Editing web pages with FCKeditor

EASY FIX

Small sites often have outdated information because the webmaster doesn't have time to keep all the pages up to date. Even something as simple as a spelling correction might take days to change. FCKEditor frees the webmaster by letting the user take charge. **BY JAMES MOHR**

In addition to managing personal web sites, I also volunteer as a webmaster for a local community sports club. A common complaint by everyone at the club is that no one takes the time to provide current information. Some of the sections at the site provide no information at all, simply because no one has the time to keep the pages up to date. I have the technical expertise to manage the site, but I can't devote the time to maintaining every individual page. I needed an easy way for the club members to put the information online themselves.

The simplest way for a novice user to maintain a website is to write the text with some text editor and then upload the text files using an FTP client. Unfortunately, most of the people in our club have trouble even with that. So, to get as many people as possible to provide current information, I needed to make updating the information as simple as possible. I created a database for news items and upcoming events, allowing the users to add information to web-based forms. However, I didn't have an easy solution for allowing users to modify the actual content pages; that is, until I ran into FCKEditor.

FCKEditor is a text editor that resides on the server and is accessible through an ordinary browser window. By integrating FCKEditor into the website, I created a convenient means for users to edit their own web pages. Even the most inexperienced users were comfortable working in a text editor, so FCKEditor provided a solution that was even easier than a comparable Zope or Wiki-based editing environment. The challenge, of course, was figuring out how to integrate FCKEditor into the website. In this article, I'll describe how to build in online editing with FCKEditor.

Full-Featured Editor

FCKeditor (Figure 1) has all of the features most people need in a word processing tool, including a full-featured toolbar. Below the FCKEditor toolbar is the WYSIWYG text area. FCKEditor works with almost any browser and in any environment.

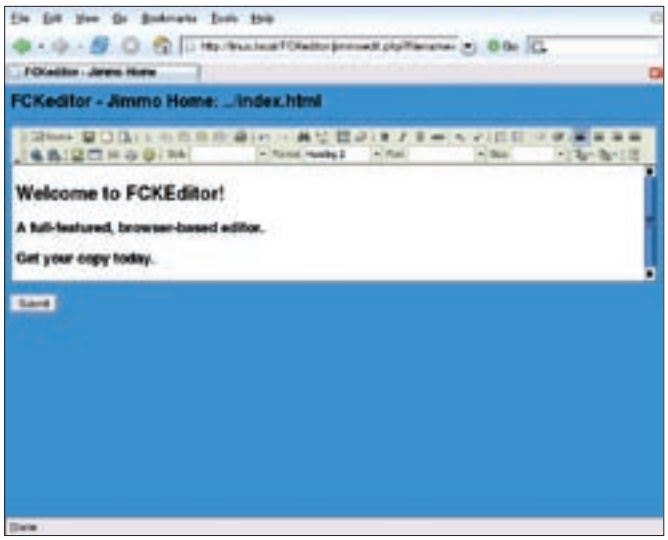

Figure 1: The FCKEditor main window.

On the server, FCKeditor provides support for the more common web development languages, such as PHP, Perl, ColdFusion, Java, and even ASP.Net. This support for popular web development languages allows seamless integration into your existing site.

FCKeditor is available in over two dozen languages. By default, the language is detected through your browser settings, but it is easy enough to set a specific language in the configuration file.

The biggest downside of FCKEditor is the lack of documentation. What little there is makes repeated references to the samples provided (see the sidebar titled "Sample Files"). While the samples help you see what the program can do, this approach to documentation does require you to dig through a lot of code to figure out how to customize the program.

Installation

You can get the latest version at the FCKEditor homepage [1]. As of this writing, the latest is Version 2 Release Candidate 2. You should definitely get RC2 if you are running an Apache server on Linux, as there were some problems with RC1.

Once you have obtained the installation package, first create an *FCKEditor* directory underneath the document root of your web server and unzip the package file into this directory. You can change the location, but to do so, you will need to make several changes to the configuration files (see the sidebar titled "Setting the Location.")

