Theme-related Software Concept and Design

{maketoc}

Overview

Tiki uses the Smarty template engine, so pages generally have a PHP file (.php) and a Smarty template (.tpl) associated with them. These produce the XHTML for the page, which is then given visual styles with CSS. The CSS is provided by a combination of some feature-specific stylesheets, default layout and design stylesheets, and theme stylesheets.

Concept (Tiki14)

There are several features that determine which theme is displayed for a given screen. See also http://doc.tiki.org/Themes
About the changes for Tiki14 see a bit more info here:
https://dev.tiki.org/Tiki14#New_theme_for_tiki.org_sites.

Site theme

Site theme setting can be made at the **Look and Feel** settings panel. Themes can have options.

Overrides:
- none

Overridden by:
- user theme
- group theme
- theme control
- perspective

User theme

If preference change_theme is enabled than users are allowed to change theme.

Overrides:
- site theme
- perspective

Overridden by:

- group theme
- admin theme
- theme control

Limited by:
- available themes

Group theme

Define a theme for a group (tiki-admingroups.php)

Overrides:
- site theme
- user theme
- perspective

Overridden by:
- admin theme
- theme control

Admin theme

Themes and theme options for the settings control panels (tiki-admin.php?page=XXX).

Overrides:
- site theme
- user theme
- group theme

Overridden by:
- perspective setting

Not affected by:
- theme control

Theme Control

Allows to have a specific theme for objects, categories and sections, see Documentation.

Overrides:
- site theme
- user theme
- group theme
- perspective

Overridden by:
- none

No effect for:
- admin theme

Perspective

Perspectives are to override a preference, see Documentation

Overrides:
- site theme
- admin theme

Overridden by:
- user theme

Edit CSS

It is possible to try, view and edit css files from a built in CSS editor. Using the "Try" functionality the selected theme/option is applied
and shown for all screens in Tiki for the duration of your login session.
Overrides:
- everything

Overridden by:
- none

How Tiki decides which theme to display

0) If a CSS try theme session is active than that theme will be used, so in this case none of the below points apply

1) Check for user's theme preference

2) If there is a group theme defined, override user's theme

3) If Theme Control feature is enabled, then override the above like this:
- If a theme is assigned to the individual object that theme is used
- If not then if a theme is assigned to the object's category that theme is used: but if the object is assigned to multiple categories and this causes a conflict in the theme choice then they are all skipped and the next logical choice in this hierarchy is chosen
- If not then a theme for the section is used

4) If an admin page (Settings control panel) is displayed, check if there is any admin theme set to override. When in a perspective, check if there is a perspective specific setting and override if available.

5) If none of the above applies, use the site theme. When in a perspective, see if there is a perspective specific setting and override if available.

Theme related CSS files

Basics

1) Tiki always loads themes/base_files/css/tiki_base.css, no matter which theme is selected. This file contains rules specific to Tiki that Bootstrap doesn't have awareness of.

2) The css file of a theme (or theme option) must always have to be named like the name of the theme's or option's folder name, for example for fivealive-lite theme it is called fivealive-lite.css. This isn't necessary if the theme is selected by specifying the Custom theme URL, but if it's selected as a theme in the "themes" directory, then it needs to follow that naming scheme.

Create a new theme or theme option

New theme

If you want to create a new theme for Tiki, you have place a css file under the css folder in the theme's directory (eg: for a new theme called "mytheme": themes/mytheme/css/mytheme.css).

New theme option

If you want to create a simple new option called "myoption" (eg: for the "mytheme" theme), than proceed like this:

- step1) create a new subfolder under the "themes/mytheme/options" folder called "myoption/css" (so path is: "themes/mytheme/options/myoption/css/")
- step2) place in the new folder a css file that contains the changed or additional css for your new option (eg: "themes/mytheme/options/myoption/css/myoption.css") - this is loaded in addition to the theme.css file in the main css folder eg themes/mytheme/css/mytheme.css
- step3) optionally place in the new folder a custom.css file that contains your specific local customizations for that one Tiki instance (eg: "themes/mytheme/options/myoption/css/custom.css") - this too is loaded in addition to any custom.css file that is placed in the main css folder eg themes/mytheme/css/custom.css

For more complex options you can also have the same folder structure as for a normal theme (fonts, icons, images, templates, etc).

Customizaton of themes and options

If you want to slightly modify (change some colors, backgrounds, etc) a theme or a theme option shipped with Tiki then it may be easier for you, if you don't edit the theme files directly, but create a custom.css file in the same directory of the theme or theme option and store your customizations there.

If you have a main theme with theme options, you can have a custom.css for the main theme and another custom.css for the theme option, both will be loaded (first the main theme's custom.css, then the theme option's custom.css)

To create a highly customized theme however it may be better, from an onward change management point of view, to add your own option to an existing theme. This then lets you use all the css from the main theme as a 'base' for your customised theme and this will be automatically maintained/updated for subsequent Tiki version changes. All you have to do then is to develop your option css and change this from time to time as may be necessary due to Tiki version upgrades and you can use your own naming convention for the option and its subsequent versions for your change management process.

Loading css files

- For a main theme Tiki gets first the theme's main css file (eg: from themes/mytheme/css/mytheme.css), then a custom.css file containing local customizations if found in the theme's css folder (eg: /themes/mytheme/css/custom.css)
- If a theme option is enabled, then Tiki gets first the main theme's css file (eg: from "themes/mytheme/css/mytheme.css") then it gets the theme option's css file ("themes/mytheme/options/myoption/css/myoption.css"), and then it loads the main theme's custom.css (eg: from "themes/mytheme/css/custom.css") and finally the theme option's custom.css file containing local customizations if found in the option's folder (eg: "themes/mytheme/options/myoption/css/custom.css")

Theme related smarty template files

Basics

- Tiki uses smarty templates
- Tiki ships all basic template files in the "templates/" folder.

