Pyrepo Documentation

Release 0.1.2

Dalton Hubble

Sep 27, 2017
## Contents

1 Features 3

2 User Guide 5
   2.1 Introduction ............................................ 5
   2.2 Installation ............................................. 5
   2.3 Usage ................................................... 5

3 API 7
   3.1 API Documentation ...................................... 7

Python Module Index 11
Pyrepo is a repository abstraction package which provides a Python API for fetching and managing a variety of repositories.
CHAPTER 1

Features

• Support for repositories managed by commands
  – Git

• Supports source hosts:
  – github.com
  – bitbucket.org

• Utility to map import paths to repositories
CHAPTER 2

User Guide

This part of the documentation describes Pyrepo and how to get started using it to interact with repositories.

Introduction

Installation

Usage
API Documentation

This documentation covers the interfaces exposed by Pyrepo.

Repositories

class pyrepo.repo.Repository (command=None, url=None, import_path=None)
   Representation of a repository which is managed by a Command object, and hosted at some url.
   A Repository object may be constructed from a command and url directly, or a repository import_path can be provided which will resolve the command or url, if they aren’t provided.

   Usage:

   ```python
   from pyrepo import git_command as git
   repo = Repository(command=git,
                      url='https://github.com/dghubble/pyrepo.git')

   # autodetects the correct repository url
   repo = Repository(command=git,
                      import_path='github.com/dghubble/pyrepo')

   # autodetects the correct command and repository url
   repo = Repository(import_path='github.com/dghubble/pyrepo')
   ```

   Provide either the command and url or provide an import_path.

   Parameters

   • command (Command) – (optional) Command for managing the repository
     • url (str) – (optional) url where the repo can be found
• **import_path** *(str)* – (optional) import path of the repository (e.g. github.com/dghubble/pyrepo)

Raises ValueError, **ImportPathError**

**clone**(*args, **kwargs*)

Calls `Command.clone` with the repository *url* positional argument, followed by the given positional args and keyword kwargs.

Usage:

```python
target_dir = os.path.expanduser('~')
# clone the repository into the target directory (e.g. home)
repo.clone(target_dir)
```

**update**(*args, **kwargs*)

Arguments are the same as `Command.update` arguments.

**tag_list**(*args, **kwargs*)

Arguments are the same as `Command.tag_list` arguments.

**class** pyrepo.repo.RepoImporter(*hosts=None, commands=None*)

Utility for resolving an import path to a *url* and a `Command`.

**Parameters**

- **hosts** *(list)* – list of `Host` objects to compare import paths against. Defaults to `default_hosts`.
- **commands** *(list)* – list of `Command` objects to consider. Defaults to `default_commands`.

**resolve**(*import_path*)

Resolves an import path to a `(Command, url)` tuple.

**Parameters**

- **import_path** *(str)* – import path identifying a repository.

**Returns** `(Command, url)` tuple corresponding to the *import_path*

**Raises** **ImportPathError**

**_match_host**(*import_path*)

Checks the import_path against hosts by matching starting characters against the host’s *prefix* and matching the full import_path against the host’s *pattern*.

**Parameters**

- **import_path** *(str)* – import path identifying a repository

**Returns** matching `Host`

**Raises** **ImportPathError**

**_match_command**(*import_path, host*)

Determines the name of the command that should be used to interact with the host’s import_path repository. Simple hosts have a *command_name* that should be used (e.g. Github -> git). More complex hosts call `command_name_func` to determine the command name to use for the repository.

**Parameters**

- **import_path** *(str)* – import path identifying a repository
- **host** – `Host` of the repository, based on the *import_path*

**Returns** *name* of the `Command` for interacting with the repository

**Raises** **ImportPathError**
_build_url(import_path, host, scheme)
  Constructs the url from which the import_path repository can be fetched.

Parameters
  • import_path (str) – import path identifying a repository
  • host – Host of the repository, based on the import_path
  • scheme (str) – url scheme of the repository

exception pyrepo.repo.ImportPathError
  Raised when a RepoImporter cannot resolve an import_path.

Commands
class pyrepo.commands.Command(**kwargs)
  Represents a repository management command (e.g. Git, Mercurial) and its usage.
  Make a repository management command.

