Trac Installation Guide for 0.12

Error: Macro TracGuideToc(None) failed

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

Trac is written in the Python programming language and needs a database, SQLite, PostgreSQL, or MySQL. For HTML rendering, Trac uses the Genshi templating system.

Since version 0.12, Trac can also be localized, and there's probably a translation available for your language. If you want to be able to use the Trac interface in other languages, then make sure you have installed the optional package Babel. Pay attention to the extra steps for localization support in the Installing Trac section below. Lacking Babel, you will only get the default english version, as usual.

If you're interested in contributing new translations for other languages or enhance the existing translations, then please have a look at TracL10N.

What follows are generic instructions for installing and setting up Trac and its requirements. While you may find instructions for installing Trac on specific systems at TracInstallPlatforms? on the main Trac site, please be sure to **first read through these general instructions** to get a good understanding of the tasks involved.

Installation Steps

1. Dependencies
    1. Mandatory Dependencies
    2. Optional Dependencies
2. Installing Trac
    1. Using `easy_install`
    2. Using `pip`
    3. From source
    4. Advanced Options
3. Creating a Project Environment
4. Running the Standalone Server
5. Running Trac on a Web Server
6. Configuring Authentication
7. Automatic reference to the SVN changesets in Trac tickets
8. Using Trac

Dependencies

Mandatory Dependencies

To install Trac, the following software packages must be installed:

- Python, version >= 2.4 and < 3.0 (note that we dropped the support for Python 2.3 in this release)
- setuptools, version >= 0.6
- Genshi, version >= 0.6

You also need a database system and the corresponding python bindings. The database can be either SQLite, PostgreSQL or MySQL.

For the SQLite database

If you're using Python 2.5 or 2.6, you already have everything you need.

If you're using Python 2.4 and need pysqlite, you can download from google code the Windows installers or the tar.gz archive for building from source:

```
$ tar xvfz <version>.tar.gz
$ cd <version>
$ python setup.py build_static install
```

This will extract the SQLite code and build the bindings.

SQLite 2.x is no longer supported. For SQLite 3.x, the pysqlite 1.1.x bindings are also no longer supported, use pysqlite 2.x.

See additional information in PySqlite.

For the PostgreSQL database

You need to install the database and its Python bindings:

- PostgreSQL
- psycopg2

See DatabaseBackend for details.

For the MySQL database

Trac can now work quite well with MySQL, provided you follow the guidelines.

- MySQL, version 5.0 or later
- MySQLdb, version 1.2.2 or later

It is **very** important to read carefully the MySqlDb page before creating the database.

Optional Dependencies

Version Control System

Subversion

- Subversion, 1.5.x or 1.6.x and the ***corresponding*** Python bindings. Older versions starting from 1.0, like 1.2.4, 1.3.2 or 1.4.2, etc. should still work. For troubleshooting information, check the TracSubversion page.

There are pre-compiled SWIG bindings available for various platforms. Note that Trac **doesn't** use PySVN, neither does it work yet with the newer `ctype`-style bindings

Please note: if using Subversion, Trac must be installed on the **same machine**. Remote repositories are currently not supported.

Others

Support for other version control systems is provided via third-parties. See PluginList and VersioningSystemBackend.

Web Server

A web server is optional because Trac is shipped with a server included, see the Running the Standalone Server section below.

Alternatively you configure Trac to run in any of the following environments.

- Apache with
  - mod_wsgi, see TracModWSGI and http://code.google.com/p/modwsgi/wiki/IntegrationWithTrac
  - mod_python 3.3.1, see TracModPython)
- a FastCGI-capable web server (see TracFastCgi)
- an AJP-capable web server (see TracOnWindowsIisAjp)
- a CGI-capable web server (see TracCgi), **but usage of Trac as a cgi script is highly discouraged**, better use one of the previous options.

Other Python Packages

- Babel, version >= 0.9.5, needed for localization support
- docutils, version >= 0.3.9 for WikiRestructuredText.
- Pygments for syntax highlighting. SilverCity and/or Enscript may still be used but are deprecated and you really should be using Pygments.
- pytz to get a complete list of time zones, otherwise Trac will fall back on a shorter list from an internal time zone implementation.