Each of the four PHP samples provided with the package (see the sidebar titled "Sample Files") demonstrates different aspects of the editor. For example, one sample lets you choose the toolbar, another the language, and so forth. Each of the components can be added to any page, so you can build a custom editor that allows you to select any of the options.

I chose one of the sample files and used it as a starting point for my configuration. The sample I chose seemed to be the easiest configuration to adapt. In my case, I don't want the users to be able to tamper with the configuration of the editor itself because I am trying to keep the system as simple as possible. So, I stripped out all of the extra code related to configuring the editor. I also turned off the setting that provides autodetection of the language. The website is in Germany, so I set the default language to German:

```
$oFCKeditor->Config ➥
['AutoDetectLanguage'] =➥
 false ;
$oFCKeditor->Config ➥
['DefaultLanguage'] = 'de' ;
```

If you are setting up FCKEditor for your own environment, choose a sample file that fits your situation and adapt the settings as necessary.

Loading Text into the Editor

The first step in integrating FCKEditor with an existing website is to define a means for loading text into the editor. The *Value* attribute contains the actual text that appears in the editor window. You can define the text that will appear in the FCKEditor window by simply setting the *Value* attribute:

```
$oFCKeditor->Value = ➥
'Your text' ;
```

You could set this value to an empty string to start up with a completely empty browser, but one of the best features of FCKEditor is the ability to edit existing pages. So, we need to figure out how to get the contents of an existing file into the editor. The simplest method is to use the PHP function *file_get_contents()*:

```
$oFCKeditor->Value = ➥
file_get_contents➥
("path_to_file") ;
```

Warning: file_get_contents() can load a URL or load the file directly. The URL might contain server-side includes, inline frames, or any number of other things. Thus, the file that ends up in your browser, might not be the one you really want to edit. For me, the safest approach is to avoid URLs. Before passing the filename to file_get_contents(), I checked to make sure it was loading a local file.

Even using this method, the contents are still more or less static. We need a way to dynamically pass in a file name that we can load. You can do this easily by passing the filename as part of a query string:

```
http://linux.local/FCKeditor➥
/jimmoedit.php?filename=➥
index.html
```

Then, inside the editor script(*jimmoedit.php* in this example), you read the filename variable:

```
$filename = ➥
$_GET['filename'];
```

Sample Files

You will find several samples for each development language in the *_samples* directory. I use PHP, so I used the files in the *php* directory and copied one of the sample files from *_samples/php* into the *FCKEditor* root directory. I then copied the *sampleposteddata.php* and *_samples/sample.css* files. In my case, I used *sample2.php* and called it *jimmoedit.php* (for simplicity's sake). This is the file that you would need to load, and the file where you need to start making any changes. I also renamed the css file to *jimmo.css* and changed the *jimmoedit.php* file accordingly.

Be careful here, because it is possible to create a query string that does a lot more than just load a file on your web server. For example, a malicious user could specify the file *.htaccess* or any other sensitive file on your system. Therefore, you should always check to see that the value written to *$filename* is valid and logical before you load the file.

Access to the Editor

More than likely, having your users type in the query string by hand is too much work, so we need to think of an easier way to pass the file. How you do it depends on how the page is generated, who is allowed to edit the pages, and so on. On one site that I manage, the pages are generated using PHP, so I can check whether the user is logging in or not. If so, an "Edit this page" link contains the full URL to the editor, including the relative path for the appropriate file.

On another site, things are a bit more complex because I am not using PHP or any other scripting language to help me figure out if the user is logged in. The solution is to put the editor into a password protected directory on the server using the Apache standard basic authentication. I can determine the name of the user who is logged in, like this:

```
$current_user = ➥
$_SERVER["PHP_AUTH_USER"];
```

After verifying that the user is logged in, I determine the group to which the user belongs. I need to determine the group because the system determines the files the user is allowed to edit based on the user's group affiliations. For example, a person in the volleyball group can only edit pages within the volleyball directory.

