Report Skins

Including

Introduction

Report skins can be used to manage the look and feel and various settings of multiple reports. Skins are basically a collection of:

- CSS stylesheets, JavaScript libraries.
- Configuration entries (settings).

Defining Skins

Skins are defined be creating skin configuration files in the reports directory. Forena comes with a few example skin configuration files such as **default_skin.skinfo**, located in Forena's subdirectory repos/reports and which can be used as a template for creating new skins.

Skin info (.skinfo) files are created in the same basic syntax as .info files for themes or modules. The following example illustrates the syntax:

```
; The name indicates the name displayed in the skin select box
; on the Format tab when editing reports.
name = Default Skin
description = Default skin for use with Forena.
; JavaScript include example
scripts[] = dataTables/media/js/jquery.dataTables.min.js
; css Include example
stylesheets[all][] = table_padding.css
```

After creating each new skin information file be sure to clear the Drupal cache.

Assigning Skins to reports

Configuring the default skin

The **default report skin** can be configured via admin option admin/config/content/forena.

Selecting a skin for a specific report

For each report you can select a skin via the Format tab of the report editor (try it out via a sample report). Alternatively, you can specify the skin directly in the .frx file by specifying the **skin="skin_file_name"** attribute in the frx:options element in the head section of the .frx file as follows (whereas skin_file_name is the filename of your skin, without the .skinfo extension of it):

```
<html id="frxsrc-1">
<head>
  <frx:options skin="skin_file_name"/>
</head>
<body>
...
</body>
</html>
```

Core Libraries

Libraries provided by core (e.g. JQuery UI) can be added if you know the providing module name as well as the library name as advertised by hook_library. Inlcuding a libraries[]=system/jquery.ui will cause the jquery.ui library provided by the sytem module to be loaded for all reports that use this skin.

CSS Libraries

Stylesheets can be included using the same syntax for Drupal themes. Including a stylesheets[all][]=sheet.css line in your .skinfo file, will cause the sheet.css file to be loaded for any media types.

If you are using a PDF generator (MPDF or Prince), understand that you can specify stylesheets[pdf][] entries to include particular stylesheets only in the PDF transformation. Forena looks first in the reports directory for the stylesheets and then at the site root level, so you can specify theme css files by fully qualifying the path to the theme. This can be particularly useful when you want to include a typography stylesheet in your PDF translations.

JavaScript Libraries

JavaScript libraries are included using the same syntax as is used in the theme info file. In the above example the scripts[]=dataTables/media/js/jquery.dataTables.js is used to load the JQuery dataTables library. Forena will search for these libraries first in the report directory and then in the sites/all/libraries folder (checkout Enable DataTables for instructions about how to install the dataTables library). This is particularly useful if you want to load additional JQuery plugins for a set but not all reports. You can add additional JavaScript libraries without needing to write custom module or theme code.

Example

Here is how a custom version of a skin info file might look like, cloned from the delivered default_skin.skinfo :

```
; The name indicates the name displayed in the skin select box
; on the Format tab when editing reports.
name = Custom Skin
description = Custom skin for use with Forena.
libraries[] = system/jquery.ui
; Javascript include example
scripts[] = dataTables/media/js/jquery.dataTables.min.js
scripts[] = custom_skin.js
; css Include example
stylesheets[all][] = table_padding.css
stylesheets[all][] = custom_skin.css
```

The line containing **stylesheets[all][] = custom_skin.css** adds this css to any reports using that skin.

The line containing **scripts[] = custom_skin.js** to the .skinfo file, adds the content of this custom_skin.js file to any report that is using this skin. In our example the JavaScript needed to actually use various features provided by the dataTables plugin. Adding the line **scripts[] = dataTables/media/js/jquery.dataTables.min.js** to the .skinfo file only makes this plugin available, so we need to add our own JavaScript to make sure that this plugin gets invoked.

According to the dataTables documentation, the content of the custom_skin.js file should look similar to this example:

```
<script id="frxsrc-2">
  $(document).ready(function() {
      $('#example').dataTable();
  } );
</script>
```

However, this doesn't take into account that in Drupal the **$(document).ready** is already used by Drupal. Rather than overwrite $(document).ready we need to implement a **Drupal 7 behavior**. Start from a copy of this code snippet mentioned on Managing JavaScript in Drupal 7 in the provided custom.js:

```
<script id="frxsrc-3">
  (function ($) {
    Drupal.behaviors.exampleModule = {
      attach: function (context, settings) {
    $('.example', context).click(function () {
    $(this).next('ul').toggle('show');
  });
      }
    };
  })(jQuery);
</script>
```

Then apply these changes to the copied code snippet:

- The behavior name **exampleModule** should be a variable name unique to our implementation so we change it to **CustomSkin**.
- Change the attach function to be code based on the dataTables example.
- Change **#example** to **table**, to use the dataTables plugin for all tables.

After these changes are applied, the updated custom_skin.js file should look like this:

```
<script id="frxsrc-4">
  (function ($) {
    Drupal.behaviors.CustomSkin = {
      attach: function (context, settings) {
  $('table').dataTable();
      }
    };
  })(jQuery);
</script>
```

For more info about this topic, checkout the video about Report Skins - Create skins that control graphing defaults and integrate JQuery plugin.

Defining Configuration Settings

Skins also facilitate assigning values to specific variables, which can then be used in all reports using the skin. This is typically done for various flavors of settings, such as:

- Establishing defaults for SVGGraph settings.
- Settings that can be used to control how reports are rendered.
- Arbitrary variables (and their values) that can be referenced in a report using the skin data context.

The following example (which can be added anywhere in a .skinfo file) contains an illustration of how to do so:

```
; ********************************************************************
; Set SVGGraph defaults:
;
; FrxSVGGraph[colors][] = red
; FrxSVGGraph[colors][] = blue
;
; ********************************************************************
; Settings that can be used to control how reports are rendered:
;
; Disable helper classes such as even and odd:
; FrxReport[noHelperClasses] = true
;
; Control the root element tag name:
; FrxXMLDoc[rootElementName] = node
;
; ********************************************************************
```

```
; Arbitrary variables referenced in reports using the skin data context:
;
; Make skin.my_variable available to be referenced anywhere in a report:
; my_variable = Value of my own variable
```