



<http://www.flexcapacitor.com>
<http://www.drumbeatinsight.com>

The HTML component allows you to display HTML in your Flex application. Add the component to the stage in Flex Builder and specify the source URL or HTML content and the component displays your HTML. You can also use the rich text editor features of FCKEditor with your Flex applications.

I do not like reading manuals. But I have learned from experience that sometimes I have to. I wrote this manual for people like me. Since this is a special component there are a few things you need to know to get it to work properly. I have put that information in here.

If any steps or information is missing please contact us at the site above.

Requirements

Flex Builder 2.0.1 or Flex SDK 2.0.1

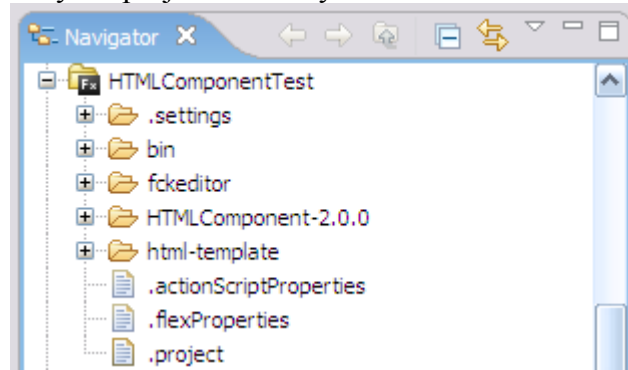
Note: If you enjoy this component please consider supporting [Drumbeat Insight](http://www.drumbeatinsight.com) to help pay for development costs. Source is available.

Installation

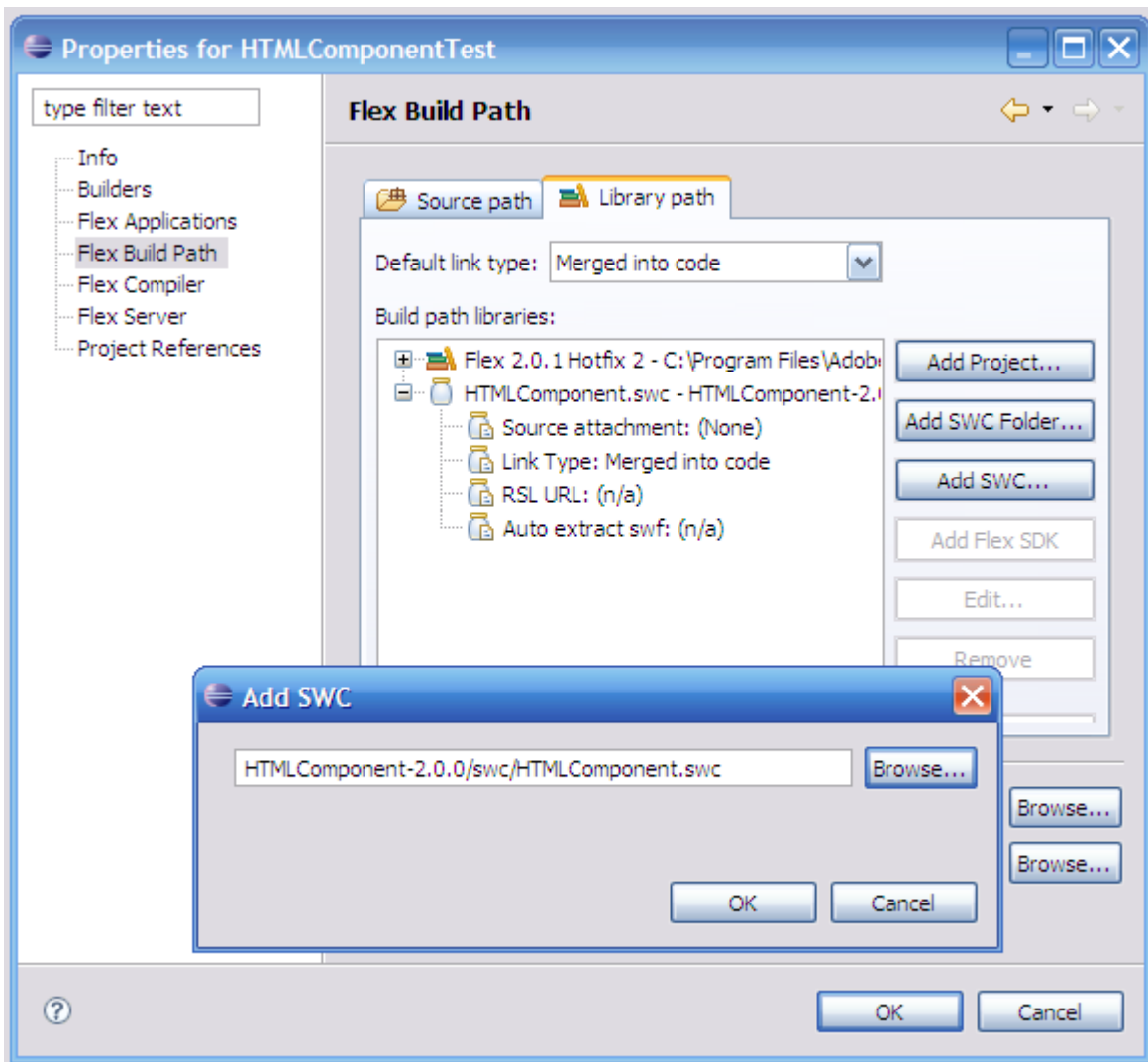
Please follow the next to steps *exactly* to make the HTML Component work in your project.