Since the directory names are the same as the group names, it is easy to figure out the correct directory. I can then use various PHP functions to search the directory and create an HTML list of available files, generating corresponding links that then pass the file name to edit. One key advantage of this approach is that I can filter out any files that users should not have access to or should not be editing.

Using the Editor

The editor itself behaves the same way as any other text editor. For example, by selecting text and then clicking the appropriate button, you can change the text to bold or italic, change the font size, and so forth. This does not just apply to standard tags – FCKEditor will insert various styles into the tag. The editor even provides a source code mode, which allows you to edit the HTML code directly.

One feature that really impressed me was the table editor. When you click on the appropriate button, you get the popup window in Figure 2. As with other HTML editors, you can define the dimensions of the tables (both number of cells and pixels), border, cell padding, and so forth.

Once the table is inserted, you can change a wide range of table properties. If you select the table by left-clicking it, you can drag the side or corners to change the size. Right-clicking the table allows you to insert or delete rows, columns, and even cells. You can also split rows or columns, and even change the basic properties of the entire table. Other features let you create lists, add links and images, or change the background colors.

Browsing the Server

FCKEditor provides a mechanism for browsing files on the server. This browsing is accomplished by using "connectors" for the various languages. For example, the connector and the related code for the PHP editor are located in *editor/filemanager/browser/default/connectors/php*.

You define which connector the browser should use in the *fckconfig.js* file. There are two variables in the *fckconfig.js* file you need to set: *FCKConfig.LinkBrowserURL* and *FCKConfig.ImageBrowserURL*. By default, these variables point to the ASP connector, so you will also need to uncomment the PHP connector.

By default FCKEditor will look for files in different directories based on their type. For example, you may wish to create a special directory for files and another for images. So that the path is not dependent on the resource type, you need to change the *connectors/php/io.php* file. First, in the function *GetUrlFromPath()* change the line

```
return $GLOBALS➥
["UserFilesPath"] ➥
. $resourceType . $folderPath ;
```

to

```
return $GLOBALS➥
["UserFilesPath"] ➥
. $folderPath ;
```

Then, in the function *ServerMapFolder()* change the line

Setting the Location

Although the instructions say to unzip the package into the *FCKEditor* directory under your web server's DocumentRoot, you can basically place the package anywhere. To change the location, you need to set the *BasePath* property of the editor object. For example, in the *jimmoedit.php* file you'll have an entry like this:

```
$oFCKeditor->BasePath = ➥
"/FCKeditor/";
```

Note that prior to this entry, you have already created the new instance of the editor. So here you are specifying the *BasePath* attribute for that instance (since $oFCKeditor is the current instance of the editor object).

If you look at the comments in the sample, you will see that *BasePath* is determined based on the *_samples* directory. Either you set the property here or within the file that creates the editor class. For PHP, this would be the *fckeditor.php* file. (In addition to creating the *FCKEditor* object, the *fckeditor.php* file is also responsible for creating the HTML that defines the inline frame that is the editor itself, plus the necessary code to create the toolbar, and so forth.) Within the constructor function for the editor class, you set the *BasePath* attribute (as well as several other useful attributes). In this example, since the path is hardcoded in the constructor, you should comment out or remove the extra code within the *jimmoedit.php* file.

Next, you have to change the path to the actual editor code. This location is included at the top of the *jimmoedit.php* file using a relative path:

```
include("../../fckeditor.php") ;
```

Since the editor file is now in the FCKEditor root directory (the same as the *fckeditor.php* file), you need to remove the relative path.

