Trac Installation Guide for 0.11

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, ?MySQL. For HTML rendering, Trac uses the ?Genshi templating system.

What follows are generic instructions for installing and setting up Trac and its requirements. While you can 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.

See TracUpgrade for instructions on how to upgrade an existing installation.

Quick Install a Released Version

For a quick install, first make sure you have ?Python (2.3-2.6) and ?easy_install.

Then enter (*omitting 'sudo' if not applicable*)

```
sudo easy_install Trac
```

to install Trac, SQLite, and Genshi.

Requirements

The hardware requirements for running Trac obviously depend on the expected data volume (number of wiki pages, tickets, revisions) and traffic. Very small projects will run fine with a 500MHz processor and 128MB RAM using SQLite. In general, the more RAM, the better. A fast hard disk also helps.

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

- ?Python, version >=2.3 (<3.0)
  - if using mod_python together with xml-related things, use python-2.5. expat is namespaced there and does not cause apache to crash any more(see ?here for details).
  - For RPM-based systems you might also need the `python-devel` and `python-xml` packages.
  - See instructions in ?TracOnWindows/Python2.5
- setuptools?, version >= 0.6
- ?Genshi, version >= 0.5 (was version >= 0.4.1 on previous 0.11 release candidates)
- You also need a database system and the corresponding python drivers for it. The database can be either SQLite, PostgreSQL or MySQL.
- Optional if some plugins require it: ?ClearSilver

For SQLite

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

If you're using Python 2.3 or 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
```

That way, the latest SQLite version will be downloaded and built into the bindings.

If you're still using SQLite 2.x, you'll need pysqlite 1.0.x, although this package is not easy to find anymore. For SQLite 3.x, try not to use pysqlite 1.1.x, which has been deprecated in favor of pysqlite 2.x.

See additional information in ?PySqlite.

For PostgreSQL

- ?PostgreSQL
- ?psycopg2
- See ?DatabaseBackend

Warning: PostgreSQL 8.3 uses a strict type checking mechanism. To use Trac with the 8.3 Version of PostgreSQL, you will need ?trac-0.11 or later.

For MySQL

- ?MySQL, version 4.1 or later (?MariaDB might work as well)
- ?MySQLdb, version 1.2.1 or later

See ?MySqlDb for more detailed information. It is *very* important to read carefully that page before creating the database.

Optional Requirements

Version Control System

Please note: if using Subversion, Trac must be installed on the **same machine**. Remote repositories are currently not supported (although Windows UNC paths such as `\\machine_name\path\to\svn` do work).

- ?Subversion, version >= 1.0. (versions recommended: 1.2.4, 1.3.2 or 1.4.2) and the ***corresponding*** Python bindings. For troubleshooting, check ?TracSubversion
    - Trac uses the ?SWIG bindings included in the Subversion distribution, **not** ?PySVN (which is sometimes confused with the standard SWIG bindings).
    - If Subversion was already installed without the SWIG bindings, on Unix you'll need to re-`configure` Subversion and `make swig-py`, `make install-swig-py`.
    - There are ?pre-compiled bindings available for win32.
- Support for other version control systems is provided via third-parties. See ?PluginList and ?VersioningSystemBackend.

Web Server

- A CGI-capable web server (see TracCgi), or
- a ?FastCGI-capable web server (see TracFastCgi), or
- an ?AJP-capable web server (see ?TracOnWindowsIisAjp), or
- ?Apache with ?mod_wsgi (see TracModWSGI or ?http://code.google.com/p/modwsgi/wiki/IntegrationWithTrac) or
    - This should work with Apache 1.3, 2.0 or 2.2 and promises to deliver more performance than using mod_python. A little less mature than mod_python.
- ?Apache with ?mod_python 3.1.3+ (see TracModPython)

- When installing mod_python the development versions of Python and Apache are required (actually the libraries and header files)

For those stuck with Apache 1.3, it is also possible to get Trac working with ?mod_python 2.7 (see ?TracModPython2.7). This guide hasn't been updated since 0.84, so it may or may not work.

Other Python Utilities

- ?docutils, version >= 0.3.9 for WikiRestructuredText.
- ?Pygments for **syntax highlighting**, although ?SilverCity >= 0.9.7 and/or ?GNU Enscript are also possible. Refer to TracSyntaxColoring for details.
- ?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

One way to install Trac is using `setuptools`. With setuptools you can install Trac from the subversion repository; for example, to install release version 0.11 do:

```
easy_install http://svn.edgewall.org/repos/trac/tags/trac-0.11
```

But of course the python-typical setup at the top of the source directory also works:

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

Note: 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.

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
```

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, the type and the path to an existing source code repository, the database connection string, and so on. If you're not sure what to specify for one of these options, just leave it blank to use the default value. The database connection string in particular will always work as long as you have SQLite installed. Leaving the path to the source code repository empty will disable any functionality related to version control, but you can always add that back when the basic system is running.

Also note that the values you specify here can be changed later by directly editing the TracIni configuration file.

Note: The user account under which the web server runs will require write permissions to the environment directory and all the files inside. 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 three options for connecting to a "real" web server: CGI, FastCGI and mod_python. For decent performance, it is recommended that you use either FastCGI or mod_python.

If you're not afraid of running newer code, you can also try running Trac on mod_wsgi. This should deliver even better performance than mod_python, but the module isn't as extensively tested as mod_python.

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 or mod_python, you need to have a trac.cgi file. This is an executable which loads the appropriate Python code. It 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.

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 methods.
- TracModPython if you use the mod_python method.

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 files 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*

All you have to do is to edit the *post-commit* hook in your SVN repository and make it execute *trac-post-commit-hook* coming with Trac.

If you are editing the *post-commit* hook for the first time you need to navigate to your SVN repository's hooks subfolder and rename existing there *post-commit* template:

```
$ cd /path/to/svn/repository/hooks
$ mv post-commit.tmpl post-commit
$ chmod 755 post-commit
```

Next open it in any text editor and add a line with path to the Trac environment connected with this SVN repository and another line executing the *trac-post-commit-hook* script:

```
REPOS="$1"
REV="$2"
TRAC_ENV="/path/to/your/trac/project"

/usr/bin/python /usr/local/bin/trac-post-commit-hook -p "$TRAC_ENV" -r "$REV"
```

Make sure that *trac-post-commit-hook* exists in above path with execution permissions for the same user which SVN is running from. This script can be found in contrib subfolder of your Trac distribution and the latest version can be always downloaded from trunk/contrib/trac-post-commit-hook.

Platform-specifics installations

- See ?TracInstallPlatforms

Using Trac

Once you have your Trac site up and running, you should be able to browse your subversion repository, create tickets, view the timeline, 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