Customizing the Trac Interface

Error: Macro TracGuideToc(None) failed

```
'NoneType' object has no attribute 'find'
```

Contents

This page gives suggestions on how to customize the look of Trac. Topics include editing the HTML templates and CSS files, but not the program code itself. The topics 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 your environment's `htdocs` directory. 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.

Next, 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 literal prefix. 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 are small images displayed by your web browser next to the site's URL and in the `Bookmarks` menu. 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:

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

Custom Navigation Entries

The `[mainnav]` and `[metanav]` sections of trac.ini be used to customize the navigation items' text and link, or even 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 navigation.

Site Appearance

Trac is using ?Genshi as the templating engine. 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`), eg `/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('@*')">
    ${select('*|comment()|text()')}
    <link rel="stylesheet" href="${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>
```

Notice that XSLT bears some similarities with Genshi templates. However, there are some Trac specific features, for example the `${href.chrome('site/style.css')}` attribute references `style.css` in the environment's `htdocs/` directory. In a similar fashion `${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 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 into specific sections. See ?this thread for a detailed explanation of the above example `site.html`. A `site.html` can contain any number of `py:match` sections. 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.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.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.path_info == '/timeline'` as the condition in a `<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.

Note that the `site.html`, despite its name, can be put in a shared templates directory, see the [inherit] templates_dir option. This could provide easier maintainence 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:

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, ie instance of a project, can be customized independently of other projects, even those hosted on the same server. The recommended way is to use a `site.html` template whenever possible, see #SiteAppearance. 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 in the Trac egg or wheel, such as `/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 or wheel's `trac/htdocs` directory.

However, do not edit templates or site resources inside the Trac egg/wheel. Reinstalling Trac overwrites your modifications. Instead use one of these 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 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 or wheel.

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

See also TracIni, TracNavigation