

Trac プラグイン

バージョン 0.9 以降、Trac は [プラグイン機能](#) に対応し、組み込み機能を拡張できるようになりました。プラグインの機能は [コンポーネント設計](#) に基づいています。

必要条件

Trac で Python-egg ベースのプラグインを使用するためには、[setuptools](#) (バージョン 0.6) をインストールしなければなりません。

setuptools をインストールするために、ブートストラップモジュールである [ez_setup.py](#) をダウンロードし、以下に示すように実行して下さい:

```
$ python ez_setup.py
```

もし ez_setup.py スクリプトが setuptools のインストールに失敗したら、[PyPI](#) をダウンロードして手動でインストールしてください。

また、単一の .py ファイルで成り立つプラグインは、各 [TracEnvironment](#) がグローバルの plugin ディレクトリに配置します。 ([milestone:0.10](#) 以降) [TracIni#GlobalConfiguration](#) を参照してください。

Trac プラグインのインストール

単一のプロジェクト

プラグインは [Python eggs](#) としてパッケージ化されています。つまり、拡張子が .egg となっている ZIP アーカイブのファイルです。

plugin のディストリビューションをダウンロードして .egg ファイルをビルドしたいのであれば、以下の通りして下さい:

- ソースをアンパックして下さい。それにより setup.py が提供されるでしょう。
- 以下のようにして実行して下さい:

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

*.egg ファイルが出力されているでしょう。実行した python の出力を調べて .egg ファイルがどこに作成されたか見つけてください。

一度、プラグインアーカイブを作成したら、[TracEnvironment](#) の plugins ディレクトリにコピーする必要があります。また、Web サーバーが egg プラグインを読み取るのに必要なパーミッションをつけてください。

この方法でインストールしたプラグインをアンインストールする場合、plugins ディレクトリから egg を削除し、Web サーバを再起動してください。

Python egg をビルドするための Python のバージョンと Trac を動かしている Python のバージョンが一致しなければなりません。例えば、Trac を Python のバージョン 2.3 以前で動かしていて、2.4 にアップグレードしたときに、Python egg は認識されなくなるでしょう。

マルチプロジェクトを設定している場合、Python

インタプリタインスタンスのプールはプロジェクトの必要に応じて動的にアロケートされ、プラグインは Python

のモジュールシステムの一定の位置を占有しますので、最初にロードされたバージョンのプラグインが、他のすべてのプロジェクトでも使用されます。言い換えれば (下記の方法で) すべてのプロジェクト向けにインストールし、個々のプロジェクトで有効/無効を設定する方が安全でしょう。

すべてのプロジェクト

単一の .egg ファイル

いくつかのプラグイン (例えば [SpamFilter](#)) は .egg ファイルとしてダウンロードし、easy_install プログラムでインストールすることができます:

```
easy_install TracSpamFilter
```

もし、システムに easy_install がなくてプラグインをインストールするには上記の必要条件のセクションを見て下さい。Windows ユーザは Scripts ディレクトリを Python をインストールしたディレクトリ (例えば、C:\Python23\Scripts) を環境変数 PATH に加えなければなりません。(より詳しい情報は、[easy_install の Windows Notes](#) を参照して下さい。)

Zip で固められた Python egg をインストールし、Web サーバに Python egg のキャッシュディレクトリに書き込み権限をつけているのにも関わらず、Trac がパーミッションエラーを出したら、解凍した Python egg を単に置き直すことによって回避できます。easy_install に --always-unzip オプションを付けるだけです:

```
easy_install --always-unzip TracSpamFilter-0.2.1dev_r5943-py2.4.egg
```

zip で固められた Python egg ファイルと同じ名前のディレクトリ (拡張子が .egg で終わっているもの) ができ、そのなかに解凍した中身が入っているでしょう。

Trac はまた、グローバルにインストールされたプラグインを検索します (0.10 以降)。 [TracIni#GlobalConfiguration](#) を参照してください。

ソースから

easy_install をソースからインストールする方法を以下に示します。Subversion のリポジトリか、tarball/zip のソースを引数に与えてください。

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

プラグインを有効にする

個別の [TracEnvironment](#) にプラグインをインストールする場合と異なり、システム全体にインストールしたプラグインを有効にするためには、[trac.ini](#) ファイルで明示的に指定しなければいけません。これは設定ファイルの [components] セクションに次のような記載を行います:

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

オプションの名前はプラグインの Python

パッケージ名です。これはプラグインのドキュメンテーションに指定されていなければいけません。ソースを見れば簡単に見つけることができます。(最上位のディレクトリにあるファイル __init__.py を探してください。)

Note: プラグインのインストール後、Web サーバを再起動する必要があります。

アンインストール

easy_install や python setup.py ではアンインストール機能は提供されていません。しかし、グローバルにインストールされた egg や参照を削除するための簡単な方法があります:

1. setuptools でインストールした場合、easy_install -m [plugin name] を実行し、\$PYTHONLIB/site-packages/easy-install.pth から参照を削除します。
2. 実行可能ファイルを /usr/bin, /usr/local/bin, C:\\Python*\\Scripts などから削除します。実行可能ファイルが分からない場合、setup.py の [console-script] を参照します。
3. インストールされた場所 (通常 \$PYTHONLIB/site-packages/ 配下) の .egg ファイルやディレクトリを削除します。
4. Web サーバを再起動する。

egg の場所が分からない場合、探すためには以下の方法を使います (この方法は、どのようなパッケージでも使えます) - myplugin の箇所は、プラグインのネームスペースで置き換えてください。ネームスペースはプラグインの有効化の時に使用した名前と同じになります。

