Basic PHP Example Documentation

This article provides information on how to integrate EditLive! into a Web page using PHP and JavaScript.
The complete source code for this example can be found in the *EDITLIVE_INSTALL/webfolder/php/basic/ directory*, where *EDITLIVE_INSTALL* is the location that the EditLive! SDK has been installed.

Getting Started

Required Skills

The following skills are required prior to working with this example:

- Basic client-side JavaScript
- Basic PHP

Overview

In this sample EditLive! is embedded into a Web page using PHP and JavaScript.

This example demonstrates how to perform the following with EditLive! and PHP:

- Embed an instance of EditLive! in a Web page using JavaScript.
- Invoke methods and set parameters effecting the appearance of EditLive!.
- Load a document into EditLive! from a PHP variable.

Integrating EditLive!

To embed EditLive! within a Web page several steps are required. Each of these steps is explained here and code samples are provided.

1. Include the editlivejava.js file

```
<script src="../../redistributables/editlivejava/editlivejava.js">
```

The editlivejava.js file contains the Ephox EditLive! JavaScript library. This library provides the interface between the browser and the EditLive! .jar file (editlivejava.jar) which contains the code for the EditLive! applet. The JavaScript library file can be found in the *EDITLIVE_INSTALL/redistributables /editlivejava* directory of the EditLive! install.

2. Declare the EditLive! JavaScript object.

```
<script language="JavaScript"> var editliveInstance;
```

3. Create a new instance of the EditLive! object. When creating the EditLive! object, the name of the form field for the applet, in addition to the width and height, are declared. In this example, the form field for the applet is ELApplet1, the width of the applet is 700 pixels, and the height is 600 pixels.

```
// Create a new EditLive! instance with the name
// "ELApplet1", a height of 600 pixels and a width of 700 pixels.
editliveInstance = new EditLiveJava("ELApplet1", 700, 600);
```

4. Set the path to the source files for EditLive!. These can be found in the *INSTALL_HOME/webfolder/redistributables/editlivejava* directory.

```
// This sets a relative path to the directory where
// the EditLive! redistributables can be
// found e.g. editlivejava.jar
editliveInstance.setDownloadDirectory( "../../redistributables/editlivejava");
```

5. Load the EditLive! configuration file into a string and set the configuration text.

```
<?
   //load the XML file into the string "$xmlConfig"
   // this helps to speed up the ELJ load time

   $filename = "sample_eljconfig.xml";
   $fd = fopen($filename,"r");
   $xmlConfig = fread ($fd, filesize($filename));
   fclose ($fd);
?>

editliveInstance.setConfigurationText("<?=rawurlencode($xmlConfig)?>");
```

Note: the PHP function rawurlencode must be used to output the variable. For more information, see the article on Encoding Content for Use with EditLive!.

6. Set the content for the applet via the setBody Method (the setDocument Method may also be used). The content must be URL encoded. The following example will first set the variable $pageContent to "* Document contents of EditLive! *" and then set the content of EditLive! to the contents of the variable $pageContent.

```
// This sets the initial content to be displayed within
// EditLive!
<?php
   $pageContent = "<p>Document contents of EditLive!</p>"
?>

editliveInstance.setBody("<?=rawurlencode($pageContent)?>");
```

The $pageContent variable can contain any HTML fragment, for example content loaded from a database. Note that rawurlencode must be used to output the variable. For more information see the article on Encoding Content for Use with EditLive!.

7. Display the EditLive! applet and close the script element.

```
   // .show is the final call and instructs the JavaScript
   // library (editlivejava.js) to insert a new EditLive!
   // at the this location.
   editliveInstance.show();
</script>
```

This section of code creates an instance of EditLive! within the page and sets properties which affect how EditLive! will be presented within the page. For more information on each of these load-time properties (the JavaScript constructor, setConfigurationFile, setBody, and show), see the EditLive! Load Time Methods. After each of the properties have been set the show Method is called. This method causes the instance of EditLive! to be displayed in the Web page.