sheffercool / PinegrowDevelopersDocumentation

forked from Pinegrow/PinegrowDevelopersDocumentation

Code Pull requests Actions Projects Wiki Security Insights Settings

master

PinegrowDevelopersDocumentation / Templates and Resources.md

BoDonkey Fixes typo and adds to Templates and Resources.md

1 contributor

Raw Blame

143 lines (117 sloc) 8.78 KB

Adding Templates and Resources

Overview

Plugins can optionally offer a variety of resources to the user. This can include HTML, CSS, PHP, and JavaScript files, basically any file that you want to use in your project. These resources are copied from the plugin either upon activation of the library from the "Manage libraries & plugins..." menu or when a plugin HTML template is saved as a project.

Templates

Overview

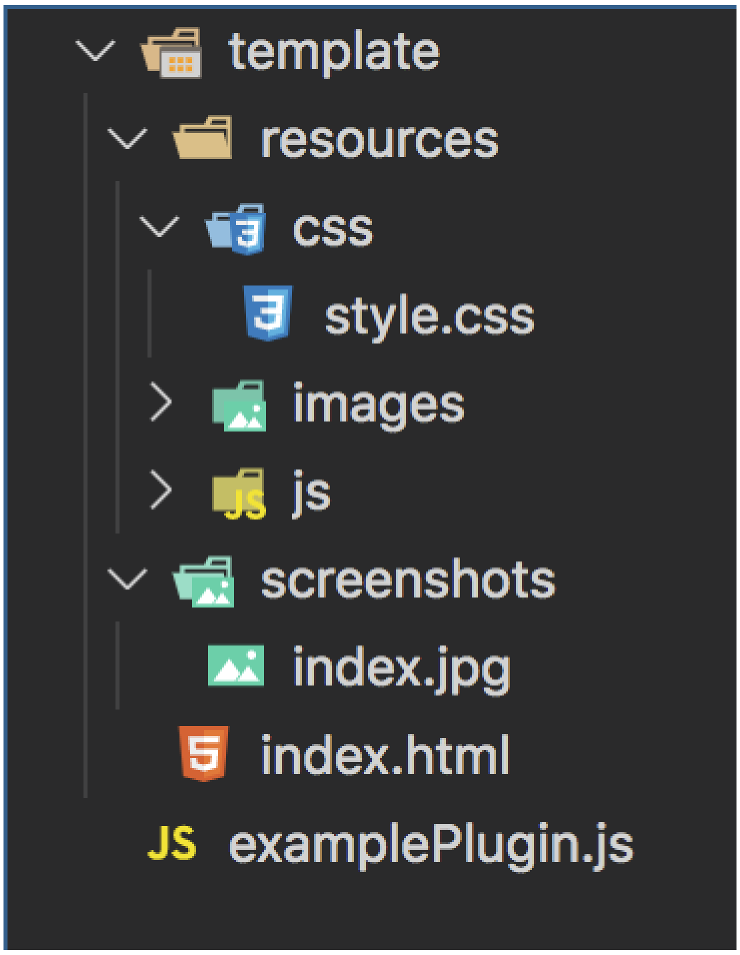
Templates are optional pre-made HTML pages that can include anything from a basic document, to a page that is already populated with a large amount of code. Typically, at minimum, a template will contain the basic HTML boilerplate, plus plugin specific CSS links (either local or remote) and/or JavaScript (either in page scripts or links to local or remote files). These templates are added using the `addTemplateProjectFromResourceFolder` constructor.

addTemplateProjectFromResourceFolder (template_folder, done, index, filter_func, skip_add_to_templates, absolute_folder)

This function is used when constructing a plugin that provides one or more templates as a base for the user. For example, the stock Bootstrap 4 framework provides 19 starter templates when the user clicks on "New page or project".

To utilize this in a plugin, create a folder that contains each HTML template file plus an additional subfolder that must be named "screenshots" and contain an image that is typically a screenshot of that template to display to the user. Each image must have the same name as the template file. Any other files that should be included when the template is saved as a project should also be in this folder. If desired, additional resources can be placed into a sub-folder(s).

