
Annize

Release 6.0.6477

Author name not set

Jul 30, 2025

CONTENTS

1	License	3
2	Up-To-Date?	5
3	Dependencies	7
4	User Manual	9
4.1	Installation	9
4.2	First Steps	9
4.3	Project Configuration	9
4.4	Graphical User Interface	9
4.5	The Annize Programming Interface	9
5	Command Line Interface Reference	11
5.1	Positional Arguments	11
5.2	Named Arguments	11
5.3	Sub-commands	11
6	API Reference	13
6.1	annize namespace	13
	Python Module Index	129
	Index	131

TODO Anise is a Python-based execution engine for automation tasks. Automation tasks exist in software development and probably all kinds of other sectors. They typically require the execution of different smaller and larger tools. Complex tasks often need a sequence of many steps to execute, with some steps having dependencies to each other. Manually triggering all these steps in the graphical interfaces of all the involved tools is possible in theory, but will generate errors and frustration after some cycles. The automation interfaces of those tools are sometimes easier, but sometimes they are error-prone. Some tasks may also need to ask the user for some information in an interactive way. Some smaller parts might also be machine-specific (e.g. filesystem paths or the code how to access a password vault), while the entire task must be runnable on some different machines. In some situations, this can lead to a rather intransparent forest of different tools, with unique oddnesses and special conventions. As the number of different project increases, you will see more and more different tools, often doing a similar job, but for different platforms or frameworks and, of course, with different usage conventions. Spontaneously written glue scripts help in the beginning, but will explode as the complexity exceeds some threshold. Typical tasks in software development could be:

- Generating documentation- Testing- Automatic code generation- Creating packages- Creating a homepage, automatically built from the available version information, the packages, the documentation and so on- Deploying this homepage to a web server- Handling version information - e.g. print it in the manual- and many moreThe Anise framework allows you to implement all those tasks in a structured but generic way in a combination of XML and Python code. Once you have created this stuff at a defined place in your project, Anise lets you easily execute your tasks from command line (or from any editor if you embed it somehow). This gives you a common and easy interface to all your 'tool glue' code.The Anise engine executes arbitrary Python sourcecode and provides some additional services like logging, parameter passing from command line, basic graphical userinterface support, a plugin interface, a flexible event system, injecting code and data from other place,dependencies between code fragments, and more.On top of this engine, Anise comes with a bunch of implementations that fulfill tasks (or parts of them) of software development. There is a testing module, a documentation- and homepage-generator, some package building methods and a lot more. The implementations use the event system in many places in order to allow customization in a somewhat technical but very flexible way. Even so, those implementations are rather specific and it depends on the particular case, if, and how many of those implementations are useful.

LICENSE

Annize is distributed under the terms of the AGPL 3 license. This also affects all included files without a license header (non-source files like images), unless they are explicitly mentioned as third-party content. Read the ‘Dependencies’ section for included third-party stuff.

UP-TO-DATE?

Are you currently reading from another source than the homepage? If you are in doubt whether your package is up-to-date, you should visit the project homepage and check that. You are currently reading the documentation for version 6.0.6477.

DEPENDENCIES

Annize makes use of some third-party parts.



Required: **Python 3.13**



Required: **Python package hallyd** `~= 0.90`



Required: **Python package klovve[graphical]** `~= 1.6`



Required: **Python package lxml** `~= 6.0`



Required: **Python package pycountry** `~= 24.6`



Recommended: **GNU/Linux**



Included: **background image** (License: <https://creativecommons.org/licenses/by-sa/3.0>; from [here](#))



Included: **font ‘Inconsolata’** (for websites; by Raph Levien; License: OFL; from [here](#))



Included: **font ‘Khula’** (for websites; by Erin McLaughlin; License: OFL; from [here](#))



Included: **font ‘Symbola’** (for logo symbol; License: free for use; from [here](#))



Included: **icon set ‘Oxygen’** (some icons; [from here](#))



Included: **third-party project logos** ([from here](#))

4.1 Installation

TODO zz

4.2 First Steps

TODO zz TODO zz cli TODO zz file format TODO zz gui

4.3 Project Configuration

TODO zz TODO zz classes TODO zz features/xml namespaces TODO zz names, references and append_to

4.3.1 Model

TODO zz

4.3.2 File Format

TODO zz

4.4 Graphical User Interface

TODO zz

4.5 The Annize Programming Interface

TODO zz TODO zz i18n TODO zz fs

4.5.1 Features

TODO zz

COMMAND LINE INTERFACE REFERENCE

```
usage: annize [-h] [--project PROJECT]
              [--with-answers-from-json-file WITH_ANSWERS_FROM_JSON_FILE]
              [--with-answers-from-json-string WITH_ANSWERS_FROM_JSON_STRING]
              [--with-answer WITH_ANSWER WITH_ANSWER]
              [command] ...
```

5.1 Positional Arguments

[command] Possible choices: do
 The command to execute.

5.2 Named Arguments

--project Project directory or Annize configuration file (otherwise: the current working directory). Annize will automatically try to find it in parent directories as well.

--with-answers-from-json-file TODO
 Default: []

--with-answers-from-json-string TODO
 Default: []

--with-answer TODO
 Default: []

5.3 Sub-commands

5.3.1 do

Execute a task.

```
annize do [-h] [task_name]
```

Positional Arguments

task_name	Task name.
	Default: ''

API REFERENCE

6.1 annize namespace

6.1.1 Subpackages

annize.asset package

Submodules

annize.asset.data module

`annize.asset.data.readme_pdf(culture)`

Parameters

culture (*str*)

Return type

Path

annize.asset.project_info module

annize.data package

Submodules

annize.data.color module

class `annize.data.color.Color`(* (*Keyword-only parameters separator (PEP 3102)*), *red*, *green*, *blue*)

Bases: `object`

Parameters

- **red** (*float*)
- **green** (*float*)
- **blue** (*float*)

property `red`: `float`

property `green`: `float`

property `blue`: `float`

property `hue`: `float`

property lightness: float

property saturation: float

with_modified(**, red=None, green=None, blue=None, hue=None, lightness=None, saturation=None*)

Parameters

- **red** (*float | None*)
- **green** (*float | None*)
- **blue** (*float | None*)
- **hue** (*float | None*)
- **lightness** (*float | None*)
- **saturation** (*float | None*)

Return type

[Color](#)

property html_color_spec: str

scalehue(**, brightness, saturation=None*)

Parameters

- **brightness** (*float*)
- **saturation** (*float | None*)

Return type

[Color](#)

transformed(**, brightness=None, saturation=None*)

Parameters

- **brightness** (*float | None*)
- **saturation** (*float | None*)

Return type

[Color](#)

annize.data.container module

class annize.data.container.**Basket**(**args, **kwargs*)

Bases: list

annize.data.unique module

class annize.data.unique.**UniqueId**(*localid=None*)

Bases: object

Parameters

localid (*str | None*)

__uniqueid_counter = 0

__uniqueid_lock = <unlocked _thread.lock object>

```

property long_str: str
property short_str: str
property processonly_long_str: str
property processonly_short_str: str
__long_str(txt)

```

Parameters
 txt (str)

Return type
 str

annize.data.version module

```
class annize.data.version.AbstractVersionPatternSegment
```

Bases: ABC

```
property segment_names: list[str]
```

```
abstractmethod regexp_string()
```

Return type
 str

```
abstractmethod segments_tuples_to_text(segments_tuples)
```

Parameters
 segments_tuples (list[Tuple[str, str]])

Return type
 str

```
_name_anon(namep)
```

Parameters
 namep (str)

Return type
 None

```
_str_to_val(txt)
```

Parameters
 txt (str)

Return type
 object

```
_abc_impl = <_abc._abc_data object>
```

```
class annize.data.version.NumericVersionPatternSegment(*, partname=None)
```

Bases: [AbstractVersionPatternSegment](#)

Parameters
 partname (str | None)

```
property partname: str | None
```

```
property segment_names: list[str]
```

```
regexp_string()
```

Return type

str

```
segments_tuples_to_text(segments_tuples)
```

Parameters

`segments_tuples` (*list*[*Tuple*[*str*, *str*]])

Return type

str

```
_name_anon(namep)
```

```
_str_to_val(txt)
```

Parameters

`txt` (*str*)

Return type

object

```
_abc_impl = <_abc._abc_data object>
```

```
class annize.data.version.SeparatorVersionPatternSegment(*, text)
```

Bases: *AbstractVersionPatternSegment*

Parameters

`text` (*str*)

```
regexp_string()
```

Return type

str

```
segments_tuples_to_text(segments_tuples)
```

Parameters

`segments_tuples` (*list*[*Tuple*[*str*, *str*]])

Return type

str

```
_abc_impl = <_abc._abc_data object>
```

```
class annize.data.version.OptionalVersionPatternSegment(*, segments)
```

Bases: *AbstractVersionPatternSegment*

Parameters

`segments` (*Iterable*[*AbstractVersionPatternSegment*])

```
property segment_names
```

```
regexp_string()
```

Return type

str

```

segments_tuples_to_text(segments_tuples)

    Parameters
        segments_tuples (list[Tuple[str, str]])

    Return type
        str

_name_anon(namep)

_abc_impl = <_abc._abc_data object>

class annize.data.version.ConcatenatedVersionPatternSegment(*, segments)
    Bases: AbstractVersionPatternSegment

    Parameters
        segments (Iterable[AbstractVersionPatternSegment])

    property segment_names

    regexp_string()

    Return type
        str

    segments_tuples_to_text(segments_tuples)

        Parameters
            segments_tuples (list[Tuple[str, str]])

        Return type
            str

        _name_anon(namep)

        _abc_impl = <_abc._abc_data object>

class annize.data.version.VersionPattern(*, segments)
    Bases: object

    Parameters
        segments (Iterable[AbstractVersionPatternSegment])

    property segments: list[AbstractVersionPatternSegment]

    property segment_names

    text_to_segments_tuples(text)

        Parameters
            text (str)

        Return type
            list[Tuple[str, str]]

    segments_tuples_to_text(segments_tuples)

        Parameters
            segments_tuples (list[Tuple[str, str]])

        Return type
            str

```

```
class annize.data.version.Version(*, text=None, pattern=None, **segment_values)
```

Bases: object

Parameters

- **text** (*str*)
- **pattern** (*VersionPattern*)

property **segments_tuples**

property **segments_values**: dict[str, Any]

property **text**: str

property **pattern**: *VersionPattern*

annize.features namespace

Subpackages

annize.features.changelog namespace

Submodules

annize.features.changelog.common module

Changelogs.

```
class annize.features.changelog.common.Item(*, text)
```

Bases: object

Parameters

text (*TrStr*)

property **text**: *TrStr*

```
class annize.features.changelog.common.Entry(*, version, time, items)
```

Bases: object

Parameters

- **version** (*Version* | *None*)
- **time** (*datetime* | *None*)
- **items** (*Iterable*[*TrStr* | *Item*])

property **items**: list[*Item*]

property **time**: datetime | None

property **version**: *Version* | None

```
class annize.features.changelog.common.Changelog(*, entries)
```

Bases: object

Parameters

entries (*Iterable*[*Entry*])

property **entries**: list[*Entry*]

```
class annize.features.changelog.common.ByVersionControlSystemCommitMessagesChangelog
```

Bases: [Changelog](#)

```
_S_CHANGE = '##CHANGE:'
```

```
_S_LABEL = '##LABEL:'
```

property entries

```
annize.features.changelog.common.default_changelog()
```

Return type

[Changelog](#)

annize.features.dependencies namespace

Submodules

annize.features.dependencies.common module

Project dependency handling.

```
class annize.features.dependencies.common.Kind(*, label, importance)
```

Bases: object

Parameters

- **label** ([TrStr](#))
- **importance** (*int*)

property label: [TrStr](#)

property importance: *int*

```
class annize.features.dependencies.common.Dependency(*, kind, label, comment, icon, importance=0)
```

Bases: object

Parameters

- **kind** ([Kind](#) / *None*)
- **label** ([TrStr](#) / *None*)
- **comment** ([TrStr](#) / *None*)
- **icon** (*str* / *None*)
- **importance** (*int*)

property kind: [Kind](#)

property label: [TrStr](#)

property comment: [TrStr](#)

property icon: *str*

property importance: *int*

```
class annize.features.dependencies.common.Required(**b)
```

Bases: [Kind](#)

```
class annize.features.dependencies.common.Recommended(**b)
    Bases: Kind

class annize.features.dependencies.common.Included(**b)
    Bases: Kind

class annize.features.dependencies.common.GnuLinux(*, kind=None, comment)
    Bases: Dependency

class annize.features.dependencies.common.Artwork(*, kind=None, label, origin, comment, license)
    Bases: Dependency

    Parameters
        • origin (str)
        • license (str | None)

annize.features.dependencies.common.dependencies_to_rst_text(dependencies)

    Parameters
        dependencies (list[Dependency])

    Return type
        str
```

annize.features.dependencies.python module

Python project dependency handling.

```
class annize.features.dependencies.python.Python(*, version, kind, comment)
    Bases: Dependency

    Parameters
        version (str)

class annize.features.dependencies.python.PythonPackage(*, name, version, kind, comment)
    Bases: Dependency

    Parameters
        • name (str)
        • version (str)
        • kind (Kind | None)
        • comment (TrStr | None)

    property name: str

    property version: str

class annize.features.dependencies.python.FromRequirementsFile(*, require-
                                                                ments_file='requirements.txt',
                                                                kind=None, comment=None)

    Bases: Basket

    Parameters
        requirements_file (str | Path)
```


annize.features.distributables namespace

Subpackages

annize.features.distributables.store namespace

Submodules

annize.features.distributables.store.pypi module

PyPI store for Python packages.

```
class annize.features.distributables.store.pypi.Connection(*, token)
```

Bases: object

Parameters

token (*str*)

property token: *str*

```
class annize.features.distributables.store.pypi.Upload(*, source, connection)
```

Bases: object

Parameters

- **source** (*FileSystemContent*)
- **connection** (*Connection*)

Submodules

annize.features.distributables.common module

Distributables.

This defines *Group* of packages. Package groups can be provided for download on the project homepage or similar.

There is also *PackageStore* for keeping a version history of packages (e.g. somewhere on disk).

```
class annize.features.distributables.common.PackageStore
```

Bases: ABC

Base class for a package store.

```
abstractmethod store_package(items, *, name)
```

Store package items.

Parameters

- **items** (*Sequence*[*FileSystemContent*]) – The items to store.
- **name** (*str*) – The item name.

Return type

None

```
abstractmethod package(*, name)
```

Return package items.

Parameters

name (*str*) – The item name.

Return type*Sequence[FilesystemContent] | None***abstractmethod package_history**(*, name, limit=3)

Return the package history.

Parameters

- **name** (*str*) – The item name.
- **limit** (*int*) – The maximum number of history items to return.

Return type*Sequence[Sequence[FilesystemContent]]***_abc_impl** = <_abc._abc_data object>**class** annize.features.distributables.common.**Group**(*, title, files, description, package_store)

Bases: object

A distributable group of package files.

Often this contains only a single file, but for some kinds of packages, there can be multiple files related to each other somehow.

Parameters

- **title** (*TrStr*) – The title of this group of package files.
- **files** (*Iterable[FilesystemContent]*) – The package files.
- **description** (*TrStr | None*) – Additional description text.
- **package_store** (*PackageStore | None*) – An optional package store.

property title: *TrStr*

The title of this group of package files.

files()

Return the files of this group.

Return type*Sequence[FilesystemContent]***property description:** *TrStr*

Additional description text.

property package_store: *PackageStore | None*

An optional package store.

__files_from_package_store()**Return type***Iterable[FilesystemContent] | None***__store_files_to_package_store()****Return type**

None

__package_store_name()**Return type***str*

annize.features.distributables.debian module

Debian (.deb) packages.

```
class annize.features.distributables.debian.Category(*,debian_name,freedesktop_name)
```

Bases: object

Parameters

- **debian_name** (*str*)
- **freedesktop_name** (*str*)

property **debian_name**: *str*

property **freedesktop_name**: *str*

```
class annize.features.distributables.debian.MenuEntry(*,name,title,category,command,is_gui,icon)
```

Bases: object

Parameters

- **name** (*str*)
- **title** (*TrStr*)
- **category** (*Category*)
- **command** (*str*)
- **is_gui** (*bool*)
- **icon** (*str | Path | FilesystemContent | None*)

property **name**: *str*

property **title**: *TrStr*

property **category**: *Category | None*

property **command**: *str*

property **is_gui**: *bool*

property **icon**: *FilesystemContent | None*

```
class annize.features.distributables.debian.ExecutableLink(*,path,name)
```

Bases: object

Parameters

- **path** (*str*)
- **name** (*str | None*)

property **path**: *str*

property **name**: *str*

```
annize.features.distributables.debian._debian_category(debian_name,freedesktop_name)
```

Parameters

- **debian_name** (*str*)

- **freedesktop_name** (*str*)

Return type

type[Category]

annize.features.distributables.debian.**ApplicationsAccessibilityCategory**
alias of ACategory

annize.features.distributables.debian.**ApplicationsAmateurradioCategory**
alias of ACategory

annize.features.distributables.debian.**ApplicationsDatamanagementCategory**
alias of ACategory

annize.features.distributables.debian.**ApplicationsEditorsCategory**
alias of ACategory

annize.features.distributables.debian.**ApplicationsEducationCategory**
alias of ACategory

annize.features.distributables.debian.**ApplicationsEmulatorsCategory**
alias of ACategory

annize.features.distributables.debian.**ApplicationsFilemanagementCategory**
alias of ACategory

annize.features.distributables.debian.**ApplicationsGraphicsCategory**
alias of ACategory

annize.features.distributables.debian.**ApplicationsMobiledevicesCategory**
alias of ACategory

annize.features.distributables.debian.**ApplicationsNetworkCategory**
alias of ACategory

annize.features.distributables.debian.**ApplicationsNetworkCommunicationCategory**
alias of ACategory

annize.features.distributables.debian.**ApplicationsNetworkFiletransferCategory**
alias of ACategory

annize.features.distributables.debian.**ApplicationsNetworkMonitoringCategory**
alias of ACategory

annize.features.distributables.debian.**ApplicationsNetworkWebbrowsingCategory**
alias of ACategory

annize.features.distributables.debian.**ApplicationsNetworkWebnewsCategory**
alias of ACategory

annize.features.distributables.debian.**ApplicationsOfficeCategory**
alias of ACategory

annize.features.distributables.debian.**ApplicationsProgrammingCategory**
alias of ACategory

annize.features.distributables.debian.**ApplicationsProjectmanagementCategory**
alias of ACategory

`annize.features.distributables.debian.ApplicationsScienceCategory`
alias of ACategory

`annize.features.distributables.debian.ApplicationsScienceAstronomyCategory`
alias of ACategory

`annize.features.distributables.debian.ApplicationsScienceBiologyCategory`
alias of ACategory

`annize.features.distributables.debian.ApplicationsScienceChemistryCategory`
alias of ACategory

`annize.features.distributables.debian.ApplicationsScienceDataanalysisCategory`
alias of ACategory

`annize.features.distributables.debian.ApplicationsScienceElectronicsCategory`
alias of ACategory

`annize.features.distributables.debian.ApplicationsScienceEngineeringCategory`
alias of ACategory

`annize.features.distributables.debian.ApplicationsScienceGeoscienceCategory`
alias of ACategory

`annize.features.distributables.debian.ApplicationsScienceMathematicsCategory`
alias of ACategory

`annize.features.distributables.debian.ApplicationsScienceMedicineCategory`
alias of ACategory

`annize.features.distributables.debian.ApplicationsSciencePhysicsCategory`
alias of ACategory

`annize.features.distributables.debian.ApplicationsScienceSocialCategory`
alias of ACategory

`annize.features.distributables.debian.ApplicationsShellsCategory`
alias of ACategory

`annize.features.distributables.debian.ApplicationsSoundsCategory`
alias of ACategory

`annize.features.distributables.debian.ApplicationsSystemCategory`
alias of ACategory

`annize.features.distributables.debian.ApplicationsSystemAdministrationCategory`
alias of ACategory

`annize.features.distributables.debian.ApplicationsSystemHardwareCategory`
alias of ACategory

`annize.features.distributables.debian.ApplicationsSystemLanguageenvironmentCategory`
alias of ACategory

`annize.features.distributables.debian.ApplicationsSystemMonitoringCategory`
alias of ACategory

`annize.features.distributables.debian.ApplicationsSystemPackagemanagementCategory`
alias of ACategory

`annize.features.distributables.debian.ApplicationsSystemSecurityCategory`
alias of ACategory

`annize.features.distributables.debian.ApplicationsTerminalemulatorsCategory`
alias of ACategory

`annize.features.distributables.debian.ApplicationsTextCategory`
alias of ACategory

`annize.features.distributables.debian.ApplicationsTvandradiocategory`
alias of ACategory

`annize.features.distributables.debian.ApplicationsViewersCategory`
alias of ACategory

`annize.features.distributables.debian.ApplicationsVideoCategory`
alias of ACategory

`annize.features.distributables.debian.ApplicationsWebdevelopmentCategory`
alias of ACategory

`annize.features.distributables.debian.GamesActionCategory`
alias of ACategory

`annize.features.distributables.debian.GamesAdventureCategory`
alias of ACategory

`annize.features.distributables.debian.GamesBlocksCategory`
alias of ACategory

`annize.features.distributables.debian.GamesBoardCategory`
alias of ACategory

`annize.features.distributables.debian.GamesCardCategory`
alias of ACategory

`annize.features.distributables.debian.GamesPuzzlesCategory`
alias of ACategory

`annize.features.distributables.debian.GamesSimulationCategory`
alias of ACategory

`annize.features.distributables.debian.GamesStrategyCategory`
alias of ACategory

`annize.features.distributables.debian.GamesToolsCategory`
alias of ACategory

`annize.features.distributables.debian.GamesToysCategory`
alias of ACategory

`annize.features.distributables.debian.HelpCategory`
alias of ACategory

`annize.features.distributables.debian.ScreenSavingCategory`

alias of `ACategory`

`annize.features.distributables.debian.ScreenLockingCategory`

alias of `ACategory`

class `annize.features.distributables.debian.Section(*, name)`

Bases: `object`

property `name: str`

`annize.features.distributables.debian._debian_section(name)`

Parameters

name (*str*)

Return type

Type[[Section](#)]

