

Clearspace Theming Guide

The Clearspace user interface is designed to be easily customizable. Through small gestures, you can make changes large and small to the look and feel of your Clearspace instance.

Note: The customizations described here require access to the admin console, so you'll need administrator privileges.

The easiest change you can make is probably simply to replace the default Clearspace logo with your own HTML markup. As described in [Replacing the Clearspace Logo](#), you don't need a theme for that.

For larger changes, themes are the Clearspace way to make groups of look-and-feel changes to your instance. The following sections will describe how to make Clearspace look the way you want it to:

[About Themes](#)
[Tour of Customizations](#)
[Replacing the Clearspace Logo](#)
[Creating Custom CSS Classes](#)
[Customizing Page Structure](#)
[Theming Best Practices](#)
[Scope for Themes](#)

About Themes

With themes, you can define user interface customizations and associate them with parts of your Clearspace UI. A theme can capture changes to page structure and CSS styles.

A theme is made up of custom templates you create that are based on templates included with Clearspace. These templates are FTL files that use the [Freemarker template engine](#).

Note: If you find that the changes you want to make might require changes to the logic or Freemarker code in the FTL code, the Jive Software professional services team can help out.

Note that themes are intended as an easy way to make relatively ambitious changes to the UI. If you just want to change what appears above the user bar (the row of menus), you can do that without making a theme. See [Replacing the Clearspace Logo](#) for more information.

Tour of Customizations

The next few sections describe how you can customize the user interface in simple ways. After [replacing the Clearspace logo](#) (a change that doesn't require a theme), you'll see how to [override Clearspace CSS classes](#) with your own, then [change the structure of a page](#) by editing the template the page is based on.

Replacing the Clearspace Logo

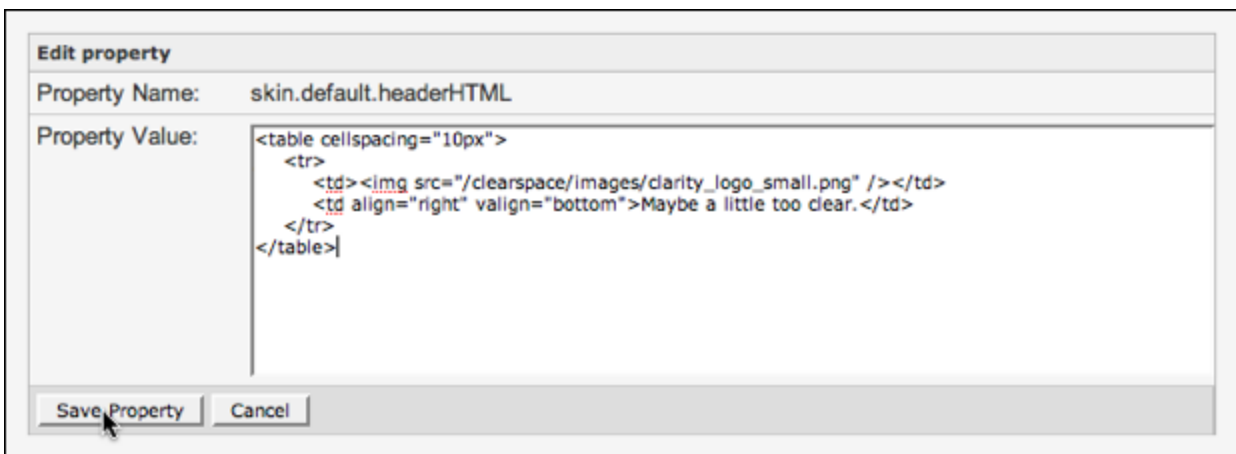
The simplest change you can make doesn't involve themes at all. If all you want to do is replace the logo at the top of Clearspace pages with something else, you can do that with a system property. Simply set the `skin.default.headerHTML` property so that its value is the HTML markup you want to appear.

Note: This property supports values up to 3500 characters.

Here are a few steps walking through the process.