Example template folder structure:



```
template
  resources
    css
      style.css
    images
    js
  screenshots
    index.jpg
  index.html
examplePlugin.js
```

Arguments

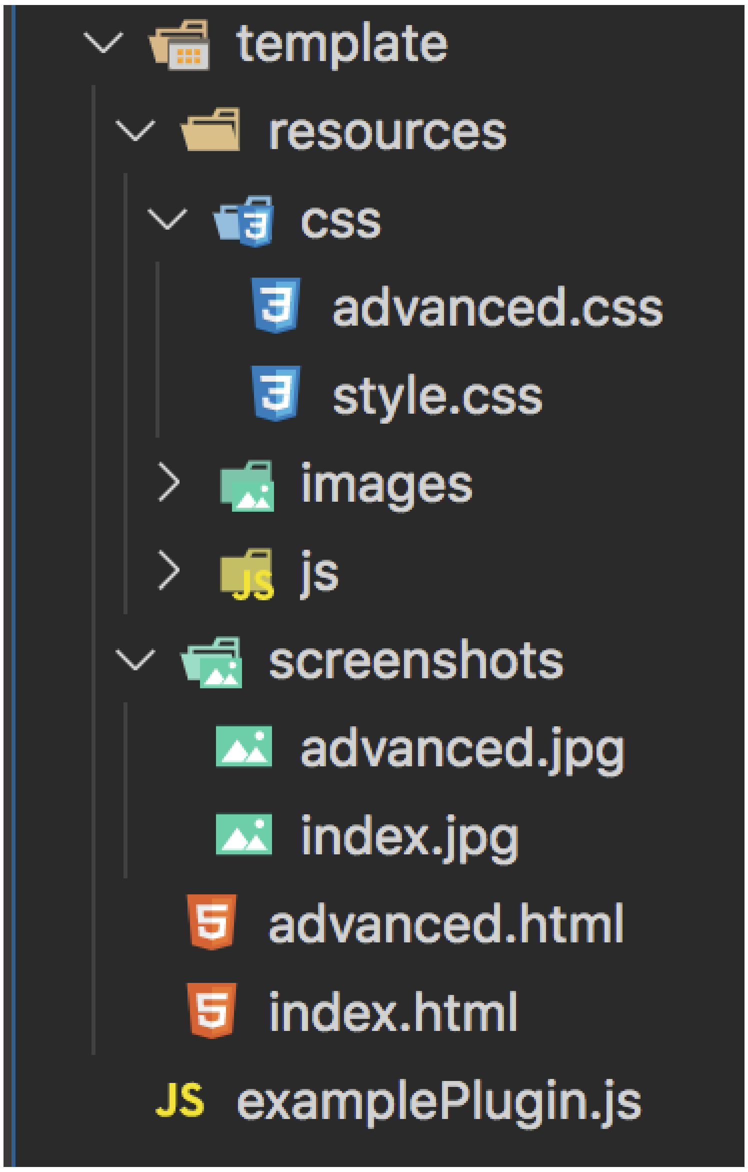
- **template_folder** - the source of the template folder relative to the main plugin file - Note: if the folder is at the same level as the main plugin file you do not need to prefix with './' to search in the current directory.
- **done** - typically passed null, takes a callback function that is triggered once the framework is created and is passed the framework as an argument
- **index** - determines the order of framework display - typically set to 100 or higher to add the plugin framework at the end
- **filter_func** - receives a function that can edit the files added to the template - optional, typically used when adding multiple templates, each with specific resources
- **skip_add_to_template** - optional, internal development use
- **absolute_folder** - optional, internal development use

Simple usage:

```
framework.addTemplateProjectFromResourceFolder('template', null, 100);
```

Advanced usage

In this example there is a template specific CSS file ("advanced.css") that should only be added if a template ("advanced.html") requires that particular file. This same method can be used to include a specific JavaScript or other file.
First, add the additional template, CSS, and screenshot to the template folder.



```
template
  resources
    css
      advanced.css
      style.css
    images
    js
  screenshots
    advanced.jpg
    index.jpg
  advanced.html
  index.html
examplePlugin.js
```

