**[Allow different GWT visual themes for different users](http://stackoverflow.com/questions/7960685/allow-different-gwt-visual-themes-for-different-users)**

My question is this: how do you allow a different GWT visual theme depending on the user that logs in? I would like to decide which theme to use when the customer logs in (that is before the GWT app gets loaded, so I am pretty sure it should be possible). I have attempted to use class replacement based on a custom-property, but that failed because only the last inherited module's set of images become visible, even though I can select the right css file... I have searched everywhere and can't find the answer!

**THomos ans :**

I would try the following general approach:

1. Define one CSS file for each of the visual themes.
2. Put them all in a [ClientBundle](http://google-web-toolkit.googlecode.com/svn/javadoc/2.4/com/google/gwt/resources/client/ClientBundle.html" \o "GWT 2.4 Javadoc) as described [here](http://code.google.com/webtoolkit/doc/latest/DevGuideClientBundle.html#CssResource).
3. Hold off injecting the themed CSS until you've authenticated the user. You can inject the general CSS you need for displaying the login screen.

---------------------------------------------------------------------------

Thank you for your suggestion Thomas, but the problem with this solution is that you're assuming the CSS stylesheet is available for me to add to a ClientBundle (I tried that but unless you copy the css file and accompanying pics to your project, you can't do that). The themes come from external GWT modules. And I would like to keep it this way for modularity (it would be painful to import a whole bunch of resources into my project every time we needed a new theme).

The work-around I came up with was to write the injection code myself (just inject a link tag in the HTML head) at run-time. For completeness, here's the code to do it:

protected void doInjection(String cssFilePath) {

// <link type="text/css" rel="stylesheet" href="sol.css">

Element headEl = Document.get().getElementsByTagName("head").getItem(0);

HeadElement head = HeadElement.as(headEl);

LinkElement link = Document.get().createLinkElement();

link.setType("text/css");

link.setRel("stylesheet");

link.setHref(GWT.getModuleBaseURL() + cssFilePath);

head.appendChild(link);

}

And you call this method with something like this:

doInjection("gwt/standard/standard.css");

Then, inherit all Resources modules from your project's GWT module file. For example:

<inherits name='com.google.gwt.user.theme.standard.StandardResources'/>

<inherits name='com.google.gwt.user.theme.dark.DarkResources'/>

Inheriting the \*Resources version of the Module avoids automatically injecting the style-sheet.

To decide which theme to use, I created a custom GWT property in the module file, based on the value of this property, I replace a default Java class (which would just insert the default theme) with a different Java class (which subclasses the default class) if a different theme should be used. This has the added bonus that I can include my own ResourceBundle resources within each theme, because the replacement Java class used with a theme, besides injecting the right css file, can also provide alternative Resources to my GWT code.

**EDIT**

I would like to add one important note: The solution described above works quite well. But if your app uses different Locales or other GWT properties, this approach may cause the number of compilation permutations to explode! With only 6 different themes and 3 different Locales, on top of the standard 6 different browser versions you normally have, the GWT compiler will create 6 x 3 x 6 = 108 different compilations!! This is pretty crazy!!

A better solution, which I decided to follow after all, is to set an attribute into the HttpSession once the user logs in, and then based on the value of this attribute, load the appropriate css file (first thing in the onModuleLoad() of my entry-point class). The only difference from the solution described above is on how you select the theme.