`annize.features.distributables.debian.AdministrationUtilitiesSection`

alias of `ASection`

`annize.features.distributables.debian.MonoCliSection`

alias of `ASection`

`annize.features.distributables.debian.CommunicationProgramsSection`

alias of `ASection`

`annize.features.distributables.debian.DatabasesSection`

alias of `ASection`

`annize.features.distributables.debian.DebianInstallerUdebPackagesSection`

alias of `ASection`

`annize.features.distributables.debian.DebugPackagesSection`

alias of `ASection`

`annize.features.distributables.debian.DevelopmentSection`

alias of `ASection`

`annize.features.distributables.debian.DocumentationSection`

alias of `ASection`

`annize.features.distributables.debian.EditorsSection`

alias of `ASection`

`annize.features.distributables.debian.EducationSection`

alias of `ASection`

`annize.features.distributables.debian.ElectronicsSection`

alias of `ASection`

`annize.features.distributables.debian.EmbeddedSoftwareSection`

alias of `ASection`

`annize.features.distributables.debian.FontsSection`

alias of `ASection`

`annize.features.distributables.debian.GamesSection`
alias of ASection

`annize.features.distributables.debian.GnomeSection`
alias of ASection

`annize.features.distributables.debian.GnuRSection`
alias of ASection

`annize.features.distributables.debian.GnustepSection`
alias of ASection

`annize.features.distributables.debian.GraphicsSection`
alias of ASection

`annize.features.distributables.debian.HamRadioSection`
alias of ASection

`annize.features.distributables.debian.HaskellSection`
alias of ASection

`annize.features.distributables.debian.WebServersSection`
alias of ASection

`annize.features.distributables.debian.InterpretersSection`
alias of ASection

`annize.features.distributables.debian.IntrospectionSection`
alias of ASection

`annize.features.distributables.debian.JavaSection`
alias of ASection

`annize.features.distributables.debian.JavascriptSection`
alias of ASection

`annize.features.distributables.debian.KdeSection`
alias of ASection

`annize.features.distributables.debian.KernelsSection`
alias of ASection

`annize.features.distributables.debian.LibraryDevelopmentSection`
alias of ASection

`annize.features.distributables.debian.LibrariesSection`
alias of ASection

`annize.features.distributables.debian.LispSection`
alias of ASection

`annize.features.distributables.debian.LanguagePacksSection`
alias of ASection

`annize.features.distributables.debian.MailSection`
alias of ASection

`annize.features.distributables.debian.MathematicsSection`
alias of ASection

`annize.features.distributables.debian.MetaPackagesSection`
alias of ASection

`annize.features.distributables.debian.MiscellaneousSection`
alias of ASection

`annize.features.distributables.debian.NetworkSection`
alias of ASection

`annize.features.distributables.debian.NewsgroupsSection`
alias of ASection

`annize.features.distributables.debian.OcamlSection`
alias of ASection

`annize.features.distributables.debian.OldLibrariesSection`
alias of ASection

`annize.features.distributables.debian.OtherOSsAndFsSections`
alias of ASection

`annize.features.distributables.debian.PperlSection`
alias of ASection

`annize.features.distributables.debian.PhpSection`
alias of ASection

`annize.features.distributables.debian.PythonSection`
alias of ASection

`annize.features.distributables.debian.RubySection`
alias of ASection

`annize.features.distributables.debian.RustSection`
alias of ASection

`annize.features.distributables.debian.ScienceSection`
alias of ASection

`annize.features.distributables.debian.ShellsSection`
alias of ASection

`annize.features.distributables.debian.SoundSection`
alias of ASection

`annize.features.distributables.debian.TasksSection`
alias of ASection

`annize.features.distributables.debian.TexSection`
alias of ASection

`annize.features.distributables.debian.TextProcessingSection`
alias of ASection

`annize.features.distributables.debian.UtilitiesSection`
alias of ASection

`annize.features.distributables.debian.VersionControlSystemsSection`
alias of ASection

`annize.features.distributables.debian.VideoSection`

alias of `ASection`

`annize.features.distributables.debian.VirtualPackagesSection`

alias of `ASection`

`annize.features.distributables.debian.WebSoftwareSection`

alias of `ASection`

`annize.features.distributables.debian.XWindowSystemSoftwareSection`

alias of `ASection`

`annize.features.distributables.debian.XfceSection`

alias of `ASection`

`annize.features.distributables.debian.ZopePloneFrameworkSection`

alias of `ASection`

class `annize.features.distributables.debian.ServiceDescription(name, command)`

Bases: `object`

Description for Debian services to be included in a package.

Parameters

- **name** (*str*) – The display name.
- **command** (*str*) – The command to be executed.

class `annize.features.distributables.debian.Package(*, source, menuentries, executable_links, packagename, description, summary, section, homepage_url, version, documentation, authors, prerm="", postinst="", architecture='all')`

Bases: `FilesystemContent`

Parameters

- **generate_func** – The content generator function. It has no parameters and returns an absolute path to the content (usually inside some temporary directory).
- **source** (`FilesystemContent`)
- **menuentries** (*list* [`MenuEntry`])
- **executable_links** (*list* [`ExecutableLink`])
- **packagename** (*str* | `None`)
- **description** (`TrStr` | `None`)
- **summary** (`TrStr` | `None`)
- **section** (`Section` | `None`)
- **homepage_url** (*str* | `None`)
- **version** (`Version` | `None`)
- **documentation** (`FilesystemContent` | `None`)
- **authors** (*list* [`Author`])
- **prerm** (*str*)
- **postinst** (*str*)

- **architecture** (*str*)

_path()

```
class _BuildInfo(source: annize.fs.FilesystemContent, executable_links:
    list[annize.features.distributables.debian.ExecutableLink], menuentries:
    list[annize.features.distributables.debian.MenuEntry], services:
    list[annize.features.distributables.debian.ServiceDescription], description: str, name:
    str, version: annize.data.version.Version, homepage: str, author:
    annize.features.authors.Author, licensename: str, section:
    annize.features.distributables.debian.Section | None, summary: str,
    documentation_source: annize.fs.FilesystemContent | None, prerm: str, postinst: str,
    architecture: str, authorstring: str = None, pkgrootpath: str = None, pkgpath_debian:
    str = None, pkgpath_documentation: str = None, pkgpath_pixmap: str = None,
    pkgpath_usrbin: str = None, config_files: list[str] = None, pkgsize: int = None, result:
    annize.fs.FilesystemContent = None)
```

Bases: object

Parameters

- **source** (*FilesystemContent*)
- **executable_links** (*list*[*ExecutableLink*])
- **menuentries** (*list*[*MenuEntry*])
- **services** (*list*[*ServiceDescription*])
- **description** (*str*)
- **name** (*str*)
- **version** (*Version*)
- **homepage** (*str*)
- **author** (*Author*)
- **licensename** (*str*)
- **section** (*Section* | *None*)
- **summary** (*str*)
- **documentation_source** (*FilesystemContent* | *None*)
- **prerm** (*str*)
- **postinst** (*str*)
- **architecture** (*str*)
- **authorstring** (*str*)
- **pkgrootpath** (*str*)
- **pkgpath_debian** (*str*)
- **pkgpath_documentation** (*str*)
- **pkgpath_pixmap** (*str*)
- **pkgpath_usrbin** (*str*)
- **config_files** (*list*[*str*])
- **pkgsize** (*int*)

```
    • result (FilesystemContent)
source: FilesystemContent
executable_links: list[ExecutableLink]
menuentries: list[MenuEntry]
services: list[ServiceDescription]
description: str
name: str
version: Version
homepage: str
author: Author
licensename: str
section: Section | None
summary: str
documentation_source: FilesystemContent | None
prerm: str
postinst: str
architecture: str
authorstring: str = None
pkgrootpath: str = None
pkgpath_debian: str = None
pkgpath_documentation: str = None
pkgpath_pixmap: str = None
pkgpath_usrbin: str = None
config_files: list[str] = None
pkgsize: int = None
result: FilesystemContent = None
```

```
classmethod _mkpackage(build)
```

Parameters

```
    build (_BuildInfo)
```

Return type

```
    FilesystemContent
```

classmethod `_mkpackage_prepareinfos(build, tmpdir)`

Parameters

- `build` (`_BuildInfo`)
- `tmpdir` (`str` | `Path`)

Return type

None

classmethod `_mkpackage_mkcopyright(build)`

Parameters

`build` (`_BuildInfo`)

Return type

None

classmethod `_mkpackage_mkchangelog(build)`

Parameters

`build` (`_BuildInfo`)

Return type

None

classmethod `_mkpackage_mkexeclinks(build)`

Parameters

`build` (`_BuildInfo`)

Return type

None

classmethod `_mkpackage_mkmenuentries(build)`

Parameters

`build` (`_BuildInfo`)

Return type

None

classmethod `_mkpackage_mkservices(build)`

Parameters

`build` (`_BuildInfo`)

Return type

None

classmethod `_mkpackage_determinesize(build)`

Parameters

`build` (`_BuildInfo`)

Return type

None

classmethod `_mkpackage_mkprepostcmds(build)`

Parameters

`build` (`_BuildInfo`)

Return type

None

classmethod `_mkpackage_mkdebiancontrolfile(build)`**Parameters**`build` (`_BuildInfo`)**Return type**

None

classmethod `_mkpackage_mkdebianconffilesfile(build)`**Parameters**`build` (`_BuildInfo`)**Return type**

None

classmethod `_mkpackage_correctbuildsourcepermissions(build)`**Parameters**`build` (`_BuildInfo`)**Return type**

None

classmethod `_mkpackage_dpkg(build)`**Parameters**`build` (`_BuildInfo`)**Return type**

None

annize.features.distributables.flatpak module

Flatpaks.

class `annize.features.distributables.flatpak.MenuEntry(*, name, title, category, command, is_gui, icon)`Bases: `object`**Parameters**

- **name** (`str`)
- **title** (`TrStr`)
- **category** (`Tuple`)
- **command** (`str`)
- **is_gui** (`bool`)
- **icon** (`FilesystemContent` / `None`)

property `name`: `str`**property** `title`: `TrStr`**property** `category`: `Tuple`

```

    property command: str

    property is_gui: bool

    property icon: FilesystemContent | None

class annize.features.distributables.flatpak.Filesystem(**b)
    Bases: object

class annize.features.distributables.flatpak.Share(**b)
    Bases: object

class annize.features.distributables.flatpak.EnvironmentVariable(**b)
    Bases: object

class annize.features.distributables.flatpak.Repository(*, public_url, friendly_name_suggestion)
    Bases: object

        Parameters
            • public_url (str)
            • friendly_name_suggestion (str | None)

    upload(source)

        Parameters
            source (FilesystemContent)

    property public_url: str

    property friendly_name_suggestion: str

class annize.features.distributables.flatpak.LocalRepository(*, public_url,
                                                             friendly_name_suggestion,
                                                             upload_path, upload_fsentry)
    Bases: Repository

        Parameters
            • public_url (str)
            • friendly_name_suggestion (str | None)
            • upload_path (str | None)
            • upload_fsentry (FilesystemContent | None)

    upload(source)

        Parameters
            source (FilesystemContent)

class annize.features.distributables.flatpak.Group(*, source, title, description, repository,
                                                    package_name, project_short_hint_name,
                                                    package_short_name)
    Bases: Group

        Parameters
            • title (str) – The title of this group of package files.
            • files – The package files.

```

- **description** ([TrStr](#) / *None*) – Additional description text.
- **package_store** – An optional package store.
- **source** ([FilesystemContent](#))
- **repository** ([Repository](#))
- **package_name** (*str*)
- **project_short_hint_name** (*str* / *None*)
- **package_short_name** (*str* / *None*)

files()

Return the files of this group.

property description

Additional description text.

```
class annize.features.distributables.flatpak.FlatpakrefFile(*, refname, package_name, title,
                                                            branch='master',
                                                            runtime_repository_url, gpgkey,
                                                            repository)
```

Bases: [FilesystemContent](#)

Parameters

- **generate_func** – The content generator function. It has no parameters and returns an absolute path to the content (usually inside some temporary directory).
- **refname** (*str* / *None*)
- **package_name** (*str*)
- **title** (*str* / *None*)
- **branch** (*str*)
- **runtime_repository_url** (*str* / *None*)
- **gpgkey** (*bytes* / *None*)
- **repository** ([Repository](#))

_path()

```
class annize.features.distributables.flatpak.GpgFile(*, refname=None, gpgkey=None)
```

Bases: [FilesystemContent](#)

Parameters

- **generate_func** – The content generator function. It has no parameters and returns an absolute path to the content (usually inside some temporary directory).
- **refname** (*str*)
- **gpgkey** (*bytes*)

_path()

```
class annize.features.distributables.flatpak.FlatpakImage(*, source, package_name,
                                                           sockets=('x11'), filesystems=('home'),
                                                           shares=('network'))
```

Bases: [FilesystemContent](#)

Parameters

- **generate_func** – The content generator function. It has no parameters and returns an absolute path to the content (usually inside some temporary directory).
- **source** (`FilesystemContent`)
- **package_name** (`str`)
- **sockets** (`Iterable[str]`)
- **filesystems** (`Iterable[str]`)
- **shares** (`Iterable[str]`)

_path()

```
class _BuildInfo(source: annize.fs.FilesystemContent, name: str, sdk: str, platform: str, kitversion: str |
    None, command: str | None, sockets: Iterable[str], filesystems: Iterable[str], shares:
    Iterable[str], menu_entries: Iterable[annize.features.distributables.flatpak.MenuEntry],
    environment: dict[str, str], pkgrootpath: str = None, pkgpath_share: str = None,
    pkgpath_share_applications: str = None, pkgpath_share_icons: str = None, result:
    annize.fs.FilesystemContent = None)
```

Bases: `object`

Parameters

- **source** (`FilesystemContent`)
- **name** (`str`)
- **sdk** (`str`)
- **platform** (`str`)
- **kitversion** (`str | None`)
- **command** (`str | None`)
- **sockets** (`Iterable[str]`)
- **filesystems** (`Iterable[str]`)
- **shares** (`Iterable[str]`)
- **menu_entries** (`Iterable[MenuEntry]`)
- **environment** (`dict[str, str]`)
- **pkgrootpath** (`str`)
- **pkgpath_share** (`str`)
- **pkgpath_share_applications** (`str`)
- **pkgpath_share_icons** (`str`)
- **result** (`FilesystemContent`)

source: `FilesystemContent`

name: `str`

sdk: `str`

platform: `str`

```
kitversion: str | None
command: str | None
sockets: Iterable[str]
filesystems: Iterable[str]
shares: Iterable[str]
menu_entries: Iterable[MenuEntry]
environment: dict[str, str]
pkgrootpath: str = None
pkgpath_share: str = None
pkgpath_share_applications: str = None
pkgpath_share_icons: str = None
result: FilesystemContent = None
```

```
classmethod _mkpackage_prepareinfos(build, tmpdir)
```

Parameters

- `build` (`_BuildInfo`)
- `tmpdir` (`Path`)

Return type

None

```
classmethod _mkpackage_flatpak_build_init(build)
```

Parameters

`build` (`_BuildInfo`)

Return type

None

```
classmethod _mkpackage_applysource(build)
```

Parameters

`build` (`_BuildInfo`)

Return type

None

```
classmethod _mkpackage_flatpak_build_finish(build)
```

Parameters

`build` (`_BuildInfo`)

Return type

None

classmethod `_mkpackage_share(build)`

Parameters

`build` (`_BuildInfo`)

Return type

None

classmethod `_mkpackage_flatpak_build_export(build)`

Parameters

`build` (`_BuildInfo`)

Return type

None

classmethod `_mkpackage(build)`

Parameters

`build` (`_BuildInfo`)

Return type

`Path`

annize.features.distributables.python_wheel module

Python Wheels.

class `annize.features.distributables.python_wheel.ExecutableLink(*, link_name, module_name, method_name, is_gui)`

Bases: `object`

Parameters

- `link_name` (`str`)
- `module_name` (`str`)
- `method_name` (`str`)
- `is_gui` (`bool`)

property `link_name`: `str`

property `module_name`: `str`

property `method_name`: `str`

property `is_gui`: `bool`

class `annize.features.distributables.python_wheel.ExtraDependencies(*, extra_name, dependencies)`

Bases: `object`

Parameters

- `extra_name` (`str`)
- `dependencies` (`Iterable[PythonPackage]`)

property `extra_name`: `str`

property dependencies: Sequence[[PythonPackage](#)]

```
class annize.features.distributables.python_wheel.Package(*, source, executable_links,
                                                         packagename, description,
                                                         homepage_url, long_description,
                                                         version, keywords, dependencies,
                                                         extra_dependencies, license, authors)
```

Bases: [FilesystemContent](#)

Parameters

- **generate_func** – The content generator function. It has no parameters and returns an absolute path to the content (usually inside some temporary directory).
- **source** ([FilesystemContent](#))
- **executable_links** ([Iterable](#)[[ExecutableLink](#)])
- **packagename** (*str* | *None*)
- **description** ([TrStr](#) | *None*)
- **homepage_url** (*str* | *None*)
- **long_description** ([TrStr](#) | *None*)
- **version** ([Version](#) | *None*)
- **keywords** ([Keywords](#) | *None*)
- **dependencies** ([Iterable](#)[[PythonPackage](#)])
- **extra_dependencies** ([Iterable](#)[[ExtraDependencies](#)])
- **license** ([License](#) | *None*)
- **authors** ([Iterable](#)[[Author](#)])

_path()

```
class _BuildInfo(source: annize.fs.FilesystemContent, description: str, long_description: str, keywords:
                 annize.features.base.Keywords, name: str, version: annize.data.version.Version,
                 homepage: str, author: annize.features.authors.Author, license:
                 annize.features.licensing.License, executable_links:
                 list[annize.features.distributables.python_wheel.ExecutableLink], dependencies: list,
                 extra_dependencies: list, pkgrootpath: str = None, pkgpath_setuppy: str = None,
                 setuppy_conf: dict[str, Any | None] = None, result: annize.fs.FilesystemContent = None)
```

Bases: [object](#)

Parameters

- **source** ([FilesystemContent](#))
- **description** (*str*)
- **long_description** (*str*)
- **keywords** ([Keywords](#))
- **name** (*str*)
- **version** ([Version](#))
- **homepage** (*str*)
- **author** ([Author](#))

- `license` (`License`)
- `executable_links` (`list[ExecutableLink]`)
- `dependencies` (`list`)
- `extra_dependencies` (`list`)
- `pkgrootpath` (`str`)
- `pkgpath_setuppy` (`str`)
- `setuppy_conf` (`dict[str, Any | None]`)
- `result` (`FilesystemContent`)

`source:` `FilesystemContent`

`description:` `str`

`long_description:` `str`

`keywords:` `Keywords`

`name:` `str`

`version:` `Version`

`homepage:` `str`

`author:` `Author`

`license:` `License`

`executable_links:` `list[ExecutableLink]`

`dependencies:` `list`

`extra_dependencies:` `list`

`pkgrootpath:` `str = None`

`pkgpath_setuppy:` `str = None`

`setuppy_conf:` `dict[str, Any | None] = None`

`result:` `FilesystemContent = None`

`classmethod _mkpackage`(`build`)

Parameters

`build` (`_BuildInfo`)

Return type

`FilesystemContent`

`classmethod _mkpackage_prepareinfos`(`build`, `tmpdir`)

Parameters

- `build` (`_BuildInfo`)
- `tmpdir` (`Path`)

Return type

`None`

classmethod `_mkpackage_setuppyconf_prepare(build)`

Parameters

`build` (`_BuildInfo`)

Return type

None

classmethod `_mkpackage_setuppyconf_install_requires(build)`

Parameters

`build` (`_BuildInfo`)

Return type

None

classmethod `_mkpackage_setuppyconf_classifiers(build)`

Parameters

`build` (`_BuildInfo`)

Return type

None

classmethod `_mkpackage_mkexeclinks(build)`

Parameters

`build` (`_BuildInfo`)

Return type

None

classmethod `_mkpackage_mksetuppyconf(build)`

Parameters

`build` (`_BuildInfo`)

Return type

None

classmethod `_mkpackage_bdist_wheel(build)`

Parameters

`build` (`_BuildInfo`)

Return type

None

classmethod `_mkpackage_mkmanifestin(build)`

Parameters

`build` (`_BuildInfo`)

Return type

None

annize.features.distributables.tar module

Tarballs.

```
class annize.features.distributables.tar.Package(*, packagename, packagenamepostfix, source,
                                              version, license, documentation)
```

Bases: [FilesystemContent](#)

Parameters

- **generate_func** – The content generator function. It has no parameters and returns an absolute path to the content (usually inside some temporary directory).
- **packagename** (*str* / *None*)
- **packagenamepostfix** (*str* / *None*)
- **source** ([FilesystemContent](#))
- **version** ([Version](#) / *None*)
- **license** ([License](#) / *None*)
- **documentation** ([FilesystemContent](#) / *None*)

_path()

annize.features.documentation namespace

Subpackages

annize.features.documentation.sphinx namespace

Subpackages

annize.features.documentation.sphinx.output namespace

Submodules

annize.features.documentation.sphinx.output.common module

Sphinx-based documentation output.

```
annize.features.documentation.sphinx.output.common.register_output_generator(outputgenerator)
```

```
annize.features.documentation.sphinx.output.common.find_output_generator_for_outputspec(output_spec)
```

```
class annize.features.documentation.sphinx.output.common.OutputGenerator(output_spec)
```

Bases: [ABC](#)

Base class for documentation output specifications. See [render\(\)](#).

property **output_spec**: [OutputSpec](#)

abstractmethod classmethod **is_compatible_for**(*output_spec*)

Parameters

output_spec ([OutputSpec](#))

Return type

bool

abstractmethod **formatname**()

Returns the Sphinx format name.

Return type

str

prepare_generate(*geninfo*)

Parameters

geninfo (*documentationsphinx.Document.GenerateInfo*)

Return type

None

postproc(*preresult*)

Parameters

preresult (*FilesystemContent*)

Return type

FilesystemContent

multilanguage_frame(*document*)

Parameters

document (*documentationsphinx.Document*)

Return type

DocumentGenerateAllCulturesResult

_abc_impl = <_abc._abc_data object>

annize.features.documentation.sphinx.output.html module

Sphinx-based HTML documentation output.

```
class annize.features.documentation.sphinx.output.html.HtmlOutputSpec(*, is_homepage=False,
    short_title=None,
    short_desc=None,
    theme=None,
    masterlink=None,
    logo_image=None, background_image=None)
```

Bases: *HtmlOutputSpec*

Parameters

- **theme** (*str* / *None*) – The sphinx html theme name.
- **short_title** (*TrStr* / *None*) – The short html title.
- **short_desc** (*TrStr* / *None*) – The short description. Ignored by most themes.
- **masterlink** (*str* / *None*) – Url that overrides the target of the main heading (which is also a link).
- **is_homepage** (*bool*)
- **logo_image** (*str* / *Path* / *FilesystemContent* / *None*)
- **background_image** (*str* / *Path* / *FilesystemContent* / *None*)

property short_title: *TrStr* | *None*

property short_desc: *TrStr* | *None*

property theme: *str* | *None*


```

property masterlink: str | None

property logo_image: FileSystemContent | None

property background_image: FileSystemContent | None

_abc_impl = <_abc._abc_data object>

```

```

class annize.features.documentation.sphinx.output.html.HtmlOutputGenerator(output_spec)
    Bases: OutputGenerator
    HTML documentation output.

    classmethod is_compatible_for(output_spec)

    formatname()
        Returns the Sphinx format name.

    prepare_generate(geninfo)

    multilanguage_frame(document)

    _abc_impl = <_abc._abc_data object>

```

annize.features.documentation.sphinx.output.pdf module

Sphinx-based PDF documentation output.

```

class annize.features.documentation.sphinx.output.pdf.PdfOutputGenerator(output_spec)
    Bases: OutputGenerator
    PDF documentation output.

    classmethod is_compatible_for(output_spec)

    formatname()
        Returns the Sphinx format name.

    postproc(preresult)

        Parameters
            preresult (str | Path)

        Return type
            Path

    _abc_impl = <_abc._abc_data object>

```

annize.features.documentation.sphinx.output.plaintext module

Sphinx-based plaintext documentation output.

```

class annize.features.documentation.sphinx.output.plaintext.PlaintextOutputGenerator(output_spec)
    Bases: OutputGenerator
    Plaintext documentation output.

    classmethod is_compatible_for(output_spec)

```

formatname()

Returns the Sphinx format name.