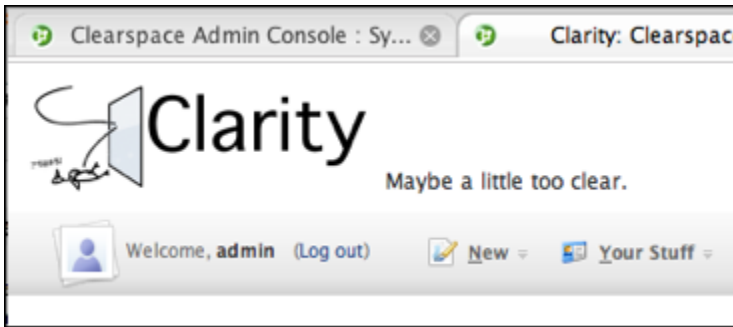
1. Log in to the admin console and navigate to System > Management > System Properties.
2. Edit the `skin.default.headerHTML` property.
 - If the property is already listed, click its edit icon to change its value.
 - If the property is not listed, scroll to the bottom of the page and enter values for the following:
 - **Property Name:** `skin.default.headerHTML`
 - **Property Value:** HTML markup you want displayed at the top of the page. If you want to link to other files, such as images, use fully-qualified URLs or a path that's relative to your application's context root. In other words, if your application's home page URL is <http://myhost:8080/mycompany>, then your file path would be something like `/mycompany/mydirectory/myfile.gif`.
3. Display your Clearspace instance in a browser to see the results. You don't need to restart Clearspace.

For example, the following sets the property so that the page displays a logo (`clarity_logo_small.png`) and tagline ("Maybe a little too clear."):



```
Property Name: skin.default.headerHTML
Property Value: <table cellspacing="10px">
  <tr>
    <td></td>
    <td align="right" valign="bottom">Maybe a little too clear.</td>
  </tr>
</table>
Save Property Cancel
```

After saving the property, simply refresh a browser displaying the Clearspace page to view the results:



Creating Custom CSS Classes

You customize the CSS of your Clearspace instance by adding or overriding CSS classes in a custom CSS FTL template file. You create a theme, create the custom template, add your CSS classes to the template, then map the theme to the UI. This section walks through the process of making a small change, but larger changes work essentially the same way. Be sure to read [Scope for Themes](#) for more information on how to map your themes.

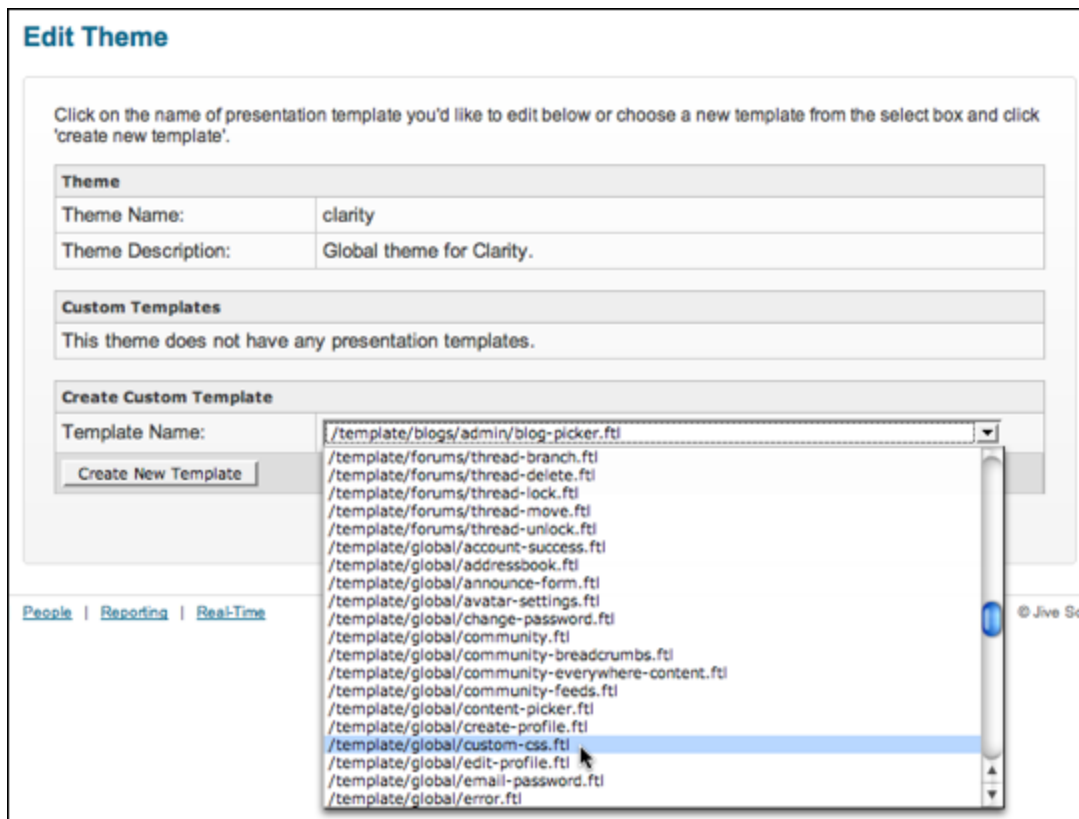
If you create *new* CSS classes, you'll also need to update page-specific FTL files to use the classes you define. See [Customizing Page Structure](#) for information.

