Capturing Content Before Submit Tutorial

Getting Started

Required Skills

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

- Basic client-side JavaScript

Required Tutorials Completed

The following tutorials are required to be undertaken before attempting this tutorial:

- Instantiation Tutorial
- Setting the Document in the Applet Tutorial
- Getting the Document in the Applet Tutorial

Tutorial

Step 1. Simulating a Simple Content Editing Interface

This tutorial simulates integrating EditLive! into the editing interface of a content management system.

The following code represents a simple interface for a content management system. The webpage provides a textarea allowing an author to write some HTML content, and a Submit button to submit the content to a server-side script (myScript.asp in this example).

```
<html>
	<body>
		<form name="exampleForm" action="myScript.asp" method="POST">
			<p><textarea id="documentContents" cols="80" rows="5"></textarea></p>
			<p><input type="submit" value="Submit the HTML Form"/></p>
		</form>
	</body>
</html>
```

This tutorial will extend this page to integrate EditLive! as the editing interface.

Save this webpage as *capturingSubmit.html*

Step 2. Create an Instance of EditLive! in a Webpage and Set the Document

As shown in the Setting the Document in the Applet Tutorial, create an instance of EditLive! in a webpage and set the Document.

```
<html>
	<body>
		<form name="exampleForm" action="myScript.asp" method="POST">
			<p><textarea id="documentContents" cols="80" rows="5"></textarea></p>
			<script src="../../redistributables/editlivejava/editlivejava.js" language="JavaScript"><
/script>
			<script>
				var editlivejava = new EditLiveJava("ELJApplet", "700", "400");
				editlivejava.setConfigurationFile("../../redistributables/editlivejava
/sample_eljconfig.xml");
				editlivejava.setBody(encodeURIComponent("<p>Original <i>HTML</i> loaded into
EditLive!</p>"));
				editlivejava.show();
			</script>
			<p><input type="submit" value="Submit the HTML Form"/></p>
		</form>
	</body>
</html>
```

Step 3. Set the AutoSubmit Property for EditLive! to False

For this tutorial, we're assuming the setAutoSubmit Method for EditLive! causes problems with our submit architecture. So, this property should be set to *false*.

```
<html>
	<body>
		<form name="exampleForm" action="myScript.asp" method="POST">
			<p><textarea id="documentContents" cols="80" rows="5"></textarea></p>
			<script src="../../redistributables/editlivejava/editlivejava.js" language="JavaScript"><
/script>
			<script>
				var editlivejava = new EditLiveJava("ELJApplet", "700", "400");
				editlivejava.setConfigurationFile("../../redistributables/editlivejava
/sample_eljconfig.xml");
				editlivejava.setBody(encodeURIComponent("<p>Original <i>HTML</i> loaded into
EditLive!</p>"));
				editlivejava.setAutoSubmit(false);
				editlivejava.show();
			</script>
			<p><input type="submit" value="Submit the HTML Form"/></p>
		</form>
	</body>
</html>
```

Step 4. Add an onsubmit Method to the HTML Form

In order to extract the contents of EditLive! when the user presses the *Submit the HTML Form* button, add a call to a Javascript method for the HTML form's **onsubmit** method.

```
<html>
	<body>
		<form name="exampleForm" action="myScript.asp" method="POST" onsubmit="assignFields()">
			<p><textarea id="documentContents" cols="80" rows="5"></textarea></p>
			<script src="../../redistributables/editlivejava/editlivejava.js" language="JavaScript"><
/script>
			<script>
				var editlivejava = new EditLiveJava("ELJApplet", "700", "400");
				editlivejava.setConfigurationFile("../../redistributables/editlivejava
/sample_eljconfig.xml");
				editlivejava.setBody(encodeURIComponent("<p>Original <i>HTML</i> loaded into
EditLive!</p>"));
				editlivejava.setAutoSubmit(false);
				editlivejava.show();
			</script>
			<p><input type="submit" value="Submit the HTML Form"/></p>
		</form>
	</body>
</html>
```

