# Configuring Flare Projects for Tooltip Help

Skittles Help is a form of Tooltip Help. When a user clicks, it opens this help on desktop browsers. This document is an [update to this article](http://www.madcapsoftware.com/blog/2012/01/09/using-flare-to-build-field-level-context-sensitive-help/), with further info and testing.

I am creating a sample help project in Flare and following these steps. Some of these steps only have to be completed once (like creating a skin). Others will need to be done for every created topic (like adding an Alias ID).

[Here is a guide](http://docs.madcapsoftware.com/FlareV10/FlareCSHGuide.pdf) to Flare Context-Sensitive Help (CSH).

Another helpful guide: [Snippets](http://docs.madcapsoftware.com/FlareV9/FlareSnippetsGuide.pdf) in Flare used for content that is replicated through a system.

**IMPORTANT:** If you import the Volusion Flare project, don’t worry about any of this! All of that work is done and ready to go. Continue to the creating topics doc.

The overall process includes:

* **Step 1: Creating the folder for TooltipHelp**. All of your topics will live here. Putting them into further folders should not be a problem, but can be determined later.
* **Step 2: Creating a master page for the TooltipHelp**. You use this to create a new topic. This sets up the styles and structure.
* **Step 3: Creating a skin** for the topics to load into. This controls the look and feel when opened on a click of link.
* **Step 4: Creating an Alias file** for the developer to link to topics. Assigning Alias IDs to help content. NOTE: Alias is used for all context sensitive help, including the Help for this page option.
* **Step 5: Testing the CSH**. You can test the links and samples at any time using one of many skins you create.
* **Step 6: Hand off to Dev** with header files of IDs. You don’t have to document all IDs in excel. Flare does it for you.

## Step 1: Creating the Folder

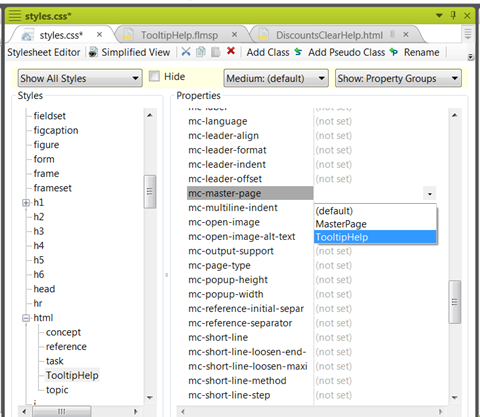
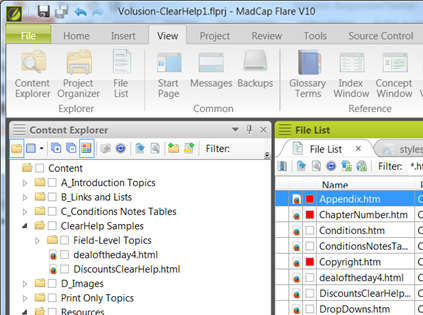
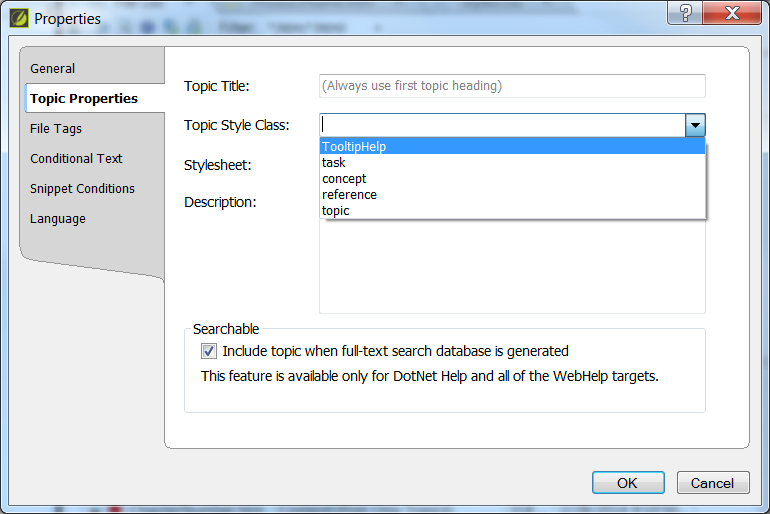
1. Create a folder in the content Explorer called TooltipHelp (for example).
2. Create a topic to test as you go.

It may be useful to keep these separate from existing topics or other field-level topics by storing them in a separate folder.  You can style them differently than the rest of your content, exclude them from the search database so they do not appear in search results, etc.

## Step 2: Creating the Master Page

The master page determines the component pieces within your created topic, like a template.

