Trac Installation Guide for 1.2

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.

Trac can also be localized, and there is probably a translation available in your language. If you want 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.

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

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

Table of Contents

Dependencies

Mandatory Dependencies

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

- Python, version >= 2.6 and < 3.0 (note that we dropped the support for Python 2.5 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

As you must be using Python 2.6 or 2.7, you already have the SQLite database bindings bundled with the standard distribution of Python (the `sqlite3` module).

Optionally, you may install a newer version of pysqlite than the one provided by the Python distribution. See PySqlite for details.

For the PostgreSQL database

You need to install the database and its Python bindings:

- PostgreSQL, version 8.0 or later
- psycopg2, version 2.0 or later

See DatabaseBackend for details.

For the MySQL database

Trac works well with MySQL, provided you follow the guidelines:

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

Given the caveats and known issues surrounding MySQL, read carefully the MySqlDb page before creating the database.

Optional Dependencies

Subversion

Subversion, 1.6.x or later and the ***corresponding*** Python bindings.

There are pre-compiled SWIG bindings available for various platforms. (Good luck finding precompiled SWIG bindings for any Windows package at that listing. TracSubversion points you to Alagazam, which works for me under Python 2.6.)

For troubleshooting information, see the TracSubversion page.

Note:

- Trac **doesn't** use PySVN, nor does it work yet with the newer `ctype`-style bindings.
- If using Subversion, Trac must be installed on the **same machine**. Remote repositories are currently not supported.

Git

Git 1.5.6 or later is supported. More information is available on the TracGit page.

Other Version Control Systems

Support for other version control systems is provided via third-party plugins. See PluginList#VersionControlSystems and VersionControlSystem.

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 can configure Trac to run in any of the following environments:

- ■Apache with
  - ■mod_wsgi, see TracModWSGI and ■ModWSGI IntegrationWithTrac.
  - ■mod_python 3.5.0, see TracModPython
- a ■FastCGI-capable web server (see TracFastCgi)
- an ■AJP-capable web server (see ■TracOnWindowsIisAjp)
- Microsoft IIS with FastCGI and a FastCGI-to-WSGI gateway (see ■IIS with FastCGI)
- 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 0.9.6 or >= 1.3, needed for localization support
- ■docutils, version >= 0.3.9 for WikiRestructuredText.
- ■Pygments for syntax highlighting.
- ■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 available versions of these dependencies are not necessarily interchangeable, so please pay attention to the version numbers. 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*.

Installing Trac

The trac-admin command-line tool, used to create and maintain project environments, as well as the tracd standalone server are installed along with Trac. There are several methods for installing Trac.

It is assumed throughout this guide that you have elevated permissions as the `root` user or by prefixing commands with `sudo`. The umask `0002` should be used for a typical installation on a Unix-based platform.

Using `easy_install`

Trac can be installed from PyPI or the Subversion repository using ■setuptools.

A few examples:

- Install the latest stable version of Trac:

  ```
  $ easy_install Trac
  ```

- Install latest development version:

  ```
  $ easy_install http://download.edgewall.org/trac/Trac-latest-dev.tar.gz
  ```

  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

More information can be found on the ■setuptools page.

> **Setuptools Warning:** If the version of your setuptools is in the range 5.4 through 5.6, the environment variable `PKG_RESOURCES_CACHE_ZIP_MANIFESTS` must be set in order to avoid significant performance degradation. More information may be found in Deploying Trac.

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 install trac psycopg2
```

or

```
$ pip 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.), download the latest packages from 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

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-1.0.tar.gz`) from the ■TracDownload page, or you can get the source directly from the repository. See ■TracRepositories for details.

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

You will 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`.

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

Using installer

On Windows, Trac can be installed using the exe installers available on the ■TracDownload page. Installers are available for the 32-bit and 64-bit versions of Python. Make sure to use the installer that matches the architecture of your Python installation.

Using package manager

Trac may be available in your platform's package repository. Note however, that the version provided by your package manager may not be the latest release.

Advanced `easy_install` 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 on a Mac OS X system:

```
$ easy_install --prefix=/usr/local --install-dir=/Library/Python/2.6/site-packages
```

Mac OS X Note: On Mac OS X 10.6, running `easy_install trac` will install into `/usr/local` and `/Library/Python/2.6/site-packages` by default.