Figure 2: HTML table configuration.

```
$sResourceTypePath = ➥
$GLOBALS["UserFilesDirectory"]➥
 . $resourceType . '/' ;
```

to

```
$sResourceTypePath = ➥
$GLOBALS➥
["UserFilesDirectory"] . '/' ;
```

Next you need to change the path for the user files. This is done in the *connector.php* file. Near the top of the file, you will find a reference that looks like this:

```
$GLOBALS["UserFilesPath"] = ➥
'/UserFiles/' ;
```

Change the location to a directory of your choice relative to the Document-Root of your server.

Storing the Edited File

When you are done editing the file, click the *Submit* button, which sends your text to the defined action (just like any web form). The default is *sampleposteddata.php*, which I simply renamed *posteddata.php*.

If you look at the sample post file, all it is doing is taking the content of the input area from the form and displaying it. However, there is no reference to the original file, quite simply because the samples weren't written to read the contents from files. To allow the script to store the file, we must pass it the name of the file. I did this by setting a hidden variable within the form:

```
<input type="hidden" name=➥
"filename" value=➥
"<?=$filename?>">
```

Then on the post page I know which file has been edited.

One thing to note is that all the sample post script does is provide a framework for you to create your own mechanism to save the file. In the sample post script, any backslashes added to the form are removed and the HTML code is converted to HTML entities by the following line:

```
$postedValue = ➥
htmlspecialchars➥
( stripslashes( $value ) ) ;
```

That means all of your tags lose their special meaning. For example, the less-than character (<) is converted to *<*. This is done so that you can see the HTML code that was produced. Otherwise, the code is interpreted by your browser.

Although this is good as a demonstration and good for testing, I find no real value in displaying the HTML code, particularly if your users are not tech-savvy. Instead, you can do what I do and provide the editor code as a preview. You can create a form underneath the preview with just a single button that actually saves the file.

Keep in mind that once you have saved the file, the old version is no longer available. To prevent irreparable damage to the original, I create an *archive* sub-directory in every directory. Prior to saving the file, I copy the original to the *archive* directory, adding a time stamp to the file name. Thus, if someone messes up the file, the original can be easily restored.

Also, keep in mind the environment in which the file will be displayed. In cases where the file is loaded directly, someone could add a server-side include or some PHP code to gain improper access. You should therefore parse the code before you save it.

Saving the file at this point is very simple. You have the filename in the *$filename* variable and the contents in *$FCKeditor1*. Use the appropriate PHP function to save the file.

Configuring the Editor

You can configure the appearance and functionality of the editor to suit your specific site. For example, you can use your own style sheet to change the appearance. By default, the editor uses the style sheet *_samples/sample.css*, which is accessed in the sample files via the relative path *../sample.css*.

In my case, I felt that the default toolbar with all of the wonderful features was just too much for my users. The goal for us was to provide a simple way of editing files online, so we needed a way of limiting what was available. By default, FCKEditor provides two different tool sets: Default and Basic. The Basic tool set only provides a half-dozen options, so that was really too few. We needed something in the middle.

Fortunately, FCKEditor provides a very simple way of changing which tools are available. In the *fckconfig.js* file, you will find that the tool bars are defined as follows:

```
FCKConfig.ToolbarSets➥
["Basic"] =  [ ... ];
```

The simplest way to create your own toolbar is to copy an existing toolbar, rename it accordingly, and then add or remove tools as appropriate. Within the square brackets, you will see the features that are provided by this specific toolbar. By including a new set of square brackets, you insert a separator between the toolsets (don't forget the comma between the tool sets). By including '-' *within* a tool set, you insert a somewhat smaller separator.

Conclusion

Although FCKEditor provides many more features, this article should provide you with enough of the basics to get started. By digging through the various configuration files, you will find many options for changing the FCKEditor to suit your needs. ■

INFO

[1] FCKEditor homepage: *www.fckeditor.net*

THE AUTHOR

James Mohr is responsible for the monitoring of several datacenters for a business solutions provider in Coburg, Germany. In addition to running the Linux Tutorial web site *http://www.linux-tutorial.info*, James is the author of several books and dozens of articles on a wide range of topics.