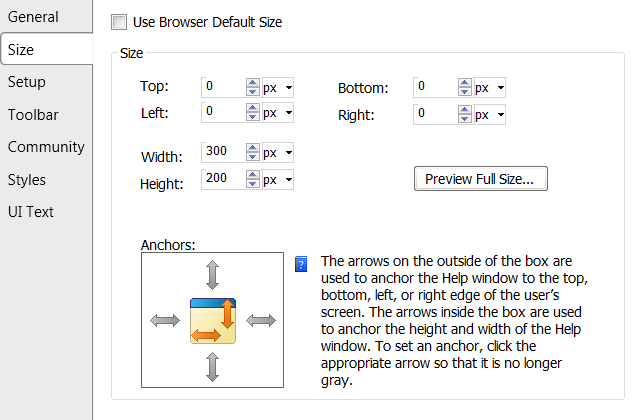
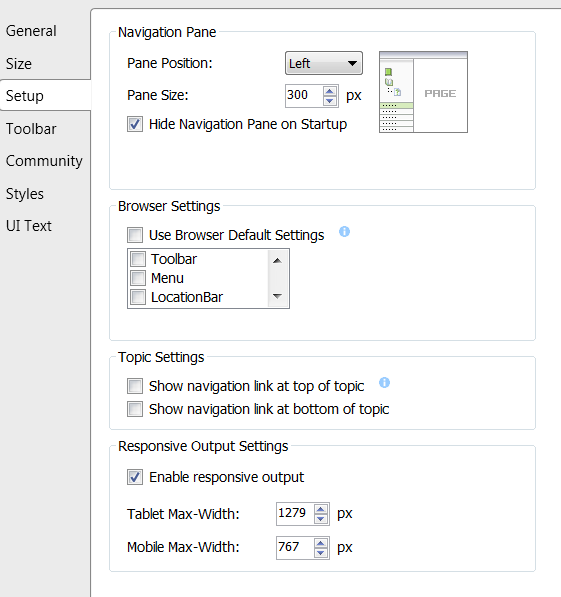
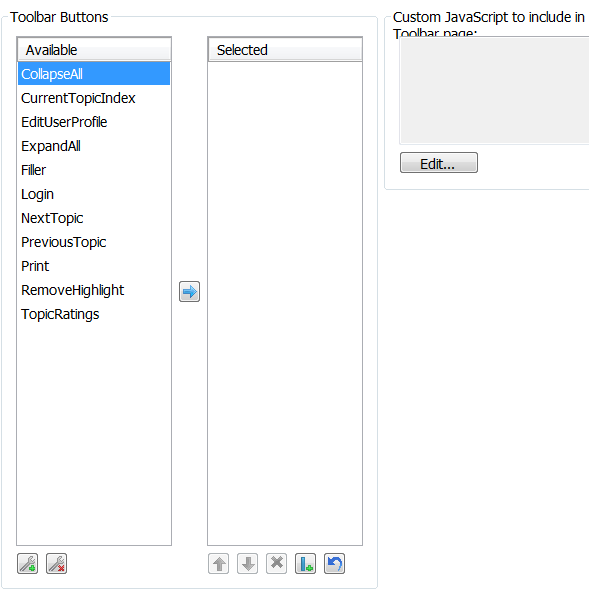
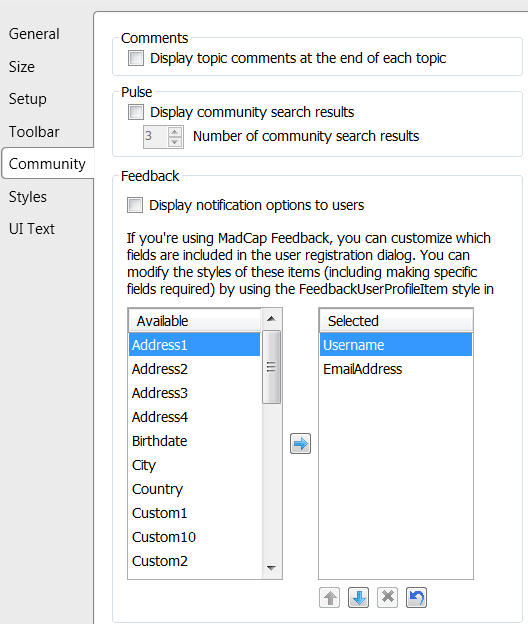
More info on Master Pages [here](http://webhelp.madcapsoftware.com/flare10/Default.htm#Master_Pages/About_Master_Pages.htm?Highlight=master%20pages).