Attention: The various available versions of these dependencies are not necessarily interchangable, so please pay attention to the version numbers above. If you are having trouble getting Trac to work please double-check all the dependencies before asking for help on the MailingList or IrcChannel.

Please refer to the documentation of these packages to find out how they are best installed. In addition, most of the platform-specific instructions also describe the installation of the dependencies. Keep in mind however that the information there *probably concern older versions of Trac than the one you're installing* (there are even some pages that are still talking about Trac 0.8!).

Installing Trac

Using `easy_install`

One way to install Trac is using `setuptools`. With setuptools you can install Trac from the subversion repository;

A few examples:

- install Trac 0.12:

  ```
  easy_install Trac==0.12
  ```

- install latest development version 0.12dev:

  ```
  easy_install Trac==dev
  ```

  Note that in this case you won't have the possibility to run a localized version of Trac; either use a released version or install from source

Using `pip`

'pip' is an easy_install replacement that is very useful to quickly install python packages. To get a trac installation up and running in less than 5 minutes:

Assuming you want to have your entire pip installation in /opt/user/trac

- `pip -E /opt/user/trac install trac psycopg2`

or

- `pip -E /opt/user/trac install trac mysql-python`

Make sure your OS specific headers are available for pip to automatically build PostgreSQL (libpq-dev) or MySQL (libmysqlclient-dev) bindings.

pip will automatically resolve all dependencies (like Genshi, pygments, etc.) and download the latest packages on pypi.python.org and create a self contained installation in /opt/user/trac

All commands (tracd, trac-admin) are available in /opt/user/trac/bin . This can also be leveraged for mod_python (using PythonHandler? directive) and mod_wsgi (using WSGIDaemonProcess directive)

Additionally, you can install several trac plugins (listed here) through pip.

From source

Of course, using the python-typical setup at the top of the source directory also works.

You can obtain the source for a .tar.gz or .zip file corresponding to a release (e.g. Trac-0.12.tar.gz), or you can get the source directly from the repository (see Trac:SubversionRepository for details).

```
$ python ./setup.py install
```

You'll need root permissions or equivalent for this step.

This will byte-compile the python source code and install it as an .egg file or folder in the `site-packages` directory of your Python installation. The .egg will also contain all other resources needed by standard Trac, such as htdocs and templates.

The script will also install the trac-admin command-line tool, used to create and maintain project environments, as well as the tracd standalone server.

If you install from source and want to make Trac available in other languages, make sure Babel is installed. Only then, perform the `install` (or simply redo the `install` once again afterwards if you realize Babel was not yet installed):

```
$ python ./setup.py install
```

Alternatively, you can do a `bdist_egg` and copy the .egg from dist/ to the place of your choice, or you can create a Windows installer (`bdist_wininst`).

Advanced Options

To install Trac to a custom location, or find out about other advanced installation options, run:

```
easy_install --help
```

Also see Installing Python Modules for detailed information.

Specifically, you might be interested in:

```
easy_install --prefix=/path/to/installdir
```

or, if installing Trac to a Mac OS X system:

```
easy_install --prefix=/usr/local --install-dir=/Library/Python/2.5/site-packages
```

Note: If installing on Mac OS X 10.6 running `easy_install http://svn.edgewall.org/repos/trac/trunk` will install into `/usr/local` and `/Library/Python/2.6/site-packages` by default

The above will place your `tracd` and `trac-admin` commands into `/usr/local/bin` and will install the Trac libraries and dependencies into `/Library/Python/2.5/site-packages`, which is Apple's preferred location for third-party Python application installations.

Creating a Project Environment

A Trac environment is the backend storage where Trac stores information like wiki pages, tickets, reports, settings, etc. An environment is basically a directory that contains a human-readable configuration file, and various other files and directories.

A new environment is created using trac-admin:

```
$ trac-admin /path/to/myproject initenv
```

trac-admin will prompt you for the information it needs to create the environment, such as the name of the project and the database connection string. If you're not sure what to specify for one of these options, just press `<Enter>` to use the default value.

Using the default database connection string in particular will always work as long as you have SQLite installed. For the other database backends? you should plan ahead and already have a database ready to use at this point.