For Flex Builder:

1. Unzip the download to your project directory.

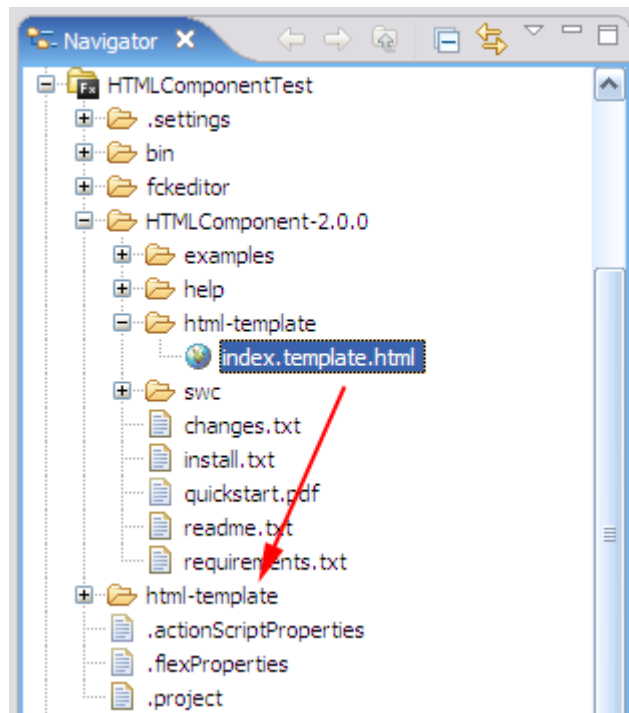


2. Open the project properties by right clicking on your project and clicking properties.
3. Goto Flex Build Path > Library Path tab and click "Add SWC".
In the dialog add the path directly to the HTMLComponent.swc file. It will look like, "HTMLComponent-2.0.0/swc/HTMLComponent.swc".