Step 5. Write the onsubmit Method

The **onsubmit** method for HTML form needs to call the getDocument Method for EditLive!.

```
<html>
	<body>
		<form name="exampleForm" action="myScript.asp" method="POST" onsubmit="assignFields()">
			<p><textarea id="documentContents" cols="80" rows="5"></textarea></p>
			<script src="../../redistributables/editlivejava/editlivejava.js" language="JavaScript"><
/script>
			<script>
				var editlivejava = new EditLiveJava("ELJApplet", "700", "400");
				editlivejava.setConfigurationFile("../../redistributables/editlivejava
/sample_eljconfig.xml");
				editlivejava.setBody(encodeURIComponent("<p>Original <i>HTML</i> loaded into
EditLive!</p>"));
				editlivejava.setAutoSubmit(false);
				editlivejava.show();

				/** Function tells EditLive! to send it's HTML Document to the
getEditLiveDocument function.
				 */
				function assignFields() {
					editlive.getDocument('getEditLiveDocument');
				}
			</script>
			<p><input type="submit" value="Submit the HTML Form"/></p>
		</form>
	</body>
</html>
```


The GetDocument property of EditLive! is an asynchronous callback property. This means that the GetDocument method takes an unknown time interval to call the specified method passes as a parameter. Furthermore, any code following the call to GetDocument will be called during this time.

In the code above, after the call to GetDocument, the assignFields() methods ends. This would then cause the HTML form to submit its contents to the myScript.asp server-side script.

In order to ensure the HTML form does not post until the *getEditLiveDocument* method is passed the contents of EditLive!, we need to ensure the *assignFields* method is treated like a boolean and returns *false*.

```
<html>
	<body>
		<form name="exampleForm" action="myScript.asp" method="POST" onsubmit="return assignFields()">
			<p><textarea id="documentContents" cols="80" rows="5"></textarea></p>
			<script src="../../redistributables/editlivejava/editlivejava.js" language="JavaScript"><
/script>
			<script>
				var editlivejava = new EditLiveJava("ELJApplet", "700", "400");
				editlivejava.setConfigurationFile("../../redistributables/editlivejava
/sample_eljconfig.xml");
				editlivejava.setBody(encodeURIComponent("<p>Original <i>HTML</i> loaded into
EditLive!</p>"));
				editlivejava.setAutoSubmit(false);
				editlivejava.show();

				/** Function tells EditLive! to send it's HTML Document to the
getEditLiveDocument function.
				 */
				function assignFields() {
					editlive.getDocument('getEditLiveDocument');

					// return false to cause the form to not submit.
					return false;
				}
			</script>
			<p><input type="submit" value="Submit the HTML Form"/></p>
		</form>
	</body>
</html>
```

Step 6. Write the getEditLiveDocument method

As seen in the Getting the Document in the Applet Tutorial, the contents of EditLive! need to be copied into the textarea.

Now that the submit process has been canceled, we also need to manually call the HTML form's submit method to pass the contents of the textarea to the myScript.asp server-side script.

```
<html>
	<body>
		<form name="exampleForm" action="myScript.asp" method="POST" onsubmit="return assignFields()">
			<p><textarea id="documentContents" cols="80" rows="5"></textarea></p>
			<script src="../../redistributables/editlivejava/editlivejava.js" language="JavaScript"><
/script>
			<script>
				var editlivejava = new EditLiveJava("ELJApplet", "700", "400");
				editlivejava.setConfigurationFile("../../redistributables/editlivejava
/sample_eljconfig.xml");
				editlivejava.setBody(encodeURIComponent("<p>Original <i>HTML</i> loaded into
EditLive!</p>"));
				editlivejava.setAutoSubmit(false);
				editlivejava.show();

				/** Function tells EditLive! to send it's HTML Document to the
getEditLiveDocument function.
				 */
				function assignFields() {
					editlive.getDocument('getEditLiveDocument');

					// return false to cause the form to not submit.
					return false;
				}

				function getEditLiveDocument(src) {
					// copying source from EditLive! into textarea
					document.exampleForm.documentContents.value = src;

					// manually trigger the form submission
					document.exampleForm.submit();
				}
			</script>
			<p><input type="submit" value="Submit the HTML Form"/></p>
		</form>
	</body>
</html>
```