SilverFrost 目次

粒度が細かいパーミッション

Trac 0.11 より前は、リポジトリブラウザ サブシステムだけで「粒度が細かいパーミッション (fine grained permissions)」を定義することができました。

0.11 以降、カスタマイズした パーミッションポリシーのプラグイン を各所に使用するための共通のメカニズムが導入されたので、すべての種類の Trac リソースのあらゆるアクションについて、そのリソースの特定バージョンのレベルまで含めて許可/拒否を設定できるようになりました。

Note: Trac 0.12 では、 authz_policy はオプションモジュールとして実装されました(tracopt.perm.authz_policy.* 配下) 。したがって、デフォルトでインストールされ、 Trac の管理 Web インタフェースの プラグイン パネルで簡単に有効にすることができます。

パーミッションポリシー

様々なパーミッションポリシーを実装することができます。 Trac にはいくつかの例を同梱しています。

現在有効なポリシーは TracIni の中で設定されているコンフィグレーションによって決定します: 例

```
[trac]
permission_policies = AuthzSourcePolicy, DefaultPermissionPolicy, LegacyAttachmentPolicy
```

このリストの1番目の #AuthzSourcePolicy ポリシーについては、下記に記載しています。続く DefaultPermissionPolicy では、 TracPermissions に記載されている従来型の粒度が粗いパーミッションチェックを行ないます。そして3番目の LegacyAttachmentPolicy は添付ファイルに対して、粒度の粗いパーミッションチェックを行ないます。

使用可能なオプションの選択肢として、 Authz 形式のシステムにてとても一般的なパーミッションポリシーを提供する #AuthzPolicy があります。詳細については、 authz_policy.py を参照して下さい。

もう一つの評判のよいパーミッションポリシーである、 #AuthzSourcePolicy は pre-0.12 で再実装され、新しいシステムでは、 Subversion のリポジトリに限定して粒度の細かいパーミッション設定をサポートするようになりました。

その他の例については、 sample-plugins/permissions を参照して下さい。

AuthzPolicy

設定方法

- ConfigObj をインストールする (0.12 でも必要)
- authz_policy.py を plugins にコピーする (Trac 0.11でのみ必要)
- authzpolicy.conf ファイルを適当な場所 (望ましくは、 Web サーバ起動ユーザ以外が読み取りできないセキュアな領域) に配置する。ファイルに非ASCII文字が含まれる場合は UTF-8 で保存してください

  trac.ini ファイルをアップデートする:

  [trac] セクションの permission_policies を編集する

  ```
  [trac]
  ...
  permission_policies = AuthzPolicy, DefaultPermissionPolicy, LegacyAttachmentPolicy
  ```

  新規に [authz_policy] セクションを追加する

  ```
  [authz_policy]
  authz_file = /some/trac/env/conf/authzpolicy.conf
  ```

  WebAdminでプラグインを有効にするか、 [components] のセクションを編集する

  ```
  [components]
  ...
  # Trac 0.12
  tracopt.perm.authz_policy.* = enabled
  # for Trac 0.11 use this
  #authz_policy.* = enabled
  ```

使用方法

パーミッションポリシーを指定する順序はとても重要です。 ポリシーは設定された順序で評価されます。

個々のポリシーはパーミッションチェックに対して `True`, `False`, `None` を返します。 ポリシーが明示的にパーミッションを許可する場合は、 `True` を返します。明示的に拒否する場合は、 `False` を返します。そして、パーミッションを許可も拒否もできない場合、 `None` が返されます。

Note: 戻り値が `None` の場合のみ、 次の パーミッションポリシーに問い合わせを行います。
どのポリシーも明示的にパーミッションを許可しない場合、最終的な結果は `False` となります（つまり、権限なしとみなされます)。

`authzpolicy.conf` は `.ini` スタイルの設定ファイルです:

```
[wiki:PrivatePage@*]
john = WIKI_VIEW, !WIKI_MODIFY
jack = WIKI_VIEW
* =
```

- config ファイルの各セクションは Trac のリソース記述子との照合に用いる グローバルなパターンです。記述子は以下のような形式です:

  ```
  <realm>:<id>@<version>[/<realm>:<id>@<version> ...]
  ```

  リソースを親から子の順に、左から右へ記述します。不特定な コンポーネントがある場合は、 `*` で置き換えられます。バージョンのパターンが明示的に記されていなければ、すべてのバージョン（`@*`）が暗黙的に追加されます。