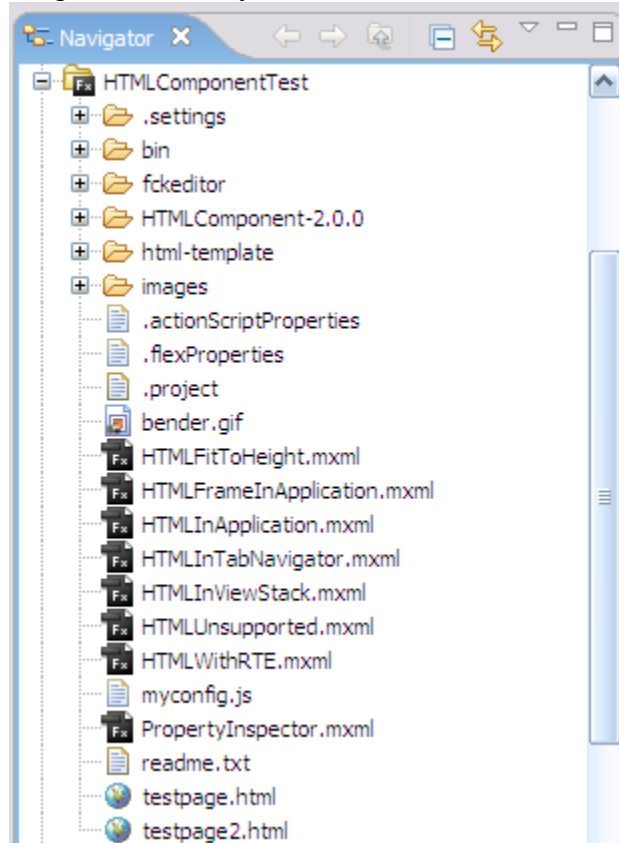


NOTE: If you have the source code you do not need to add this component to your project.

4. Close out the dialogs.
5. In the unzipped html-template directory copy the "index-template.html" file into your project's html-template directory.



6. Copy the files in the examples directory to the root of your project. Run the examples in Firefox. If you run them in Internet Explorer some examples will need to be uploaded to a server due to Internet Explorers security model. See notes below for more information.



7. Read the rest of this document. 5 more minutes.

Usage

To use the HTML component drag the component from the Component Panel (under the category Flexcapacitor) to the stage. Set the size and location according to your needs and set the source path to an HTML page or define the HTML markup in the htmlText property. Define iframe or division respectively.

There are two methods to display HTML content in your Flex application. The first method is to use the browser to render content of a URL (can be any valid URL, ie html, php, jsp, etc) in an iframe. The second method is to use the browser to render the HTML code you define in the htmlText property inside a div element. Both methods render use the browser to render an HTML element and that element is positioned, sized and responds as if it were a Flex component in your application.

NEW! You can also display the a rich text editor by setting the elementType to "editor".

Each of these methods require following the instructions below and taking note of the important messages below this component to work.

Method 1 – Set Source (in an iframe)

In the Flex Properties view enter the URL you wish to display in the "source" property as shown below. Set the elementType property to "iframe". Test your project.

```
<fc:HTML x="10" y="20" source="http://www.google.com" elementType="iframe"/>
```

See example *HTMLFrameInApplication.mxml* for a working example.

Method 2 – Display HTML Code (in a div)

After adding the HTML component to the stage go into source view and add the htmlText property as a child tag to the HTML component. Enter your HTML code as shown below. Set the "elementType" property to "division". Test your project.

```
<fc:HTML x="33" y="37" elementType="division">
  <fc:htmlText>
    <![CDATA[<h1>Here is an H1 Element</h1>
      <input name="myInput" id="myInput" size="41" title="Search" type="text">
      <input name="btn1" value="Search" type="submit"><br/><br/>
      <form><input type="submit" value="Reload"></form>
      
    ]]>
  </fc:htmlText>
</fc:HTML>
```

See example *HTMLInApplication.mxml* for a working example.

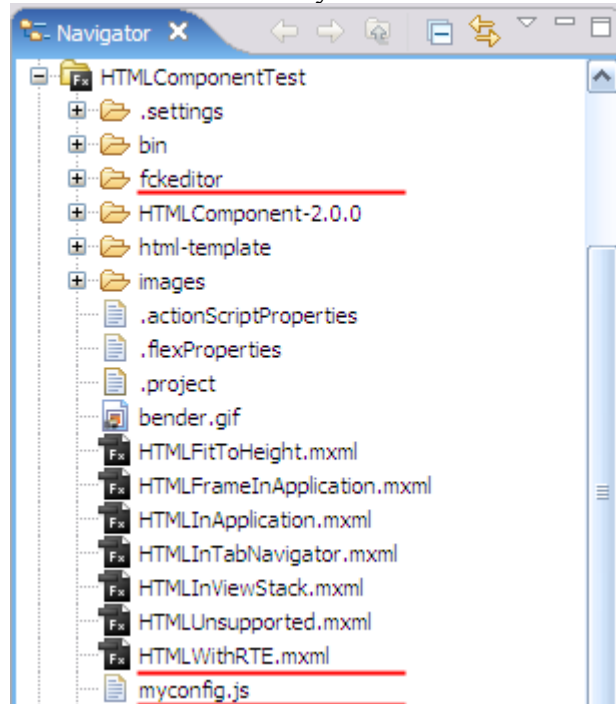
Method 3 – Display a Rich Text Editor (in a div)