_abc_impl = <_abc._abc_data object>

Submodules

annize.features.documentation.sphinx.common module

Sphinx-based documentation.

class annize.features.documentation.sphinx.common.**Document**(**project_name*, *title*, *authors*, *version*, *release*)

Bases: *Document*, ABC

Parameters

- **project_name** (*TrStr* / *None*)
- **title** (*TrStr* / *None*)
- **authors** (*list*[*Author*])
- **version** (*Version* / *None*)
- **release** (*TrStr* / *None*)

class **GenerateInfo**(*main_document*: 'Document', *intstruct*: annize.fs.Path, *outdir*: annize.fs.FilesystemContent, *confdir*: annize.fs.FilesystemContent, *culture*: annize.i18n.Culture, *configvalues*: dict, *configlines*: list, *entry_path*: str = "")

Bases: object

Parameters

- **main_document** (*Document*)
- **intstruct** (*Path*)
- **outdir** (*FilesystemContent*)
- **confdir** (*FilesystemContent*)
- **culture** (*Culture*)
- **configvalues** (*dict*)
- **configlines** (*list*)
- **entry_path** (*str*)

main_document: *Document*

intstruct: *Path*

outdir: *FilesystemContent*

confdir: *FilesystemContent*

culture: *Culture*

configvalues: dict

configlines: list

```

    entry_path: str = ''

    _to_dict()

abstractmethod _generate_sources(geninfo)

    Parameters
        geninfo (GenerateInfo)

    Return type
        str

property projectname__original: TrStr | None
property project_name: TrStr
property title__original: TrStr | None
property title: TrStr
property authors: list[Author]
property version: Version | None
property release: TrStr | None
generate_all_cultures(output_spec)
generate(output_spec, *, culture=None)
__generate_set_misc(geninfo)
__generate_prepare_shortsnippets()
__generate_prepare_annizeicons()
__generate_set_culture()
__generate_set_version_and_release(geninfo)
__generate_geninfo_to_confpy()
_abc_impl = <_abc._abc_data object>

class annize.features.documentation.sphinx.common.CompositeDocument(*, documents, **kwargs)
    Bases: Document

    Parameters
        documents (Iterable[Document])

    __get_inner_generateinfo(innerdoc)

    Parameters
        • geninfo (GenerateInfo)
        • innerdoc (Document)

    _generate_sources(geninfo)

    _abc_impl = <_abc._abc_data object>

```

class annize.features.documentation.sphinx.common.**ApiReferenceLanguage**

Bases: ABC

Base class for a programming language in api references. See `ApiReferencePiece`.

class **ApiReferenceGenerateInfo**(*geninfo*, *source*, *heading*)

Bases: [GenerateInfo](#)

Parameters

- **geninfo** ([GenerateInfo](#))
- **source** ([FilesystemContent](#))
- **heading** ([TrStr](#))

property **source**: [FilesystemContent](#)

property **heading**: [TrStr](#)

abstractmethod **generate_sources**(*rngeninfo*)

Parameters

rngeninfo ([ApiReferenceGenerateInfo](#))

Return type

str

_abc_impl = <_abc._abc_data object>

class annize.features.documentation.sphinx.common.**ApiReferenceDocument**(*, *language*, *heading*,
source, *cultures*,
***kwargs*)

Bases: [Document](#)

An api reference.

Parameters

- **language** ([ApiReferenceLanguage](#))
- **heading** ([TrStr](#) | *None*)
- **source** (*str* | *Path* | [FilesystemContent](#))
- **cultures** (*list*[[Culture](#)])

available_cultures()

__get_refgeninfo(*geninfo*)

Parameters

geninfo ([GenerateInfo](#))

_generate_sources(*geninfo*)

_abc_impl = <_abc._abc_data object>

```
class annize.features.documentation.sphinx.common.ArgparseCommandLineInterfaceDocument(*,
                                                                                       parser_factory,
                                                                                       pro-
                                                                                       gram_name,
                                                                                       head-
                                                                                       ing,
                                                                                       source_directory,
                                                                                       cul-
                                                                                       tures,
                                                                                       **kwargs)
```

Bases: *Document*

Parameters

- **parser_factory** (*str*)
- **program_name** (*str*)
- **heading** (*str* | *None*)
- **source_directory** (*str* | *Path* | *FilesystemContent*)
- **cultures** (*list*[*Culture*])

available_cultures()

_generate_sources(*geninfo*)

_abc_impl = *<_abc._abc_data object>*

```
class annize.features.documentation.sphinx.common.RstDocumentVariant(*, cul-
                                                                                       ture=<annize.i18n.UnspecifiedCulture
                                                                                       object>, source)
```

Bases: *object*

Parameters

- **culture** (*Culture*)
- **source** (*str* | *Path* | *FilesystemContent*)

property culture: *Culture*

property source: *FilesystemContent*

```
class annize.features.documentation.sphinx.common.RstDocument(*, variants, **kwargs)
```

Bases: *Document*

A reStructuredText formatted file or a directory of such files.

Parameters

variants (*list*[*RstDocumentVariant*])

available_cultures()

__get_variant(*culture*)

Parameters

culture (*Culture*)

Return type

RstDocumentVariant | *None*

```
_generate_sources(geninfo)
```

```
_abc_impl = <_abc._abc_data object>
```

```
class annize.features.documentation.sphinx.common.AboutProjectDocument(*, dependencies,  
                                                                    cultures, **kwargs)
```

Bases: *Document*

Parameters

- **dependencies** (*list*[*Dependency*])
- **cultures** (*list*[*Culture*])

```
available_cultures()
```

```
_generate_sources(geninfo)
```

```
_abc_impl = <_abc._abc_data object>
```

```
class annize.features.documentation.sphinx.common.ReadmeDocument(*, project_name, title, authors,  
                                                                    version, release, documents,  
                                                                    dependencies, cultures)
```

Bases: *CompositeDocument*

A reStructuredText formatted file or a directory of such files.

Parameters

- **documents** (*Iterable*[*Document*])
- **dependencies** (*list*[*Dependency*])
- **cultures** (*list*[*Culture*])

```
_ABOUT_NAME = 'annize..about'
```

```
_abc_impl = <_abc._abc_data object>
```

```
available_cultures()
```

property title

annize.features.documentation.sphinx.cpp module

Sphinx-based C/C++ documentation.

```
class annize.features.documentation.sphinx.cpp.CppApiReferenceLanguage(**kwargs)
```

Bases: *DoxygenSupportedApiReferenceLanguage*

C++ language support for api references.

```
_abc_impl = <_abc._abc_data object>
```

annize.features.documentation.sphinx.doxygen_compat module

Sphinx-based documentation with Doxygen support.

```

class annize.features.documentation.sphinx.doxygen_compat.DoxygenSupportedApiReferenceLanguage(*,
    _doxygenopts=dict[str, str] | None,
    extract_all=bool,
    extract_private=bool,
    extract_package=bool,
    extract_static=bool,
    extract_local_classes=bool,
    extract_local_methods=bool,
    extract_anon_nspaces=bool,
    extract_priv_virtual=bool,
    file_patterns=str,
    inline_inherited_members=bool,
    inherit_docs=bool,
    hide_undocumented_members=bool,
    exclude_patterns=str,
    pre_declarations=None,
    )

```

Bases: [ApiReferenceLanguage](#)

Language support for api references powered by Doxygen.

Parameters

- **_doxygenopts** (*dict[str, str] | None*)
- **extract_all** (*bool*)
- **extract_private** (*bool*)
- **extract_package** (*bool*)
- **extract_static** (*bool*)
- **extract_local_classes** (*bool*)
- **extract_local_methods** (*bool*)
- **extract_anon_nspaces** (*bool*)
- **extract_priv_virtual** (*bool*)
- **file_patterns** (*str*)
- **inline_inherited_members** (*bool*)
- **inherit_docs** (*bool*)
- **hide_undocumented_members** (*bool*)

- **hide_undoc_classes** (*bool*)
- **exclude_patterns** (*str*)
- **predefined** (*list[str] | None*)

__info(*outpath*)

__prepare_generate(*rootpath, outpath*)

generate_sources(*srcpath, intstructpath, confdirpath, heading*)

_abc_impl = <_abc._abc_data object>

annize.features.documentation.sphinx.javascript module

Sphinx-based Javascript documentation.

class annize.features.documentation.sphinx.javascript.**JavaScriptApiReferenceLanguage**

Bases: [ApiReferenceLanguage](#)

JavaScript language support for api references.

__jsfiles(*dirf*)

Parameters

dirf (*str*)

Return type

list[str]

__scanjsfile(*f*)

Parameters

f (*str*)

Return type

str

generate_sources(*rgeinfo*)

_abc_impl = <_abc._abc_data object>

annize.features.documentation.sphinx.python module

Sphinx-based Python documentation.

class annize.features.documentation.sphinx.python.**Python3ApiReferenceLanguage**(*,
show_undoc_members=True,
show_protected_members=True)

Bases: [ApiReferenceLanguage](#)

Python 3 language support for api references.

Parameters

- **show_undoc_members** (*bool*)
- **show_protected_members** (*bool*)


```
__patch_property_types_in_docstrings(pdir)
```

Parameters

pdir (*str*)

Return type

None

```
generate_sources(rgeinfo)
```

```
_abc_impl = <_abc._abc_data object>
```

annize.features.documentation.sphinx.rst module

Sphinx-based reStructuredText documentation.

```
class annize.features.documentation.sphinx.rst.RstGenerator
```

Bases: object

Generator for some documentation source parts.

```
static heading(text, *, variant='=', sub=False, anchor=None)
```

Generates documentation source for a section heading.

Parameters

- **text** (*TrStr* / *str*)
- **variant** (*str*)
- **sub** (*bool*)
- **anchor** (*str* / *None*)

Return type

str

Submodules

annize.features.documentation.common module

Documentation.

```
class annize.features.documentation.common.DocumentGenerateResult(file, entry_path)
```

Bases: object

Parameters

- **file** (*FilesystemContent*)
- **entry_path** (*str*)

property file: *FilesystemContent*

property entry_path: *str*

```
_set_entry_path(entry_path)
```

Parameters

entry_path (*str*)

Return type

None

```
class annize.features.documentation.common.DocumentGenerateAllCulturesResult(file, entry_path,  
                                                                           en-  
                                                                           try_paths_for_languages)
```

Bases: [DocumentGenerateResult](#)

Parameters

- **file** ([FilesystemContent](#))
- **entry_path** (*str*)
- **entry_paths_for_languages** (*dict[str, str]*)

entry_path_for_language (*language*)

Parameters

language (*str*)

Return type

str

property culture_names: *Sequence[str]*

```
class annize.features.documentation.common.Document
```

Bases: [ABC](#)

abstractmethod available_cultures()

Return type

Sequence[Culture]

abstractmethod generate (*output_spec, *, culture=<annize.i18n.UnspecifiedCulture object>*)

Parameters

culture ([Culture](#))

Return type

[DocumentGenerateResult](#)

abstractmethod generate_all_cultures (*output_spec*)

Return type

[DocumentGenerateAllCulturesResult](#)

_abc_impl = *<_abc._abc_data object>*

```
class annize.features.documentation.common.OutputSpec
```

Bases: [ABC](#)

Base class for documentation output specifications. See `render()`.

_abc_impl = *<_abc._abc_data object>*

```
class annize.features.documentation.common.HtmlOutputSpec(*, is_homepage=False)
```

Bases: [OutputSpec](#)

HTML documentation output.

Parameters

is_homepage (*bool*) – If to render output for a homepage with slight different stylings and behavior.

property `is_homepage`

`_abc_impl = <_abc._abc_data object>`

class `annize.features.documentation.common.PdfOutputSpec`

Bases: *OutputSpec*

PDF documentation output.

`_abc_impl = <_abc._abc_data object>`

class `annize.features.documentation.common.PlaintextOutputSpec`

Bases: *OutputSpec*

Plaintext documentation output.

`_abc_impl = <_abc._abc_data object>`

class `annize.features.documentation.common.GeneratedDocument(*, document, output_spec, culture, filename)`

Bases: *FilesystemContent*

Parameters

- **generate_func** – The content generator function. It has no parameters and returns an absolute path to the content (usually inside some temporary directory).
- **document** (*Document*)
- **output_spec** (*OutputSpec*)
- **culture** (*Culture* | *None*)
- **filename** (*str* | *None*)

`_path()`

annize.features.files namespace

Subpackages

annize.features.files.transfer namespace

Submodules

annize.features.files.transfer.common module

File transfers.

class `annize.features.files.transfer.common.Endpoint`

Bases: *ABC*

abstractmethod `access_filesystem(rootpath)`

Parameters

rootpath (*str*)

Return type

ContextManager[*Path*, *bool* | *None*]

`_abc_impl = <_abc._abc_data object>`

```
class annize.features.files.transfer.common.FsEndpoint(*, fsentry, path)
```

Bases: [Endpoint](#)

Parameters

- **fsentry** ([Path](#) | *None*)
- **path** (*str* | *None*)

```
access_filesystem(rootpath)
```

Parameters

rootpath (*str*)

```
_abc_impl = <_abc._abc_data object>
```

```
class annize.features.files.transfer.common.Upload(*, source, destination_endpoint,  
                                                    destination_path)
```

Bases: *object*

Parameters

- **source** ([FilesystemContent](#))
- **destination_endpoint** ([Endpoint](#))
- **destination_path** (*str* | *Path* | *None*)

annize.features.files.transfer.ssh module

ssh-based file transfers.

```
class annize.features.files.transfer.ssh.Endpoint(*, host, port=22, username, identity_file,  
                                                  has_shell_access=False)
```

Bases: [Endpoint](#)

Parameters

- **host** (*str*)
- **port** (*int*)
- **username** (*str*)
- **identity_file** (*str* | *None*)
- **has_shell_access** (*bool*)

```
property host: str
```

```
property port: int
```

```
property username: str
```

```
property identity_file: str | None
```

```
property has_shell_access: bool
```

```
access_filesystem(rootpath)
```

locationstring(*path=None*)

Parameters

path (*str* | *None*)

Return type

str

exec(*cmdline*)

Parameters

cmdline (*str*)

Return type

TODO

_abc_impl = <_abc._abc_data object>

Submodules

annize.features.files.common module

Files and directories.

class annize.features.files.common.**FsEntry**(*, *path*, *root*)

Bases: *FilesystemContent*

A filesystem location, either relative to the Annize project root directory or another root.

If it is already known whether the entry is a file or a directory, consider using *File* or *Directory* instead. Special files (e.g. symlinks) can also be represented by a *File*.

Parameters

- **path** (*str* | *Path* | *None*) – The path that points to the referenced content (relative to *root*).
- **root** (*str* | *Path* | *FilesystemContent* | *None*) – The root directory. If unset, it is the Annize project root directory.

property root: *FilesystemContent*

The root directory.

relative_path is considered to be relative to this one.

property relative_path: *Path*

The path that points to the referenced content (relative to *root*).

_path()

class annize.features.files.common.**File**(*, *path*, *root*)

Bases: *FsEntry*

A file location, either relative to the Annize project root directory or another root.

Parameters

- **path** (*str* | *Path* | *None*) – The path that points to the referenced content (relative to *root*).
- **root** (*str* | *Path* | *FilesystemContent* | *None*) – The root directory. If unset, it is the Annize project root directory.

class annize.features.files.common.**Exclude**(*, *by_path_pattern*, *by_path*, *by_name_pattern*, *by_name*)

Bases: object

An exclusion definition. Usually used with *Directory* and *DirectoryPart*.

Parameters

- **by_path_pattern** (*str* / *None*) – Exclude by this regexp pattern on the full path.
- **by_path** (*str* / *None*) – Exclude this path.
- **by_name_pattern** (*str* / *None*) – Exclude by this regexp pattern on the file name.
- **by_name** (*str* / *None*) – Exclude this file name.

__does_exclude(*by_text*, *by_pattern*)

Parameters

- **text** (*str*)
- **by_text** (*str*)
- **by_pattern** (*Pattern*)

Return type

bool

does_exclude(*relative_path*, *source*, *destination*)

Return whether a given location is excluded by this exclusion definition.

Parameters

- **relative_path** (*Path*) – The relative path to check for exclusion.
- **source** (*Path*) – The absolute source path.
- **destination** (*Path*) – The absolute destination path.

Return type

bool

class annize.features.files.common.**ExcludeAllBut**(*, *excludes*)

Bases: *Exclude*

A negative exclusion definition.

It will exclude an item whenever *_none_* of the given inner exclude definitions match.

Parameters

excludes (*list* [*Exclude*]) – List of inner exclude definitions.

does_exclude(*relative_path*, *source*, *destination*)

Return whether a given location is excluded by this exclusion definition.

Parameters

- **relative_path** – The relative path to check for exclusion.
- **source** – The absolute source path.
- **destination** – The absolute destination path.

```
class annize.features.files.common.DirectoryPart(*, excludes, root, source_path=None,
                                                destination_path=None, path=None,
                                                destination_is_parent=False)
```

Bases: object

A part of a directory. Used in [Directory](#).

Parameters

- **excludes** (*Iterable*[[Exclude](#)]) – List of exclusion definitions.
- **root** (*str* | *Path* | [FilesystemContent](#) | *None*) – The root directory. If unset, it is the root directory specified for the owning [Directory](#).
- **source_path** (*str* | *Path* | *None*) – The path that points to the referenced content (relative to root).
- **destination_path** (*str* | *Path* | *None*) – The relative destination path inside the owning [Directory](#).
- **path** (*str* | *Path* | *None*) – Shorter way to set `source_path` and `destination_path` to the same path.
- **destination_is_parent** (*bool*) – Whether to consider the destination path as the parent of the new destination (instead of the new destination itself). The actual destination will have the same basename as the source then.

property excludes: [Sequence](#)[[Exclude](#)]

Exclusion definitions.

property root: [FilesystemContent](#) | *None*

The root directory (or *None* for the owning [Directory](#).root).

[source_path](#) is considered to be relative to this one.

property source_path: [Path](#)

The path that points to the referenced content (relative to [root](#)).

property destination_path: [Path](#)

The relative destination path inside the owning [Directory](#).

See also [destination_is_parent](#).

property destination_is_parent: *bool*

Whether to consider the destination path as the parent of the new destination (instead of the new destination itself).

```
class annize.features.files.common.Directory(*, path, root, excludes, parts, name)
```

Bases: [FsEntry](#)

A directory location, either relative to the Annize project root directory or another root.

Depending on how it is configured, this might point to a dynamically generated temporary location (e.g. if it is composed of parts or excludes are specified).

Parameters

- **path** (*str* | *None*) – The path that points to the referenced directory (relative to root). If set, do not set `parts`!
- **root** (*str* | *Path* | [FilesystemContent](#) | *None*) – The root directory. If unset, it is the Annize project root directory.

- **excludes** (*Iterable*[[Exclude](#)]) – Exclusion specifications. If some are specified, this directory will be dynamically generated. If set, do not set **parts**!
- **parts** (*Iterable*[[DirectoryPart](#)]) – Directory parts. If some are specified, this directory will be dynamically generated. Also, do not set **path** or **excludes**!
- **name** (*str* / *None*) – The name that this directory shall have (instead of its original name). If specified, this directory will be dynamically generated. It must not contain directory separator characters (mostly `"/"`).

property parts: [Sequence](#)[[DirectoryPart](#)]

property excludes: [Sequence](#)[[Exclude](#)]

_path()

__transfer_filter_for_exclude()

Parameters

exclude ([Exclude](#))

Return type

`annize.fs.Path.TTransferFilter`

class `annize.features.files.common.ProjectDirectory`

Bases: [FilesystemContent](#)

The Annize project root directory.

Parameters

generate_func – The content generator function. It has no parameters and returns an absolute path to the content (usually inside some temporary directory).

_path()

class `annize.features.files.common.MachineRootDirectory`

Bases: [FilesystemContent](#)

The machine root directory, i.e. `/`.

Parameters

generate_func – The content generator function. It has no parameters and returns an absolute path to the content (usually inside some temporary directory).