Note: It's a good idea to plan your customizations ahead of time -- their scope as well as the details of the customizations themselves -- because how you map your theme determines the scope of the changes applied. With those plans in hand, a good way to start is by creating your theme.

1. Create a new theme. In the admin console, go to System > Settings > Themes, then click Create New Theme. Give the new theme a name and description, then click Create Theme.

| Create Theme | |
|---|--------------------------|
| Theme Name: | Clarity |
| Theme Description: | Global theme for Clarity |
| <input type="button" value="Create Theme"/> <input type="button" value="Cancel"/> | |

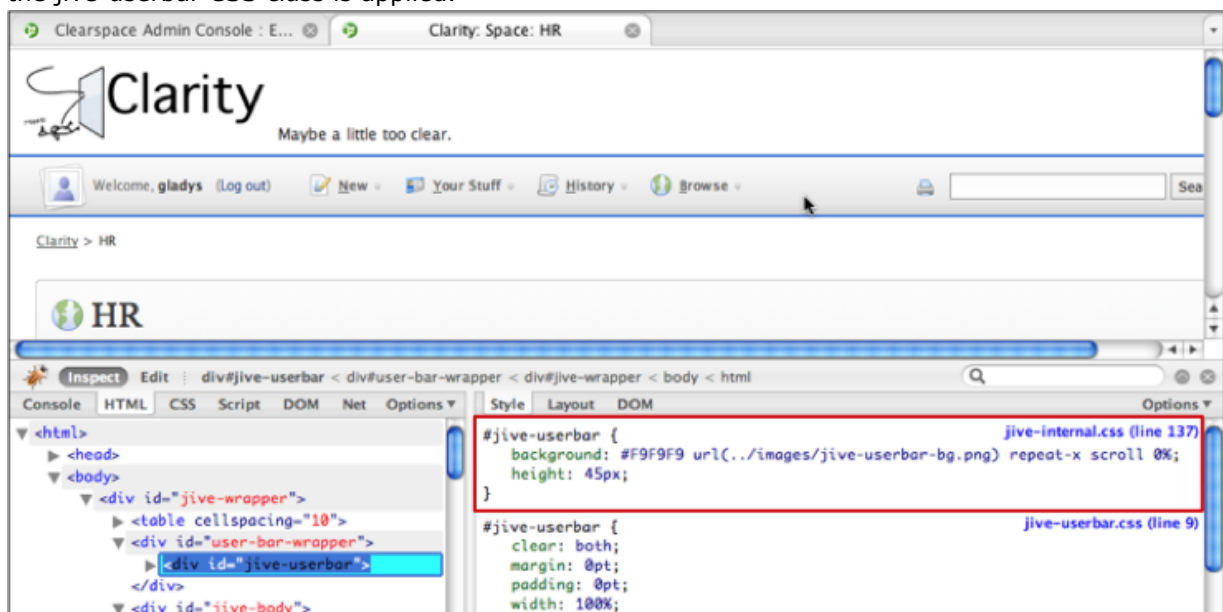
2. Begin editing the theme by clicking its edit icon.
3. Under Create Custom Template, select `/template/global/custom-css.ftl`, then click Create New Template.