After adding the HTML component to the stage go into source view and add the htmlText property as a

child tag to the HTML component. Enter your HTML code you would like to edit in the `htmlText` property as shown below. Set the "elementType" property to "editor".

Complete by following these steps:

1. Download FCKEditor from <http://www.fckeditor.net/>. (This example is using 2.4.2)
2. Unzip the download to a directory in your project called "fckeditor"
3. Set the `editorPath` to the "fckeditor" directory



4. Set the `configPath` to your custom fckeditor config file (if one exists). We should store this outside of the fckeditor directory.

```
<fc:HTML id="rte" elementType="editor"
  borderStyle="solid" borderThickness="4" borderColor="#605669"
  top="180" bottom="20" right="20" left="20"
  configPath="../../myconfig.js"
  editorPath="fckeditor/">
  <fc:htmlText>
    <![CDATA[<h1>Lorem ipsum dolor</h1>
    Lorem ipsum dolor sit amet, consectetur adipiscing elit.
    ]]>
  </fc:htmlText>
</fc:HTML>
```

Test your project.

You can set the options of the FCKEditor by using a custom `config.js` file. One is included in the examples and more information can be found on the <http://www.fckeditor.net/> website.

To get the HTML that the user has edited use the `getHTML()` method. To set the HTML of the editor set the `htmlText` property or call `setHTML()`.

See example HTMLWithRTE.xml for a working example.

A note about important notes. These notes are important to read and follow because we are communicating with the browser to produce HTML for us. Doing so we have to take into account additional requirements.

IMPORTANT NOTE – VIEWING IN BROWSER

When you are testing the source property in Internet Explorer you will not be able to view local html files from the filesystem because of Internet Explorer's security policy. If you try you will not see anything. You can get around this by viewing the application in another browser like Firefox, on view on localhost or a remote server. For example, <file://myhtmlpage.html> will not work in IE but <http://localhost/myhtmlpage.html> will.

IMPORTANT NOTE – HTML TEXT PROPERTY

Be sure to add CDATA delimiters around your htmlText. These are not added by default when you use the htmlText property because I don't know how to do it. Sorry, sue me. But what you can do is nest the `<mx:Script>` tag inside the htmlText tag, it will automatically add the CDATA tags and then you can erase the begin and end Script tags.

IMPORTANT NOTE – HTML TEXT PROPERTY

When using the htmlText property in the HTML component be sure to use the correct namespace. IE, do not use `mx:htmlText`, use `namespace:htmlText`, like `fc:htmlText`.

IMPORTANT NOTE – INSIDE CONTAINER

When using the HTML component inside of a container you need to make sure the container or component is the min-width of the HTML Component. You can also do this by setting the container height to 100% or the HTML Component height to 100%. The reason is that when the container that holds the HTML component is smaller than the HTML Component then duplicate scroll bars appear and HTML content can overlap. You may never experience this problem but be aware of it when using containers.

IMPORTANT NOTE – RICH TEXT EDITOR

When using the HTML Component to display a rich text editor certain browsers will not render the rich text editor at all or without the images when viewed on the local file system. This because of browser security mechanisms. Please view the same page on a local or remote server to see it properly.

IMPORTANT NOTE – VIEWING IN BROWSER

If you do not see anything rendered in the browser make sure you have copied all the files you need including the custom html-template that you installed in the Installation section and the normal .js files Flex uses. Test in a different browser and on the server. Test that the default applications work. The best advice – install [Firebug](#) and test in Firefox (click the errors link in the bottom right hand corner). It will show you a list of errors for you to debug. Most of these errors should be documented on <http://www.judahfrangipane.com>.

