Customizing the Trac Interface

Table of Contents

Customizing the Trac Interface **1**
Introduction 2
Project Logo and Icon 2
Logo 2
Icon 2
Custom Navigation Entries 2
Site Appearance 3
Project List 4
Project Templates 5

Introduction

This page is meant to give users suggestions on how they can customize the look of Trac. Topics on this page cover editing the HTML templates and CSS files, but not the program code itself. The topics are intended to show users how they can modify the look of Trac to meet their specific needs. Suggestions for changes to Trac's interface applicable to all users should be filed as tickets, not listed on this page.

Project Logo and Icon

The easiest parts of the Trac interface to customize are the logo and the site icon. Both of these can be configured with settings in trac.ini.

The logo or icon image should be put in a folder named "htdocs" in your project's environment folder. (*Note: in projects created with a Trac version prior to 0.9 you will need to create this folder*)

> *Note: you can actually put the logo and icon anywhere on your server (as long as it's accessible through the web server), and use their absolute or server-relative URLs in the configuration.*

Now configure the appropriate section of your trac.ini:

Logo

Change the `src` setting to `site/` followed by the name of your image file. The `width` and `height` settings should be modified to match your image's dimensions (the Trac chrome handler uses "`site/`" for files within the project directory `htdocs`, and "`common/`" for the common `htdocs` directory belonging to a Trac installation). Note that 'site/' is not a placeholder for your project name, it is the actual prefix that should be used (literally). For example, if your project is named 'sandbox', and the image file is 'red_logo.gif' then the 'src' setting would be 'site/red_logo.gif', not 'sandbox/red_logo.gif'.

```
[header_logo]
src = site/my_logo.gif
alt = My Project
width = 300
height = 100
```

Icon

Icons should be a 32x32 image in `.gif` or `.ico` format. Change the `icon` setting to `site/` followed by the name of your icon file. Icons will typically be displayed by your web browser next to the site's URL and in the `Bookmarks` menu.

```
[project]
icon = site/my_icon.ico
```

Note though that this icon is ignored by Internet Explorer, which only accepts a file named favicon.ico at the root of the host. To make the project icon work in both IE and other browsers, you can store the icon in the document root of the host, and reference it from trac.ini as follows:

```
[project]
icon = /favicon.ico
```

Should your browser have issues with your favicon showing up in the address bar, you may put a "?" (less the quotation marks) after your favicon file extension.

```
[project]
icon = /favicon.ico?
```

Custom Navigation Entries

The new [mainnav] and [metanav] can now be used to customize the text and link used for the navigation items, or even to disable them (but not for adding new ones).

In the following example, we rename the link to the Wiki start "Home", and hide the "Help/Guide". We also make the "View Tickets" entry link to a specific report .

```
[mainnav]
wiki.label = Home
tickets.href = /report/24

[metanav]
help = disabled
```

See also TracNavigation for a more detailed explanation of the mainnav and metanav terms.

Site Appearance

Trac is using ?Genshi as the templating engine. Documentation is yet to be written, in the meantime the following tip should work.

Say you want to add a link to a custom stylesheet, and then your own header and footer. Save the following content as `site.html` inside your projects `templates/` directory (each Trac project can have their own `site.html`), e.g. `/path/to/env/templates/site.html`:

```
<html xmlns="http://www.w3.org/1999/xhtml"
     xmlns:py="http://genshi.edgewall.org/"
     py:strip="">

 <!--! Add site-specific style sheet -->
 <head py:match="head" py:attrs="select('@*')">
   ${select('*|comment()|text()')}
   <link rel="stylesheet" type="text/css"
         href="${href.chrome('site/style.css')}" />
 </head>

 <body py:match="body" py:attrs="select('@*')">
   <!--! Add site-specific header -->
   <div id="siteheader">
     <!--! Place your header content here... -->
   </div>

   ${select('*|text()')}

   <!--! Add site-specific footer -->
   <div id="sitefooter">
     <!--! Place your footer content here... -->
   </div>
 </body>
</html>
```

Those who are familiar with XSLT may notice that Genshi templates bear some similarities. However, there are some Trac specific features - for example `${href.chrome('site/style.css')}` attribute references a CSS file placed into environment's `htdocs/` directory. In a similar fashion `${chrome.htdocs_location}` is used to specify the common `htdocs/` directory belonging to a Trac installation. That latter location can however be overriden using the [trac] htdocs_location configuration setting.

`site.html` is one file to contain all your modifications. It usually works using the `py:match` directive (element or attribute), and it allows you to modify the page as it renders - the matches hook onto specific sections depending on what it tries to find and modify them. See ?this thread for a detailed explanation of the above example `site.html`. A `site.html` can contain any number of such `py:match` sections for whatever you need to modify. This is all Genshi, so the ?docs on the exact syntax can be found there.