This creates a custom template to which you can add CSS classes.

Note: When you save your CSS custom template, Clearspace will copy as `<jiveHome>/themes/<theme_name>/global/custom-css.ftl` in your Clearspace installation. You might find that it's easier to simply create and save the template, then open the saved template in your favorite editor.

- Using a tool such as [Firebug](#), figure out the styles you want to add or override. For example, the following illustration shows Firebug focused on the user bar (the part with the menus in it), where the `jive-userbar` CSS class is applied.



- To override the style you've identified, simply add a class of the same name to your custom CSS template. (You can even start by copying the classes from the Clearspace instance, then paste them into your custom CSS template.)

The following example shows the user bar style updated with a different color and no background

image.

Edit Theme Template

Enter the Freemarker template code into the textbox below and click 'save template' when you're finished. If your changes break the template, you always revert back to the system default by clicking the 'restore default' button or by deleting this template.



| Template | |
|-----------------|--|
| Template Name: | /template/global/custom-css.ftl |
| Template Value: | <pre>/* custom-css */ #jive-userbar{ background:#05E3EE repeat-x scroll 0%; height:45px; }</pre> |

Save Template Cancel

6. Save the template when you've added the classes you want.
7. In order to see your changes, you'll need to map the new theme to some aspect of the user interface. For example, if you want the your theme to impact UI everywhere in your Clearspace instance, you'll want to set it as the global theme, as shown below. Note that you can have only one global theme, so plan to add all of your global template changes to a single theme for global mapping.

Themes

Available Themes:

| Name | Description | Actions |
|---------|---------------------------|--|
| clarity | Global theme for Clarity. |   |

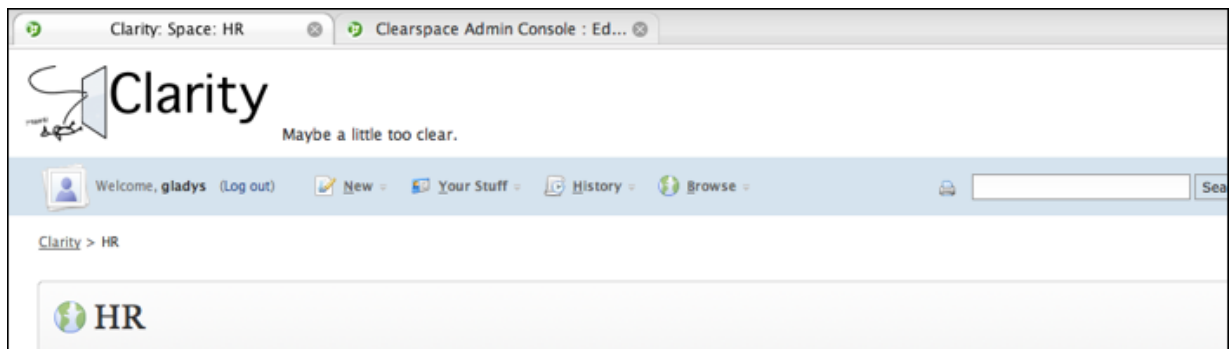
Create New Theme

Global Theme Map:

The global theme will be used throughout the application as the default theme. You can change the global theme using the form below.

clarity Set Global Theme

Here's the user bar with its background color set to blue.



Customizing Page Structure

You change the structure of a Clearspace page by customizing the Freemarker FTL template file that the page is rendered from. An FTL file defines the order of UI elements on a page (or a section of a page), along with the data displayed on the page. The custom template is part of a theme that you map to the user interface.

When adding custom templates to a theme, be sure you're picking a theme that is (or will be) mapped to the part of the UI you want to affect. For more information, see [Scope for Themes](#).

