Profiles Open Source Branding

The Profiles Open Source branding system aims to help you to use configurations in order to switch between different variations for the overall appearance (along with some smaller details), i.e., the 'branding', of the pages of Open Source Profiles.

Profiles contains two Branding Configurations.

- The default configuration, OpenSource branding, is intended for sites wanting to get up and running quickly with a simple and functional interface.
- The Foo configuration represents a fictional (satirical) institution. Foo is a good sandbox for experimenting with the features of the branding setup, and we use it below to illustrate the branding system.

  NB: If you are not sure which branding configuration you are using you can check by running (in a shell) the script, checkConfigurationFolder. For example, if you are currently using the Foo configuration then

  ```
  % checkConfigurationFolder.bash   Foo
  ```

  should indicate three empty diff results. If you are using a different configuration, then you will likely see many diff results.

For your institution's branding, you can use your own headers and footers, and your own particular values for properties, css definitions and text-snippets, by modifying these files:

1. myBranding.json
2. myBranding.js
3. myBranding.css

   NB: For the text-snippets in myBranding.json, you might use an editor with text-wrapping, as each entry must occur on one (possibly long) line.

Some Examples from the Foo Configuration

The following examples assume that the files of the included 'Foo' branding -- that is, the three 'myBranding' files from the Foo folder -- have been copied up into the Configuration folder.

The data and functions from the three myBranding.* configuration files get used across the application, so we shall refer to a few files ('general-application files') beyond those three.

The first few examples below refer to properties that are set in myBranding.json.

Text Snippets

Single Snippets: aboutProfiles-whatIsIt

In Foo's myBranding.json we have

> "aboutProfiles-whatIsIt": "Foo \"Profiles\" creates searchable and interactive online profiles of all faculty at Foo University, Foo School of Dental Medicine, and Foo School of Public Health. Foo Profiles is home to several faculty whose expertise covers nearly the entire breadth and depth of biomedicine. But in such a large community, how can you find collaborators to work with on your research? This is where Profiles can help."

In the general-application file brandingUtil.js, the function loadBrandingConstants() associates the label "aboutProfiles-whatIsIt" with the given text-snippet, using the global variable gBrandingConstants.

> **NB: loadBrandingConstants() looks for your myBranding.json relative to the 'fundamental' value of gBrandingConstants.staticRoot, which defaults to '/StaticFiles/'.**
>
> **If the folder that holds your 'static files' is not called StaticFiles, change your myBranding.js near the top:**

```
...
// Set staticRoot to the folder that holds your 'static files', ie
//   the {Coonfiguration, css, html-templates, etc.} folders
gBrandingConstants.staticRoot = "/StaticFiles/";
…
```

As a result of loading the branding constants, the code can then refer to them by index in the global variable gBrandingConstants[]. In the case of 'aboutProfiles-whatIsIt', the general-application file aboutAndHelp.js says

```
$('#whatIsIt').html(gBrandingConstants["aboutProfiles-whatIsIt"]);
```

This code fills in the targeted div in aboutOpenSource.html:

```
<div id="whatIsIt"></div>
```

Grouped Snippets: 'howToEdit' Header and Blurbs

As an 'advanced' example of text-snippets, you can have grouped blurbs that automatically hide their topic-header if myBranding.json fails to supply values for the blurbs. For example, in help.html, we have (with emphasis added here)

...
<div class="topic" **sharedAttr**="howToEdit">How do I edit my profile?</div>
<div class="**blurbForTopic**" **sharedAttr**="howToEdit" blurb="help-*howToEdit1*"></div>
<div class="**blurbForTopic**" **sharedAttr**="howToEdit" blurb="help-*howToEdit15*"></div>
<div class="**blurbForTopic**" **sharedAttr**="howToEdit" blurb="help-*howToEdit2*"></div>
...

In Foo's myBranding.json, we have

```
…
"help-howToEdit1": "",
"help-howToEdit2": "",
…
```

Since the div's in the html use both **'sharedAttr'** and '**blurbForTopic'** attributes, the group- or 'systematic'-oriented function applySystematicBlurbs(), in brandingUtil.js, will fill in the blurbs and show the topic / header, UNLESS the blurbs are all empty or absent.

Our Foo has no string at all for *…Edit15*, and empty strings for the other blurbs. So in addition to displaying nothing for those blurbs, the topic paragraph, 'How do I edit my profile?' is hidden on Foo's help page.

Other Properties

The data from myBranding.json can be used not only as text-blurbs. For example, in the general-application file, common.js, we set up favicons on our pages, with the following:

```
let faviconHref = `href="${gBrandingConstants.faviconUrl}"`;
if (faviconHref) {
  let head = $('head');
  head.append(`<link rel="icon" type="image/x-icon" ${faviconHref}>`);
  head.append(`<link rel="shortcut icon" type="image/x-icon"
${faviconHref}>`);
}
```

This code utilizes the faviconUrl supplied in Foo's myBranding.json

```
"faviconUrl": "https://fooCollege.edu/favicon.ico"
```

Sadly, that particular favicon does not exist, but yours should.

Custom Css

In addition to the data in myBranding.json, the branding configuration includes styling definitions in myBranding.css. In Foo's myBranding.css, you can see the setup for the background-color, etc. used in Foo's headers and footers.

Custom Implementation For Required Functions

The configuration also includes myBranding.js, where you may supply your own implementation of

```
setupHeadIncludesAndTabTitle()
emitBrandingHeader()
emitBrandingFooter()
```

For instance, Foo's emitBrandingFooter() sets up 'larger' and 'smaller' footer div's, providing potential for 'responsive' differences between the appearance on larger and smaller screen-widths. You could simplify it to use just one div.

Indeed, the current Foo function treats both large and small footers the same. You could use different text for large/small either by baking it directly into the logic of emitBrandingFooter(), or indirectly, by supplying large and small blurbs in the Json, and then having emitBrandingFooter() access the blurbs via gBrandingConstants, as in the earlier example above.

Other Resources

You might also supply other resources, e.g., images that your own *.js or *.html can reference. These should get placed in appropriate folders. E.g., the general-application has folders …./img/branding, …./js, etc.