_path()

[annize.features.homepage namespace](#)

[Subpackages](#)

[annize.features.homepage.sections namespace](#)

[Submodules](#)

[annize.features.homepage.sections.about module](#)

Homepage about section.

class `annize.features.homepage.sections.about.Section`(***, *head*=<*annize.i18n.ProvidedTrStr object*>, *sort_index*=10000)

Bases: [HomepageSection](#)


```
generate_content(info)

_abc_impl = <_abc._abc_data object>
```

annize.features.homepage.sections.changelog module

Homepage changelog section.

```
class annize.features.homepage.sections.changelog.Section(*, changelog,
                                                         head=<annize.i18n.ProvidedTrStr
                                                         object>, sort_index=60000)
```

Bases: [HomepageSection](#)

Parameters

changelog ([Changelog](#) / None)

```
generate_content(info)
```

```
_abc_impl = <_abc._abc_data object>
```

annize.features.homepage.sections.documentation module

Homepage documentation section.

```
class annize.features.homepage.sections.documentation.Section(*, documentation,
                                                             head=<annize.i18n.ProvidedTrStr
                                                             object>, sort_index=30000)
```

Bases: [HomepageSection](#)

Parameters

documentation ([list](#) [[Document](#)])

```
pre_process_generate(info)
```

```
generate_content(info)
```

```
_abc_impl = <_abc._abc_data object>
```

annize.features.homepage.sections.download module

Homepage download section.

```
class annize.features.homepage.sections.download.Section(*, distributables, dependencies,
                                                         head=<annize.i18n.ProvidedTrStr
                                                         object>, sort_index=40000)
```

Bases: [HomepageSection](#)

Parameters

- **distributables** ([list](#) [[Group](#)])
- **dependencies** ([list](#) [[Dependency](#)])

```
__generate_packagelist(info)
```

Parameters

info ([_GenerateInfo](#))

```
pre_process_generate(info)
```

```
post_process_generate(info)
```

```
generate_content(info)
```

```
_abc_impl = <_abc._abc_data object>
```

```
annize.features.homepage.sections.download.friendly_filesize(isize)
```

Parameters

isize (*int*)

Return type

str

```
annize.features.homepage.sections.download.filehash(filepath)
```

Parameters

filepath (*str*)

Return type

str

annize.features.homepage.sections.gallery module

Homepage media gallery section.

```
class annize.features.homepage.sections.gallery.Section(*, head=<annize.i18n.ProvidedTrStr
                                                         object>, sort_index=50000,
                                                         media_galleries)
```

Bases: [HomepageSection](#)

Parameters

media_galleries (*list[Gallery]*)

```
pre_process_generate(info)
```

```
generate_content(info)
```

```
_abc_impl = <_abc._abc_data object>
```

annize.features.homepage.sections.imprint module

Homepage imprint section.

```
class annize.features.homepage.sections.imprint.Section(*, imprint,
                                                         head=<annize.i18n.ProvidedTrStr object>,
                                                         sort_index=70000)
```

Bases: [HomepageSection](#)

Parameters

imprint (*TrStr*)

```
generate_content(info)
```

```
_abc_impl = <_abc._abc_data object>
```

annize.features.homepage.sections.license module

Homepage license section.

```
class annize.features.homepage.sections.license.Section(*, head=<annize.i18n.ProvidedTrStr
object>, sort_index=20000)
```

Bases: [HomepageSection](#)

`generate_content`(*info*)

`_abc_impl` = <_abc._abc_data object>

Submodules

annize.features.homepage.common module

Homepage.

```
class annize.features.homepage.common.HomepageSection(*, head, sort_index=0)
```

Bases: [ABC](#)

Parameters

- `head` ([TrStr](#))
- `sort_index` (*int*)

```
class _GenerateInfo(culture: annize.i18n.Culture, custom_arg: object | None, document_root_url: str,
document_variant_directory: annize.fs.Path, document_variant_url: str)
```

Bases: [object](#)

Parameters

- `culture` ([Culture](#))
- `custom_arg` (*object | None*)
- `document_root_url` (*str*)
- `document_variant_directory` ([Path](#))
- `document_variant_url` (*str*)

`culture`: [Culture](#)

`custom_arg`: *object | None*

`document_root_url`: *str*

`document_variant_directory`: [Path](#)

`document_variant_url`: *str*

```
class _PrePostProcGenerateInfo(document_root_directory: annize.fs.Path, document_root_url: str,
custom_arg: object | None)
```

Bases: [object](#)

Parameters

- `document_root_directory` ([Path](#))
- `document_root_url` (*str*)

- `custom_arg` (*object* | *None*)

`document_root_directory:` *Path*

`document_root_url:` *str*

`custom_arg:` *object* | *None*

class `Content`(*, *rst_text=""*, *media_files=()*)

Bases: *object*

Parameters

- `rst_text` (*str*)
- `media_files` (*list*[*FilesystemContent*])

property `rst_text:` *str*

property `media_files:` *list*[*FilesystemContent*]

append_rst(*rst_text*)

Return type
None

attach_media_file(*file*)

Parameters
file (*FilesystemContent*)

Return type
None

property `head:` *TrStr*

property `sort_index:` *int*

pre_process_generate(*info*)

Parameters
info (*_PrePostProcGenerateInfo*)

Return type
None

abstractmethod `generate_content`(*info*)

Parameters
info (*_GenerateInfo*)

Return type
Content

post_process_generate(*info*)

Parameters
info (*_PrePostProcGenerateInfo*)

Return type
None

`_abc_impl = <_abc._abc_data object>`

```
class annize.features.homepage.common.Homepage(*, title, short_desc, sections, cultures)
```

Bases: object

Parameters

- **title** ([TrStr](#) / *None*)
- **short_desc** ([TrStr](#) / *None*)
- **sections** (*list* [[HomepageSection](#)])
- **cultures** (*list* [[Culture](#)])

property **cultures**: *list* [[Culture](#)]

property **sections**: *list* [[HomepageSection](#)]

property **title**: [TrStr](#)

property **short_desc**: [TrStr](#)

_append_section(*section*)

Parameters

section ([HomepageSection](#))

Return type

None

generate()

__generate_pre_post_proc(*, *is_pre*, *document_root_directory*, *document_root_url*)

Parameters

- **custom_args** (*dict*)
- **is_pre** (*bool*)
- **document_root_directory** ([Path](#))
- **document_root_url** (*str*)

__generate_section(*, *custom_args*, *culture*, *document_root_directory*, *document_root_url*, *document_variant_directory*)

Parameters

- **custom_args** (*dict*)
- **culture** ([Culture](#))
- **document_root_directory** ([Path](#))
- **document_root_url** (*str*)
- **document_variant_directory** ([Path](#))

```
class annize.features.homepage.common.SimpleProjectHomepage(*, changelog, dependencies,
                                                             distributables, documentation,
                                                             imprint, media_galleries, **kwargs)
```

Bases: [Homepage](#)

Parameters

- **changelog** ([Changelog](#) / *None*)

- **dependencies** (*list*[*Dependency*])
- **distributables** (*list*[*Group*])
- **documentation** (*list*[*Document*])
- **imprint** (*TrStr* | *None*)
- **media_galleries** (*list*[*Gallery*])

class `annize.features.homepage.common.GeneratedHomepage`(*, *homepage*)

Bases: *FilesystemContent*

Parameters

- **generate_func** – The content generator function. It has no parameters and returns an absolute path to the content (usually inside some temporary directory).
- **homepage** (*Homepage*)

`_path()`

`annize.features.i18n` namespace

Submodules

`annize.features.i18n.common` module

Internationalization, i.e. translation and similar tasks.

class `annize.features.i18n.common._ProjectDefinedTranslationProvider`

Bases: *TranslationProvider*

Internally created translation provider for backing *String* instances.

translate(*string_name*, *, *culture*)

Return the translation of a given text for a given culture (or *None* if there is no translation for it).

Note: This does NOT obey the culture's fallbacks (see `Culture.fallback_cultures`)! That functionality is implemented in higher level parts of the API.

Parameters

- **string_name** – The string name.
- **culture** – The culture.

add_translations(*string_name*, *variants*)

Parameters

- **string_name** (*str*)
- **variants** (*dict*[*str*, *str*])

Return type

None

_ProjectDefinedTranslationProvider__translations_for_string_name(*string_name*)

Parameters

string_name (*str*)

Return type

dict[*str*, *str*]

`_abc_impl = <_abc._abc_data object>`

class `annize.features.i18n.common.String(*, string_name, stringtr, **variants)`

Bases: [ProvidedTrStr](#)

A translatable text defined in an Annize project.

Do not use directly. See `TrStr.tr()`.

Parameters

- **string_name** (*str* / *None*) – The string name.
- **stringtr** (*str* / *None*)
- **variants** (*str*)

`_abc_impl = <_abc._abc_data object>`

class `annize.features.i18n.common.Culture(*, iso_639_1_language_code, region_code, fallback_cultures)`

Bases: [Culture](#)

A culture defined in an Annize project.

Do not use directly. See e.g. `get_from_iso_639_1_lang_code()` and `culture_by_spec()`.

Parameters

- **english_lang_name** – The language name in English.
- **iso_639_1_language_code** (*str*) – The ISO-639-1 language code, like "en".
- **region_code** (*str* / *None*) – Optional language variant region_code, like "US".
- **fallback_cultures** (*list*[[Culture](#)]) – List of fallback cultures. See `fallback_cultures`.

class `annize.features.i18n.common.ProjectCultures(*, cultures)`

Bases: `list`

Definition of an Annize project's target cultures.

Parameters

cultures (*list*[[Culture](#)])

`annize.features.i18n.common.project_cultures()`

Return a list of the current Annize project's target cultures. See also [ProjectCultures](#).

Return type

Sequence[[Culture](#)]

`annize.features.i18n.gettext` module

gettext-based internationalization.

class `annize.features.i18n.gettext.UpdatePOs(*, po_directory)`

Bases: `object`

Parameters

po_directory (*str* / *Path*)

```
class annize.features.i18n.gettext.GenerateMOs(*, po_directory, mo_directory, file_name)
```

Bases: object

Parameters

- **po_directory** (*str* | *Path* | *FileSystemContent*)
- **mo_directory** (*str* | *Path*)
- **file_name** (*str* | *None*)

```
class annize.features.i18n.gettext.TextSource(*, mo_directory, priority=0)
```

Bases: object

Parameters

- **mo_directory** (*str* | *Path*)
- **priority** (*int*)

annize.features.injections namespace

Submodules

annize.features.injections.common module

Injections.

```
class annize.features.injections.common.Injection
```

Bases: ABC

```
    abstractmethod inject(destination)
```

Parameters

destination (*Path*)

```
    _abc_impl = <_abc._abc_data object>
```

```
class annize.features.injections.common.FileSystemContentInjection(*, content)
```

Bases: *Injection*

Parameters

content (*str* | *Path* | *FileSystemContent*)

property content: *FileSystemContent*

```
    inject(destination)
```

Parameters

destination (*Path*)

```
    _abc_impl = <_abc._abc_data object>
```

```
class annize.features.injections.common.Inject(*, injection, destination)
```

Bases: object

Parameters

- **injection** (*Injection*)
- **destination** (*Path*)

annize.features.injections.python module

Python injections.

```
class annize.features.injections.python.ProjectInfoInjection(*, filename='project_info.py',
                                                             version)
```

Bases: [Injection](#)

Parameters

- **filename** (*str*)
- **version** ([Version](#) | *None*)

```
inject(destination)
```

Parameters

destination ([Path](#))

```
_abc_impl = <_abc._abc_data object>
```

annize.features.testing namespace

Submodules

annize.features.testing.common module

Testing.

```
class annize.features.testing.common.RunTests(**b)
```

Bases: [object](#)

```
class annize.features.testing.common.Test
```

Bases: [ABC](#)

abstractmethod [run\(\)](#)

```
_abc_impl = <_abc._abc_data object>
```

```
class annize.features.testing.common.TestGroup(*, tests)
```

Bases: [Test](#)

Parameters

tests (*list*[[Test](#)])

run()

```
_abc_impl = <_abc._abc_data object>
```

annize.features.testing.pylint module

pylint testing.

```
class annize.features.testing.pylint.Test(*, source_directory)
```

Bases: [Test](#)

Parameters

source_directory (*str* | [Path](#))

run()

```
_abc_impl = <_abc._abc_data object>
```

annize.features.testing.pytest module

pytest testing.

```
class annize.features.testing.pytest.Test(*, test_directory, source_directory)
```

Bases: *Test*

Parameters

- **test_directory** (*str* / *Path*)
- **source_directory** (*str* / *Path*)

```
run()
```

```
_abc_impl = <_abc._abc_data object>
```

annize.features.version_control namespace

Submodules

annize.features.version_control.common module

Version control.

```
class annize.features.version_control.common.VersionControlSystem
```

Bases: *ABC*

```
abstractmethod get_current_revision()
```

Return type

str

```
abstractmethod get_revision_list()
```

Return type

list[str]

```
abstractmethod get_commit_message(revision)
```

Parameters

revision (*str*)

Return type

str

```
abstractmethod get_commit_time(revision)
```

Parameters

revision (*str*)

Return type

datetime

```
abstractmethod get_revision_number(revision)
```

Parameters

revision (*str*)

Return type

int

`_abc_impl = <_abc._abc_data object>`**class** annize.features.version_control.common.**BuildVersion**(*, *base_version*, *vcs*)Bases: *Version***Parameters**

- **base_version** (*Version*)
- **vcs** (*VersionControlSystem*)

`__effversion()`**property** segments_tuples**property** text`annize.features.version_control.common.default_version_control_system()`**Return type***VersionControlSystem* | None**annize.features.version_control.git module**

git-based version control.

class annize.features.version_control.git.**VersionControlSystem**(*, *path*)Bases: *VersionControlSystem***Parameters****path** (*str* | None)**property** path: *Path*`__call_git(cmdline)`**Parameters****cmdline** (*list[str]*)**Return type**

str

`get_current_revision()``get_revision_list()``get_commit_message(revision)``get_commit_time(revision)``get_revision_number(revision)``_abc_impl = <_abc._abc_data object>`**class** annize.features.version_control.git.**ExcludeByGitIgnores**Bases: *Exclude***Parameters**

- **by_path_pattern** – Exclude by this regexp pattern on the full path.

- **by_path** – Exclude this path.
- **by_name_pattern** – Exclude by this regexp pattern on the file name.
- **by_name** – Exclude this file name.

does_exclude(*relative_path, source, destination*)

Return whether a given location is excluded by this exclusion definition.

Parameters

- **relative_path** – The relative path to check for exclusion.
- **source** – The absolute source path.
- **destination** – The absolute destination path.

Submodules

annize.features.authors module

Project author information.

class annize.features.authors.**Author**(*, *full_name, email*)

Bases: object

Parameters

- **full_name** (*TrStr*)
- **email** (*str* | *None*)

property full_name: *TrStr*

property email: *str* | *None*

annize.features.authors.**project_authors**()

Return type

list[*Author*]

annize.features.authors.**join_authors**(*authors*)

Parameters

authors (*list*[*Author*])

Return type

Author

annize.features.base module

Project base information.

class annize.features.base.**Data**(*, *project_name=None, pretty_project_name=None, summary=None, long_description=None, homepage_url=None, imprint=None, project_directory=None*)

Bases: object

Parameters

- **project_name** (*str*)
- **pretty_project_name** (*TrStr* | *None*)

```

        • summary (TrStr / None)
        • long_description (TrStr / None)
        • homepage_url (str / None)
        • imprint (TrStr / None)
        • project_directory (str / None)

property project_name: str

property pretty_project_name: TrStr

property summary: TrStr

property long_description: TrStr

property homepage_url: str

property imprint: TrStr

property project_directory: str

class annize.features.base.BrandColor(*, red, green, blue)
    Bases: Color

        Parameters

            • red (float)

            • green (float)

            • blue (float)

class annize.features.base.DateTime(*, iso)
    Bases: datetime

        Parameters

            iso (str)

class annize.features.base.Keywords(*, from_string="", split_by=' ', keywords=())
    Bases: object

        Parameters

            • from_string (str)

            • split_by (str)

            • keywords (list[str])

        property keywords: list[str]

class annize.features.base.Keyword(text)
    Bases: Keywords

        Parameters

            text (str)

annize.features.base.project_keywords()

        Return type

            Keywords

```

class annize.features.base.Basket(*, items)

Bases: [Basket](#)

Parameters

items (*list[object]*)

class annize.features.base.FirstOf(*, objects)

Bases: [Basket](#)

Parameters

objects (*list[object]*)

annize.features.base.brand_color(*, none_on_undefined=False)

Parameters

none_on_undefined (*bool*)

Return type

[Color](#)

annize.features.base._get_data(key, default)

Parameters

- **key** (*str*)
- **default** (*Any*)

Return type

Any

annize.features.base.project_name()

Return type

str

annize.features.base.pretty_project_name()

Return type

[TrStr](#)

annize.features.base.summary()

Return type

[TrStr](#)

annize.features.base.long_description()

Return type

[TrStr](#)

annize.features.base.homepage_url()

Return type

str

annize.features.base.imprint()

Return type

[TrStr](#)

annize.features.base.project_directory()

Return type

[Path](#)

annize.features.licensing module

Project licensing information.

class annize.features.licensing.**License**(*, name, text, **additional_info)

Bases: object

Parameters

- **name** ([TrStr](#))
- **text** ([TrStr](#) | *None*)

property name: [TrStr](#)

property text: [TrStr](#) | *None*

additional_info(key, *, default=*None*)

Parameters

- **key** (*str*)
- **default** (*Any*)

Return type

Any

annize.features.licensing.**_license**(name)

Parameters

name (*str*)

Return type

Type[[License](#)]

annize.features.licensing.**AFLv3**

alias of [ALicense](#)

annize.features.licensing.**AGPLv3**

alias of [ALicense](#)

annize.features.licensing.**Apache2**

alias of [ALicense](#)

annize.features.licensing.**Artistic1**

alias of [ALicense](#)

annize.features.licensing.**BSD2clause**

alias of [ALicense](#)

annize.features.licensing.**BSD3clause**

alias of [ALicense](#)

annize.features.licensing.**Cc0v1**

alias of [ALicense](#)

annize.features.licensing.**CcBy3**

alias of [ALicense](#)

annize.features.licensing.**CcByNc3**

alias of [ALicense](#)

`annize.features.licensing.CcByNcNd3`
alias of `ALicense`

`annize.features.licensing.CcByNcSa3`
alias of `ALicense`

`annize.features.licensing.CcByNd3`
alias of `ALicense`

`annize.features.licensing.CcBySa3`
alias of `ALicense`

`annize.features.licensing.GPLv3`
alias of `ALicense`

`annize.features.licensing.LGPLv3`
alias of `ALicense`

`annize.features.licensing.MIT`
alias of `ALicense`

`annize.features.licensing.MPLv11`
alias of `ALicense`

`annize.features.licensing.MPLv2`
alias of `ALicense`

`annize.features.licensing.PublicDomain`
alias of `ALicense`

`annize.features.licensing.project_licenses()`

Return type

`list[License]`

annize.features.media_galleries module

Media galleries.

class `annize.features.media_galleries.MediaType(*values)`

Bases: `Enum`

IMAGE = `'image'`

VIDEO = `'video'`

class `annize.features.media_galleries.Gallery(*, source, title)`

Bases: `object`

Parameters

- **source** (`FilesystemContent`)
- **title** (`TrStr` | `None`)

class `Item(file, description, mediatype)`

Bases: `object`

Parameters

- **file** (`FilesystemContent`)

- **description** (`TrStr` | `None`)
- **mediatype** (`MediaType`)

property file: `FilesystemContent`

property description: `TrStr` | `None`

property mediatype: `MediaType`

property items: `list[Item]`

property title: `TrStr`

_description_for_mediafile(*itemfile*)

Parameters**itemfile** (`FilesystemContent`)**Return type**`TrStr`**annize.features.task module**

Tasks.

class `annize.features.task.Task`(*, *innertasks*, *is_advanced=False*)Bases: `object`**Parameters**

- **innertasks** (`list[object]`)
- **is_advanced** (`bool`)

property is_advanced: `bool`**annize.features.version module**

Project versioning.

class `annize.features.version.Line`(*, *version*)Bases: `object`**Parameters****version** (`Version`)**property version:** `Version`**class** `annize.features.version.Version`(*, *text*, *pattern*, ***segment_values*)Bases: `Version`**Parameters**

- **text** (`str` | `None`)
- **pattern** (`VersionPattern` | `None`)

`annize.features.version.default_version_pattern`()**Return type**`VersionPattern`

`annize.features.version.project_versions()`

Return type

`list[Version]`

class `annize.features.version.CommonVersionPattern`

Bases: `VersionPattern`

annize.flow package

Execution of Annize projects.

See e.g. `annize.flow.runner.Runner` and `annize.flow.run_context.RunContext`.

Submodules

annize.flow.run_context module

Run contexts. Typically used by the infrastructure in the course of the execution of an Annize project.

See also `RunContext` and `current()`.

class `annize.flow.run_context.RunContext`

Bases: `object`

Holds data for a single execution of an Annize project (usually happening in a `annize.flow.runner.Runner`).

Beyond a few fixed data, like the root path of the Annize project configuration files, it stores every object that was created by definition in the Annize project configuration. Many parts of this API (e.g. many method names) use the term ‘object’ for all data items stored in a run context.

Each of those objects has at least one name. This can be a “friendly name”, i.e. a name that was explicitly specified in the project. If no name was specified, there will at least be an dynamically generated one, so every object is uniquely addressable by name. The dynamically generated names will differ between one project execution and another one, while friendly names are stable by nature.

There are further ways to access stored data, which do not involve names. See this class’ methods.

For each object in the store, arbitrary metadata can be stored as well.

See also `current()`.

This run context needs to be entered (with-block) during project execution.

Do not use directly.

`_IS_TOPLEVEL_OBJECT__METADATA_KEY = 'annize..is_toplevel_object'`

`ANNIZE_CONFIG_ROOTPATH__NAME = 'annize..annize_config_rootpath'`

prepare(`*`, `annize_config_rootpath`)

Prepare the execution.

Needs to be called once, before the actual execution begins, in order to make some basic data available.

Parameters

annize_config_rootpath (*Path*) – The Annize project configuration root path.

Return type

`None`

object_by_name(*name*, *default=None*, *, *create_nonexistent=False*)

Return an object by one of its names (or a default value).

See also [set_object_name\(\)](#).

Parameters

- **name** (*str*) – An object name.
- **default** (*Any*) – The default value to return when no object exists with the given name.
- **create_nonexistent** (*bool*) – (If the default value is going to be returned because no object existed with the given name) Whether to store the default value in the data store with the given name, so it can be found later.

Return type

Any

object_names(*obj*)

Return all object names for a given object (with friendly names first).

This method always returns a non-empty list. Even if the given object was not stored at all yet, it automatically gets added to the store implicitly.

See also [set_object_name\(\)](#).

Parameters

obj (*Any*) – The object.

Return type

list[str]

object_name(*obj*)

Return one object name for a given object (preferably a friendly one).

This method always returns a valid name. Even if the given object was not stored at all yet, it automatically gets added to the store implicitly.

See also [set_object_name\(\)](#).

Parameters

obj (*Any*) – The object.

Return type

str

set_object_name(*obj*, *name*)

Assign a name to an object.

All names assigned earlier remain valid as well.

See also [object_by_name\(\)](#), [object_names\(\)](#) and others.

Parameters

- **obj** (*Any*) – The object.
- **name** (*str*) – The new name.

Return type

None

objects_by_type(*obj_type*, *toplevel_only*=True)

Return all stored objects that are instance of a given type.

See also [add_object\(\)](#) and others.

Parameters

- **obj_type** (*type[T]*) – The type.
- **toplevel_only** (*bool*) – Whether to return only objects that are defined on project level.

Return type

list[T]

add_object(*obj*)

Add an object to the store and return its name.

If the object already is the store, and already has a friendly name, this one is returned. So, in fact, this method has the same effect as [object_name\(\)](#). It might just express your intent better than that one in some cases.

See also [object_by_name\(\)](#), [objects_by_type\(\)](#) and others.

Parameters

obj (*Any*) – The object to add.

Return type

str

is_friendly_name(*name*)

Returns whether the given name is a friendly one.

Parameters

name (*str*) – The name to check.

Return type

bool

is_toplevel_object(*obj*)

Return whether a given object represents the definition of an object on the Annize project file root level (e.g. by the top level tags in .xml configuration files).

Parameters

obj (*Any*) – The object to check.

Return type

bool

mark_object_as_toplevel(*obj*)

Mark an object as a toplevel one. See [is_toplevel_object\(\)](#).

Parameters

obj (*Any*) – The object.

Return type

None

object_metadata(*obj*, *key*, *default*=None)

Return a piece of metadata for a given object (or a default value if there is no value by the given key).

See also [set_object_metadata\(\)](#).