1. Identify the part of the UI that you want to customize, then identify the FTL file that corresponds to it.
2. Create or edit the theme that will contain the custom template.
3. Under Create Custom Template, select the FTL file you're going to customize, then click Create Template.

Note: As with the CSS template, Clearspace will copy your saved template to the `<jiveHome>/themes/<theme_name>` directory of your Clearspace installation. You can edit the FTL file in the admin console, of course, but you'll most likely find it easier to edit in an editor that is designed for code (such as an HTML editor).

4. Make changes to your custom template.

The following code takes a blog post title from elsewhere in the page and moves it so that the title follows the blog name.

```

<#include "/template/global/include/form-message.ftl" />

<#if (blog.userCount > 1)>
  <span class="jive-groupblog-info">
<#else>
  <span class="jive-blog-info">
</#if>
  <span class="jive-blog-info-padding">

    <h3><a href="{BlogUtils.getBlogURL(blog)}" style="text-decoration:none">${blog.name}</a>:
    <a href="{BlogUtils.getPostPermalink(post)}" style="text-decoration:none">${post.subject}</a></h3>

    <span class="jive-blog-info-details">

    <!-- <strong>${blog.blogPostCount} Posts</strong> -->

    <span class="jive-pagination">
      <span class="jive-pagination-prevnext">
        <#if post.previousPost?exists || post.nextPost?exists >

```

5. If you haven't already, map the theme that contains the custom template to the user interface. This example maps the theme to a URL in order to have it effect all blogs (which aren't contained within a community).

Custom Theme Maps:

Themes can be mapped to various application data such as Spaces and URL patterns. This enables a single Clearspace instance to display multiple skins throughout the application. Theme maps can be applied without a server restart and can be modified anytime.

| Theme | Type | Value | Delete |
|---|------|-------|--------|
| No theme maps are installed at this time. | | | |

Theme:

Map to URL:

Map to Communities:

Root Space

- R&D
- Products
- HR

You can choose multiple spaces.

Before

Clarity > Blogs > HR Blog > 2007 > April > 23



HR Blog

Vacation policy changes

Posted by [glady](#) Apr 23, 2007 0

I blogged last week on the possibility of changing our holiday schedule. Turns out that quite a few of you thought our old schedule just didn't match up with our seasonal holiday fun (I agree!). But I couldn't help noticing that a lot of the comments I got were about our vacation policy, too. In particular, folks are wondering if they can redistribute that week of vacation we offer at the end of the year.

So this week I thought I'd explain the current policy a bit and offer a suggestion or two. As always, I hope you'll chime in if you have thoughts. (Then again, I know I can count on you for that 😊).

More to come!

Tags: [vacation](#), [time_off](#), [hr_policies](#)

After

[Clarity](#) > [Blogs](#) > [HR Blog](#) > [2007](#) > [April](#) > [23](#)

HR Blog: Vacation policy changes

Posted by [glady](#) Apr 23, 2007

 0

I blogged last week on the possibility of changing our holiday schedule. Turns out that quite a few of you thought our old schedule just didn't match up with our seasonal holiday fun (I agree!). But I couldn't help noticing that a lot of the comments I got were about our vacation policy, too. In particular, folks are wondering if they can redistribute that week of vacation we offer at the end of the year.

So this week I thought I'd explain the current policy a bit and offer a suggestion or two. As always, I hope you'll chime in if you have thoughts. (Then again, I know I can count on you for that 😊).

More to come!

 Tags: [vacation](#), [time_off](#), [hr_policies](#)

Theming Best Practices

As you make your changes to the UI, try the following to make your work easy and straightforward.

Make Your Plans

Think about your changes in terms of the UI regions they'll touch, then plan your themes as groups of templates corresponding to those UI regions. For more information, see [Scope for Themes](#).

Also, make a list of the FTL files you'll want to customize for each theme. Note that this could be different custom versions of the same file for different themes.

Use the Right Tools

Use browser-based tools such as the [Firefox Web Developer](#) and [Firebug](#) plugins for the Firefox web browser. These tools are useful for figuring out which CSS classes you should edit to change a particular part of the UI. Here's a shot of Firebug displaying HTML markup and CSS style corresponding to a blog post title.