  例: WikiStart ページを照合する

  ```
  [wiki:*]
  [wiki:WikiStart*]
  [wiki:WikiStart@*]
  [wiki:WikiStart]
  ```

  例: WikiStart の 添付ファイル `wiki:WikiStart@117/attachment/FOO.JPG@*` を照合する

  ```
  [wiki:*]
  [wiki:WikiStart*]
  [wiki:WikiStart@*]
  [wiki:WikiStart@*/attachment/*]
  [wiki:WikiStart@117/attachment/FOO.JPG]
  ```

- セクションは設定ファイルに書かれている 順に 現在の Trac のリソース記述子に対して チェックされます。順番は重要です
  一度 セッションにマッチすれば、順に 現在のユーザ名がセッションの キー（ユーザ名）と照合されます
  - キー（ユーザ名）の前に `@` を付けると、グループとして処理されます
  - 値（パーミッション）の前に `!` を付けると、そのパーミッションは 拒否されます

  通常の Trac パーミッションのルールを適用していれば、ユーザ名は、 'anonymous', 'authenticated', <username> '*'
  等とマッチするはずです。

Note: ユーザによって作成された（例えば、ブラウザ上から 管理 / 権限（英語版では Admin / Permissions）の '権限グループの追加'
（英語版では 'adding subjects to groups'））グループには使えません。詳細については #5648 を参照してください。

例えば、 `authz_file` が次の内容を含み:

```
[wiki:WikiStart@*]
* = WIKI_VIEW

[wiki:PrivatePage@*]
john = WIKI_VIEW
* = !WIKI_VIEW
```

デフォルトパーミッションが次のような内容の場合:

```
john           WIKI_VIEW
jack           WIKI_VIEW
# anonymous ■ WIKI_VIEW ■■■■■■■■■■
```

結果:

- **WikiStart の全てのバージョンは、（匿名ユーザも含む）全員が閲覧できます**
- **PrivatePage は john が表示可能です**
- **他のページは john と jack が表示可能です**

Groups:

```
[groups]
admins = john, jack
devs = alice, bob

[wiki:Dev@*]
@admins = TRAC_ADMIN
@devs = WIKI_VIEW
* =

[*]
@admins = TRAC_ADMIN
* =
```

結果:

- **すべてのアクセスがブロックされます（ホワイトリストアプローチ)。しかし**
- **admins グループはすべてにおいて TRAC_ADMIN 権限を取得しており、**
- **devs グループは Wiki ページを閲覧可能です**

リポジトリの例（閲覧ソースの詳細設定):

```
# ■■■■■■■■■:
[repository:test_repo@*]
john = BROWSER_VIEW, FILE_VIEW
# John ■ test_repo ■■■■■ BROWSER_VIEW ■ FILE_VIEW ■■■■■■■

# ■■■■■■■■■■:
[repository:*@*]
john = BROWSER_VIEW, FILE_VIEW
# John ■■■■■■■■■■■■■■■ BROWSER_VIEW ■ FILE_VIEW ■■■■■■■
```

より詳細なリポジトリのアクセス許可:

```
# John ■  trunk/src/some/location/ ■■■■■■■■■■■■■■BROWSER_VIEW ■ FILE_VIEW ■■■■■■■
[repository:test_repo@*/source:trunk/src/some/location/*@*]
john = BROWSER_VIEW, FILE_VIEW


# John ■ trunk/src/some/location ■■■■■■■ 1 ■■■■■■■■■■■■■■BROWSER_VIEW ■ FILE_VIEW ■■■■■■■
[repository:test_repo@*/source:trunk/src/some/location/*@1]
john = BROWSER_VIEW, FILE_VIEW


# John ■ trunk/src/some/location ■ 'somefile' ■■■■■■■■■■■■■■BROWSER_VIEW ■ FILE_VIEW ■■■■■■■
[repository:test_repo@*/source:trunk/src/some/location/somefile@*]
john = BROWSER_VIEW, FILE_VIEW
```

```
# John ■ trunk/src/some/location ■■■■■■■ 1 ■ 'somefile' ■■■■■■■■■■■■■BROWSER_VIEW ■ FILE_VIEW ■■■■■■■
[repository:test_repo@*/source:trunk/src/some/location/somefile@1]
john = BROWSER_VIEW, FILE_VIEW
```