Parameters
  • name (str) – executable binary command name (e.g. ‘hg’)
  • long_name (str) – long command name (e.g. ‘Mercurial’)
  • init_cmd (str) – command to initialize a repository
  • add_cmd (str) – command to add a {path} to staging
  • commit_cmd (str) – commit a changeset to the repo with a commit {message}
  • create_cmd (str) – command to clone a fresh repository copy from {repo_url} to {target_path}
  • update_cmd (str) – command to download updates into existing repository
  • tag_list_cmd (TagCommand) – command to list repository tags
  • tag_sync_cmd (str) – command to sync to a specific tag
  • tag_sync_default_cmd (str) – command to sync to the default tag
  • schemes (list) – scheme names
  • ping_cmd (str) – command to ping for scheme

init (dir=None)
  Initializes a repository in the given directory.

clone(repo_url, target_path)
  Runs the command to clone/create a new repository. :params repi_url: url to remote repo or path to local repo :params target: path to intended destination for repo

update (dir=None)
  Runs the command to update a repo, from the given cwd directory

tag_list (cwd=None)
  Runs the command to list tags, from the given cwd directory.

ping (scheme, repo, cwd=’.’)
class pyrepo.commands.TagCommand(cmd, pattern)
  Represents a command to list tags to sync onto.
Parameters

- **cmd** – command to list tags
- **pattern** – regex to extract tags

```python
pyrepo.commands.git_command = <pyrepo.commands.Command object>
Git command definition
```

```python
pyrepo.commands.hg_command = <pyrepo.commands.Command object>
Mercurial command definition
```

```python
pyrepo.commands.default_commands = [<pyrepo.commands.Command object>, <pyrepo.commands.Command object>]
default set of commands for repositories
```

**Hosts**

```python
class pyrepo.hosts.Host(command_name_func=None, **kwargs)
Represents a host of repositories and how import_path identifiers of repositories are mapped to the host’s urls.
```

Parameters

- **name (str)** – name of the repository host
- **prefix (str)** – prefix to matching import paths should have
- **pattern (str)** – regex pattern for the import path
- **command_name (str)** – name of the command for interacting with hosted repositories
- **command_name_func (func)** – func(import_path) which returns the str name of a command to be used for the host’s import_path repository. May raise ImportPathError. Defaults to None and command_name is used directly.
- **url_format (str)** – format string for the repository url, which should contain a import_path format specifier.

Raises ValueError

```python
pyrepo.hosts.github_host = <pyrepo.hosts.Host object>
github.com repository host definition
```

```python
pyrepo.hosts.bitbucket_host = <pyrepo.hosts.Host object>
bitbucket.org repository host definition
```

```python
pyrepo.hosts.default_hosts = [<pyrepo.hosts.Host object>, <pyrepo.hosts.Host object>]
default set of repository hosts
```
Python Module Index

p
pyrepo, 7
Symbols

_build_url() (pyrepo.repo.RepoImporter method), 8
_match_command() (pyrepo.repo.RepoImporter method), 8
_match_host() (pyrepo.repo.RepoImporter method), 8

B
bitbucket_host (in module pyrepo.hosts), 10

C
clone() (pyrepo.commands.Command method), 9
clone() (pyrepo.repo.Repository method), 8
Command (class in pyrepo.commands), 9

default_commands (in module pyrepo.commands), 10
default_hosts (in module pyrepo.hosts), 10

G
git_command (in module pyrepo.commands), 10
github_host (in module pyrepo.hosts), 10

H
hg_command (in module pyrepo.commands), 10
Host (class in pyrepo.hosts), 10

I
ImportPathError, 9
init() (pyrepo.commands.Command method), 9

P
ping() (pyrepo.commands.Command method), 9
pyrepo (module), 7

R
RepoImporter (class in pyrepo.repo), 8
Repository (class in pyrepo.repo), 7
resolve() (pyrepo.repo.RepoImporter method), 8

tag_list() (pyrepo.commands.Command method), 9
tag_list() (pyrepo.repo.Repository method), 8
TagCommand (class in pyrepo.commands), 9

T
update() (pyrepo.commands.Command method), 9
update() (pyrepo.repo.Repository method), 8