The `tracd` and `trac-admin` commands will be placed in `/usr/local/bin` and will install the Trac libraries and dependencies into `/Library/Python/2.6/site-packages`, which is Apple's preferred location for third-party Python application installations.

Creating a Project Environment

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

A new environment is created using trac-admin:

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

You will be prompted for the information needed to create the environment: the name of the project and the database connection string. If you're not sure what to specify for any of these options, just press `<Enter>` to use the default value.

Using the default database connection string 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.

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

Filesystem Warning: When selecting the location of your environment, make sure that the filesystem on which the environment directory resides supports sub-second timestamps (i.e. **not** `ext2` or `ext3` on Linux, or HFS+ on OSX), as the modification time of the `conf/trac.ini` file will be monitored to decide whether an environment restart is needed or not. A too coarse-grained timestamp resolution may result in inconsistencies in Trac < 1.0.2. The best advice is to opt for a platform with sub-second timestamp resolution, regardless of the Trac version.

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
```

The actual username and groupname of the apache server may not be exactly `apache`, and are specified in the Apache configuration file by the directives `User` and `Group` (if Apache `httpd` is what you use).

Warning: Please only use ASCII-characters for account name and project path, unicode characters are not supported there.

Deploying Trac

Setuptools Warning: If the version of your setuptools is in the range 5.4 through 5.6, the environment variable `PKG_RESOURCES_CACHE_ZIP_MANIFESTS` must be set in order to avoid significant performance degradation.

If running `tracd`, the environment variable can be set system-wide or for just the user that runs the `tracd` process. There are several ways to accomplish this in addition to what is discussed here, and depending on the distribution of your OS.

To be effective system-wide a shell script with the `export` statement may be added to `/etc/profile.d`. To be effective for a user session the `export` statement may be added to `~/.profile`.

```
export PKG_RESOURCES_CACHE_ZIP_MANIFESTS=1
```

Alternatively, the variable can be set in the shell before executing `tracd`:

```
$ PKG_RESOURCES_CACHE_ZIP_MANIFESTS=1 tracd --port 8000 /path/to/myproject
```

If running the Apache web server, Ubuntu/Debian users should add the `export` statement to `/etc/apache2/envvars`. RedHat/CentOS/Fedora should can add the `export` statement to `/etc/sysconfig/httpd`.

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:

- FastCGI
- Apache with mod_wsgi
- Apache with mod_python
- CGI *(should not be used, as the performance is far from optimal)*

Trac also supports ■AJP which may be your choice if you want to connect to IIS. Other deployment scenarios are possible: ■nginx, ■uwsgi, ■Isapi-wsgi etc.

Generating the Trac cgi-bin directory

Application scripts for CGI, FastCGI and mod-wsgi can be generated using the trac-admin `deploy` command:

```
deploy <directory>

   Extract static resources from Trac and all plugins
```

Grant the web server execution right on scripts in the `cgi-bin` directory.

For example, the following yields a typical directory structure:

```
$ mkdir -p /var/trac
$ trac-admin /var/trac/<project> initenv
$ trac-admin /var/trac/<project> deploy /var/www
$ ls /var/www
cgi-bin htdocs
$ chmod ugo+x /var/www/cgi-bin/*
```

Mapping Static Resources

Without additional configuration, Trac will handle requests for static resources such as stylesheets and images. For anything other than a TracStandalone deployment, this is not optimal as the web server can be set up to directly serve the static resources. For CGI setup, this is **highly undesirable** as it causes abysmal performance.

Web servers such as ■Apache allow you to create *Aliases* to resources, giving them a virtual URL that doesn't necessarily reflect their location on the file system. We can map requests for static resources directly to directories on the file system, to avoid Trac processing the requests.

There are two primary URL paths for static resources: `/chrome/common` and `/chrome/site`. Plugins can add their own resources, usually accessible at the `/chrome/<plugin>` path.

A single `/chrome` alias can used if the static resources are extracted for all plugins. This means that the `deploy` command (discussed in the previous section) must be executed after installing or updating a plugin that provides static resources, or after modifying resources in the `$env/htdocs` directory. This is probably appropriate for most installations but may not be what you want if, for example, you wish to upload plugins through the *Plugins* administration page.

The `deploy` command creates an `htdocs` directory with:

- `common/` - the static resources of Trac
- `site/` - a copy of the environment's `htdocs/` directory
- `shared` - the static resources shared by multiple Trac environments, with a location defined by the `[inherit] htdocs_dir` option
- `<plugin>/` - one directory for each resource directory provided by the plugins enabled for this environment

The example that follows will create a single `/chrome` alias. If that isn't the correct approach for your installation you simply need to create more specific aliases:

