Wikiprint Book

Title: Trac plugins

Subject: TechTIDE-Wiki - TracPlugins

Version: 1

Date: 11/25/25 03:27:46

Table of Contents

Frac plugins		3	i
	Plugin discovery	3	į
	Installing a Trac plugin	3	i
	For a single project	3	i
	For all projects	3	i
	With an .egg file	3	
	From source	4	
	Enabling the plugin	4	
	Upgrading the environment	4	
	Redeploying static resources	4	
	Upgrading a Plugin	4	
	Uninstalling	5	
	Setting up the plugin cache	5	
	Web-based plugin administration	5	
	Troubleshooting	5	
	Is setuptools properly installed?	5	
	Did you get the correct version of the Python egg?	5	
	Is the plugin enabled?	6	
	Check the permissions on the .egg file	6	
	Check the log files	6	
	Verify you have the proper permissions	6	i
	Is the wrong version of the plugin loading?	6	i
	If all of the above failed	6	i

Trac plugins

Trac is extensible with plugins. Plugin functionality is based on the component architecture, with special cases described in the plugin development page.

Plugin discovery

From the user's point of view, a plugin is either a standalone .py file or a package (egg or wheel). Trac looks for plugins in Python's site-packages directory, the global shared plugins directory and the project environment plugins directory. Components defined in globally-installed plugins must be explicitly enabled in the [components] section of the trac.ini file. Components defined in the plugins directory of the project environment are enabled, unless explicitly disabled in the [components] section of the trac.ini file.

Installing a Trac plugin

The instructions below are applicable to a plugin packaged as an egg. Plugins implemented as a single py file should be downloaded and copied to the project environment plugins directory or the global shared plugins directory.

For a single project

If you have downloaded a source distribution of a plugin, and want to build the .egg file:

- Unpack the source. It should provide setup.py.
- · Run:

```
$ python setup.py bdist_egg
```

You should now have an *.egg file. Examine the output of running Python to find where this was created.

Once you have the plugin archive, copy it into the plugins directory of the project environment. Also, make sure that the web server has sufficient permissions to read the plugin egg. Then restart the web server. If you are running as a "tracd" standalone server, restart tracd, ie kill the process and run again.

To uninstall a plugin installed this way, remove the egg from the plugins directory and restart the web server.

Note: the Python version that the egg is built with *must* match the Python version with which Trac is run. For example, if you are running Trac under Python 2.6, but have upgraded your standalone Python to 2.7, the eggs won't be recognized.

Note: in a multi-project setup, a pool of Python interpreter instances will be dynamically allocated to projects based on need; since plugins occupy a place in Python's module system, the first version of any given plugin to be loaded will be used for all projects. In other words, you cannot use different versions of a single plugin in two projects of a multi-project setup. It may be safer to install plugins for all projects (see below), and then enable them selectively on a project-by-project basis.

For all projects

With an .egg file

Some plugins, such as TracTags, are downloadable as an .egg file that can be installed with easy_install or pip:

```
$ easy_install TracTags
```

```
$ pip install TracTags
```

If easy_install is not on your system, see the Trac setuptools documentation.

pip is included in Python 2.7.9. In earlier versions of Python it can be installed through the package manager of your OS (e.g. apt-get install python-pip) or using the <u>leget pip.py</u>.

If Trac reports permission errors after installing a zipped egg, and you would rather not bother providing an egg cache directory writable by the web server, you can get around it by simply unzipping the egg. Just pass --always-unzip to easy_install:

```
$ easy_install --always-unzip TracTags
```

You should end up with a directory having the same name as the zipped egg, complete with .egg extension, and containing its uncompressed contents.

Trac also searches for plugins installed in the shared plugins directory, see <u>TracIni#GlobalConfiguration</u>. This is a convenient way to share the installation of plugins across several, but not all, environments.

From source

easy_install and pip make installing from source a snap. Just give it the URL to either a repository or a tarball/zip of the source:

- \$ easy_install https://trac-hacks.org/svn/tagsplugin/trunk
- \$ pip install svn+https://trac-hacks.org/svn/tagsplugin/trunk

When installing from a repository using pip, be sure to use the repository type in the protocol. For example, svn+https for Subversion and git+https for Git.

Enabling the plugin

Unlike plugins installed per environment, you'll have to explicitly enable globally installed plugins via <u>trac.ini</u>. This also applies to plugins installed in the shared plugins directory, ie the path specified in the <u>[inherit] plugins_dir</u> configuration option.

This is done in the [components] section of the configuration file trac.ini. For example:

```
[components]
tractags.* = enabled
```

The name of the option is the Python package of the plugin. This should be specified in the documentation of the plugin, but can also be easily discovered by looking at the source: look for a top-level directory that contains a file named __init__.py.

After installing the plugin, you must restart your web server.

Upgrading the environment

Some plugins may require an environment upgrade. This will typically be necessary for plugins that implement IEnvironmentSetupParticipant.

Common reasons for requiring an environment upgrade are to add tables to the database or add configuration parameters to trac.ini. A notification will be displayed when accessing Trac for the first time after installing a plugin and restarting the web server. To upgrade the environment, run the command:

```
$ trac-admin /path/to/env upgrade
```

A database backup will be made before upgrading the environment, unless the --no-backup option is specified. For more information, refer to the documentation output by trac-admin /path/to/env help upgrade.

Redeploying static resources

If you mapped static resources so they are served by the web server, and the plugin contains static resources (CSS, JavaScript and image files), the resources will need to be deployed to the location on the filesystem that is served by the web server.