Note: Timeline での通知を John に表示するためには、上記パーミッションリストに CHANGESET_VIEW を追加する必要があります。

利用不可となる機能

粒度が細かいパーミッションでは!DefaultPermissionPolicyで行っていたような（管理画面での）グループ機能は備わっていません(#9573, #5648 参照)。パッチは一部利用可能です(#6680 にある authz_policy.2.patch を参照してください)

利用不可となる機能:

```
[groups]
team1 = a, b, c
team2 = d, e, f
team3 = g, h, i
departmentA = team1, team2
```

パーミッショングループも同様にサポートされていません。下記のことができません:

```
[groups]
permission_level_1 = WIKI_VIEW, TICKET_VIEW
permission_level_2  = permission_level_1, WIKI_MODIFY, TICKET_MODIFY
[*]
@team1 = permission_level_1
@team2 = permission_level_2
@team3 = permission_level_2, TICKET_CREATE
```

AuthzSourcePolicy (mod_authz_svn のようなパーミッションポリシー)

この文書が書かれている時点では、 Trac 0.11
以前にリポジトリへの厳密なアクセス制御に使用されていた、古い「粒度が細かいパーミッション」システムは、パーミッションポリシーのコンポーネントにコンバ

「粒度が細かいパーミッション」の制御に定義ファイルを必要とします。この定義ファイルは Subversion の mod_authz_svn
で使用しているものを使います。 このファイルの形式と Subversion での用法に関する情報は、 svn book の Server Configuration (サーバ設定)
の章にある Path-Based Authorization (ディレクトリごとのアクセス制御) の項を参照してください。

例:

```
[/]
* = r

[/branches/calc/bug-142]
harry = rw
sally = r

[/branches/calc/bug-142/secret]
harry =
```

- / = 全員 read アクセスが可能です。これはデフォルトの動作となります
- /branches/calc/bug-142 = harry は read/write アクセス権を持ち、 sally は read アクセス権のみを持ちます
- /branches/calc/bug-142/secret = harry はアクセス権を持たず、 sally は read アクセス権を持ちます
  (パーミッションはサブフォルダに継承されます)

Trac の設定

「粒度が細かいパーミッション」を有効にするには、 trac.ini ファイルの [trac] セクションに authz_file オプションを
設定しなければなりません 。オプションが空値に設定されていたり、そもそも指定されていない場合、パーミッションは適用されません。

```
[trac]
authz_file = /path/to/svnaccessfile
```

auth_file 内でシンタックス [modulename:/some/path] を使用する場合、以下の設定を追加してください:

```
authz_module_name = modulename
```

modulename には、 [trac] セクション中の repository_dir に設定したリポジトリと同じものを設定します。例えば [trac] セクション内の repository_dir に /srv/active/svn/blahblah を設定している場合は次のように設定します:

```
[trac]
authz_file = /path/to/svnaccessfile
authz_module_name = blahblah
...
repository_dir = /srv/active/svn/blahblah
```

Subversion の Authz ファイル /path/to/svnaccessfile では、 [blahblah:/some/path] のようにエントリを記載します。

Note: Authz ファイルで使用するユーザ名と、 Trac で使用するユーザ名は 同じでなければなりません。

0.12 では、 trac.ini の permission_policies に AuthzSourcePolicy を必ず含めて下さい。さもないと、 authz のパーミッションファイルは無視されます。

```
[trac]
permission_policies = AuthzSourcePolicy, DefaultPermissionPolicy, LegacyAttachmentPolicy
```

Subversion の設定

通常は同じアクセスファイルを対応する Subversion リポジトリに適用します。 Apache のディレクティブには以下のように設定してください:

```
<Location /repos>
 DAV svn
 SVNParentPath /usr/local/svn

 # our access control policy
 AuthzSVNAccessFile /path/to/svnaccessfile
</Location>
```

複数のプロジェクト Environment において、プロジェクト全体にどのようにアクセス制限を行うかについての情報は TracMultipleProjectsSVNAccess を参照してください。

デバッグ用パーミッション

trac.ini の設定:

```
[logging]
log_file = trac.log
log_level = DEBUG
log_type = file
```

ウォッチコマンド:

```
tail -n 0 -f log/trac.log | egrep '\[perm\]|\[authz_policy\]'
```

どんなチェックが行なわれているか見ることができます。より詳細な情報については、プラグインのソースに添付されているドキュメントを参照して下さい。

See also: TracPermissions, TracHacks:FineGrainedPageAuthzEditorPlugin は設定を編集するプラグインです。