Customization of templates

If you don't like how, for example, a blog post (templates/tiki-view_blog_post.tpl) looks, you can have your own, redesigned template.
You have a number of ways of implementing this however as follows:
1) If you want to have a custom template available for **all** your themes, you should put the altered or new template file in the "themes/templates/" folder.

2) If you want to have a custom template for a given theme and all its potential options, put the template file in the "themes/mytheme/templates" folder
3) If you want to have a custom template for just one specific theme option, put the template file in the "themes/mytheme/options/myoption/templates" folder

In this way you can manage a hierarchy of choices so that the same .tpl file can have different content depending upon whether it is available at a specific level in this choice hierarchy.

How Tiki decides which template to display

1) first try to use the theme option's template folder (eg: "themes/mytheme/options/myoption/templates/tiki-view_blog_post.tpl)
2) then try to use the theme's template folder (eg: "themes/mytheme/templates/tiki-view_blog_post.tpl)
3) then try to use "themes/templates" folder (eg: "themes/templates/tiki-view_blog_post.tpl)
4) finally use "templates/" folder ((eg: "templates/tiki-view_blog_post.tpl))

Some other considerations

Newsletter css

As discussed in the Newsletter documentation a customised newsletter.css file can be defined to ensure that the css used in the Newsletter generated emails is as compact as possible and to ensure better presentation in a range of email clients.

The customised newsletter.css file should be placed in either the /themes/mytheme/css/ folder or the /themes/mytheme/options/myoption/css/ folder.

custom.js

Custom javascript can be defined using the Customization tab of the main Look & Feel admin screen - but often this is better managed in a separate file called custom.js that is held on the server.

There are number of choices where the file with the custom javascript can be held on the server depending upon whether the script is theme dependent or not:

- to use a file independent of any specific theme it should be placed in the root of the /themes folder
- to use a file for all options of a specific theme it should be placed in the root of the main folder for the theme eg in /themes/mytheme
- finally to use a file for a specific option within a theme it should be placed in the root of the option folder eg in themes/mytheme/options/myoption/

In all instances the file will be automatically loaded after the main set of javascript definitions.

Developer info

The different theme settings are stored as preferences and can be accessed as smarty variables.

Theme selection logic

Theme selection is primarily done at lib/setup/theme.php. If you enable Theme Control, then after lib/setup/theme.php has finished tiki-tc.php kicks in and makes sure that Theme Control settings are applied.

Template selection logic

Template selection is done using lib/init/smarty.php.

Most of this was developed in the tiki14_themes experimental branch

It was merged back into trunk on 19 jan 2015

Here are the details of the merge back to trunk commit

[+]
Merge #2 from tiki14_themes

Merged from tiki14_themes [MOD] iconsets: Change the theme specific iconset naming convention to match css and other files, so now should be: themes/your-theme/icons/your_theme.php and themes/your-theme/options/your-option/icons/your_option.php Note: hyphens need to be replaced with underscores [from revision 53595] [FIX] restore possibility to override an icon img with another img just by having the same name. Also change the default icon set icon class to btn-link [from revision 53616] [FIX] typo [from revision 53617] [MOD] basic things to get themegenerator somewhat functional again [from revision 53618] [MOD] themegenlib at new dir [from revision 53619] [MOD] Some visual and loading changes for brosho CSS assistant. This plugin kindof works, altough its source has not been modified since 2010. (https://github.com/mustardamus/brosho-plugin) Maybe for Tiki15 look for something more up-to-date [from revision 53620] [MOD] remove and modify tips [from revision 53621] [FIX] themegen: A few more fixes for the theme generator dialog [from revision 53628] [REF] icons: Further refactoring of iconsets, so that all the icons available in the chosen set can be used in templates - glyphicons defaults to come soon [from revision 53629] [REF] icons: Example theme and option icons as demos, possibly should be removed or replaced with better (useful) examples [from revision 53630] [FIX] icons: Add glyphicons defaults [from revision 53631] [KIL] Remove redundant layout_section feature. Hasn't been connected to anything for a long time and should be done with module visibility, perspectives and/or theme control these days. [from revision 53632]

Upgrade to Tiki14

The structure of themes changed singificantly for Tiki14, so please note the below points:
- your old css files will probably need to be reviewed modified as many selectors have been adjusted to bootstrap
- an upgrade patch is provided that modifies your database. The change is focusing on removing ".css" from the end of the theme values in as shown below:

```
UPDATE `tiki_preferences` SET `name` = 'theme' WHERE `name` = 'theme_active'; UPDATE `tiki_preferences` SET `name` = 'theme_option', `value` = REPLACE(`value`, '.css', '') WHERE `name` = 'style_option'; UPDATE `tiki_user_preferences` SET `value` = REPLACE(`value`, '.css', '') WHERE `prefName` = 'theme'; UPDATE `tiki_user_preferences` SET `prefName` = 'theme_option', `value` = REPLACE(`value`, '.css', '') WHERE `prefName` = 'theme-option'; UPDATE `tiki_theme_control_categs` SET `theme` = REPLACE(`theme`, '.css', ''); UPDATE `tiki_theme_control_objects` SET `theme` = REPLACE(`theme`, '.css', ''); UPDATE `tiki_theme_control_sections` SET `theme` = REPLACE(`theme`, '.css', ''); UPDATE `users_groups` SET `groupTheme` = REPLACE(`groupTheme`, '.css', '');
```