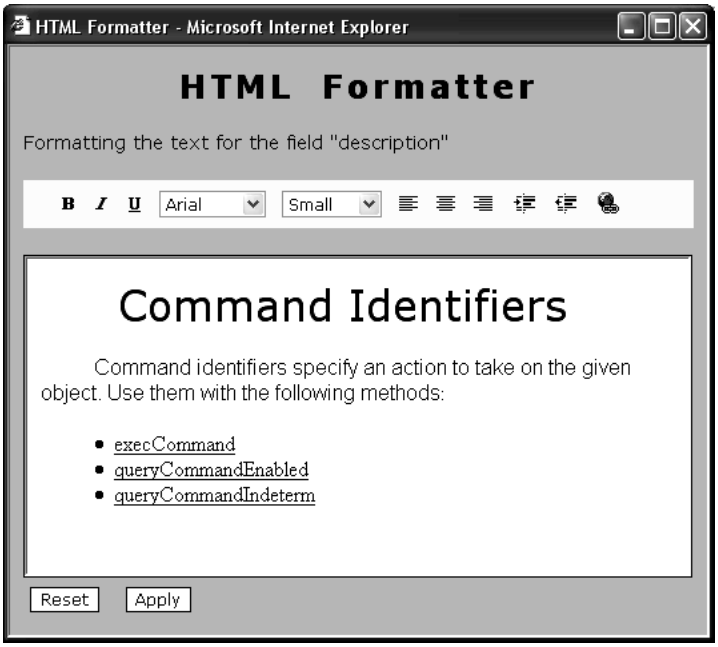

Figure1. The online HTML editor

If the user had selected some text in the editor's pane, then only that text's formatting will change. If the user hasn't selected any text, then any text that he types in after applying the new formatting command will appear with the new properties.

Each formatting action performed by execCommand() inserts automatically into the text all necessary HTML formatting tags. When all desired formatting is done, the content of the editor's pane is saved back in the database, and the text comes along with the HTML formatting tags. Afterwards the contents of each field can be re-edited, in this case the editor shows the existing text layout correctly, and it can be changed again and again.

2. Importing the HTML layout into Flash

Flash supports HTML formatting tags, so if the text which is to be loaded into a text component comes along with the HTML formatting tags in it, the layout will be rendered automatically [4]. There are some compatibility problems however, and they must be eliminated before the text is loaded into the Flash [2], [6]. First, Flash supports only several HTML tags <b>, <i>, <u>, <p>,
, <font>, <a> and . For this reason some capabilities of the command identifiers, like block indent and outdent cannot be directly applied in Flash. Second, there are differences between the tags used by the command identifiers and the tags supported by Flash: for example to make the text bold, the "bold" command identifier uses the tag, while Flash supports only the <b> tag. Third, Flash requires all tag attributes to be quoted, for example <p align="center">, while command identifiers do not use quotes.

On the other side, Flash supports several CSS properties [7], which can be used to replace the missing support for some HTML tags. Table 1 summarizes all the tags, used by the command identifiers, and also the corresponding elements which can be used in Flash and in the PDF converter HTMLDOC to keep the imported text layout intact.

Table 1. Comparison of text formatting elements supported by Command Identifiers, Flash and HTMLDOC

Tag supported by Command Identifiers	Tag supported by Flash	Tag supported by HTMLDOC
<STRONG>	<B>	<B>
<EM>	<I>	<I>
<U>	<U>	<U>
<BLOCKQUOTE>	<SPAN class="indent">	<BLOCKQUOTE>
<FONT face="face">	<SPAN class="face">	<FONT face="face">
<FONT size="face">	<SPAN class="size">	<FONT size="face">
<FONT color="color">	<SPAN class="color">	<FONT color="color">
<P align="align">	<P align="align">	<DIV align="align">
<A>	<A>	<A>

Because Flash does not support , , and <blockquote> tags they must be replaced. The <font> tag is supported by Flash, but replacing it with a tag with a stylesheet-defined class increases the flexibility of the system. The <font> tag supports only 7 fixed font sizes, while by defining a variety of stylesheets which are to be applied in different Flash presentations the set of font sizes can be unlimited. For the same reason <font> tags with "face" or "color" attributes are replaced by tags having different classes. These replacements also concern the case when the <font> tag has more than

one attribute, for example <font face="arial" color="green">.

The layout of all fields must be kept in the database in Flash-compatible format, because Flash gets the content of each field directly from the database. Therefore the conversion from command identifiers to Flash-compatible layout must be done before the edited field is inserted back into the database. So the PHP function which updates the field, parses the content of the editor's pane, makes all necessary replacements and then updates the field in the database. This means however that if a field, which already has been formatted, is to be reloaded back into editor, a backward conversion from Flash to command identifiers must be performed. This conversion is done by the PHP function loading the text into the editor.

3. Generating PDF documents preserving the HTML text layout

The conversion from HTML to PDF format can be done programmatically using PHP, Pearl or some other popular language. However this is not a simple task [1]. There are many ready-made tools offering conversion of HTML documents to PDF format. The PDF writers are not considered here, because they require user intervention and the purpose of this system is the fully automated PDF documents generation.

