pootle$_{f}sDocumentation$

Release latest

Nov 03, 2017

Contents

Pootle FS app provides a plugin framework for synchronizing external filesystems containing localisation files.

An FS can be either a local filesystem or a VCS system such as git, svn, hg or cvs.

The app uses a configuration syntax to create associations between Pootle `Stores` and file stores. The stores can then be synced and changes in either can be tracked.

Syncing is a 2-step process in which changes to Stores/files are initially staged with any or all of:

- `add_translations`
- `fetch_translations`
- `rm_translations`
- `merge_translations`

Changes to previously synced Stores/files are automatically staged for synchronisation, where no conflict exists.

Once the desired changes have been staged `sync_translations` is called to perform the synchronisation.

CHAPTER 1

Pootle FS Configuration

1.1 Configuring your project in Pootle

To set an FS plugin for a project, use the `set_fs` command:

```
pootle fs MYPROJECT set_fs FS_TYPE FS_URL
```

`MYPROJECT` must the name of a valid project in Pootle.

`FS_TYPE` should be an installed and registered FS plugin type - such as `git` or `local`.

`FS_URL` must be a URL specific to the type of FS plugin you are using.

1.2 Creating a .pootle.ini on your filesystem

When pootle_fs first pulls your filesystem it looks for a file `.pootle.ini` to set up the configuration of your project.

The configuration file uses the `ini` syntax.

You can see the current configuration for your project as follows:

```
pootle fs MYPROJECT config
```

1.3 Updating the configuration

If you make changes to your `.pootle.ini` file they do not take affect until you have updated the configuration:

```
pootle fs MYPROJECT config --update
```

1.4 Defining a translation_path

1.5 Defining a directory path

CHAPTER 2

Pootle filesystem Workflow

2.1 Syncing previously synced Stores/files

When a `Store` and corresponding file have been synced previously, they are automatically staged for syncing if either changes.

This is not the case however if both have changed - see resolving conflicts section for further information.

To re-sync `Stores` and files:

```
(env) $ pootle fs myproject sync_translations
```

2.2 Pulling new translation files from the filesystem to Pootle

The workflow for bringing new translations from the filesystem into Pootle is:

```
(env) $ pootle fs myproject fetch_translations
(env) $ pootle fs myproject sync_translations
```

Where `fetch_translations` will stage the new translations, and `sync_translations` will actually sync to the database.

Note: You can fetch/sync specific `Stores` or files, or groups of them using the `-P` and `-p` options to fetch_translations and sync_translations.

2.3 Pushing new translation files from Pootle to the filesystem

The workflow for sending translations from Pootle to the filesystem:

```
(env) $ pootle fs myproject add_translations
(env) $ pootle fs myproject sync_translations
```

Where `add_translations` will stage the new translations, and `sync_translations` will actually sync to the filesystem.

Note: You can add/sync specific `Stores` or files, or groups of them using the `-P` and `-p` options to add_translations and sync_translations.

2.4 Resolving conflicts

Conflicts can occur if both a Pootle `Store` and the corresponding file have changed.

Conflict can also arise if a new Pootle `Store` is added and a matched file has been added in the filesystem.

2.4.1 Resolving conflicts - overwriting Pootle with filesystem version

If you wish to keep the version that is currently on the filesystem, discarding all changes in Pootle, you can do the following:

```
(env) $ pootle fs myproject fetch_translations --force
(env) $ pootle fs myproject sync_translations
```

2.4.2 Resolving conflicts - overwriting filesystem with Pootle version

If you wish to keep the version that is currently in Pootle, discarding all changes in the filesystem, you can do the following:

```
(env) $ pootle fs myproject add_translations --force
(env) $ pootle fs myproject sync_translations
```

2.4.3 Resolving conflicts - merging

In order to merge the changes made in both Pootle and the filesystem, you can:

```
(env) $ pootle fs myproject merge_translations
(env) $ pootle fs myproject sync_translations
```

When merging if there are conflicts in translation units the default behaviour is to keep the filesystem version, and make the Pootle version into a suggestion.

You can reverse this behaviour as follows:

```
(env) $ pootle fs myproject merge_translations --pootle-wins
(env) $ pootle fs myproject sync_translations
```

2.4.4 Removing files/Stores

Sometimes a `Store` or file is unmatched on the other side, either because it is newly added or because a `Store` or file has been removed.

You can remove `Stores` or files that do not have a corresponding match:

```
(env) $ pootle fs myproject rm_translations
(env) $ pootle fs myproject sync_translations
```

This will not affect any other `Stores` or files.

CHAPTER 3

Pootle FS commands

Pootle FS commands

CHAPTER 4

`fs` command

Get FS info for all projects

```
pootle fs
```

CHAPTER 5

`set_fs` subcommand

Set the FS for a project. Project must exist in Pootle.

```
pootle fs myproject set_fs git git@github.com:translate/myprojrepo
```

CHAPTER 6

`info` subcommand

Get the FS info for a project. This is the default command - so `info` can be ommitted.

```
pootle fs myproject info
```

or...

```
pootle fs myproject
```

CHAPTER 7

`config` subcommand

Print out the project FS configuration

```
pootle fs myproject config
```

`--update -u` Update the configuration from the FS .pootle.ini file

CHAPTER 8

`status` subcommand

List the status of files in Pootle and FS

```
pootle fs myproject status
```

CHAPTER 9

`fetch_translations` subcommand

Pull the FS repository if required, and on reading the `.pootle.ini` configuration file, create *FSStore* objects to track the associations.

```
pootle fs myproject fetch_translations
```

This command is the functional opposite of the `add_translations` command.