Since 0.12, Trac doesn't ask for a source code repository anymore when creating an environment. Repositories can be added afterward, or the version control support can be disabled completely if you don't need it.

Also note that the values you specify here can be changed later by directly editing the conf/trac.ini configuration file.

Finally, make sure the user account under which the web front-end runs will have **write permissions** to the environment directory and all the files inside. This will be the case if you run `trac-admin ... initenv` as this user. If not, you should set the correct user afterwards. For example on Linux, with the web server running as user `apache` and group `apache`, enter:

```
# chown -R apache.apache /path/to/myproject
```

Running the Standalone Server

After having created a Trac environment, you can easily try the web interface by running the standalone server tracd:

```
$ tracd --port 8000 /path/to/myproject
```

Then, fire up a browser and visit `http://localhost:8000/`. You should get a simple listing of all environments that `tracd` knows about. Follow the link to the environment you just created, and you should see Trac in action. If you only plan on managing a single project with Trac you can have the standalone server skip the environment list by starting it like this:

```
$ tracd -s --port 8000 /path/to/myproject
```

Running Trac on a Web Server

Trac provides various options for connecting to a "real" web server: CGI, FastCGI, mod_wsgi and mod_python. For decent performance, it is recommended that you use either FastCGI or mod_wsgi.

Trac also supports AJP which may be your choice if you want to connect to IIS.

Generating the Trac cgi-bin directory

In order for Trac to function properly with FastCGI you need to have a `trac.fcgi` file and for mod_wsgi a `trac.wsgi` file. These are Python scripts which load the appropriate Python code. They can be generated using the `deploy` option of trac-admin.

There is, however, a bit of a chicken-and-egg problem. The trac-admin command requires an existing environment to function, but complains if the deploy directory already exists. This is a problem, because environments are often stored in a subdirectory of the deploy. The solution is to do something like this:

```
mkdir -p /usr/share/trac/projects/my-project
trac-admin /usr/share/trac/projects/my-project initenv
trac-admin /usr/share/trac/projects/my-project deploy /tmp/deploy
mv /tmp/deploy/* /usr/share/trac
```

Setting up the Plugin Cache

Some Python plugins need to be extracted to a cache directory. By default the cache resides in the home directory of the current user. When running Trac on a Web Server as a dedicated user (which is highly recommended) who has no home directory, this might prevent the plugins from starting. To override the cache location you can set the PYTHON_EGG_CACHE environment variable. Refer to your server documentation for detailed instructions on how to set environment variables.

Configuring Authentication

The process of adding, removing, and configuring user accounts for authentication depends on the specific way you run Trac. The basic procedure is described in the Adding Authentication section on the TracCgi page. To learn how to setup authentication for the frontend you're using, please refer to one of the following pages:

- TracStandalone if you use the standalone server, `tracd`.
- TracCgi if you use the CGI or FastCGI web front ends.
- TracModWSGI if you use the Apache mod_wsgi web front end.
- TracModPython if you use the Apache mod_python web front end.

Automatic reference to the SVN changesets in Trac tickets

You can configure SVN to automatically add a reference to the changeset into the ticket comments, whenever changes are committed to the repository. The description of the commit needs to contain one of the following formulas:

- **`Refs #123`** - to reference this changeset in `#123` ticket
- **`Fixes #123`** - to reference this changeset and close `#123` ticket with the default status *fixed*

This functionality requires a post-commit hook to be installed as described in TracRepositoryAdmin, and enabling the optional commit updater components by adding the following line to the `[components]` section of your trac.ini, or enabling the components in the "Plugins" admin panel.

```
tracopt.ticket.commit_updater.* = enabled
```

For more information, see the documentation of the `CommitTicketUpdater` component in the "Plugins" admin panel.

Using Trac

Once you have your Trac site up and running, you should be able to create tickets, view the timeline, browse your version control repository if configured, etc.

Keep in mind that anonymous (not logged in) users can by default access most but not all of the features. You will need to configure authentication and grant additional permissions to authenticated users to see the full set of features.

Enjoy!

The Trac Team

See also: TracInstallPlatforms, TracGuide, TracCgi, TracFastCgi, TracModPython, TracModWSGI, TracUpgrade, TracPermissions