1. If you are using master pages (yes you will), you may not want the field-level topics to have breadcrumbs, headers, footers, topics toolbars, etc.  It is a good idea to create a new master page and apply it to just these topics.
   1. Create a new master page and name it something meaningful: Project > Add Master Page > Choose ‘MasterPage.flmsp’. In this sample it was named **TooltipHelp**.
   2. In the master page editor, remove every proxy except the “topic body proxy”. Save. You can always go back, add more style components, etc.   
      Master Page
2. Open the stylesheet from the Content Explorer and find the HTML style, select it.  Right-click it and create a class called **TooltipHelp**. In the **Unclassified** section, find mc-master-page and select then new master page (**TooltipHelp**) from the drop-down. Save.  
   
3. Now you can apply the **TooltipHelp** class to field-level topics.  At this point you can also remove them from the search database too.
   1. Open the **File List**: View > File List  
      
   2. Select all of the topics that are in the TooltipHelp folder, right-click and select **Properties**. An easier way to locate them is to sort by path.
   3. In the **Properties** dialog, click the **Topic Properties** tab. For the **Topic Style Class**, select **TooltipHelp**. Click **OK** to update.  
      

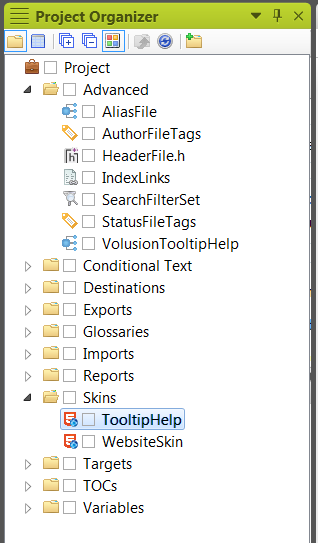
## Step 3: Creating the Skin

Skins are the wrappers for loading web helps and pages. These skins are associated to the pages, and used to load when links are selected.

Learn more about Skins [here](http://docs.madcapsoftware.com/FlareV9/FlareSkinsGuide.pdf).

1. Create a skin for the the field-level Help calls: **Project > New > Add Skin…** Name it “**TooltipHelp**“.
2. Set the following settings in the new skin. Note: some of these are optional and may or may not work for you, depending on how your developers define the popup.  These are the settings that I used for a simple JavaScript popup.
   1. **General tab:** Only check the “TOC” option in the Features section.  All of the other settings in this section should be un-checked.
   2. **Size tab:** Unselect Use Browser Default Size. Set the Width to be 300 px and the Height to be 200 px.  Everything else should be “0″ or un-checked.  
      
   3. **About tab:** Un-check this option.
   4. **Setup tab:** Check the options “Hide Navigation Pane on Startup”. Un-check any of the browser settings that you do not want, most likely all of them. Turn off responsive output, or it will display the TOC slider!   
      
   5. **Toolbar tab:** For the Selected area, click the first option and the Delete X to move it back to Available. This removes any toolbars for this topic.  
      
   6. **Community tab:** Unselect all options!   
      
   7. **Styles tab:** This section controls all visible styles used on these pages. For example, you may want to strip the company logo.
   8. **Save** the skin.

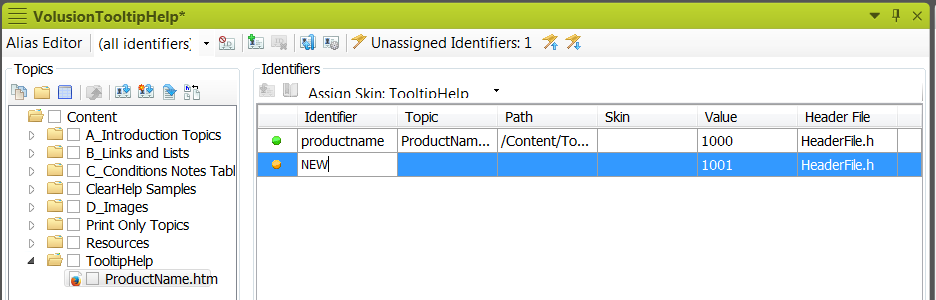
You can find your skin files in the Project Organizer > Skins:



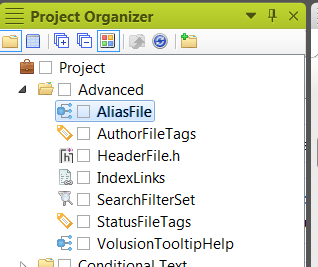
## Step 4: Creating the Alias File

You will add 1 Alias file, and many Identifiers to it. These IDs are exported in Header Files for dev for their linking.

[Here is a guide](http://docs.madcapsoftware.com/FlareV10/FlareCSHGuide.pdf) to Flare Context-Sensitive Help (CSH).

