The Trac Environment

Trac uses a directory structure and a database for storing project data. The directory is referred to as the ?environment?.

Creating an Environment

A new Trac environment is created using trac-admin's initenv:

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

`trac-admin` will ask you for the name of the project and the database connection string (explained below).

Some Useful Tips

- The user under which the web server runs will require file system write permission to the environment directory and all the files inside. Please remember to set the appropriate permissions. The same applies to the source code repository, although the user under which Trac runs will only require write access to a Subversion repository created with the BDB file system; for other repository types, check the corresponding plugin's documentation.
- `initenv`, when using an svn repository, does not imply that trac-admin will perform `svnadmin create` for the specified repository path. You need to perform the `svnadmin create` prior to `trac-admin initenv` if you're creating a new svn repository altogether with a new trac environment, otherwise you will see a message "Warning: couldn't index the repository" when initializing the environment.
- Non-ascii environment paths are not supported
- Also, it seems that project names with spaces can be problematic for authentication (see ?#7163).
- TracPlugins located in a shared plugins folder that is defined in an inherited configuration are currently not loaded during creation, and hence, if they need to create extra tables for example, you'll need to upgrade the environment before being able to use it.

Database Connection Strings

Since version 0.9, Trac supports both ?SQLite and ?PostgreSQL database backends. Preliminary support for ?MySQL was added in 0.10. The default is to use SQLite, which is probably sufficient for most projects. The database file is then stored in the environment directory, and can easily be backed up together with the rest of the environment.

SQLite Connection String

The connection string for an SQLite database is:

```
sqlite:db/trac.db
```

where `db/trac.db` is the path to the database file within the Trac environment.

PostgreSQL Connection String

If you want to use PostgreSQL or MySQL instead, you'll have to use a different connection string. For example, to connect to a PostgreSQL database on the same machine called `trac`, that allows access to the user `johndoe` with the password `letmein`, use:

```
postgres://johndoe:letmein@localhost/trac
```

Note that due to the way the above string is parsed, the "/" and "@" characters cannot be part of the password.

If PostgreSQL is running on a non-standard port (for example 9342), use:

```
postgres://johndoe:letmein@localhost:9342/trac
```

On UNIX, you might want to select a UNIX socket for the transport, either the default socket as defined by the PGHOST environment variable:

```
postgres://user:password@/database
```

or a specific one:

```
postgres://user:password@/database?host=/path/to/socket/dir
```

Note that with PostgreSQL you will have to create the database before running `trac-admin initenv`.

See the ?PostgreSQL documentation for detailed instructions on how to administer ?PostgreSQL. Generally, the following is sufficient to create a database user named `tracuser`, and a database named `trac`.

```
createuser -U postgres -E -P tracuser
createdb -U postgres -O tracuser -E UTF8 trac
```

When running `createuser` you will be prompted for the password for the user 'tracuser'. This new user will not be a superuser, will not be allowed to create other databases and will not be allowed to create other roles. These privileges are not needed to run a trac instance. If no password is desired for the user, simply remove the `-P` and `-E` options from the `createuser` command. Also note that the database should be created as UTF8. LATIN1 encoding causes errors trac's use of unicode in trac. SQL_ASCII also seems to work.

Under some default configurations (debian) one will have run the `createuser` and `createdb` scripts as the `postgres` user. For example:

```
sudo su - postgres -c 'createuser -U postgres -S -D -R -E -P tracuser'
sudo su - postgres -c 'createdb -U postgres -O tracuser -E UTF8 trac'
```

Trac uses the `public` schema by default but you can specify a different schema in the connection string:

```
postgres://user:pass@server/database?schema=yourschemaname
```

MySQL Connection String

If you want to use MySQL instead, you'll have to use a different connection string. For example, to connect to a MySQL database on the same machine called `trac`, that allows access to the user `johndoe` with the password `letmein`, the mysql connection string is:

```
mysql://johndoe:letmein@localhost:3306/trac
```

Source Code Repository

Since version 0.12, a single Trac environment can be connected to more than one repository. There are many different ways to connect repositories to an environment, see TracRepositoryAdmin. This page also details the various attributes that can be set for a repository (like `type`, `url`, `description`).

In Trac 0.12 `trac-admin` no longer asks questions related to repositories. Therefore, by default Trac is not connected to any source code repository, and the *Browse Source* toolbar item will not be displayed. You can also explicitly disable the `trac.versioncontrol.*` components (which are otherwise still loaded)

```
[components]
trac.versioncontrol.* = disabled
```

For some version control systems, it is possible to specify not only the path to the repository, but also a *scope* within the repository. Trac will then only show information related to the files and changesets below that scope. The Subversion backend for Trac supports this; for other types, check the corresponding plugin's documentation.

Example of a configuration for a Subversion repository used as the default repository:

```
[trac]
repository_type = svn
repository_dir = /path/to/your/repository
```

The configuration for a scoped Subversion repository would be:

```
[trac]
repository_type = svn
repository_dir = /path/to/your/repository/scope/within/repos
```

Directory Structure

An environment directory will usually consist of the following files and directories:

- `README` - Brief description of the environment.
- `VERSION` - Contains the environment version identifier.
- `attachments` - Attachments to wiki pages and tickets are stored here.
- `conf`
  - `trac.ini` - Main configuration file. See TracIni.
- `db`
  - `trac.db` - The SQLite database (if you're using SQLite).
- `htdocs` - directory containing web resources, which can be referenced in Genshi templates using `/htdocs/site/...` URLs. *(since 0.11)*
- `log` - default directory for log files, if logging is turned on and a relative path is given.
- `plugins` - Environment-specific plugins (Python eggs or single file plugins, since ?0.10)
- `templates` - Custom Genshi environment-specific templates. *(since 0.11)*
  - `site.html` - method to customize header, footer, and style, described in TracInterfaceCustomization#SiteAppearance

Caveat: don't confuse a *Trac environment directory* with the *source code repository directory*

This is a common beginners' mistake. It happens that the structure for a Trac environment is loosely modelled after the Subversion repository directory structure, but those are two disjoint entities and they are not and *must not* be

located at the same place.

See also: TracAdmin, TracBackup, TracIni, TracGuide