During the system development many converters were tested, starting from open-source free ones, up to these requiring 500 Euro licence per server. Finding a good solution proved to be a difficult task. Most of these tools lack Linux support, which is a serious disadvantage. Some of them (mostly open-source) showed strange bugs and proved to be unusable.

Finally the tool called HTMLDOC [5] was chosen because of the following capabilities: Windows and Linux versions, command-line interface, support for all HTML tags, bug-free performance.

Although HTMLDOC supports all HTML tags, there are some peculiarities in the PDF layout constructing algorithm which make another conversion obligatory: from the Flash-compatible format stored in the database, to a format, allowing the HTMLDOC tool to preserve the text layout intact in the resulting PDF document. The differences between these two formats are summarized in Table 1.

HTMLDOC has a graphical interface, but can also be used from the command line. This allows the fully automatic usage of this tool from the CMS operator's point of view. The tool is called by a PHP function using the following command line:

```
htmldoc -f outfile.pdf -t pdf14 --charset cp-1252 –webpage infile.html
```

Here "infile.html" is the name of a temporary created HTML file containing the layout which is to be converted, and "outfile.pdf" is the resulting PDF file.

On Figure 2 the whole layout transferring mechanism is graphically presented:

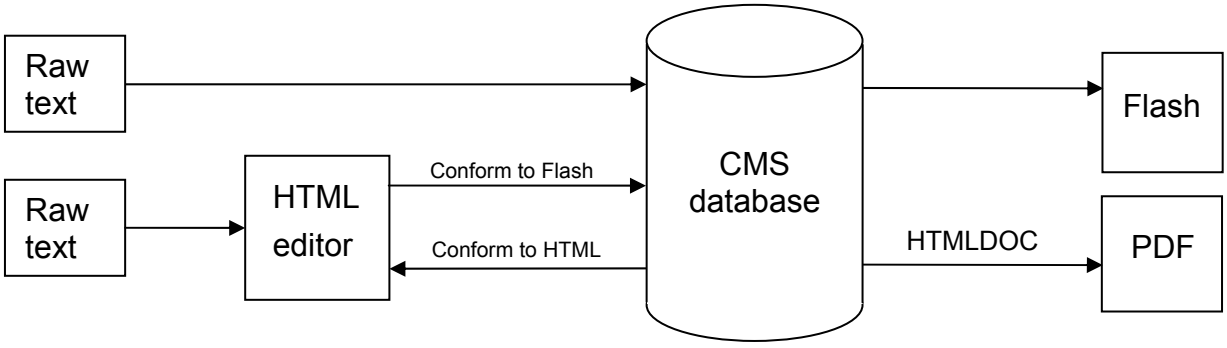

Figure 2. Overall layout transferring mechanism

The raw text is provided by the CMS operator directly to the editor, or prior to the text formatting, using the standard CMS interface. The CMS operator can associate the HTML editor to each desired database field. After that at any time the field's text can be loaded into the online editor and quickly formatted; then the produced layout is converted to the Flash-compatible format and saved into the CMS database. The editing procedure can be repeated as many times as needed. When the text is loaded from the database, it is converted from the Flash-compatible form back to the original HTML format, produced by the HTML editor. Then after the newly applied formatting operators the text is saved again into the database along with its new layout.

Each time when a database update operation is performed, the PDF converter is automatically called. Thus all changes made to the content or to the layout of a field are immediately reflected in the corresponding PDF file.

The layouts saved into the CMS database along with the generated PDF files can be directly used by Flash movies. In principle the idea is that the Flash movie should present each information field with the corresponding format in a text component, and nearby should stay a link to the PDF version, but of course this is not obligatory – the flash designer can utilize all fields the way he desires.

CONCLUSIONS AND FUTURE WORK

The online layout formatting and transferring system described above is a fully operational, in-browser editing system capable of building and formatting HTML contents on the fly, followed by automatic transferring of the issued layout into Flash and PDF documents. The practical application of this system can intensify and simplify the exploitation of content management systems using tools for formatted presentation of textual information. The realization is entirely based on PHP and JavaScript, it is fully independent and can be attached to every PHP-enabled system as an external module.

The ideas for the future development of the system are basically pointed at extending the CSS support to take advantage of all possible Flash and PDF capabilities. In fact the capabilities of the system as a whole are determined by the limited set of functions, supported by the command identifiers. More diverse results could be achieved with the usage of a suitable set of predefined styles.

REFERENCES

[1] Steward, S. PDF Hacks. Cambridge, O'Reilly, 2004

[2] Yeung, R. Macromedia Flash MX 2004 Hands-On Training. Berkeley, New Riders, 2004.

[3] Http://msdn.microsoft.com/library/default.asp?url=
/workshop/author/dhtml/reference/commandids.asp

[4] Http://visualintensity.com/flash-tutorials/mx2004/textarea-load/

[5] Http://www.easysw.com/htmldoc/

[6] Http://www.macromedia.com/cfusion/knowledgebase/index.cfm?id=tn_14808

[7] Http://www.whatdoiknow.org/archives/001181.shtml

ABOUT THE AUTHOR

Nikolaj Ivanov Cholakov, PhD, Department of Information technologies, University of Veliko Turnovo, Phone: +359 62 649831, E-mail: n.cholakov@uni-vt.bg.