Use the admin console to create and save your templates, but edit them with your own editor (preferably one that features syntax coloring). The Clearspace admin console provides a way to edit these files, but it's pretty basic. After you save templates, you'll find them at a path like the following:

```
<jive_home>/themes/<theme_name>/<template_path>
```

For example, if you create a custom template based on custom-css.ftl and associate it with a my_theme theme, you'll find it here:

```
<jive_home>/themes/my_theme/template/global/custom-css.ftl
```

Scope for Themes

As you plan your customizations, their scope will mean a lot. Do you want the changes to be seen across the UI, only in a particular community, or only in selected patches of the UI reached by certain URLs? The scope decision will be one of the first you make.

You'll use scope to map your themes to the UI so you can test your changes. Without this mapping, it won't be possible to really see how things are shaping up. The scope decisions are built on your design goals. If you're making more than just a few minor changes, you'll find it's useful to think in terms of UI boundaries. Will your changes impact certain kinds of content or certain communities, or certain content *in* certain communities? Your conceptual decision related to UI boundaries translates into implementation decisions about themes and their mappings.

In other words, group your change ideas into batches that correspond to the UI globally, or to communities, or to specific URLs. Then create themes corresponding to those groups, add custom templates to the themes, and map the themes to the UI scope you defined.

Template Hierarchy

Clearspace applies custom templates using the following hierarchy in which more specific levels override broader ones. The precedence from specific to general is URL > Community > Global, as described here:

- If you map a theme to a particular URL, templates in that theme are the only customization used for UI at that URL. All templates from themes mapped to a containing community or from global themes are ignored.
- If you map a theme to a particular community, templates in that theme are used unless you've mapped a URL-specific theme in the scope of that community. Templates in a global theme are ignored.
- If you make the theme the global theme, its templates are used except in any part of the UI where you've mapped a community- or URL-specific theme.

Global

Make a theme the global theme when you want its templates to impact UI across the application. Keep in mind that if you separately customize those same templates in themes that are mapped to communities or URLs, those more specific mappings will take precedence.

Note that the word "global" is used in the path names for some templates. This can be a bit confusing. The name doesn't mean that the templates are always applied globally, but rather that they guide UI that's not specific to certain content types. A "global" template will be applied only to the parts of the UI its containing theme is mapped to, as described above.

Map to a Community

Map a theme to a community for community-specific look and feel. Using custom templates for blogs, communities, or discussions, you can have those content look like they belong together in a particular community. "Global" custom templates will guide the look of their UI elements only in that community.

Keep in mind that if you separately customize those same templates in themes that are mapped to URLs, the more specific mappings will take precedence.

Map to a URL

Mapping a theme to a URL is a great way to customize very specific parts of the UI, such as blogs or user profiles (or even just one person's blog and profile).

When specifying the URL to map to, you use [regular expression](#) syntax. For example, a dot (.) means "any character" and an asterisk (*) means "any number of characters."

For example, blog URLs follow this form:

```
<context_root>/blogs/<blog_name>/<post_year>/<post_month>/<post_date>/<post_title>
```

So the following would map a theme to "any blog posts from 2007":

```
.*blogs/.*/2007.*
```

You can imagine how you might fill a theme with custom templates that guide blog appearance, along with some cross-content UI, then map that theme to this URL to call out content in blog years.

Here are a few more examples in which the Clearspace instance root is merely "clearspace":

All blogs

This reads as "any number of any characters followed by 'clearspace/blogs' followed by any number of any characters."

```
.*clearspace/blogs.*  
  
// Or more universally:  
.*blogs.*
```

Just the blogs main page

This reads as "any number of any characters followed by 'clearspace/blogs' and nothing else."

```
.*clearspace/blogs  
  
// Or more universally:  
.*blogs
```

A specific user's blog

This reads as "any number of any characters followed by 'clearspace/blogs/gladys' and any number of any characters." In other words, this maps to all posts in Gladys' blog.

```
.*clearspace/blogs/gladys.*  
  
// Or more universally:  
.*blogs/gladys.*
```