Example snippet of adding introduction text to the new ticket form (but not shown during preview):

```
<form py:match="div[@id='content' and @class='ticket']/form" py:attrs="select('@*')">
 <py:if test="req.environ['PATH_INFO'] == '/newticket' and (not 'preview' in req.args)">
   <p>Please make sure to search for existing tickets before reporting a new one!</p>
 </py:if>
 ${select('*')}
</form>
```

This example illustrates a technique of using `req.environ['PATH_INFO']` to limit scope of changes to one view only. For instance, to make changes in `site.html` only for timeline and avoid modifying other sections - use `req.environ['PATH_INFO'] == '/timeline'` condition in `<py:if>` test.

More examples snippets for `site.html` can be found at ?CookBook/SiteHtml.

Example snippets for `style.css` can be found at ?CookBook/SiteStyleCss.

If the environment is upgraded from 0.10 and a `site_newticket.cs` file already exists, it can actually be loaded by using a workaround - providing it contains no ClearSilver processing. In addition, as only one element can be imported, the content needs some sort of wrapper such as a `<div>` block or other similar parent container. The XInclude namespace must be specified to allow includes, but that can be moved to document root along with the others:

```
<form py:match="div[@id='content' and @class='ticket']/form" py:attrs="select('@*')"
       xmlns:xi="http://www.w3.org/2001/XInclude">
 <py:if test="req.environ['PATH_INFO'] == '/newticket' and (not 'preview' in req.args)">
   <xi:include href="site_newticket.cs"><xi:fallback /></xi:include>
 </py:if>
 ${select('*')}
</form>
```

Also note that the `site.html` (despite its name) can be put in a common templates directory - see the [inherit] templates_dir option. This could provide easier maintainence (and a migration path from 0.10 for larger installations) as one new global `site.html` file can be made to include any existing header, footer and newticket snippets.

Project List

You can use a custom Genshi template to display the list of projects if you are using Trac with multiple projects.

The following is the basic template used by Trac to display a list of links to the projects. For projects that could not be loaded it displays an error message. You can use this as a starting point for your own index template.

```
<!DOCTYPE html
   PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
     xmlns:py="http://genshi.edgewall.org/"
     xmlns:xi="http://www.w3.org/2001/XInclude">
 <head>
   <title>Available Projects</title>
 </head>
 <body>
   <h1>Available Projects</h1>
   <ul>
     <li py:for="project in projects" py:choose="">
       <a py:when="project.href" href="$project.href"
          title="$project.description">$project.name</a>
       <py:otherwise>
         <small>$project.name: <em>Error</em> <br /> ($project.description)</small>
       </py:otherwise>
     </li>
   </ul>
 </body>
</html>
```

Once you've created your custom template you will need to configure the webserver to tell Trac where the template is located (pls verify ... not yet changed to 0.11):

For mod_wsgi:

```
os.environ['TRAC_ENV_INDEX_TEMPLATE'] = '/path/to/template.html'
```

For FastCGI:

```
FastCgiConfig -initial-env TRAC_ENV_PARENT_DIR=/parent/dir/of/projects \
              -initial-env TRAC_ENV_INDEX_TEMPLATE=/path/to/template
```

For mod_python:

```
PythonOption TracEnvParentDir /parent/dir/of/projects
PythonOption TracEnvIndexTemplate /path/to/template
```

For CGI:

```
SetEnv TRAC_ENV_INDEX_TEMPLATE /path/to/template
```

For TracStandalone, you'll need to set up the `TRAC_ENV_INDEX_TEMPLATE` environment variable in the shell used to launch tracd:

- Unix

  ```
  $ export TRAC_ENV_INDEX_TEMPLATE=/path/to/template
  ```

- Windows

  ```
  $ set TRAC_ENV_INDEX_TEMPLATE=/path/to/template
  ```

Project Templates

The appearance of each individual Trac environment (that is, instance of a project) can be customized independently of other projects, even those hosted by the same server. The recommended way is to use a `site.html` template (see #SiteAppearance) whenever possible. Using `site.html` means changes are made to the original templates as they are rendered, and you should not normally need to redo modifications whenever Trac is upgraded. If you do make a copy of `theme.html` or any other Trac template, you need to migrate your modifiations to the newer version - if not, new Trac features or bug fixes may not work as expected.

With that word of caution, any Trac template may be copied and customized. The default Trac templates are located inside the installed Trac egg (`/usr/lib/pythonVERSION/site-packages/Trac-VERSION.egg/trac/templates, .../trac/ticket/templates, .../trac/wiki/templates, ++`). The #ProjectList template file is called `index.html`, while the template responsible for main layout is called `theme.html`. Page assets such as images and CSS style sheets are located in the egg's `trac/htdocs` directory.

However, do not edit templates or site resources inside the Trac egg - installing Trac again can completely delete your modifications. Instead use one of two alternatives:

- For a modification to one project only, copy the template to project `templates` directory.
- For a modification shared by several projects, copy the template to a shared location and have each project point to this location using the `[inherit] templates_dir =` trac.ini option.

Trac resolves requests for a template by first looking inside the project, then in any inherited templates location, and finally inside the Trac egg.

Trac caches templates in memory by default to improve performance. To apply a template you need to restart the server.

See also TracGuide, TracIni