```
Alias /trac/chrome/common /path/to/trac/htdocs/common
Alias /trac/chrome/site /path/to/trac/htdocs/site
Alias /trac/chrome/shared /path/to/trac/htdocs/shared
Alias /trac/chrome/<plugin> /path/to/trac/htdocs/<plugin>
```

Example: Apache and `ScriptAlias`

Assuming the deployment has been done this way:

```
$ trac-admin /var/trac/<project> deploy /var/www
```

Add the following snippet to Apache configuration, changing paths to match your deployment. The snippet must be placed *before* the `ScriptAlias` or `WSGIScriptAlias` directive, because those directives map all requests to the Trac application:

```
Alias /trac/chrome /path/to/trac/htdocs

<Directory "/path/to/www/trac/htdocs">
 # For Apache 2.2
 <IfModule !mod_authz_core.c>
   Order allow,deny
   Allow from all
 </IfModule>
 # For Apache 2.4
 <IfModule mod_authz_core.c>
   Require all granted
 </IfModule>
</Directory>
```

If using mod_python, add this too, otherwise the alias will be ignored:

```
<Location "/trac/chrome/common">
 SetHandler None
</Location>
```

Alternatively, if you wish to serve static resources directly from your project's `htdocs` directory rather than the location to which the files are extracted with the `deploy` command, you can configure Apache to serve those resources. Again, put this *before* the `ScriptAlias` or `WSGIScriptAlias` for the .*cgi scripts, and adjust names and locations to match your installation:

```
Alias /trac/chrome/site /path/to/projectenv/htdocs

<Directory "/path/to/projectenv/htdocs">
 # For Apache 2.2
 <IfModule !mod_authz_core.c>
   Order allow,deny
   Allow from all
 </IfModule>
 # For Apache 2.4
 <IfModule mod_authz_core.c>
   Require all granted
 </IfModule>
</Directory>
```

Another alternative to aliasing `/trac/chrome/common` is having Trac generate direct links for those static resources (and only those), using the htdocs_location configuration setting:

```
[trac]
htdocs_location = http://static.example.org/trac-common/
```

Note that this makes it easy to have a dedicated domain serve those static resources, preferentially cookie-less.

Of course, you still need to make the Trac `htdocs/common` directory available through the web server at the specified URL, for example by copying (or linking) the directory into the document root of the web server:

```
$ ln -s /path/to/trac/htdocs/common /var/www/static.example.org/trac-common
```

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.

If you setup hook scripts that call Trac, such as the Subversion post-commit hook script provided in the `/contrib` directory, make sure you define the `PYTHON_EGG_CACHE` environment variable within these scripts as well.

Configuring Authentication

Trac uses HTTP authentication. You'll need to configure your webserver to request authentication when the `.../login` URL is hit (the virtual path of the "login" button). Trac will automatically pick the `REMOTE_USER` variable up after you provide your credentials. Therefore, all user management goes through your web server configuration. Please consult the documentation of your web server for more info.

The process of adding, removing, and configuring user accounts for authentication depends on the specific way you run Trac.

Please refer to one of the following sections:

- TracStandalone#UsingAuthentication if you use the standalone server, `tracd`.
- TracModWSGI#ConfiguringAuthentication if you use the Apache web server, with any of its front end: `mod_wsgi`, `mod_python`, `mod_fcgi` or `mod_fastcgi`.
- TracFastCgi if you're using another web server with FCGI support (Cherokee, Lighttpd, LiteSpeed, nginx)

■TracAuthenticationIntroduction also contains some useful information for beginners.

Granting admin rights to the admin user

Grant admin rights to user admin:

```
$ trac-admin /path/to/myproject permission add admin TRAC_ADMIN
```

This user will have an *Admin* navigation item that directs to pages for administering your Trac project.

Configuring Trac

TracRepositoryAdmin provides information on configuring version control repositories for your project.

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 only a few of the features, in particular they will have a read-only access to the resources. 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, TracUpgrade, TracPermissions