Execute the deploy command, as was done during install and upgrade:

```
$ trac-admin /path/to/env deploy /deploy/path
```

After executing the command, you must restart your web server.

Note: Some web browsers (IE, Opera) cache CSS and Javascript files, so you should instruct your users to manually erase the contents of their browser's cache. A forced refreshed (SHIFT + <F5>) should be enough.

Upgrading a Plugin

Normally, upgrading a plugin is simply a matter of repeating the install process. You may want to uninstall old versions of the plugin.

The pip install command has an --upgrade (-U) switch that will uninstall the old version and install the new version. The command can have some unintended side-effects though, because it will also upgrade the plugin dependencies. For example, if Trac is listed as a dependency of the plugin in setup.py, the latest version of Trac will be downloaded and installed. This may not be what you want if you are running an older version of Trac because not all your plugins are compatible with the latest version of Trac, or you simply haven't done the appropriate planning for upgrading Trac. Uninstalling and then installing the plugin can be a safer option:

```
$ pip uninstall <pluginname>
$ pip install <pluginname>
```

Alternatively you can use a requirements file and pin the versions of the packages that you don't want to implicitly upgrade.

Uninstalling

pip makes it easy to uninstall a plugin:

```
$ pip uninstall <pluginname>
```

The pip uninstall command can be used even if the plugin was installed using easy_install or python setup.py install.

Neither easy_install nor python setup.py have an uninstall feature. However, it is usually trivial to remove a globally installed egg and reference:

- i. Do easy_install -m <plugin name> to remove references from \$PYTHONLIB/site-packages/easy-install.pth when the plugin is installed by setuptools.
- ii. Delete executables from /usr/bin, /usr/local/bin, or C:\\Python*\Scripts. To find what executables are involved, refer to the [console-script] section of setup.py.
- iii. Delete the .egg file or folder from where it's installed, usually inside \$PYTHONLIB/site-packages/.
- iv. Restart the web server.

If you are uncertain about the location of the egg file, you can try to locate it by replacing myplugin with whatever namespace the plugin uses (as used when enabling the plugin):

```
>>> import myplugin
>>> print myplugin.__file__
/opt/local/python24/lib/site-packages/myplugin-0.4.2-py2.4.egg/myplugin/__init__.pyc
```

Setting up the plugin cache

Some plugins will need to be extracted by the Python egg's runtime. See TracInstall for information on setting up the egg cache.

Web-based plugin administration

The WebAdmin interface offers limited support for plugin configuration to users with TRAC_ADMIN permission:

- · enabling and disabling installed plugins
- · installing plugins by uploading them as eggs

If you wish to disable the second function for security reasons, add the following to your trac.ini file:

```
[components]
trac.admin.web_ui.PluginAdminPanel = disabled
```

This disables the whole panel, so the first function will no longer be available either.

Troubleshooting

Is setuptools properly installed?

Try this from the command line:

```
$ python -c "import pkg_resources"
```

If you get no output, setuptools is installed. Otherwise, you'll need to install it before plugins will work in Trac.

Did you get the correct version of the Python egg?

Python eggs have the Python version encoded in their filename. For example, MyPlugin-1.0-py2.5.egg is an egg for Python 2.5, and will **not** be loaded if you're running a different Python version (such as 2.4 or 2.6).

Also, verify that the egg file you downloaded is indeed a .zip archive. If you downloaded it from a Trac site, chances are you downloaded the HTML preview page instead.

Is the plugin enabled?

If you install a plugin globally, ie *not* inside the plugins directory of the Trac project environment, you must explicitly enable it in trac.ini. Make sure that:

- you actually added the necessary line(s) to the [components] section.
- the package/module names are correct and do not contain typos.
- the value is "enabled", not "enable" or "Enable".
- the section name is "components", not "component".

Check the permissions on the .egg file

Trac must be able to read the .egg file.

Check the log files

Enable logging and set the log level to DEBUG, then watch the log file for messages about loading plugins.

Verify you have the proper permissions

Some plugins require you have special permissions in order to use them. WebAdmin, for example, requires the user to have TRAC_ADMIN permissions for it to show up on the navigation bar.

Is the wrong version of the plugin loading?

If you put your plugins inside the plugins directories, and certainly if you have more than one project, you need to make sure that the correct version of the plugin is loading. Here are some basic rules:

- Only one version of the plugin can be loaded for each running Trac server, ie each Python process. The Python namespaces and module list will be shared, and it cannot handle duplicates. Whether a plugin is enabled or disabled makes no difference.
- A globally installed plugin (typically setup.py install) will override any version in the global or project plugins directories. A plugin from the global plugins directory will be located *before* any project plugins directory.
- If your Trac server hosts more than one project (as with TRAC_ENV_PARENT_DIR setups), having two versions of a plugin in two different projects will give unpredicatable results. Only one of them will load, and the one loaded will be shared by both projects. Trac will load the first plugin found, usually from the project that receives the first request.
- Having more than one version listed inside Python site-packages is fine, ie installed with setup.py install, because setuptools will make sure you get the version installed most recently. However, don't store more than one version inside a global or project plugins directory: neither the version number nor the installed date will matter at all. There is no way to determine which one will be located first when Trac searches the directory for plugins.

If all of the above failed

Okay, so the logs don't mention plugins, the egg is readable, the Python version is correct, and the egg has been installed globally (and is enabled in trac.ini)... and it still doesn't work or give any error messages or any other indication as to why. Hop on the IrcChannel or MailingList and ask away!

See also <u>TracGuide</u>, <u>■plugin list</u>, <u>■component architecture</u>.