Basic ColdFusion Example Documentation

This section of the document provides information on how to integrate EditLive! into a Web page using ColdFusion and JavaScript. The complete source code for this example can be found in the *EDITLIVE_INSTALL/webfolder/coldfusion/basic/* folder, 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 ColdFusion

Overview

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

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

- 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 ColdFusion 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"> </script>
```

The *editlivejava.js* file contains the Tiny EditLive! JavaScript library. This library provides the interface between the browser and the EditLive! .jar file (*editliv ejava.jar*) which contains the code for the EditLive! applet. The JavaScript library file can be found in the *EDITLIVE_INSTALL/redistributables/editlivejava* directory, where *EDITLIVE_INSTALL* is where the EditLive! SDK was installed.

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 *EDITLIVE_INSTALL/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 configuration file on the server to speed up loading ---> <cfset configpath=ExpandPath
("sample_eljconfig.xml")> <cffile action="read" file="#configpath#" variable="xmlConfig"> <cfoutput>
editliveInstance.setConfigurationText("#URLEncodedFormat(xmlConfig)#"); </cfoutput>
```

URLEncodedFormat 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 contents to "* Document contents of EditLive! *" and then set the content of EditLive! to the contents of the variable contents.

```
<cfset contents="<p>This is the initial source</p>">
<cfoutput>
   editliveInstance.setBody("#URLEncodedFormat(contents)#");
</cfoutput>
```

The contents variable can contain any HTML fragment - for example, content loaded from a database. Note that URLEncodedFormat 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 the load-time properties here (the JavaScript Constructor, setConfigurationFile Method, setBody Method, and show Method), see the EditLive! Load Time Methods article. After each of the properties have been set the show property is invoked. This property causes the instance of EditLive! to be displayed in the Web page.