Parameters

- **obj** (*Any*) – The object.
- **key** (*str*) – The metadata key.
- **default** (*Any*) – The default value.

Return type*Any***set_object_metadata**(*obj*, *key*, *value=None*)

Store a piece of metadata for a given object.

See also [*object_metadata\(\)*](#).**Parameters**

- **obj** (*Any*) – The object.
- **key** (*str*) – The metadata key.
- **value** (*Any*) – The metadata value to store for this object and key.

Return type

None

__put_object(*name*, *obj*)**Parameters**

- **name** (*str*)
- **obj** (*Any*)

Return type

None

__object_raw_name(*obj*)**Parameters****obj** (*Any*)**Return type**

str

__object_metadata_dict(*obj*)**Parameters****obj** (*Any*)**Return type**dict[str, *Any*]**annize.flow.run_context.current()**

Return the current run context.

Note: In most cases you do not need to use this function directly. See the other functions defined on module level.

If there is no current run context (i.e. this function is called outside the execution of an Annize project), [*OutOfContextError*](#) will be raised.

Return type[*RunContext*](#)

exception `annize.flow.run_context.OutOfContextError`

Bases: `TypeError`

`annize.flow.run_context.object_by_name(name, default=None, *, create_nonexistent=False)`

Same as `RunContext.object_by_name()` on the `_current_` run context (`current()`).

Parameters

- **name** (*str*)
- **default** (*Any*)
- **create_nonexistent** (*bool*)

Return type

Any

`annize.flow.run_context.object_names(obj)`

Same as `RunContext.object_names()` on the `_current_` run context (`current()`).

Parameters

obj (*Any*)

Return type

`list[str]`

`annize.flow.run_context.object_name(obj)`

Same as `RunContext.object_name()` on the `_current_` run context (`current()`).

Parameters

obj (*Any*)

Return type

`str`

`annize.flow.run_context.set_object_name(obj, name)`

Same as `RunContext.set_object_name()` on the `_current_` run context (`current()`).

Parameters

- **obj** (*Any*)
- **name** (*str*)

Return type

`None`

`annize.flow.run_context.objects_by_type(obj_type, toplevel_only=True)`

Same as `RunContext.objects_by_type()` on the `_current_` run context (`current()`).

Parameters

- **obj_type** (*type[T]*)
- **toplevel_only** (*bool*)

Return type

`list[T]`

`annize.flow.run_context.add_object(obj)`

Same as `RunContext.add_object()` on the `_current_` run context (`current()`).

Parameters

obj (*Any*)

Return type

str

`annize.flow.run_context.is_friendly_name(name)`Same as `RunContext.is_friendly_name()` on the `_current_run` context (`current()`).**Parameters****name** (str)**Return type**

bool

`annize.flow.run_context.is_toplevel_object(obj)`Same as `RunContext.is_toplevel_object()` on the `_current_run` context (`current()`).**Parameters****obj** (Any)**Return type**

bool

`annize.flow.run_context.object_metadata(obj, key, default=None)`Same as `RunContext.object_metadata()` on the `_current_run` context (`current()`).**Parameters**

- **obj** (Any)
- **key** (str)
- **default** (Any)

Return type

Any

`annize.flow.run_context.set_object_metadata(obj, key, value=None)`Same as `RunContext.set_object_metadata()` on the `_current_run` context (`current()`).**Parameters**

- **obj** (Any)
- **key** (str)
- **value** (Any)

Return type

None

annize.flow.runner module

The Annize project runner.

See [Runner](#).

```
class annize.flow.runner.Runner(*, project, selected_task=None, user_feedback_answers,
                                user_feedback=None)
```

Bases: ABC

TODO.

Parameters

- **project** (Project)

- `selected_task` (*str* | *None*)
- `user_feedback_answers` (*dict[str, Any]*)
- `user_feedback` (*TODO*)

`run_runner()`

Return type

None

`_dispatch(func)`

`__do_run(project)`

Parameters

project (*Project*)

`abstractmethod show_task_chooser()`

Return type

None

`abstractmethod show_task_execution()`

Return type

None

`abstractmethod show_task_execution_success()`

Return type

None

`get_tasks()`

Return type

list[str]

`__set_tasks(tasks)`

Parameters

tasks (*list[str]*)

Return type

None

`get_selected_task()`

Return type

str

`set_selected_task(task_name)`

Parameters

task_name (*str*)

Return type

None

`is_finished()`

Return type

bool

get_success_state()

Return type

Tuple[bool, str]

__set_success_state(*success*, *message*)

Parameters

- **success** (*bool*)
- **message** (*str*)

Return type

None

wait_finished()

Return type

None

_abc_impl = <_abc._abc_data object>

annize.fs package

Annize filesystem API.

Used by Annize Features.

See [Path](#), [FilesystemContent](#) and others.

class `annize.fs.FilesystemContent`(*generate_func*)

Bases: object

Base class for a source of arbitrary filesystem content.

It provides access to that content by [path\(\)](#). Some implementations will return a static path to already existing content, while other implementations will return a temporary path to ad-hoc generated content.

This content can be a file, a complete directory, or anything else. It could even return a path that points to nothing. It depends on the actual implementation what kind of content it provides.

This type is used instead of plain paths in situations where dynamic filesystem content might be exchanged (usually via some temporary files) instead of already existing files or directories. So, a `FilesystemContent` usually provides a path for reading. There is no strict rule against writing at that path, but that might lead to expected behavior (e.g. when the path points to a temporary copy of something, so changes do not take the desired effect, or when it has undesired side effects on other consumers of the same instance). There might be features whose internal code does that in order to automatically handle relative paths.

Parameters

generate_func (*Callable*[[], *TInputPath*]) – The content generator function. It has no parameters and returns an absolute path to the content (usually inside some temporary directory).

path()

Return the path that points to the content.

It always returns the same path and does not do any further processing when called more than once (so it is safe and cheap to call that multiple times).

Return type

[Path](#)

class annize.fs.Path(*args, **kwargs)

Bases: Path, [FilesystemContent](#)

A path.

This is compatible to `pathlib.Path`, but provides some convenience methods that can save a few lines of code for typical operations.

Each path is also a [FilesystemContent](#). However, since `FilesystemContent` only allows absolute paths, using a relative path as a `FilesystemContent` will fail at runtime! See also [content\(\)](#).

Parameters

args (*str* / *Path* / *FilesystemContent*) – Path parts. Often this is one string, one `pathlib.Path` or one [FilesystemContent](#). The latter one is only allowed as the first part.

_path()

Return itself (in order to implement [FilesystemContent](#)).

Return type

[Path](#)

path()

Return itself (in order to implement [FilesystemContent](#)).

Return type

[Path](#)

children()

Like `iterdir()`, but sorted by name.

Return type

Sequence[[Path](#)]

ctime()

Return the ctime for this path.

Return type

datetime

mtime()

Return the mtime for this path.

Return type

datetime

write_file(data)

Write data to a file at this path (like `write_text` or `write_bytes`).

Parameters

data (*bytes* / *TrStr* / *str*) – The data to write.

Return type

None

remove(*, missing_ok=True)

Remove the file, directory, symlink, ... at this path.

Parameters

missing_ok (*bool*) – Whether it is okay if there is nothing at this path.

Return type

None

file_size()

Return the file size in bytes.

Return type

int

temp_clone(*, temp_root_path=None, basename=None)

Return a temporary clone of the content at this path.

Parameters

- **temp_root_path** (*str* / *Path* / *None*) – Optional root directory for temporary files. If unset, an OS-default will be used.
- **basename** (*str* / *None*) – Optional new basename. If unset, the original one will be used.

Return type

Path

TTransferFilter

alias of Callable[[*Path*, *Path*, *Path*], bool]

copy_to(destination, *, destination_as_parent=False, merge=False, overwrite=False, transfer_filter=None)

Copy the file, directory, symlink, ... at this path to a given destination. All missing parent directories in the destination path get created automatically.

Parameters

- **destination** (*str* / *Path*) – The destination.
- **destination_as_parent** (*bool*) – Whether to consider the destination as the parent of the new destination (instead of the new destination itself). The actual destination will have the same basename as the source then.
- **merge** (*bool*) – Whether to merge the source content into the destination. If not, each new destination directory will replace the existing one or even fail.
- **overwrite** (*bool*) – Whether to allow overwriting of the destination.
- **transfer_filter** (Callable[[*Path*, *Path*, *Path*], bool] / *None*) – The optional transfer filter to use. It can exclude particular parts from the transfer. It is a function with three *Path* parameters: The relative path of an item, the absolute source path and the absolute destination path. It returns False to skip that item.

Return type

Path

move_to(destination, *, destination_as_parent=False, merge=False, overwrite=False, transfer_filter=None)

Move the file, directory, symlink, ... at this path to a given destination. All missing parent directories in the destination path get created automatically.

Parameters

- **destination** (*str* / *Path*) – The destination.
- **destination_as_parent** (*bool*) – Whether to consider the destination as the parent of the new destination (instead of the new destination itself). The actual destination will have the same basename as the source then.
- **merge** (*bool*) – Whether to merge the source content into the destination. If not, each new destination directory will replace the existing one or even fail.

- **overwrite** (*bool*) – Whether to allow overwriting of the destination.
- **transfer_filter** (*Callable*[[*Path*, *Path*, *Path*], *bool*] | *None*) – The optional transfer filter to use. It can exclude particular parts from the transfer. It is a function with three *Path* parameters: The relative path of an item, the absolute source path and the absolute destination path. It returns *False* to skip that item.

Return type

Path

class TransferFilters

Bases: *object*

class And(**inner_filters*)

Bases: *object*

Parameters

inner_filters (*Path.TTransferFilter*)

class _TransferHelper

Bases: *object*

static transfer_to(*source*, *destination*, *, *merge*, *overwrite*, *destination_as_parent*, *action*, *transfer_filter*=*None*)

Parameters

- **source** (*Path*)
- **destination** (*Path*)
- **merge** (*bool*)
- **overwrite** (*bool*)
- **destination_as_parent** (*bool*)
- **action** (*Callable*)
- **transfer_filter** (*Callable*[[*Path*, *Path*, *Path*], *bool*] | *None*)

Return type

Path

static transfer_action_copy(*source*, *destination*)

Parameters

- **source** (*Path*)
- **destination** (*Path*)

Return type

None

static transfer_action_move(*source*, *destination*)

Parameters

- **source** (*Path*)
- **destination** (*Path*)

Return type

None

static _TransferHelper__transfer_piece(*action*, *transfer_filter*, *source*, *destination*, *merge*, *overwrite*, *relative_path*=*"*)

Parameters

- **action** (*Callable*)
- **transfer_filter** (*Callable*[[*Path*, *Path*, *Path*], *bool*] | *None*)
- **source** (*Path*)
- **destination** (*Path*)
- **merge** (*bool*)
- **overwrite** (*bool*)

- **relative_path** (*str*)

Return type

None

`annize.fs.content(f, *, root=None)`

Return a [FilesystemContent](#) for an arbitrary given path or [FilesystemContent](#).

If the input already is a valid [FilesystemContent](#), it gets returned as-is. If the input is a string, it automatically gets interpreted as a path (like [Path](#)). If it is a relative path, this function will return a [FilesystemContent](#) that interprets it relative to the current Annize project root directory (which only makes sense when used inside an Annize project execution) or another root location.

Parameters

- **f** (*str* | *Path* | [FilesystemContent](#)) – The input path or filesystem content.
- **root** (*str* | *Path* | [FilesystemContent](#) | *None*) – The path or filesystem content to be used as root directory for relative paths in *f*.

Return type[FilesystemContent](#)

`annize.fs.fresh_temp_directory(name=None, *, temp_root_path=None)`

Return a fresh empty temporary directory for arbitrary usage.

This directory will automatically be removed after the Annize project run has been finished. It can only be used for a `with`-block, which removes it directly after this block. Each instance can only be used once in the latter way.

For usage without a `with`-block, see [annize.fs.ext.FreshTempDirectory.path](#).

Parameters

- **name** (*str* | *Path* | *None*) – The optional directory name. Otherwise, the implementation will choose a name.
- **temp_root_path** (*str* | *Path* | *None*) – Optional root directory for temporary files. If unset, an OS-default will be used.

Return type[FreshTempDirectory](#)

`annize.fs.dynamic_file(*, content, file_name=None, temp_root_path=None)`

Return a ‘filesystem content’ that provides a file with some given content.

Parameters

- **content** (*str* | *bytes* | *Callable[[], str | bytes]*) – The content of this dynamic file. This may be either direct content (*str* or *bytes*) or a function that returns content.
- **file_name** (*str* | *None*) – The optional file name. Otherwise, the implementation will choose a name.
- **temp_root_path** (*str* | *Path* | *None*) – Optional root directory for temporary files. If unset, an OS-default will be used.

Return type[FilesystemContent](#)

Submodules

annize.fs.ext module

Annize filesystem API extensions.

Note: Commonly used functionality is also available in simpler ways (e.g. somehow in [annize.fs](#)).

class `annize.fs.ext.FreshTempDirectory`(*name=None, *, temp_root_path=None*)

Bases: `object`

A fresh empty temp directory for arbitrary usage.

See [annize.fs.fresh_temp_directory\(\)](#).

Do not use directly.

Parameters

- **name** (*str* | *Path* | *None*)
- **temp_root_path** (*str* | *Path* | *None*)

property `path`: *Path*

The path of this temp directory.

It is empty after creation and will be removed automatically after usage.

`__cleanup()`

class `annize.fs.ext.DynamicFile`(**, content, file_name=None, temp_root_path=None*)

Bases: *FilesystemContent*

A filesystem content that provides a file with some given content.

See [annize.fs.dynamic_file\(\)](#).

Do not use directly.

Parameters

- **content** (*str* | *bytes* | *Callable*[[*str* | *bytes*]], *str* | *bytes*)
- **file_name** (*str* | *None*)
- **temp_root_path** (*str* | *Path* | *None*)

`_TStaticContent` = *str* | *bytes*

`_TContent`

alias of *str* | *bytes* | *Callable*[[*str* | *bytes*], *str* | *bytes*]

`_path()`

class `annize.fs.ext.Mount`(*src, dst, *, options=(), mount_command=('mount',), umount_command=('umount',))*

Bases: `object`

Mounting of a filesystem.

This mounts a filesystem as long as its context is entered (`with`-block).

Parameters

- **src** (*str* | *Path*) – The filesystem to mount. Often a device file.

- **dst** (*str* / *Path*) – The mount-point.
- **options** (*Iterable[str]*) – Additional mount options.
- **mount_command** (*Sequence[str]*) – The mount command to use.
- **umount_command** (*Sequence[str]*) – The umount command to use.

property destination: *Path*

The mount-point.

annize.i18n package

Annize i18n backend.

The most fundamental mechanism around i18n is to get a translatable text (*TrStr*) from somewhere and get a translation from it, e.g. via *TrStr.translate()* or *tr_if_trstr()*.

Usually the translation is based on the current culture (*current_culture()*) - during project execution this iterates over the project's target cultures, while in UI contexts it is equal to *annize_user_interaction_culture()*.

There can be *TrStr* coming from various sources with various implementations. A common one is *ProvidedTrStr*, which is backed by the so-called “translation providers”. One typical translation provider implementation is internally based on *gettext*. There is always at least one translation provider instance of that type, fetching translations from Annize own *gettext* translations. In general, translation providers could be based on arbitrary sources and is not restricted at all to *gettext*.

Other *TrStr* might have arbitrary other ways to translate texts, not backed by translation providers. Often they generate translations dynamically, e.g. by combining other *TrStr*.

At higher level, Annize i18n provides the following functionality:

- It hosts Annize own text translations. They are backed by *gettext* and typically referenced by *tr()* internally.
 - Annize projects are allowed to use those texts when convenient. A translation provider for them always exists, so

a project could contain nodes like `<String xmlns="annize:i18n" string_name="an_int_DebianPackage"/>`. Find all available texts in the top level directory i18n of Annize.

- It allows Annize projects to define and use own translated texts. - Either directly inside project configuration or via *gettext*.

The former can be done with a node like `<String xmlns="annize:i18n"><String.en>Yes</String.en><String.de>Ja</String.de></String>`. Usage of *gettext* involves the definition of a *annize.features.i18n.gettext.TextSource* and nodes like `<String.stringtr xmlns="annize:i18n">tr("myOwnStringName")</String.stringtr>`. More steps are needed to generate the required *.mo*-files (see below). Note: Even for texts that are directly defined in the project, if you add a *string_name* to them, you can also reference them in the same way as *gettext* based texts.

- It allows Annize projects to override Annize own text translations. - Either directly inside project configuration or via *gettext* (mostly like described above). It is also

possible add new languages or to override only some languages.

- It helps Annize projects to deal with *gettext* *.mo*- and *.po*-files; no matter whether these texts are used in the Annize project configuration or in the project's source code. See *annize.features.i18n.gettext.UpdatePOs* and *annize.features.i18n.gettext.GenerateMOs*.

class annize.i18n.TranslationProvider

Bases: ABC

Base class for objects that provide translations for some strings in some languages (here usually called: cultures).

Most translatable texts are backed by translation providers (some only indirectly or not at all). This class is a fundamental part of the Annize i18n API, although only small parts of Annize code need to deal with them directly.

See [translate\(\)](#) and also [translation_providers\(\)](#) and [add_translation_provider\(\)](#).

abstractmethod `translate(string_name, *, culture)`

Return the translation of a given text for a given culture (or None if there is no translation for it).

Note: This does NOT obey the culture's fallbacks (see [Culture.fallback_cultures](#))! That functionality is implemented in higher level parts of the API.

Parameters

- **string_name** (*str*) – The string name.
- **culture** ([Culture](#)) – The culture.

Return type*str* | None`_abc_impl = <_abc._abc_data object>`**class** annize.i18n.GettextTranslationProvider(*mo_path, domain_name=None*)Bases: [TranslationProvider](#)

A translation provider that is backed by .mo-files from gettext.

Parameters

- **mo_path** (*str* | *Path*)
- **domain_name** (*str* | None)

translate(*string_name, *, culture*)

Return the translation of a given text for a given culture (or None if there is no translation for it).

Note: This does NOT obey the culture's fallbacks (see [Culture.fallback_cultures](#))! That functionality is implemented in higher level parts of the API.

Parameters

- **string_name** – The string name.
- **culture** – The culture.

class _NoneTranslations(*fp=None*)

Bases: NullTranslations

`_abc_impl = <_abc._abc_data object>`**annize.i18n.add_translation_provider(provider, *, priority=0)**

Add a new translation provider.

See also [translation_providers\(\)](#).

Parameters

- **provider** ([TranslationProvider](#)) – The new translation provider.
- **priority** (*int*) – The priority. Providers with lower priority value are queried earlier.

Return type

None

`annize.i18n.translation_providers()`

Return all translation providers.

See also [add_translation_provider\(\)](#).**Return type**list[[TranslationProvider](#)]`annize.i18n.TCultureSpec = 'Culture|str|None'`Types that can specify a particular culture. See e.g. [culture_by_spec\(\)](#).`annize.i18n.tr(string_name, *, culture=None)`Return the translation for a text in the current culture or any other one, or raise [TranslationUnavailableError](#) if no translation is available for that culture (or its fallbacks).Instead of this function, depending on the use case, [TrStr.tr\(\)](#) might be the right choice.**Parameters**

- **string_name** (*str*) – The string name.
- **culture** ([Culture](#) | *str* | *None*) – The culture.

Return type*str***class** `annize.i18n.TrStr`Bases: [ABC](#)

Base class for translatable texts.

Each instance can hold the translation for one text for different cultures. In order to translate it to the current culture, the simplest is to just apply `str()` on it.See also [tr\(\)](#).**static** `tr(string_name)`Return a translatable text (by querying the translation providers; see [translation_providers\(\)](#)).**Parameters**

- **string_name** (*str*) – The string name.

Return type[TrStr](#)**translate**(*culture=None*)Return the translation of this text for the current culture or any other one, or raise [TranslationUnavailableError](#) if no translation is available for that culture (or its fallbacks).**Parameters**

- **culture** ([Culture](#) | *str* | *None*) – The culture.

Return type*str***abstractmethod** `get_variant(culture)`Return the translation of this text for a given culture (or *None* if there is no translation for it).Note: This is implemented by subclasses, but usually not called directly from outside. See [translate\(\)](#). This does NOT obey the culture's fallbacks.

Parameters

culture ([Culture](#)) – The culture.

Return type

str | None

format(*args, **kwargs)

Return a formatted variant of this text (i.e. similar to `Python str.format()`).

Parameters

- **args** – Formatting args.
- **kwargs** – Formatting kwargs.

Return type

[TrStr](#)

`_abc_impl = <_abc._abc_data object>`

class `annize.i18n.ProvidedTrStr(string_name)`

Bases: [TrStr](#)

Representation for a translatable text backed by the translations providers.

Do not use directly. See [TrStr.tr\(\)](#).

Parameters

string_name (str) – The string name.

property `string_name`

get_variant(culture)

Return the translation of this text for a given culture (or None if there is no translation for it).

Note: This is implemented by subclasses, but usually not called directly from outside. See `translate()`. This does NOT obey the culture's fallbacks.

Parameters

culture – The culture.

`_abc_impl = <_abc._abc_data object>`

`annize.i18n.TrStrOrStr = annize.i18n.TrStr | str`

Type annotation for something that can be either a `str` or a [TrStr](#).

`annize.i18n.tr_if_trstr(text, *, culture=None)`

Translate a given text (if it is not a plain `str`) to the current culture or any other one, or raise [TranslationUnavailableError](#) if no translation is available for that culture (or its fallbacks).

This is a convenience function that (a) can take either a translatable text or a plain `str` and (b) allows to specify the target culture. In cases this is not needed, there are probably simpler ways to do the same.

Parameters

- **text** ([TrStr](#) / str) – The text to translate.
- **culture** ([Culture](#) / str / None) – The culture.

Return type

str

`annize.i18n.to_trstr(text)`

Return a translatable text for a given text.

This is a no-op for translatable texts, but returns a (technically) translatable text for a plain `str`. In the latter case, the translation will be the input text for all cultures.

This is useful when you need a translatable text (e.g. as input parameter) but maybe only have a plain `str`.

Parameters

text (`TrStr` / `str`) – The text.

Return type

`TrStr`

`class annize.i18n._FixedTrStr(text)`

Bases: `TrStr`

Parameters

text (`str`)

`get_variant(culture)`

Return the translation of this text for a given culture (or `None` if there is no translation for it).

Note: This is implemented by subclasses, but usually not called directly from outside. See `translate()`.

This does NOT obey the culture's fallbacks.

Parameters

culture – The culture.

`_abc_impl = <_abc._abc_data object>`

`class annize.i18n.Culture(english_lang_name, iso_639_1_language_code, region_code, fallback_cultures)`

Bases: `object`

Representation for an Annize culture. This includes the specification of a language and an optional language variant.

The major purpose of Annize i18n backend is to generate culture-specific translations for some texts.