This command does not add any translation files in the FS - for that you need to `sync_translations`.

`--force` Stage files from FS that are conflicting

CHAPTER 10

`add_translations` subcommand

Add translations from Pootle into FS, using an optional `pootle_path` argument to filter which translations to add.

This command is the functional opposite of the `fetch_translations` command.

If you use the `--force` option it will add new translations from Pootle that are already present in the FS.

This command does not add any translation files in the FS - for tht you need to `push_translations`.

`--force` Stage files from Pootle that are conflicting

CHAPTER 11

`merge_translations` subcommand

Stage for merging any matched Stores/files that have either both been added or have both been updated

```
pootle fs myproject merge_translations
```

`--pootle-wins` Use the Pootle version for units that have conflicting changes.

CHAPTER 12

`rm_translations` subcommand

Stage for removal any matched Stores/files that do not have a corresponding Store/file in Pootle/FS.

```
pootle fs myproject rm_translations
```

CHAPTER 13

`sync_translations` subcommand

Synchronize translations between FS and Pootle:

- Create stores in Pootle where they dont exist already
- Update exisiting stores from FS translation file
- Create files where not present
- Update existing files where Stores have changed
- Remove files/Stores staged for removal

```
pootle fs myproject sync_translations
```

CHAPTER 14

Path options

--pootle_path -P Only show/affect files where the pootle_path matches a given file glob.

--path -p Only show/affect files where the FS path matches a given file glob.

CHAPTER 15

Pootle FS status

Possible status

`conflict` Both the pootle revision has changed since last sync and the latest_hash of the file has changed. The next step would be to `fetch_translations` or `add_translations` using `--force` to keep the FS version or Pootle version respectively.

`conflict_untracked` A conflict can also arise if a file on the FS has status `fs_untracked` and a matching `Store` has status `pootle_untracked` in this case you can use either `fetch_translations` or `add_translations` with `--force` depending on whether you want to keep the FS file or the `Store`.

`pootle_untracked` A new store has been added in Pootle and matches a `translation_path` in `.pootle.ini`, but does not have any `StoreFS` sync configuration. The next step would be to use `add_translations` to add a configuration.

`pootle_added` A new `Store` has been created in Pootle and has been staged using `add_translations`. It has not yet been synced and does not exist in the FS. The next step would be to `sync_translations` to sync this `Store`

`pootle_changed` A `Store` has changed in Pootle since the last sync. The next step would be to use `sync_translations` to push these changes to the FS.

`pootle_removed` A previously synced `Store` has been removed. The next step is would be to either use `fetch_translations --force` to restore the FS version, or to use `rm_translations` to stage for removal from FS.

`fs_untracked` A new file has been added in FS and matches a `translation_path` in `.pootle.ini`, but does not have any `StoreFS` sync configuration. The next step would be to use `fetch_translations` to add a configuration. Alternatively, you can use `rm_translations` to stage for removal from FS.

`fs_added` A new file has been created in FS and has been staged using `fetch_translations`. It has not yet been synced. The next step would be to `sync_translations` to create and sync this `Store`

`fs_changed` A file has changed in FS since the last sync. The next step would be to use `sync_translations` to push these changes to the FS.

`fs_removed` A previously synced file has been removed from the FS. The next step is would be to either use `add_translations --force` to restore the Pootle version, or to use `rm_translations` to stage for removal from Pootle.

`merge_fs` Merge Stores/files that have both been updated. If there are conflicting units use the translation target from the FS.

`merge_pootle` Merge Stores/files that have both been updated. If there are conflicting units use the translation target from the Pootle.

`to_remove` A file or Store that does not have a corresponding Store/file that has been staged for removal.

`both_removed` A previously synced file has been removed from the FS and Pootle - effectively orphaned. We may be able to use some kind of garbage collection to prevent this happening.

CHAPTER 16

Pootle FS Git plugin

16.1 Installation

Currently only available for developer install:

https://github.com/translate/pootle_fs_git

The core pootle_fs app is also required (also dev only):

https://github.com/translate/pootle_fs

Currently also requires the `no_mtime` branch of pootle:

https://github.com/phlax/pootle/tree/no_mtime

16.2 Pootle configuration

```
(env) $ pootle fs MYPROJECT set_fs git GIT_URL
```

`MYPROJECT` should be the name of a project in your Pootle site.

`GIT_URL` should be git ssh url.

16.3 Git authentication

Currently only ssh authentication is supported.

The user running the pootle commands therefore must have a working ssh environment and read/write access to the git repository in order to synchronize.

16.4 Custom .pootle.ini options

When using the git pootle_fs plugin there are some git-specific options

```
[default]
commit_message = "A custom commit message..."
author_name = "My Self"
author_email = "me@my.domain"
committer_name = "Pootle Server"
committer_email = "pootle@my.server"
```

16.5 Further reading

- Workflow
- Status
- Commands