1. To create the alias file: Click Project > New > Advanced > Add Alias File. Give it a meaningful name. For this example, using VolusionTooltipHelp.
2. The Alias Editor opens with a browseable list of topics on the left and identifiers on the right.
3. To create an ID, right click in the Identifiers section and select New Identifier. Or click the New Identifier button (circled). Enter a name for the Identifier. The value is the number.   
   
4. To associate a topic to the ID, browse to find the topic to link in the Topics area.
   1. If you have the new identifier selected in the Identifiers section, right click on the topic and select Assign Topic to Selected Identifier.
   2. If you need to add a new identifier, select Assign Topic to New Identifier. This will create a new alias id. The topic and path for the identifier update.
5. Also select each ID and click the Assign Skin, TooltipHelp. This will update the skin to associate. This is important for viewing on click.
6. Save when done.

You can find your Alias file in the Project Organizer > Advanced:

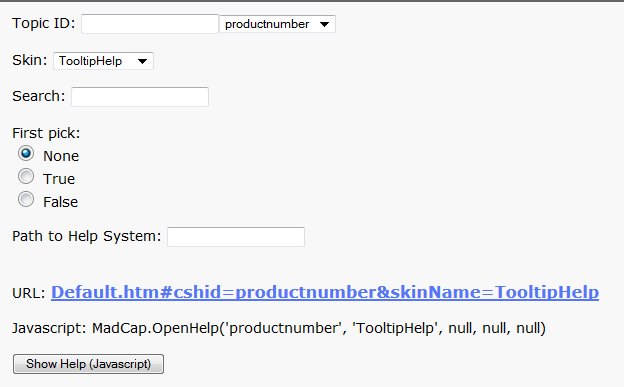


## Step 5: Testing the CSH

Before you begin testing, you will need to make sure you have Ghostscript installed for Flare: <http://www.madcapsoftware.com/downloads/ghostscript.aspx>

If you don’t have it, you will receive the following warning on compile:

You must install Ghostscript if you insert a postscript file (e.g., EPS, PS) or an equation. Ghostscript allows these elements to be converted to a supported file type in the output. Ghostscript is also used to render vector graphics in the XML Editor. To install the latest version of Ghostscript, click the link below.

1. Click Tools > Test CSH API Calls. If you need to generate a new help, it will prompt.
2. A test page opens for entering a topic id to test. Select the drop downs or type in the id to test. Select the skin TooltipHelp. The URL updates based on selections. 
3. Click Show Help (Javascription).
4. If you don’t like the output, continue tweaking the Skin, adding/updating styles.

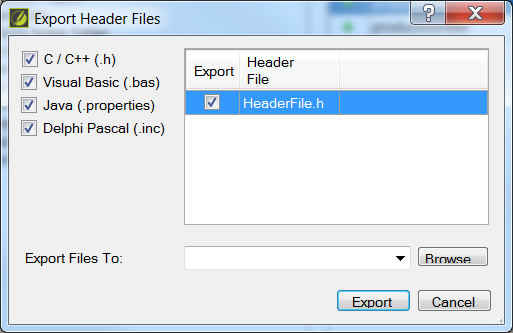
## Step 6: Hand off to Dev

Development will handle the input of IDs within the code to access the help. They can also use javascript to handle to loading of the help HTML page.

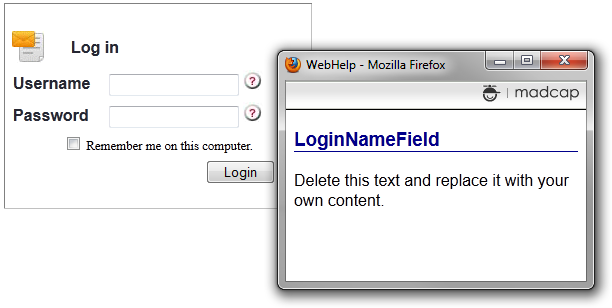
1. You will need to provide Dev with a list of the identifiers so they know which ones to use and for which fields. This is done by supplying them with the header file: **Tools > Export Header File(s)**
2. They can write a link to call the popup code and the CSH Identifier.  The following is a sample of a JavaScript popup:

<a href=”#” onclick=”return pop(‘MyWebHelp/Default\_CSH.htm#TL\_Name’)”> <img src=”Help.jpg” border=”0″></a>

Formats:



When you click on the skittle image image, a popup with the field-level Help topic will appear.

[](http://www.madcapsoftware.com/blog/wp-content/uploads/2012/01/fieldlevelHelp1.png)

*Example of help for the ? skittle for username and password.*

Developers can change the look and feel of the popup framework depending on their code and development language used for the application.   The above image just uses simple JavaScript to open a new browser window (a little more code is needed but not shown).