Enter the culture context (`with-block`) in order to make it the current culture. This can also be done in a nested way (the former current culture does not take any effect meanwhile, but becomes the current culture again after this context). This is done by the UI, but also during the execution of an Annize project (iterating over its target cultures).

Annize projects choose their target cultures by means of `annize.features.i18n.common.Culture`.

Do not use directly. See e.g. `get_from_iso_639_1_lang_code()` and `culture_by_spec()`.

Parameters

- **english_lang_name** (`str`) – The language name in English.
- **iso_639_1_language_code** (`str`) – The ISO-639-1 language code, like "en".
- **region_code** (`str` / `None`) – Optional language variant region_code, like "US".
- **fallback_cultures** (`Iterable[Culture]`) – List of fallback cultures. See `fallback_cultures`.

`static get_from_iso_639_1_lang_code(iso_639_1_language_code, region_code=None, *, fallback_cultures=())`

Return a culture by its ISO-639-1 language code (and an optional region_code).

Parameters

- **iso_639_1_language_code** (*str*) – The ISO-639-1 language code, like "en".
- **region_code** (*str* / *None*) – Optional language variant region_code, like "US".
- **fallback_cultures** (*Iterable[Culture]*) – List of fallback cultures. See [fallback_cultures](#).

Return type[Culture](#)**property english_lang_name: str**

The language name in English.

property iso_639_1_language_code: str

The ISO-639-1 language code, like "en".

property region_code: str | None

Optional language variant region_code, like "US".

property full_name: str

The full culture code (incl. the region code), like "en-US" or "en".

property fallback_cultures: Sequence[Culture]

Fallback cultures.

Most parts of the API (unless documented otherwise) try those fallback cultures (in their original order) when an operation was not possible with this culture (e.g. there was no translation available for this culture).

This can be used for cultures that ‘inherit’ from other ones, but also internally by Annize UI in order to fall back to English if there is no UI translation available for the user’s language.

Note: For a culture with a region code, fallbacks usually contain the region-less culture implicitly, so e.g. de_DE and de_CH automatically fall back to de.

culture_list()

Return a list that starts with this culture and then all fallback cultures in an expanded way, i.e. including their fallback cultures (recursively).

The result does never contain duplicates and also handles circular references of fallback cultures.

For any function that explicitly regards fallback cultures, this is the list they iterate over.

Return type[Iterable\[Culture\]](#)**__best_system_locale()****Return type**

str

__current_system_locale_setup()**Return type**[_TSystemLocaleSetup](#)**__set_system_locale_setup()****Parameters****system_locale_setup** ([_TSystemLocaleSetup](#))**Return type**

None

`__set_env__var(value)`

Parameters

- **key** (*str*)
- **value** (*str* | *None*)

Return type

None

class `_TSystemLocaleSetup`(*LC_ALL: str | None, LANGUAGE: str | None*)

Bases: `object`

Parameters

- **LC_ALL** (*str* | *None*)
- **LANGUAGE** (*str* | *None*)

LC_ALL: *str* | *None*

LANGUAGE: *str* | *None*

class `annize.i18n.UnspecifiedCulture`

Bases: `Culture`

Unspecified culture.

To be used whenever no particular culture is specified.

Do not use directly. See e.g. `get_from_iso_639_1_lang_code()` and `culture_by_spec()`.

Parameters

- **english_lang_name** – The language name in English.
- **iso_639_1_language_code** – The ISO-639-1 language code, like "en".
- **region_code** – Optional language variant region_code, like "US".
- **fallback_cultures** – List of fallback cultures. See `fallback_cultures`.

class `annize.i18n._CultureFence`

Bases: `Culture`

Do not use directly. See e.g. `get_from_iso_639_1_lang_code()` and `culture_by_spec()`.

Parameters

- **english_lang_name** – The language name in English.
- **iso_639_1_language_code** – The ISO-639-1 language code, like "en".
- **region_code** – Optional language variant region_code, like "US".
- **fallback_cultures** – List of fallback cultures. See `fallback_cultures`.

`annize.i18n._last_resort_culture = <annize.i18n.Culture object>`

The last resort culture. In some internal places, this is used as the final fallback if the specified culture (incl. its fallbacks) is not available.

`annize.i18n.culture_by_spec(culture)`

Return a culture for a given culture spec (i.e. a culture, a string representing one or None).

This is a no-op for a culture, return the current culture for None or uses `Culture.get_from_iso_639_1_lang_code()` for a string (after maybe splitting it into the language code and the region code).

Parameters

culture (`Culture` / `str` / `None`) – The culture spec.

Return type

`Culture`

`annize.i18n.current_culture()`

Return the current culture. If there is no current culture, raise `NoCurrentCultureError`.

During project execution, this is usually not the same as the `annize_user_interaction_culture` but one of the cultures targeted by that project.

Return type

`Culture`

`annize.i18n._annize_user_interaction_culture()`

Return type

`Culture`

`annize.i18n.annize_user_interaction_culture = <annize.i18n.Culture object>`

The culture for interaction with the user. During project execution, this is potentially not the same as the `current_culture()`.

`annize.i18n.friendly_join_string_list(texts)`

Return a translatable string for a list of texts. They usually get concatenated with ", " between, but with something like " and " as the last separator; like "foo, bar and baz".

Parameters

texts (`list[TrStr` / `str]`) – The input texts.

Return type

`TrStr`

exception `annize.i18n.NoCurrentCultureError`

Bases: `TypeError`

Error that occurs when the current culture was requested when there is no current culture.

exception `annize.i18n.TranslationUnavailableError(text, language)`

Bases: `TypeError`

Error that occurs when a translatable text was asked for translation to a language where no translation is available for.

Parameters

- **text** (`TrStr`)
- **language** (`str`)

annize.object package

Handling of Annize objects.

There is no particular subclass that all Annize objects inherit from! Annize objects can be of arbitrary types.

There are some decorators for optional configuration and finetuning of Annize objects' methods and attributes here.

In submodules, there are routines for object and object type handling, internally used by the infrastructure.

`annize.object.explicit_only(arg_name)`

Parameters

`arg_name` (*str*)

Submodules

annize.object.controller module

TODO.

`class annize.object.controller._CreateObjectHelper`

Bases: `object`

`class ParameterConfig(parameter_name: str, explicit_only: bool | None = None)`

Bases: `object`

Parameters

- `parameter_name` (*str*)
- `explicit_only` (*bool | None*)

`parameter_name: str`

`explicit_only: bool | None = None`

`with_updates(oconfig)`

Parameters

`oconfig` (`ParameterConfig`)

Return type

`ParameterConfig`

`static create_object(call_type, args, kwargs)`

Parameters

- `args` (*Iterable*)
- `kwargs` (*dict*)

`static _CreateObjectHelper__convert_kwargs_from_string(argument_infos, args, kwargs)`

`static _CreateObjectHelper__determine_matching_keywords_for_arg(arg, call_type, argument_infos)`

`static _CreateObjectHelper__fill_empty_lists(argument_infos, args, kwargs)`

`static _CreateObjectHelper__fill_unspecified_optionals(argument_infos, args, kwargs)`

```
static _CreateObjectHelper__get_parameter_config(call_type, arg_name)
```

Parameters

- **call_type** (*type*)
- **arg_name** (*str*)

Return type

[ParameterConfig](#)

```
static _CreateObjectHelper__put_item_into_kwargs(arg, kwargs, kwarg_name, param_type_info)
```

```
static _CreateObjectHelper__shift_args_to_kwargs(call_type, argument_infos, args, kwargs)
```

```
annize.object.controller.create_object(call_type, args, kwargs)
```

Parameters

- **args** (*Iterable*)
- **kwargs** (*dict*)

```
exception annize.object.controller.MultipleValuesForSingleArgumentError(argname)
```

Bases: [TypeError](#)

Parameters

argname (*str*)

[annize.object.parameter_info module](#)

TODO.

```
class annize.object.parameter_info.ParameterInfo(name, resolved_type, is_optional,  
                                                construct_from_string_func)
```

Bases: [object](#)

Parameters

- **name** (*str*)
- **resolved_type** (*type* | *None*)
- **is_optional** (*bool*)
- **construct_from_string_func** (*Callable*[[*str*], *Any*] | *None*)

property **name**: *str*

property **resolved_type**: *type* | *None*

matches_object(*obj*)

Parameters

obj (*object*)

Return type

bool

matches_type(*type_*)

Parameters

type_ (*type*)


```

    Return type
        bool
matches_inner_type(type_)

    Parameters
        type_ (type)

    Return type
        bool

property is_optional: bool

property inner_type_info: ParameterInfo | None

property allows_multiple_args: bool

property is_constructable_from_string: bool

construct_from_string(s)

    Parameters
        s (str)

    Return type
        Any

class annize.object.parameter_info.ListParameterInfo(name, is_optional, innertypeinfo)
    Bases: ParameterInfo

    Parameters
        • name (str)
        • is_optional (bool)

    property allows_multiple_args

    property inner_type_info

class annize.object.parameter_info.UnionParameterInfo(name, is_optional,
                                                         union_member_type_infos)
    Bases: ParameterInfo

    Parameters
        • name (str)
        • is_optional (bool)

    property resolved_type

    matches_object(obj)

annize.object.parameter_info._get_type_info(for_type, as_optional=False)

    Parameters
        as_optional (bool)

    Return type
        ParameterInfo

```

`annize.object.parameter_info.type_parameter_infos(*, for_callable)`

Parameters

`for_callable` (*Callable*)

Return type

dict[str, [ParameterInfo](#)]

annize.project package

Annize projects.

See [Project](#), [Node](#) and also the submodules.

class `annize.project.Project`(*node, annize_config_rootpath*)

Bases: `object`

An Annize project.

The configuration structure is available in [node](#).

Do not use directly.

Load a project with [load\(\)](#) or create a fresh project with [create_new\(\)](#). Save changes with [save\(\)](#).

Parameters

- `node` ([ProjectNode](#))
- `annize_config_rootpath` (*str* / *Path*)

property `node`: [ProjectNode](#)

The project node.

This contains the entire configuration structure of this project.

property `annize_config_rootpath`: `Path`

The “config root path” of this Annize project.

This is usually not the same as the project’s “root path”, but a subdirectory like ‘-meta’ inside it.

static `load`(*project_path*)

Load a project from disk. Return `None` if the given path does not point into an Annize project.

Parameters

`project_path` (*str* / *Path*) – A path somewhere inside the project to be opened.

Return type

[Project](#) | `None`

save()

Save the current state of the project back to disk.

static `create_new`(*project_root_path, subdirectory_name='meta'*)

Create a new Annize project.

This will create an initial version of the Annize project configuration on disk as well.

Parameters

- `project_root_path` (*str* / *Path*) – The project root path.
- `subdirectory_name` (*str*) – The subdirectory name where to store the Annize configuration files inside the project root directory. This is not arbitrary but must be one of the well known ones!

Return type[Project](#)**class** `annize.project.Node`Bases: `ABC`

Nodes are the building blocks of a project.

They exist in a serialized way in the project files (usually xml), and when the project is loaded to memory (see [annize.project.loader](#)) they are represented by a structure of `Node` instances.

Each `Node` has various features (see methods and properties of this class), e.g. it can be observed for changes. Each node can also have children. This is just a base class for more specific node types, though. See also its subclasses in the same module.

The most relevant subclass in many regards is [ObjectNode](#).

class `ChangeEvent(target_node)`Bases: `object`

Base class for events on a [Node](#). See subclasses and [Node.add_change_handler\(\)](#).

Parameters**target_node** ([Node](#))**property** `target_node`: [Node](#)

The target node this event is about.

class `ChildrenListChangeEvent(target_node, child_node, child_position)`Bases: [ChangeEvent](#)

Base class for events on a [Node](#) that are about changes on the list of children. See subclasses.

Parameters

- **target_node** ([Node](#))
- **child_node** ([Node](#))
- **child_position** (`int`)

property `child_node`: [Node](#)

The child node this event is about.

property `child_position`: `int`

The position of the child node in the list of children.

class `ChildAddedEvent(target_node, child_node, child_position)`Bases: [ChildrenListChangeEvent](#)

Node event that occurs when a child node was added.

Parameters

- **target_node** ([Node](#))
- **child_node** ([Node](#))
- **child_position** (`int`)

class `ChildRemovedEvent(target_node, child_node, child_position)`Bases: [ChildrenListChangeEvent](#)

Node event that occurs when a child node was removed.

Parameters

- **target_node** ([Node](#))
- **child_node** ([Node](#))
- **child_position** (*int*)

class [PropertyChangedEvent](#)(*target_node, property_name, old_value, new_value*)

Bases: [ChangeEvent](#)

Node event that occurs when a property of a node was changed.

Parameters

- **target_node** ([Node](#))
- **property_name** (*str*)
- **old_value** (*Any*)
- **new_value** (*Any*)

property **property_name**: **str**

The property name.

property **old_value**: **Any**

The old property value.

property **new_value**: **Any**

The new property value.

add_change_handler(*handler, *, also_watch_children*)

Add a function that handles changes on this node.

See also [remove_change_handler\(\)](#).

Parameters

- **handler** (*Callable[[[ChangeEvent](#)], None]*) – The handler function to add.
- **also_watch_children** (*bool*) – Whether this function shall also observe this node's children.

remove_change_handler(*handler*)

Remove a change handler function that was added by [add_change_handler\(\)](#) earlier.

If that function was added multiple times, it will remove all of them. If the function was not added, this will do nothing.

Parameters

handler (*Callable[[[ChangeEvent](#)], None]*) – The handler function to remove.

__changed__helpers(*event*)

Parameters

event ([ChangeEvent](#))

_changed__child_added(*child_node, child_position*)

Parameters

- **child_node** ([Node](#))
- **child_position** (*int*)

_changed__child_removed(*child_node*, *child_position*)

Parameters

- **child_node** (*Node*)
- **child_position** (*int*)

_changed__property_changed(*node*, *property_name*, *old_value*, *new_value*)

Parameters

- **node** (*Node*)
- **property_name** (*str*)
- **old_value** (*Any*)
- **new_value** (*Any*)

property parent: *Node* | *None*

This node's parent node.

property children: *Iterable[Node]*

This node's child nodes.

insert_child(*i*, *node*)

Insert a new child node.

Parameters

- **i** (*int*) – The position.
- **node** (*Node*) – The node to insert.

Return type

None

append_child(*node*)

Append a new child node.

Parameters

node (*Node*) – The node to append.

Return type

None

remove_child(*node*)

Remove a child node.

If that node is not a child node, it raises a *ValueError*.

Parameters

node (*Node*) – The node to remove.

Return type

None

abstractmethod classmethod _allowed_child_types()

Return a list of node types that this node type allows to have as child nodes.

Return type

Iterable[type[Node]]

clone(*with_children=True, with_marshallers=False*)

Parameters

- **with_children** (*bool*)
- **with_marshallers** (*bool*)

Return type

[Node](#)

description(*, *with_children=True, multiline=True*)

Parameters

- **with_children** (*bool*)
- **multiline** (*bool*)

Return type

str

__description(*indent, with_children, multiline*)

Parameters

- **indent** (*int*)
- **with_children** (*bool*)
- **multiline** (*bool*)

Return type

str

abstractmethod _str_helper()

Return type

Iterable[str]

_abc_impl = <_abc._abc_data object>

class annize.project.**ProjectNode**

Bases: [Node](#)

An Annize project root node.

Each project has exactly one root node. It has no parent. Its children are the Annize project configuration files. It has no direct serialized representation (or, one could argue, it is the directory that contains these files).

save()

Store the current state to the Annize project configuration files.

insert_child(*i, node*)

Insert a new child node.

Parameters

- **i** – The position.
- **node** – The node to insert.

__changed_handler(*event*)

Parameters

event ([ChangeEvent](#))

get_changes(* , since=0, until=9223372036854775807)

Return all changes that happened to the project, since the moment of loading it or any later point in time, and until now or any earlier point in time.

All timestamp arguments are based on a virtual clock (which basically increases by 1 for each change). See `TODO`.

Parameters

- **since** (*int*) – The timestamp where to start with returning changes (inclusive).
- **until** (*int*) – The timestamp where to stop with return changes (non-inclusive).

Return type

list[[ChangeEvent](#)]

__compacted_changelist()

Parameters

events (list[[ChangeEvent](#)])

Return type

list[[ChangeEvent](#) | None]

undo_changes(*since*)

Undo all changes that happened to the project since a given point in time.

Parameters

since (*int*) – The timestamp where to start with undoing changes (inclusive).

Return type

None

static load(*path*)

Parameters

path (*str* | *Path*)

Return type

[ProjectNode](#)

classmethod _allowed_child_types()

Return a list of node types that this node type allows to have as child nodes.

_str_helper()

_abc_impl = <_abc._abc_data object>

class annize.project.**FileNode**(*path*, *marshaller*)

Bases: [Node](#)

An Annize project file node.

Each project has one file node per configuration file. They are the children of the [ProjectNode](#). The children of a file node are mostly of type [ObjectNode](#), but can also be different ones.

Parameters

- **path** (*str* | *Path*)
- **marshaller** (*TODO*)

property path: Path

The file path.

property marshaler: TODO

_str_helper()

classmethod _allowed_child_types()

Return a list of node types that this node type allows to have as child nodes.

_abc_impl = <_abc._abc_data object>

class annize.project.ArgumentNode

Bases: [Node](#), ABC

Base class for nodes that can be used as an argument, usually in an [ObjectNode](#).

See subclasses.

property name: str | None

The name of this argument node.

Names are used for a few purposes (the documentation will mention that where it is important), but primarily you can refer to a named argument with a [ReferenceNode](#) and you can use it for [append_to](#).

property append_to: str | None

The name of another argument node where this argument node gets appended to its children at runtime.

This essentially makes this argument node appear twice at runtime. It will also be in the place where it was defined; just a reference to that argument is created as a result.

property arg_name: str | None

The argument name where this argument is associated to in the parent object.

Valid argument names depend on the type of object that the parent is representing.

_str_helper()

_abc_impl = <_abc._abc_data object>

class annize.project.ObjectNode

Bases: [ArgumentNode](#)

An Annize project object node.

In a typical Annize project, most nodes are object nodes. Most structure in their project files represent them (usually the tags in xml files). All the other node types are basically related to containing object nodes (like file nodes or the project root node) or have other support purposes.

Children are mostly other object nodes, [ScalarValueNode](#) or [ReferenceNode](#). They are associated to a particular parameter name (of the object type) by their [ArgumentNode.arg_name](#).

property type_name: str

The name of the type of this object.

property feature: str

The Annize feature name that provides this object.

_str_helper()


```
classmethod _allowed_child_types()
```

Return a list of node types that this node type allows to have as child nodes.

```
_abc_impl = <_abc._abc_data object>
```

```
class annize.project.ScalarValueNode
```

Bases: [ArgumentNode](#)

An Annize project scalar value node.

It represents a fixed string value.

```
property value: str
```

The string that this node represents.

```
_str_helper()
```

```
__shorten(maxlen=100)
```

Parameters

- **obj** (*Any*)
- **maxlen** (*int*)

Return type

str

```
classmethod _allowed_child_types()
```

Return a list of node types that this node type allows to have as child nodes.

```
_abc_impl = <_abc._abc_data object>
```

```
class annize.project.ReferenceNode
```

Bases: [ArgumentNode](#)

A reference node.

This node represents a reference to another node (by its [ArgumentNode.name](#))

```
class OnUnresolvableAction(*values)
```

Bases: Enum

```
FAIL = 'fail'
```

```
SKIP = 'skip'
```

```
property reference_key: str
```

The name of the node this node references to.

```
property on_unresolvable: OnUnresolvableAction
```

```
_str_helper()
```

```
classmethod _allowed_child_types()
```

Return a list of node types that this node type allows to have as child nodes.

```
_abc_impl = <_abc._abc_data object>
```

class annize.project.**BlockNode**

Bases: *Node*

A block node without any own behavior.

The purpose of that block differs for particular subclass.

_str_helper()

classmethod **_allowed_child_types()**

Return a list of node types that this node type allows to have as child nodes.

_abc_impl = **<_abc._abc_data object>**

class annize.project.**BlockNodeWithScope**

Bases: *BlockNode*

Base class for a block node with an additional scope definition.

The purpose of the block and the scope definition depend on the particular subclass.

class **Scope(*values)**

Bases: Enum

BLOCK = 'block'

FILE = 'file'

PROJECT = 'project'

_str_helper()

property **scope:** *Scope*

The scope of this block.

_abc_impl = **<_abc._abc_data object>**

class annize.project.**OnFeatureUnavailableNode**

Bases: *BlockNodeWithScope*

An Annize project on-Feature-unavailable definition node.

They control how Annize behaves when the project configuration refers to a Feature that is not available.

The scope defines whether this rule applies only for the block, for the entire file that contains it, or for the entire project, while *do* defines if and how much gets ignored when the specified Feature is not available.

class **Action(*values)**

Bases: Enum

FAIL = 'fail'

SKIP_BLOCK = 'skip_block'

SKIP_NODE = 'skip_node'

_str_helper()

property **feature:** **str**

The name of the Feature that gets checked by this node. Empty string or * (the default) means all features.

property do: [Action](#)

The action when the specified Feature is not available.

_abc_impl = <_abc._abc_data object>

exception annize.project.**FeatureUnavailableError**(*feature_name*)

Bases: ModuleNotFoundError

Parameters

feature_name (*str*)

exception annize.project.**BadStructureError**(*message*)

Bases: ValueError

Parameters

message (*str*)

exception annize.project.**MaterializerError**(*message*)

Bases: TypeError

Parameters

message (*str*)

exception annize.project.**ParserError**(*message*)

Bases: ValueError

Parsing error like bad input xml.

Parameters

message (*str*)

exception annize.project.**UnresolvableReferenceError**(*reference_key*)

Bases: [MaterializerError](#)

Parameters

reference_key (*str*)

Subpackages

annize.project.file_formats package

File formats for Annize configuration files.

See also the submodules.

class annize.project.file_formats.**FileFormat**

Bases: ABC

A file format for Annize configuration files.

class **Marshaler**

Bases: ABC

abstractmethod **add_change**(*change*)

Parameters

change (*TODO*)

Return type

None

_abc_impl = <_abc._abc_data object>

classmethod `parse_file(path)`

Read the given file and return a project file node for it.

Parameters

path (*str* | *Path*) – The file to parse.

Return type

FileNode

`_abc_impl = <_abc._abc_data object>`

`annize.project.file_formats.register_file_format(format_name)`

Parameters

format_name (*str*)

`annize.project.file_formats.get_format(format_name)`

Parameters

format_name (*str*)

Return type

FileFormat | *None*

`annize.project.file_formats.parse(path)`

Parameters

path (*str* | *Path*)

Return type

ProjectNode

Submodules

`annize.project.file_formats.xml` module

Support for Annize configuration XML files.

class `annize.project.file_formats.xml.XmlFileFormat`

Bases: *FileFormat*

classmethod `parse_file(path)`

Read the given file and return a project file node for it.

Parameters

path – The file to parse.