Second, pass a function to the `addTemplateProjectFromResourceFolder` to select any files from an array of files (`templateSpecificFiles`) to exclude from the standard template. In this example, "advanced.css".

```
var templateSpecificFiles= ['advanced.css'];

framework.addTemplateProjectFromResourceFolder('template', null, 100, function (r
        var currentFilesName = templateSpecificFiles.filter(function (fileName) {
                        return node.name == fileName;
                });

                if (currentFilesName && currentFilesName.length > 0) {
                        node.required = false;
                };
})
```

This function is passed an argument, conventionally named node, that contains information about each added resource. This function returns any resources that match a filename included in the array of file names not to include automatically and then sets the `required` key of the resource to false. This ensures that the file will only be saved in the project if there is a link in the HTML template. This same function can also be extended to place the templates in a specific order using the `order` key.

```
var templatesOrder = ['index.html', 'advanced.html'];

        //Add our template into the project
        framework.addTemplateProjectFromResourceFolder('template', null, 100, fur
                //remainder of function from above

                var templateIndex = templatesOrder.indexOf(node.name);
                if (templateIndex >= 0) node.order = templateIndex;
        });
```

Resources

Overview

Plugin resources are CSS and JavaScript files that can be loaded onto any page that activates the plugin. They can be provided as local files that get copied into the project folder, or as remote links that get added into the current HTML page(s). Additionally, resources can be used to upgrade existing project or page files and links.

PgComponentTypeResource (resource_url, code)

The PGComponentTypeResource constructor is used to add javascript and CSS files or links to a project. It recieves addition to the two arguments:

- **resource_url** This argument takes the URL of the file or folder to be added. For local files the URL can be generated using a helper function, `framework.getResourceUrl(path_to_resource)` .
- **code** This argument is typically not used and defaults to null.

getResourceUrl(path_to_resource)

This helper function takes the path to the resource file or folder relative to the base plugin file.

```
framework.getResourceUrl('template/resources/css/style.css');
```

In addition to the arguments passed in at instantiation it can accept a number of additional key:value pairs.

key	value
type	mime type - this allows the Pinegrow app to determine the correct way to embed the file on the HTML page, <script> and src for JavaScript, or <link> and href for CSS. Automatically set to correct mime type by lookup.
detect	regex string - determine if another file (like jQuery), or an earlier version exists on the page -- typically not used
footer	boolean - determines whether the item should be added to the head (false) or the bottom of the body (true) -- default: false
project	internal use only
isFolder	boolean - used to indicate if the resource is a folder or not -- default: false
source	url - typically the same as the resource_url -- default: null
relative_url	resource file location relative to the template -- default: null
replace	boolean (or function returning boolean) If the `detect` key is used and a match is found then this indicates if the found resource should be replaced with the file at resource_url -- default: false -- typically used to determine if a resource needs to be replaced during an update

The newly created resource object is returned as a value to the framework in the `resources` key.

```
framework.resources.add(new_resource)
```

Typical example code using file structure from above.

```
var resource_files = [
                'css/index.css',
                'js/my-js.js'
        ];
resource_files.forEach(function (resource_file) {
                var file = framework.getResourceUrl('template/resources/' + resou
                var resource = new PgComponentTypeResource(file);
                resource.relative_url = resource_file;
                resource.source = file;
                resource.footer = resource_file.indexOf('.js') >= 0;
                resource.type = resource_file.indexOf('.js') >= 0 ? 'application/
                framework.resources.add(resource);
        });
```

The `PgComponentTypeResource` constructor can also be used to add in remote files, such as CDN hosted framework files or Google fonts.

```
var resource = new PgComponentTypeResource('https://cdnjs.cloudflare.com/ajax/lib
resource.type = 'application/javascript';
framework.resources.add(resource);

var resource = new PgComponentTypeResource('https://fonts.googleapis.com/css?fami
resource.type = 'text/css';
framework.resources.add(resource);
```

Framework key:value pairs for Templates and Resources

ignore_css_files

This key is used if the plugin includes customized CSS files within a template or as a stand-alone resource that shouldn't be altered by the user. It receives an array containing a single regex string or multiple comma-separated regex strings. Any CSS file in the page resources that is matched by these string will be locked for editing under the stylesheet dropdown.
Single string

```
framework.ignore_css_files = [/my_plugin_style\.css/i];
```

Multiple regex strings

```
framework.ignore_css_files = [/my_plugin_style\.css/i, /my_other_styling\.css/i];
```

Next: Components