Features

The HTML component has many features to make your life easier. You can treat the HTML component as you would any other UIComponent. You can move it, resize it, show it, hide it, set the source to a different url, set the html text to different html text content, set the scroll policy and add borders and programmatic skins. You can even automatically resize the component to fit the height of the HTML content! SWEET!!! You can also provide alternate HTML content or link or redirect to another page when a lame browser does not support the HTML component. Note: most modern browsers are supported.

One of the features mentioned is **Fit to Content Height** (`fitToContentHeight`). This feature, when enabled will check the height of the HTML content (a page or html text) and resize the component to fit, thus removing the *HTML* scrollbars. This is nice if you are for example displaying a post of unknown size from Wordpress but do not want to make the user scroll a small little box where your content is. Occasionally you may need to offset the height by adding a few pixels. You can offset this value by setting the `fitToHeightOffset`. Usually 1 or 2 pixels should suffice.

Note: You cannot get the height of a URL that is not on your domain. Don't even think of trying. Its impossible. Not even Neo from the Matrix can do it. This is due to cross domain security restrictions in the browser. What is an example you ask? Well I'm glad you asked. If you set the source of your HTML Component to <http://www.google.com> and you set the `fitToContentHeight` property to true the height will return -1 and a `heightFault` event will be generated. Why? Because unless your page is hosted on google.com you cannot *cross script* into their domain. You *can* check the height of HTML text content and pages hosted on the same domain as the html wrapper (not the domain of the swf but the domain of the HTML page the swf is in). So if your HTML wrapper page is www.mysite.com/mypage.html and you set the HTML component source to www.mysite.com/myotherpage.html or "myotherpage.html" (using a relative url) you *can* get the height of the page. These are security restrictions of the browser and not Flash or the HTML component.

Failing the HTML Component

Not all browsers support the use of the HTML component. You can gracefully bail on these browsers by using the `externalInterfaceFunction` function. The component automatically fails silently (no runtime error) when no JavaScript connection can be made. The `htmlText` if available is rendered to the best of the `TextArea` components ability. That is not always good enough though. Using this function allows you to handle it anyway *you* want. You have the power. You're the boss of it.

Note: You can also use this function to change or modify the `htmlText` before it is displayed.

The `externalInterfaceFunction` function passes in one argument. That argument is a boolean value that indicates if JavaScript communication is available. If it is not available then the HTML content, the value of the `htmlText` property, is rendered by the `TextArea.htmlText` field. This property supports some but not all HTML tags and does not render anything if you displaying the contents of a URL via the `source` property. If this is not the desired behavior you can set the `HTML.htmlText` property to a new value. Remember, you're in charge of what is displayed. An example is shown below.

```
<fc:HTML elementType="division" id="html1"
externalInterfaceFunction="{checkBrowserSupport2}" x="327" y="223">
```

```
private function checkBrowserSupport(available:Boolean):Boolean {
    var status:String = available ? "available" : "not available";
    //trace("Flash reports JavaScript communication is " + status);
    html1.htmlText = "The browser you have sucks. This feature is not
supported.";
    return false;
}
```

See example HTMLUnsupported.mxml for a working example.

Or you could be nicer and show them the HTML content in a new window. It is up to you decide what to do if anything.

```
private function checkBrowserSupport2(available:Boolean):Boolean {
    var status:String = available ? "available" : "not available";
    //trace("Flash html2 reports JavaScript communication is " + status);
    html2.htmlText = "The browser you have does not support viewing HTML content
on top of Flash content. ";
    html2.htmlText += "<br/><br/>To view the HTML content click <font
color='#0000ff'>";
    // we create a dynamic or static link to the content. could be an php or jsp
page
    // it is up to you how you point to the html content
    html2.htmlText += "<a href='somepage.html?whatever=1'
target='_blank'>here</a></font>...";
    return false;
}
```

See example HTMLUnsupported.mxml for a working example.

Properties, Methods and Events

For api documentation open the index.html page in the help directory of the unzipped folder.

For more information visit <http://www.drumbeatinsight.com>

@Copyright 2000+ Drumbeat Insight.