`_abc_impl = <_abc._abc_data object>`

class `annize.project.file_formats.xml._XmlParser`

Bases: *object*

class `_Context`

Bases: *object*

property `marshaller`

in_file(*fpath*, *marshaller*)

Parameters

fpath (*str* | *Path*)

```

in_node(xnode)

static _Context__nodeshort(xnode, maxlen=100)
    Parameters
        xnode (Element)

class _TagParts(name, namespace="")
    Bases: object
        Parameters
            namespace (str)

ATTRIBUTE_COMMAND_START = '~'
ATTRIBUTE_COMMAND_END = '~'

classmethod escape_attribute_string(txt)
    Parameters
        txt (str)
    Return type
        str

parse_file(fpath)
    Parameters
        fpath (str | Path)
    Return type
        FileNode

classmethod _XmlParser__interpret_attribute_string(txt)
    Parameters
        txt (str)
    Return type
        Tuple[bool, str]

classmethod _XmlParser__parse_attr(key, value)
    Parameters
        • key (str)
        • value (str)
    Return type
        Node

_XmlParser__parse_child(node, xnode)
    Parameters
        • node (Node)
        • xnode (Element)
    Return type
        Tuple[Node, Element]

```

`_XmlParser__parse_children(node, xparent)`

Parameters

- `node` ([Node](#))
- `xparent` ([Element](#))

`_XmlParser__parse_tag(node, argname, callname, feature, xnode)`

Parameters

`node` ([Node](#))

`class annize.project.file_formats.xml.Marshaler`

Bases: [Marshaler](#)

`class XmlDocumentLocation(element: <cyfunction Element at 0x7f84829360c0>, attr_name: str = "")`

Bases: `object`

Parameters

- `element` ([Element](#))
- `attr_name` (*str*)

`element`: [Element](#)

`attr_name`: *str* = ''

`add_change(change)`

`add_element(node, xelem)`

Parameters

- `node` ([Node](#))
- `xelem` ([Element](#))

`add_element_attr(node, xelem, attrname)`

Parameters

- `node` ([Node](#))
- `xelem` ([Element](#))
- `attrname` (*str*)

`_abc_impl = <_abc._abc_data object>`

`add_element_tree(node, xtree)`

Parameters

- `node` ([Node](#))
- `xtree` ([ElementTree](#))

`save_filenode_to_file(node)`

Parameters

`node` ([FileNode](#))

annize.project.materializer package

Materializing of Annize projects into a working runtime structure (usually used by the Runner application).

See [materialize\(\)](#).

All submodules are only used internally by this one. There is a core part, some preprocessor functions, and some behaviors that implement what it does for different types of project nodes.

class `annize.project.materializer.MaterializationResult`(*root_objects*, *node_association*, *problems*)

Bases: `object`

Parameters

- **root_objects** (`list[Any]`)
- **node_association** (`dict[Node, list[Any]]`)
- **problems** (`dict[Node | None, list[Exception]]`)

property `root_objects`: `list[Any]`

objects_for_node(*node*)

Parameters

node (`Node`)

Return type

`list[Any] | None`

erroneous_nodes()

Return type

`list[Node]`

errors_for_node(*node*)

Parameters

node (`Any`)

Return type

`list[Exception]`

`annize.project.materializer.materialize`(*project*, *, *feature_loader=None*)

Parameters

- **project** (`ProjectNode`)
- **feature_loader** (`FeatureLoader | None`)

Return type

[MaterializationResult](#)

`annize.project.materializer._translate_from_clone`(*real_nodes_for_clones*, *node_association*, *errors*)

`annize.project.materializer._node_clone_link`(*original*, *clone*)

Subpackages

annize.project.materializer.behaviors package

Behaviors.

See [Behavior](#).

```
class annize.project.materializer.behaviors.Behavior
```

Bases: ABC

A behavior implements what the materializer does for a given node. See subclasses in the submodules.

```
abstractmethod node_context(nodemat)
```

For a node, the materializer will enter the context returned by this function for all behaviors.

The materializer itself does that for the root node. Behaviors itself are responsible for triggering that same process on children.

So, any node gets materialized in the context of all behaviors on all parent nodes. Actual materialization logic happens in this function, in the course of setting up and taking down all these contexts.

Parameters

nodemat ([NodeMaterialization](#)) – The node materialization for the current node.

Return type

ContextManager

```
_abc_impl = <_abc._abc_data object>
```

Submodules

annize.project.materializer.behaviors.argument module

See [ArgumentBehavior](#) and [AssociateArgumentNodeBehavior](#).

```
class annize.project.materializer.behaviors.argument.ArgumentBehavior(callfct, *,
                                                                    feature_loader)
```

Bases: [Behavior](#)

Behavior that handles argument nodes (incl. creation of an object for an object node).

Parameters

feature_loader ([FeatureLoader](#))

```
node_context(nodemat)
```

For a node, the materializer will enter the context returned by this function for all behaviors.

The materializer itself does that for the root node. Behaviors itself are responsible for triggering that same process on children.

So, any node gets materialized in the context of all behaviors on all parent nodes. Actual materialization logic happens in this function, in the course of setting up and taking down all these contexts.

Parameters

nodemat – The node materialization for the current node.

```
_abc_impl = <_abc._abc_data object>
```

```
class annize.project.materializer.behaviors.argument.AssociateArgumentNodeBehavior(association)
```

Bases: [Behavior](#)

Parameters

association (*dict*[[ArgumentNode](#), *list*[*Any*]])

```
node_context(nodemat)
```

For a node, the materializer will enter the context returned by this function for all behaviors.

The materializer itself does that for the root node. Behaviors itself are responsible for triggering that same process on children.

So, any node gets materialized in the context of all behaviors on all parent nodes. Actual materialization logic happens in this function, in the course of setting up and taking down all these contexts.

Parameters

nodemat – The node materialization for the current node.

_abc_impl = <_abc._abc_data object>

annize.project.materializer.behaviors.basket module

See *BasketBehavior*.

class annize.project.materializer.behaviors.basket.**BasketBehavior**

Bases: *Behavior*

Behavior that handles baskets.

node_context(*nodemat*)

For a node, the materializer will enter the context returned by this function for all behaviors.

The materializer itself does that for the root node. Behaviors itself are responsible for triggering that same process on children.

So, any node gets materialized in the context of all behaviors on all parent nodes. Actual materialization logic happens in this function, in the course of setting up and taking down all these contexts.

Parameters

nodemat – The node materialization for the current node.

_abc_impl = <_abc._abc_data object>

annize.project.materializer.behaviors.block module

See *BlockBehavior*.

class annize.project.materializer.behaviors.block.**BlockBehavior**

Bases: *Behavior*

Behavior that handles block.

node_context(*nodemat*)

For a node, the materializer will enter the context returned by this function for all behaviors.

The materializer itself does that for the root node. Behaviors itself are responsible for triggering that same process on children.

So, any node gets materialized in the context of all behaviors on all parent nodes. Actual materialization logic happens in this function, in the course of setting up and taking down all these contexts.

Parameters

nodemat – The node materialization for the current node.

_abc_impl = <_abc._abc_data object>

annize.project.materializer.behaviors.feature_unavailable module

See *FeatureUnavailableBehavior*.

class

`annize.project.materializer.behaviors.feature_unavailable.FeatureUnavailableBehavior`

Bases: [Behavior](#)

Behavior that handles on-feature-unavailable nodes.

`__context_skipnode_featureignorelist(node)`

`__context_catchexceptions(nodemat, featureignorelist)`

`node_context(nodemat)`

For a node, the materializer will enter the context returned by this function for all behaviors.

The materializer itself does that for the root node. Behaviors itself are responsible for triggering that same process on children.

So, any node gets materialized in the context of all behaviors on all parent nodes. Actual materialization logic happens in this function, in the course of setting up and taking down all these contexts.

Parameters

`nodemat` – The node materialization for the current node.

`_abc_impl = <_abc._abc_data object>`

`annize.project.materializer.behaviors.reference` module

See [ReferenceBehavior](#).

class `annize.project.materializer.behaviors.reference.ReferenceBehavior`

Bases: [Behavior](#)

Behavior that handles reference nodes.

`node_context(nodemat)`

For a node, the materializer will enter the context returned by this function for all behaviors.

The materializer itself does that for the root node. Behaviors itself are responsible for triggering that same process on children.

So, any node gets materialized in the context of all behaviors on all parent nodes. Actual materialization logic happens in this function, in the course of setting up and taking down all these contexts.

Parameters

`nodemat` – The node materialization for the current node.

`_abc_impl = <_abc._abc_data object>`

Submodules

`annize.project.materializer.core` module

Inner core parts of the project materializer. Only used internally by the parent package.

class `annize.project.materializer.core.NodeMaterialization(materializer, node, store)`

Bases: `object`

Parameters

- **`materializer`** ([ProjectMaterializer](#))
- **`node`** ([Node](#))

```

        • store (dict)
property node: Node
set_materialized_result(resultlist)
set_problems(problems)

Parameters
    problems (Iterable[Exception])
get_materialized_children_tuples()
get_materialized_children()

Return type
    Iterable[Any]
try_get_materialization_for_node(node)

Parameters
    node (Node)
property has_result
property result
property problems: list[Exception]

class annize.project.materializer.core.ProjectMaterializer(node, *, behaviors)
Bases: object
Parameters
    • node (Node)
    • behaviors (Iterable[annize.project.materializer.behaviors.Behavior])
__materialization_for_node(node, store)

Parameters
    • node (Node)
    • store (dict)
Return type
    NodeMaterialization
__materialize(node, store)

Parameters
    • node (Node)
    • store (dict)
Return type
    None
__materialize_hlp_childobjs(node, store)

Parameters
    • node (Node)

```

- **store** (*dict*)

Return type

list[Tuple[Node, list[Any]]]

__get_erroneous_nodes (*materializationstore, old_erroneous_nodes*)

get_materialized ()

Return type

Tuple[list[Any] | None, dict[Node, list[Exception]]]

exception `annize.project.materializer.core.InternalError`

Bases: `Exception`

exception `annize.project.materializer.core.ChildrenNotMaterializableError` (*node*)

Bases: *InternalError*

Parameters

node (*Node*)

annize.project.materializer.preprocessors module

Some preprocessor functions used by the materializer.

Only used internally by the parent package.

`annize.project.materializer.preprocessors.resolve_appendtonodes` (*topnode*)

Parameters

topnode (*Node*)

Return type

Node

`annize.project.materializer.preprocessors.normalize_blockscopes` (*topnode*)

Parameters

topnode (*Node*)

Return type

Node

Submodules

annize.project.feature_loader module

Feature module loader.

See *FeatureLoader*.

class `annize.project.feature_loader.FeatureLoader`

Bases: `ABC`

Base class for a feature module loader.

abstractmethod `load_feature` (*name*)

Parameters

name (*str*)

Return type*Any* | None**abstractmethod** `get_all_available_feature_names()`**Return type**

list[str]

`_abc_impl = <_abc._abc_data object>`**class** `annize.project.feature_loader.DefaultFeatureLoader`Bases: [*FeatureLoader*](#)

Default feature module loader.

`_FEATURES_NAMESPACE = 'annize.features'``_COMMON_NAMESPACE_POSTFIX = 'common'``load_feature(name)``get_all_available_feature_names()``__find_feature_modules_in_package(package_name)`**Parameters**`package_name` (str)**Return type**

list[str]

`_abc_impl = <_abc._abc_data object>`**annize.project.inspector module**

Project inspector.

See [*Inspector*](#).**class** `annize.project.inspector.Inspector(feature_loader)`

Bases: object

Inspectors are used in order to get various additional metadata about parts of a project, which are useful e.g. for project configuration UIs.

Parameters`feature_loader` ([*FeatureLoader*](#))**class** `ArgumentMatchings(all_matchings)`

Bases: object

Parameters`all_matchings` (list[[*ArgumentMatching*](#)])**class** `ArgumentMatching(arg_name, nodes, allows_multiple_args)`

Bases: object

Parameters

- `arg_name` (str)
- `nodes` (list[[*ArgumentNode*](#)])
- `allows_multiple_args` (bool)

```
    property argname: str
    property allows_multiple_args: bool
    property nodes: list[ArgumentNode]

matching_by_argname(arg_name)
    Parameters
        arg_name (str)
    Return type
        ArgumentMatching|None

all()
    Return type
        list[ArgumentMatching]

class TypeInfo(feature, typename, ctype)
    Bases: object
    Parameters
        • feature (str | None)
        • typename (str)
        • ctype (type)
    property feature: str | None
    property typename: str
    property type: type

match_arguments(node)
    Parameters
        node (Node)
    Return type
        ArgumentMatchings

match_node(node)
    Parameters
        node (Node)
    Return type
        ArgumentMatching | None

get_all_types()
    Return type
        list[TypeInfo]

get_types_for_argument(node, argname)
    Parameters
        • node (Node)
        • argname (str)
    Return type
        list[TypeInfo]
```

get_project_node(*node*)

Parameters

node (*Node*)

Return type

ProjectNode | None

get_node_by_name(*subtree, name*)

Parameters

- **subtree** (*ProjectNode*)
- **name** (*str*)

Return type

Node | None

resolve_reference_node(*node, *, deep=True*)

Parameters

- **node** (*Node*)
- **deep** (*bool*)

Return type

ArgumentNode | None

__get_node_materialtype(*node*)

Parameters

node (*Node*)

Return type

type | None

annize.project.loader module

Loading Annize projects from disk.

See also [load_project\(\)](#).

annize.project.loader.load_project(*project_path*)

Load a project from disk. Return None if the given path does not lead to a location inside an Annize project.

Do not use it directly. See [annize.project.Project.load\(\)](#).

Parameters

project_path (*str* / *Path*) – A path to somewhere inside an Annize project.

Return type

Project | None

annize.project.loader.find_project_annize_config_root_file(*project_path*)

Return the main configuration file for an Annize project given by a path (the path may point to somewhere inside the project; not only inside the Annize configuration directory), or None if the given path does not lead to a location inside an Annize project.

Parameters

project_path (*str* / *Path*) – A path into the Annize project.

Return type

Path | None

`annize.project.loader.project_root_directory(annize_config_rootpath)`

Parameters

`annize_config_rootpath` (*str* | *Path*)

Return type

Path

annize.ui package

`annize.ui.app(app_name, **kwargs)`

Parameters

`app_name` (*str*)

Subpackages

annize.ui.apps package

Subpackages

annize.ui.apps.runner package

Subpackages

annize.ui.apps.runner.models package

Submodules

annize.ui.apps.runner.models.main module

annize.ui.apps.runner.models.task_chooser module

annize.ui.apps.runner.models.task_execution module

annize.ui.apps.runner.models.user_feedback module

annize.ui.apps.runner.views package

Submodules

annize.ui.apps.runner.views.main module

annize.ui.apps.runner.views.task_chooser module

annize.ui.apps.runner.views.task_execution module

annize.ui.apps.runner.views.user_feedback module

annize.ui.apps.studio package

Subpackages

annize.ui.apps.studio.models package

Submodules

`annize.ui.apps.studio.models.add_child` module

`annize.ui.apps.studio.models.main` module

`annize.ui.apps.studio.models.main_tab_panel` module

`annize.ui.apps.studio.models.object_editor` module

`annize.ui.apps.studio.models.problems_list` module

`annize.ui.apps.studio.models.project_config` module

`annize.ui.apps.studio.views` package

Submodules

`annize.ui.apps.studio.views.add_child` module

`annize.ui.apps.studio.views.main` module

`annize.ui.apps.studio.views.main_tab_panel` module

`annize.ui.apps.studio.views.object_editor` module

`annize.ui.apps.studio.views.problems_list` module

`annize.ui.apps.studio.views.project_config` module

`annize.user_feedback` package

class `annize.user_feedback.UserFeedbackController`

Bases: `ABC`

abstractmethod `message_dialog(message, answers, config_key)`

Parameters

- `message` (*str*)
- `answers` (*list[str]*)
- `config_key` (*str | None*)

Return type

int

abstractmethod `input_dialog(question, suggested_answer, config_key)`

Parameters

- `question` (*str*)
- `suggested_answer` (*str*)
- `config_key` (*str | None*)

Return type

str | None

abstractmethod `choice_dialog(question, choices, config_key)`

Parameters

- **question** (*str*)
- **choices** (*list[str]*)
- **config_key** (*str | None*)

Return type
int | None

_abc_impl = *<_abc._abc_data object>*

class *annize.user_feedback.NullUserFeedbackController*

Bases: *object*

message_dialog(*)

input_dialog(*)

choice_dialog(*)

exception *annize.user_feedback.UnsatisfiableUserFeedbackAttemptError*

Bases: *RuntimeError*

annize.user_feedback._controllers_tuples_for_context(context)

Parameters

context (*RunContext*)

Return type

list[Tuple[int, UserFeedbackController]]

annize.user_feedback._controllers_for_context(context)

Parameters

context (*RunContext*)

Return type

list[UserFeedbackController]

annize.user_feedback._add_controller_to_context(, controller, context, priority_index=0)*

Parameters

• **controller** (*UserFeedbackController*)

• **context** (*RunContext*)

• **priority_index** (*int*)

Return type

None

*annize.user_feedback.message_dialog(message, answers, *, config_key=None)*

Parameters

• **message** (*TrStr | str*)

• **answers** (*Iterable[TrStr | str]*)

• **config_key** (*str | None*)

Return type

int

`annize.user_feedback.input_dialog(message, *, suggested_answer, config_key=None)`

Parameters

- `message` (`TrStr` / `str`)
- `suggested_answer` (`TrStr` / `str`)
- `config_key` (`str` / `None`)

Return type

`str` | `None`

`annize.user_feedback.choice_dialog(message, choices, *, config_key=None)`

Parameters

- `message` (`TrStr` / `str`)
- `choices` (`Iterable[TrStr / str]`)
- `config_key` (`str` / `None`)

Return type

`int` | `None`

Submodules

`annize.user_feedback.static module`

`class annize.user_feedback.static.StaticUserFeedbackController(answers)`

Bases: `UserFeedbackController`

Parameters

`answers` (`dict[str, Any]`)

`add_answer(config_key, value)`

Parameters

- `config_key` (`str`)
- `value` (`Any`)

Return type

`None`

`__get_answer(config_key)`

Parameters

`config_key` (`str`)

Return type

`Any`

`message_dialog(message, answers, config_key)`

`input_dialog(question, suggested_answer, config_key)`

`choice_dialog(question, choices, config_key)`

`_abc_impl = <_abc._abc_data object>`

6.1.2 Submodules

6.1.3 annize.annize_cli module

The Annize CLI.

`annize.annize_cli.main()`

`annize.annize_cli.parser(*, only_documentation=True)`

Parameters

`only_documentation` (*bool*)

Return type

ArgumentParser

`class annize.annize_cli.Commands(project, with_answers_from_json_file, with_answers_from_json_string, with_answer, **_)`

Bases: `object`

Parameters

- `project` (*str*)
- `with_answers_from_json_file` (*Iterable[str]*)
- `with_answers_from_json_string` (*Iterable[str]*)
- `with_answer` (*Iterable[Tuple[str, str]]*)

`__initial_cwd = '/home/pino/projects/annize'`

`classmethod __answers_from_json_files(destination, with_answers_from_json_files)`

Parameters

- `destination` (*dict*)
- `with_answers_from_json_files` (*Iterable[str]*)

`classmethod __answers_from_json_strings(destination, with_answers_from_json_strings)`

Parameters

- `destination` (*dict*)
- `with_answers_from_json_strings` (*Iterable[str]*)

`classmethod __answers_from_single_answers(destination, with_answers)`

Parameters

- `destination` (*dict*)
- `with_answers` (*Iterable[Tuple[str, str]]*)

`do(task_name, **_)`

Parameters

`task_name` (*str*)

`studio(**_)`

PYTHON MODULE INDEX

a

annize, 13
annize.annize_cli, 128
annize.asset, 13
annize.asset.data, 13
annize.asset.project_info, 13
annize.data, 13
annize.data.color, 13
annize.data.container, 14
annize.data.unique, 14
annize.data.version, 15
annize.features, 18
annize.features.authors, 72
annize.features.base, 72
annize.features.changelog, 18
annize.features.changelog.common, 18
annize.features.dependencies, 19
annize.features.dependencies.common, 19
annize.features.dependencies.python, 20
annize.features.distributables, 21
annize.features.distributables.common, 21
annize.features.distributables.debian, 23
annize.features.distributables.flatpak, 34
annize.features.distributables.python_wheel, 39
annize.features.distributables.store, 21
annize.features.distributables.store.pypi, 21
annize.features.distributables.tar, 42
annize.features.documentation, 43
annize.features.documentation.common, 53
annize.features.documentation.sphinx, 43
annize.features.documentation.sphinx.common, 46
annize.features.documentation.sphinx.cpp, 50
annize.features.documentation.sphinx.doxygen_compat, 50
annize.features.documentation.sphinx.javascript, 52
annize.features.documentation.sphinx.output, 43
annize.features.documentation.sphinx.output.common, 43
annize.features.documentation.sphinx.output.html, 44
annize.features.documentation.sphinx.output.pdf, 45
annize.features.documentation.sphinx.output.plaintext, 45
annize.features.documentation.sphinx.python, 52
annize.features.documentation.sphinx.rst, 53
annize.features.files, 55
annize.features.files.common, 57
annize.features.files.transfer, 55
annize.features.files.transfer.common, 55
annize.features.files.transfer.ssh, 56
annize.features.homepage, 60
annize.features.homepage.common, 63
annize.features.homepage.sections, 60
annize.features.homepage.sections.about, 60
annize.features.homepage.sections.changelog, 61
annize.features.homepage.sections.documentation, 61
annize.features.homepage.sections.download, 61
annize.features.homepage.sections.gallery, 62
annize.features.homepage.sections.imprint, 62
annize.features.homepage.sections.license, 63
annize.features.i18n, 66
annize.features.i18n.common, 66
annize.features.i18n.gettext, 67
annize.features.injections, 68
annize.features.injections.common, 68
annize.features.injections.python, 69
annize.features.licensing, 75
annize.features.media_galleries, 76
annize.features.task, 77
annize.features.testing, 69
annize.features.testing.common, 69
annize.features.testing.pylint, 69
annize.features.testing.pytest, 70
annize.features.version, 77
annize.features.version_control, 70

- annize.features.version_control.common, 70
- annize.features.version_control.git, 71
- annize.flow, 78
- annize.flow.run_context, 78
- annize.flow.runner, 83
- annize.fs, 85
- annize.fs.ext, 90
- annize.i18n, 91
- annize.object, 99
- annize.object.controller, 99
- annize.object.parameter_info, 100
- annize.project, 102
- annize.project.feature_loader, 120
- annize.project.file_formats, 111
- annize.project.file_formats.xml, 112
- annize.project.inspector, 121
- annize.project.loader, 123
- annize.project.materializer, 115
- annize.project.materializer.behaviors, 115
- annize.project.materializer.behaviors.argument, 116
- annize.project.materializer.behaviors.basket, 117
- annize.project.materializer.behaviors.block, 117
- annize.project.materializer.behaviors.feature_unavailable, 117
- annize.project.materializer.behaviors.reference, 118
- annize.project.materializer.core, 118
- annize.project.materializer.preprocessors, 120
- annize.ui, 124
- annize.ui.apps, 124
- annize.user_feedback, 125
- annize.user_feedback.static, 127