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

プラグインのキャッシュの設定

いくつかのプラグインは Python eggs のランタイム (pkg_resources) によって解凍される必要があります。というのもそれらの内容がファイルシステム上に実際に存在する必要があるからです。通常、Python eggs は現在のユーザのホームディレクトリに解凍されますが、それにより問題が発生するかもしれません。その場合、環境変数 PYTHON_EGG_CACHE を設定してデフォルトのロケーションを上書きすることができます。

PYTHON_EGG_CACHE を SetEnv ディレクティブを使用して Apache に設定するには以下のようにします:

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

これは [CGI](#) と [mod_python](#) のどちらをフロントエンドにしようと動作します。このディレクティブに [Trac Environment](#) へのパスを設定し、例えば同じ `<Location>` ブロックにおいてください。

例 (CGI用):

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

例 (mod_python用):

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

Note: これは `mod_env` モジュールが必要です。

[FastCGI](#) で、Web サーバに設定するためには、`-initial-env` オプションやサーバが指定している方法で、環境変数を設定する必要があります。

Note: プロジェクトディレクトリを設定するために、既に `-initial-env` を使用している場合は、[TracFastCgi](#) に例示されるように、`trac.fcgi` に少なくとも一つの環境変数を加える必要があります。

Subversion の フックスクリプトについて

もし、Trac エンジンを呼び出すような Subversion の フックスクリプト - Trac の配布物の `/contrib` ディレクトリで提供されている `post-commit` フックスクリプトなど - を設定していたら、プラグインと同様にスクリプトの中で環境変数 `PYTHON_EGG_CACHE` を定義して下さい。

トラブルシューティング

setuptools は正しくインストールされていますか？

以下のコマンドを実行してみてください:

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

もし、コマンドラインから何もかえって来なければ、setuptools はインストールされています。そうでなければ、Trac を動かす前に setuptools をインストールする必要があります。

Python egg は正しいバージョンですか？

Python egg はファイル名の中で Python のバージョンをエンコードします。例えば、`MyPlugin-1.0-py2.4.egg` は Python 2.4 用の Python egg であり、異なる Python のバージョン (2.3 や 2.5) で動かそうとしても動かないでしょう。

また、ダウンロードした Python egg ファイルが本当に ZIP アーカイブであるかどうかを確認して下さい。もし Trac サイトからダウンロードしたとしたら、HTML プレビューページを代わりにダウンロードしているかもしれません。

プラグインは有効になっていますか？

グローバル領域にプラグインをインストールした場合 ([TracEnvironment](#) の `plugin` ディレクトリ内ではなく)、[_trac.ini](#) に明確に有効にする設定をしなければなりません。以下の事項を確認して下さい:

- `[components]` セクションに必要な行数を追加したか。
- パッケージ / モジュール名は正しいか
- 正しい値、"enabled" になっているかどうか。例えば、"enable" ではなく。

Python egg ファイルのパーミッションのチェック

Trac は Python egg ファイルを読めなければなりません。

ログファイルのチェック

Trac で [logging](#) を有効にし、ログレベルを `DEBUG` に設定し、プラグインがロードされる時のログメッセージを見て下さい。

必要な権限を持っていることを確認してください

いくつかのプラグインを利用するためには、あなたが特定の権限を持っている必要があります。例えば [WebAdmin](#) がナビゲーションバーに表示されるためには、`TRAC_ADMIN` の権限が必要です。

読み込むプラグインのバージョンを間違えていませんか？

複数のプロジェクトがある場合に plugins

ディレクトリにプラグインを置くなら確実に正しいバージョンのプラグインがロードされていることを確認する必要があります。ここにいくつかの基本的なルールが

- 実行中の Trac サーバ (すなわち各々の Python プロセス) にはプラグインの 1 バージョンだけがロードできます。Python の名前空間とモジュールリストは (プラグインが `■■■` (enabled) になっているか `■■■` (disabled) になっているか) 関わらず全プロジェクトで共有されるでしょう。そしてそれは複製をハンドリングすることができません。
- グローバルにインストールされたプラグイン (通常は `setup.py install`) はグローバルな `plugins` ディレクトリあるいはプロジェクト毎の `plugins` ディレクトリ内のどんなバージョンも無視するでしょう。グローバルにインストールされたプラグインは、他のどのプラグインよりも前に動作するでしょう。
- Trac サーバが (`TRAC_ENV_PARENT_DIR` セットアップのように) 複数のプロジェクトをホスティングした場合、異なるプロジェクトで異なるバージョンのプラグインを使用していると、どのバージョンのプラグインが使用されるか Trac は基本的に一番最初にリクエストを受けたプロジェクトからプラグインをロードします。
- Python の `site-packages` 内に複数のバージョンをリストしておく (すなわち `setup.py install` を使用してインストールする) のは素晴らしいことです。 - `setuptools` は最も新しいバージョンを確実にインストールします。しかし、複数バージョンのプラグインをグローバルな `plugins` ディレクトリや、プロジェクト毎の `plugins` ディレクトリに格納しないでください。 - Trac はプラグインを探すときにバージョン番号、格納された日付などを気にしません。この場合 Trac がどのプラグインを最初に見つけ出すかを確実に指定する方法はありません。

上記のすべてに失敗した場合

`plugins` に対するログが出力されず、`egg` が読み込み可能であり、Python のバージョンが正しく、その上で `egg` がグローバルにインストールされて (そして `trac.ini` で 使用可能に設定されて) いても、なぜかプラグインが動かず、なんのエラーメッセージも表示されない場合は、[IrcChannel](#) で質問してください。

See also [TracGuide](#), [プラグイン一覧](#), [コンポーネント設計](#)