Wikiprint Book

Title: Trac Plugins

Subject: Ecopath Developer Site - TracPlugins

Version: 2

Date: 2024-03-28 23:47:24

Table of Contents

Trac Plugins **3**
- Requirements 3
- Installing a Trac Plugin 3
  - For a Single Project 3
  - For All Projects 3
    - With an .egg file 3
    - From source 4
    - Enabling the plugin 4
    - Uninstalling 4
- Setting up the Plugin Cache 4
  - About hook scripts 5
- Troubleshooting 5
  - Is setuptools properly installed? 5
  - Did you get the correct version of the Python egg? 5
  - Is the plugin enabled? 5
  - Check the permissions on the egg file 5
  - Check the log files 6
  - Verify you have proper permissions 6
  - Is the wrong version of the plugin loading? 6
  - If all of the above failed 6

Trac Plugins

Since version 0.9, Trac supports ?plugins that extend the built-in functionality. The plugin functionality is based on the ?component architecture.

Plugins can be either installed globally, in a shared plugins directory (see Global Configuration) or locally for specific TracEnvironment, in its `plugins` directory. Except for the later case, the components defined in a plugin should be explicitly enabled in the [components] section of the trac.ini file.

Requirements

To use egg based plugins in Trac, you need to have ?setuptools (version 0.6) installed.

To install `setuptools`, download the bootstrap module ?ez_setup.py and execute it as follows:

```
$ python ez_setup.py
```

If the `ez_setup.py` script fails to install the setuptools release, you can download it from ?PyPI and install it manually.

Plugins can also consist of a single `.py` file dropped into either the environment or the shared plugins directory.

Installing a Trac Plugin

For a Single Project

Plugins are packaged as ?Python eggs. That means they are ZIP archives with the file extension `.egg`.

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

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

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

Then you will have a *.egg file. Examine the output of running python to find where this was created.

Once you have the plugin archive, you need to 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 (this requirement was not previously mentioned in this document, but in my tests it began working only after I did so).

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

Note that the Python version that the egg is built with must match the Python version with which Trac is run. If for instance you are running Trac under Python 2.5, but have upgraded your standalone Python to 2.6, the eggs won't be recognized.

Note also that in a multi-project setup, a pool of Python interpreter instances will be dynamically allocated to projects based on need, and 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 the 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 ?SpamFilter) are downloadable as a `.egg` file which can be installed with the `easy_install` program:

```
easy_install TracSpamFilter
```

If `easy_install` is not on your system see the Requirements section above to install it. Windows users will need to add the `Scripts` directory of their Python installation (for example, `C:\Python24\Scripts`) to their `PATH` environment variable (see ?easy_install Windows notes for more information).

If Trac reports permission errors after installing a zipped egg and you would rather not bother providing a 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 TracSpamFilter-0.2.1dev_r5943-py2.4.egg
```

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 *(since 0.10)*, see TracIni#GlobalConfiguration. This is a convenient way to share the installation of plugins across several but not all environments.

From source

`easy_install` makes installing from source a snap. Just give it the URL to either a Subversion repository or a tarball/zip of the source:

```
easy_install http://svn.edgewall.com/repos/trac/plugins/0.11/spam-filter
```

Enabling the plugin

Unlike plugins installed per-environment, you'll have to explicitly enable globally installed plugins via trac.ini. This also applies to plugins installed in shared plugins directory, i.e. the path specified in the `[inherit] plugins_dir` configuration option.

This is done in the `[components]` section of the configuration file, for example:

```
[components]
tracspamfilter.* = 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`.)

Note: After installing the plugin, you need to restart your web server.

Uninstalling

`easy_install` or `python setup.py` does not have an uninstall feature. Hower, it is usually quite trivial to remove a globally installed egg and reference:

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

If you are uncertain about the location of the egg, here is a small tip to help locate an egg (or any package) - replace `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 eggs runtime (`pkg_resources`), so that their contents are actual files on the file system. The directory in which they are extracted defaults to '.python-eggs' in the home directory of the current user, which may or may not be a problem. You can however override the default location using the `PYTHON_EGG_CACHE` environment variable.

To do this from the Apache configuration, use the `SetEnv` directive as follows:

```
SetEnv PYTHON_EGG_CACHE /path/to/dir
```

This works whether you are using the CGI or the mod_python front-end. Put this directive next to where you set the path to the Trac environment, i.e. in the same `<Location>` block.

For example (for CGI):

```
 <Location /trac>
  SetEnv TRAC_ENV /path/to/projenv
  SetEnv PYTHON_EGG_CACHE /path/to/dir
</Location>
```

or (for mod_python):

```
 <Location /trac>
  SetHandler mod_python
  ...
  SetEnv PYTHON_EGG_CACHE /path/to/dir
</Location>
```

Note: SetEnv requires the `mod_env` module which needs to be activated for Apache. In this case the SetEnv directive can also be used in the `mod_python` Location block.

For FastCGI, you'll need to `-initial-env` option, or whatever is provided by your web server for setting environment variables.

Note: that if you already use -initial-env to set the project directory for either a single project or parent you will need to add an additional -initial-env directive to the FastCgiConfig directive. I.e.

```
FastCgiConfig -initial-env TRAC_ENV=/var/lib/trac -initial-env PYTHON_EGG_CACHE=/var/lib/trac/plugin-cache
```

About hook scripts

If you have set up some subversion hook scripts that call the Trac engine - such as the post-commit hook script provided in the `/contrib` directory - make sure you define the `PYTHON_EGG_CACHE` environment variable within these scripts as well.

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 (i.e. *not* inside the `plugins` directory of the Trac project environment) you will have to 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
- the value is ?enabled", not e.g. ?enable?

Check the permissions on the egg file

Trac must be able to read the 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 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 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 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), then having two versions of a plugin in two different projects will give uncertain results. Only one of them will load, and the one loaded will be shared by both projects. Trac will load the first found - basically 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`) - 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 version number nor 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

OK, 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 the trac.ini) and it still doesn't work or give any error messages or any other indication as to why? Hop on the ?IrcChannel and ask away.

See also TracGuide, ?plugin list, ?component architecture