Symbols

_ABOUT_NAME (annize.features.documentation.sphinx.common.ReadmeDocument attribute), 50
 _COMMON_NAMESPACE_POSTFIX (annize.project.feature_loader.DefaultFeatureLoader attribute), 121
 _Context__nodeshort() (annize.project.file_formats.xml._XmlParser._Context static method), 113
 _CreateObjectHelper (class in annize.object.controller), 99
 _CreateObjectHelper.ParameterConfig (class in annize.object.controller), 99
 _CreateObjectHelper__convert_kwargs_from_string() (annize.object.controller._CreateObjectHelper static method), 99
 _CreateObjectHelper__determine_matching_keywords_for_arg() (annize.object.controller._CreateObjectHelper static method), 99
 _CreateObjectHelper__fill_empty_lists() (annize.object.controller._CreateObjectHelper static method), 99
 _CreateObjectHelper__fill_unspecified_optionals() (annize.object.controller._CreateObjectHelper static method), 99
 _CreateObjectHelper__get_parameter_config() (annize.object.controller._CreateObjectHelper static method), 99
 _CreateObjectHelper__put_item_into_kwargs() (annize.object.controller._CreateObjectHelper static method), 100
 _CreateObjectHelper__shift_args_to_kwargs() (annize.object.controller._CreateObjectHelper static method), 100
 _CultureFence (class in annize.i18n), 97
 _FEATURES_NAMESPACE (annize.project.feature_loader.DefaultFeatureLoader attribute), 121
 _FixedTrStr (class in annize.i18n), 95
 _IS_TOPLEVEL_OBJECT__METADATA_KEY (annize.flow.run_context.RunContext attribute), 78
 _ProjectDefinedTranslationProvider (class in annize.features.i18n.common), 66
 _ProjectDefinedTranslationProvider__translations_for_string() (annize.features.i18n.common._ProjectDefinedTranslationProvider method), 66
 _S_CHANGE (annize.features.changelog.common.ByVersionControlSystemControl attribute), 19
 _S_LABEL (annize.features.changelog.common.ByVersionControlSystemControl attribute), 19
 _TContent (annize.fs.ext.DynamicFile attribute), 90
 _TStaticContent (annize.fs.ext.DynamicFile attribute), 90
 _TransferHelper__transfer_piece() (annize.fs.Path._TransferHelper static method), 88
 _XmlParser (class in annize.project.file_formats.xml), 112
 _XmlParser._Context (class in annize.project.file_formats.xml), 112
 _XmlParser._TagParts (class in annize.project.file_formats.xml), 113
 _XmlParser__interpret_attribute_string() (annize.project.file_formats.xml._XmlParser class method), 113
 _XmlParser__parse_attrib() (annize.project.file_formats.xml._XmlParser class method), 113
 _XmlParser__parse_child() (annize.project.file_formats.xml._XmlParser method), 113
 _XmlParser__parse_children() (annize.project.file_formats.xml._XmlParser method), 113
 _XmlParser__parse_tag() (annize.project.file_formats.xml._XmlParser method), 114
 __answers_from_json_files() (annize.annize_cli.Commands class method), 128
 __answers_from_json_strings() (annize.annize_cli.Commands class method), 128

<code>__answers_from_single_answers()</code> (annize.annize_cli.Commands class method), 128	<code>__generate_set_culture()</code> (annize.features.documentation.sphinx.common.Document method), 47
<code>__best_system_locale()</code> (annize.i18n.Culture method), 96	<code>__generate_set_misc()</code> (annize.features.documentation.sphinx.common.Document method), 47
<code>__call_git()</code> (annize.features.version_control.git.VersionControlSystem method), 71	<code>__generate_set_version_and_release()</code> (annize.features.documentation.sphinx.common.Document method), 47
<code>__changed_helpers()</code> (annize.project.Node method), 104	<code>__get_answer()</code> (annize.user_feedback.static.StaticUserFeedbackControl method), 127
<code>__changed_handler()</code> (annize.project.ProjectNode method), 106	<code>__get_erroneous_nodes()</code> (annize.project.materializer.core.ProjectMaterializer method), 120
<code>__cleanup()</code> (annize.fs.ext.FreshTempDirectory method), 90	<code>__get_inner_generateinfo()</code> (annize.features.documentation.sphinx.common.CompositeDocument method), 47
<code>__compacted_changelist()</code> (annize.project.ProjectNode method), 107	<code>__get_node_materialtype()</code> (annize.project.inspector.Inspector method), 122
<code>__context_catchexceptions()</code> (annize.project.materializer.behaviors.feature_unavailable.FeatureUnavailableBehavior method), 118	<code>__get_refgeninfo()</code> (annize.features.documentation.sphinx.common.ApiReferenceDocument method), 48
<code>__context_skipnode_featureignorelist()</code> (annize.project.materializer.behaviors.feature_unavailable.FeatureUnavailableBehavior method), 118	<code>__get_variant()</code> (annize.features.documentation.sphinx.common.RstDocument method), 49
<code>__current_system_locale_setup()</code> (annize.i18n.Culture method), 96	<code>__info()</code> (annize.features.documentation.sphinx.doxygen_compat.DoxygenSupplement method), 52
<code>__description()</code> (annize.project.Node method), 106	<code>__init_varargs_cwd()</code> (annize.annize_cli.Commands attribute), 128
<code>__do_run()</code> (annize.flow.runner.Runner method), 84	<code>__jsfiles()</code> (annize.features.documentation.sphinx.javascript.JavaScriptDocument method), 52
<code>__does_exclude()</code> (annize.features.files.common.Exclude method), 58	<code>__long_str()</code> (annize.data.unique.UniqueId method), 15
<code>__effversion()</code> (annize.features.version_control.common.BuildVariable method), 71	<code>__materialization_for_node()</code> (annize.project.materializer.core.ProjectMaterializer method), 119
<code>__files_from_package_store()</code> (annize.features.distributables.common.Group method), 22	<code>__materialize()</code> (annize.project.materializer.core.ProjectMaterializer method), 119
<code>__find_feature_modules_in_package()</code> (annize.project.feature_loader.DefaultFeatureLoader method), 121	<code>__object_metadata_dict()</code> (annize.flow.run_context.RunContext method), 81
<code>__generate_geninfo_to_confpy()</code> (annize.features.documentation.sphinx.common.Document method), 47	<code>__object_raw_name()</code> (annize.flow.run_context.RunContext method), 81
<code>__generate_packagelist()</code> (annize.features.homepage.sections.download.Section method), 61	<code>__package_store_name()</code> (annize.features.distributables.common.Group method), 22
<code>__generate_pre_post_proc()</code> (annize.features.homepage.common.Homepage method), 65	<code>__patch_property_types_in_docstrings()</code> (annize.features.documentation.sphinx.python.Python3ApiReferenceDocument method), 52
<code>__generate_prepare_annizeicons()</code> (annize.features.documentation.sphinx.common.Document method), 47	<code>__prepare_generate()</code> (annize.features.documentation.sphinx.doxygen_compat.DoxygenSupplement method), 52
<code>__generate_prepare_shortsnippets()</code> (annize.features.documentation.sphinx.common.Document method), 47	
<code>__generate_section()</code> (annize.features.homepage.common.Homepage method), 65	

method), 52

__put_object() (annize.flow.run_context.RunContext method), 81

__scanjsfile() (annize.features.documentation.sphinx.javaApiReferenceDocumentation.sphinx.common.ReadmeDocument attribute), 52

__set_env__var() (annize.i18n.Culture method), 96

__set_success_state() (annize.flow.runner.Runner method), 85

__set_system_locale_setup() (annize.i18n.Culture method), 96

__set_tasks() (annize.flow.runner.Runner method), 84

__shorten() (annize.project.ScalarValueNode method), 109

__store_files_to_package_store() (annize.features.distributables.common.Group method), 22

__transfer_filter_for_exclude() (annize.features.files.common.Directory method), 60

__uniqueid_counter (annize.data.unique.UniqueId attribute), 14

__uniqueid_lock (annize.data.unique.UniqueId attribute), 14

_abc_impl (annize.data.version.AbstractVersionPatternSegment attribute), 15

_abc_impl (annize.data.version.ConcatenatedVersionPatternSegment attribute), 17

_abc_impl (annize.data.version.NumericVersionPatternSegment attribute), 16

_abc_impl (annize.data.version.OptionalVersionPatternSegment attribute), 17

_abc_impl (annize.data.version.SeparatorVersionPatternSegment attribute), 16

_abc_impl (annize.features.distributables.common.PackageStore attribute), 22

_abc_impl (annize.features.documentation.common.Document attribute), 54

_abc_impl (annize.features.documentation.common.HtmlOutput attribute), 55

_abc_impl (annize.features.documentation.common.Output attribute), 54

_abc_impl (annize.features.documentation.common.PdfOutput attribute), 55

_abc_impl (annize.features.documentation.common.PlaintextOutput attribute), 55

_abc_impl (annize.features.documentation.sphinx.common.ApiReference attribute), 50

_abc_impl (annize.features.documentation.sphinx.common.ApiReferenceDocumentation attribute), 48

_abc_impl (annize.features.documentation.sphinx.common.ApiReferenceDocumentation attribute), 48

_abc_impl (annize.features.documentation.sphinx.common.ApiReferenceDocumentation attribute), 49

_abc_impl (annize.features.documentation.sphinx.common.ApiReferenceDocumentation attribute), 49

_abc_impl (annize.features.documentation.sphinx.common.Document attribute), 47

_abc_impl (annize.features.documentation.sphinx.common.Document attribute), 47

_abc_impl (annize.features.documentation.sphinx.common.ReadmeDocument attribute), 50

_abc_impl (annize.features.documentation.sphinx.common.RstDocument attribute), 50

_abc_impl (annize.features.documentation.sphinx.cpp.CppApiReferenceLa attribute), 50

_abc_impl (annize.features.documentation.sphinx.doxygen_compat.Doxygen attribute), 52

_abc_impl (annize.features.documentation.sphinx.javascript.JavaScriptAp attribute), 52

_abc_impl (annize.features.documentation.sphinx.output.common.OutputC attribute), 44

_abc_impl (annize.features.documentation.sphinx.output.html.HtmlOutput attribute), 45

_abc_impl (annize.features.documentation.sphinx.output.html.HtmlOutput attribute), 45

_abc_impl (annize.features.documentation.sphinx.output.pdf.PdfOutputGe attribute), 45

_abc_impl (annize.features.documentation.sphinx.output.plaintext.Plaintex attribute), 46

_abc_impl (annize.features.documentation.sphinx.python.Python3ApiRefer attribute), 53

_abc_impl (annize.features.files.transfer.common.Endpoint attribute), 55

_abc_impl (annize.features.files.transfer.common.FsEndpoint attribute), 56

_abc_impl (annize.features.files.transfer.ssh.Endpoint attribute), 57

_abc_impl (annize.features.homepage.common.HomepageSection attribute), 64

_abc_impl (annize.features.homepage.sections.about.Section attribute), 61

_abc_impl (annize.features.homepage.sections.changelog.Section attribute), 61

_abc_impl (annize.features.homepage.sections.documentation.Section attribute), 61

_abc_impl (annize.features.homepage.sections.download.Section attribute), 62

_abc_impl (annize.features.homepage.sections.gallery.Section attribute), 62

_abc_impl (annize.features.homepage.sections.imprint.Section attribute), 62

_abc_impl (annize.features.homepage.sections.license.Section attribute), 63

_abc_impl (annize.features.i18n.common.String attribute), 67

_abc_impl (annize.features.i18n.common._ProjectDefinedTranslationProv attribute), 66

_abc_impl (annize.features.i18n.common._ProjectDefinedTranslationProv attribute), 66

_abc_impl (annize.features.injections.common.FileSystemContentInjection attribute), 68

_abc_impl (annize.features.injections.common.Injection attribute), 68

attribute), 68
 _abc_impl (annize.features.injections.python.ProjectInfoInjection attribute), 69
 _abc_impl (annize.features.testing.common.Test attribute), 69
 _abc_impl (annize.features.testing.common.TestGroup attribute), 69
 _abc_impl (annize.features.testing.pylint.Test attribute), 69
 _abc_impl (annize.features.testing.pytest.Test attribute), 70
 _abc_impl (annize.features.version_control.common.VersionControlSystem attribute), 126
 _abc_impl (annize.features.version_control.git.VersionControlSystem attribute), 127
 _abc_impl (annize.flow.runner.Runner attribute), 85
 _abc_impl (annize.i18n.GettextTranslationProvider attribute), 92
 _abc_impl (annize.i18n.ProvidedTrStr attribute), 94
 _abc_impl (annize.i18n.TrStr attribute), 94
 _abc_impl (annize.i18n.TranslationProvider attribute), 92
 _abc_impl (annize.i18n._FixedTrStr attribute), 95
 _abc_impl (annize.project.ArgumentNode attribute), 108
 _abc_impl (annize.project.BlockNode attribute), 110
 _abc_impl (annize.project.BlockNodeWithScope attribute), 110
 _abc_impl (annize.project.FileNode attribute), 108
 _abc_impl (annize.project.Node attribute), 106
 _abc_impl (annize.project.ObjectNode attribute), 109
 _abc_impl (annize.project.OnFeatureUnavailableNode attribute), 111
 _abc_impl (annize.project.ProjectNode attribute), 107
 _abc_impl (annize.project.ReferenceNode attribute), 109
 _abc_impl (annize.project.ScalarValueNode attribute), 109
 _abc_impl (annize.project.feature_loader.DefaultFeatureLoader attribute), 121
 _abc_impl (annize.project.feature_loader.FeatureLoader attribute), 121
 _abc_impl (annize.project.file_formats.FileFormat attribute), 112
 _abc_impl (annize.project.file_formats.FileFormat.Marshaler attribute), 111
 _abc_impl (annize.project.file_formats.xml.Marshaler attribute), 114
 _abc_impl (annize.project.file_formats.xml.XmlFileFormat attribute), 112
 _abc_impl (annize.project.materializer.behaviors.Behavior attribute), 116
 _abc_impl (annize.project.materializer.behaviors.argument.AssociateArgument attribute), 117
 _abc_impl (annize.project.materializer.behaviors.basket.BasketBehavior attribute), 117
 _abc_impl (annize.project.materializer.behaviors.block.BlockBehavior attribute), 117
 _abc_impl (annize.project.materializer.behaviors.feature_unavailable.FeatureUnavailable attribute), 118
 _abc_impl (annize.project.materializer.behaviors.reference.ReferenceBehavior attribute), 118
 _abc_impl (annize.user_feedback.UserFeedbackController attribute), 126
 _abc_impl (annize.user_feedback.static.StaticUserFeedbackController attribute), 127
 _add_controller_to_context() (in module annize.user_feedback), 126
 _allowed_child_types() (annize.project.BlockNode class method), 110
 _allowed_child_types() (annize.project.FileNode class method), 108
 _allowed_child_types() (annize.project.Node class method), 105
 _allowed_child_types() (annize.project.ObjectNode class method), 108
 _allowed_child_types() (annize.project.ProjectNode class method), 107
 _allowed_child_types() (annize.project.ReferenceNode class method), 109
 _allowed_child_types() (annize.project.ScalarValueNode class method), 109
 _annize_user_interaction_culture() (in module annize.i18n), 98
 _append_section() (annize.features.homepage.common.Homepage method), 65
 _changed__child_added() (annize.project.Node method), 104
 _changed__child_removed() (annize.project.Node method), 104
 _changed__property_changed() (annize.project.Node method), 105
 _controllers_for_context() (in module annize.user_feedback), 126
 _controllers_tuples_for_context() (in module annize.user_feedback), 126
 _debian_category() (in module annize.features.distributables.debian), 23
 _debian_section() (in module annize.features.distributables.debian), 27
 _description_for_mediafile() (annize.features.media_galleries.Gallery method), 77

- `_dispatch()` (*annize.flow.runner.Runner* method), 84
- `_generate_sources()` (*annize.features.distributables.flatpak.FlatpakImage* class method), 38
- `_generate_sources()` (*annize.features.distributables.debian.Package* class method), 48
- `_generate_sources()` (*annize.features.distributables.debian.Package* class method), 49
- `_generate_sources()` (*annize.features.distributables.debian.Package* class method), 47
- `_generate_sources()` (*annize.features.distributables.debian.Package* class method), 47
- `_generate_sources()` (*annize.features.distributables.debian.Package* class method), 49
- `_get_data()` (in module *annize.features.base*), 74
- `_get_type_info()` (in module *annize.object.parameter_info*), 101
- `_last_resort_culture` (in module *annize.i18n*), 97
- `_license()` (in module *annize.features.licensing*), 75
- `_materialize_hlp_childobjs()` (*annize.project.materializer.core.ProjectMaterializer* method), 119
- `_mkpackage()` (*annize.features.distributables.debian.Package* class method), 32
- `_mkpackage()` (*annize.features.distributables.flatpak.FlatpakImage* class method), 39
- `_mkpackage()` (*annize.features.distributables.python_wheel.Package* class method), 41
- `_mkpackage_applysource()` (*annize.features.distributables.flatpak.FlatpakImage* class method), 38
- `_mkpackage_bdist_wheel()` (*annize.features.distributables.python_wheel.Package* class method), 42
- `_mkpackage_correctbuildsourcepermissions()` (*annize.features.distributables.debian.Package* class method), 34
- `_mkpackage_determinesize()` (*annize.features.distributables.debian.Package* class method), 33
- `_mkpackage_dpkg()` (*annize.features.distributables.debian.Package* class method), 34
- `_mkpackage_flatpak_build_export()` (*annize.features.distributables.flatpak.FlatpakImage* class method), 39
- `_mkpackage_flatpak_build_finish()` (*annize.features.distributables.flatpak.FlatpakImage* class method), 38
- `_mkpackage_flatpak_build_init()` (*annize.features.distributables.flatpak.FlatpakImage* class method), 38
- `_mkpackage_mkchangelog()` (*annize.features.distributables.debian.Package* class method), 33
- `_mkpackage_mkcopyright()` (*annize.features.distributables.debian.Package* class method), 33
- `_mkpackage_mkdebianconffilesfile()` (*annize.features.distributables.debian.Package* class method), 34
- `_mkpackage_mkdebiancontrolfile()` (*annize.features.distributables.debian.Package* class method), 34
- `_mkpackage_mkexeclinks()` (*annize.features.distributables.debian.Package* class method), 33
- `_mkpackage_mkexeclinks()` (*annize.features.distributables.python_wheel.Package* class method), 42
- `_mkpackage_mkmanifestin()` (*annize.features.distributables.python_wheel.Package* class method), 42
- `_mkpackage_mkmenuentries()` (*annize.features.distributables.debian.Package* class method), 33
- `_mkpackage_mkprepostcmds()` (*annize.features.distributables.debian.Package* class method), 33
- `_mkpackage_mkservices()` (*annize.features.distributables.debian.Package* class method), 33
- `_mkpackage_mksetuppyconf()` (*annize.features.distributables.python_wheel.Package* class method), 42
- `_mkpackage_prepareinfos()` (*annize.features.distributables.debian.Package* class method), 32
- `_mkpackage_prepareinfos()` (*annize.features.distributables.flatpak.FlatpakImage* class method), 38
- `_mkpackage_prepareinfos()` (*annize.features.distributables.python_wheel.Package* class method), 41
- `_mkpackage_setuppyconf_classifiers()` (*annize.features.distributables.python_wheel.Package* class method), 42
- `_mkpackage_setuppyconf_install_requires()` (*annize.features.distributables.python_wheel.Package* class method), 42
- `_mkpackage_setuppyconf_prepare()` (*annize.features.distributables.python_wheel.Package* class method), 41

_mkpackage_share() (annize.features.distributables.flatpak.FlatpakImage _str_helper() (annize.project.ReferenceNode method),
 class method), 38
 _name_anon() (annize.data.version.AbstractVersionPatternSegment_helper() (annize.project.ScalarValueNode
 method), 15
 _name_anon() (annize.data.version.ConcatenatedVersionPatternSegment_helper() (annize.data.version.AbstractVersionPatternSegment
 method), 17
 _name_anon() (annize.data.version.NumericVersionPatternSegment_helper() (annize.data.version.NumericVersionPatternSegment
 method), 16
 _name_anon() (annize.data.version.OptionalVersionPatternSegment_helper() (annize.features.documentation.sphinx.common.Document.Ge
 method), 17
 _node_clone_link() (in module annize.project.materializer), 115
 _path() (annize.features.distributables.debian.Package
 method), 31
 _path() (annize.features.distributables.flatpak.FlatpakImage
 method), 37
 _path() (annize.features.distributables.flatpak.FlatpakrefFile
 method), 36
 _path() (annize.features.distributables.flatpak.GpgFile
 method), 36
 _path() (annize.features.distributables.python_wheel.Package
 method), 40
 _path() (annize.features.distributables.tar.Package
 method), 43
 _path() (annize.features.documentation.common.GeneratedDocumentHelper
 method), 55
 _path() (annize.features.files.common.Directory
 method), 60
 _path() (annize.features.files.common.FsEntry method),
 57
 _path() (annize.features.files.common.MachineRootDirectory
 method), 60
 _path() (annize.features.files.common.ProjectDirectory
 method), 60
 _path() (annize.features.homepage.common.GeneratedHomepage
 method), 66
 _path() (annize.fs.Path method), 86
 _path() (annize.fs.ext.DynamicFile method), 90
 _set_entry_path() (annize.features.documentation.common.DocumentGenerateResult
 method), 53
 _str_helper() (annize.project.ArgumentNode method),
 108
 _str_helper() (annize.project.BlockNode method), 110
 _str_helper() (annize.project.BlockNodeWithScope
 method), 110
 _str_helper() (annize.project.FileNode method), 108
 _str_helper() (annize.project.Node method), 106
 _str_helper() (annize.project.ObjectNode method),
 108
 _str_helper() (annize.project.OnFeatureUnavailableNode
 method), 110
 _str_helper() (annize.project.ProjectNode method),
 107
 _str_helper() (annize.project.ReferenceNode method),
 109
 _str_helper() (annize.project.ScalarValueNode
 method), 109
 _str_helper() (annize.data.version.AbstractVersionPatternSegment
 method), 15
 _str_helper() (annize.data.version.NumericVersionPatternSegment
 method), 16
 _str_helper() (annize.features.documentation.sphinx.common.Document.Ge
 method), 47
 _translate_from_clone() (in module annize.project.materializer), 115
 A
 AboutProjectDocument (class in annize.features.documentation.sphinx.common),
 50
 AbstractVersionPatternSegment (class in annize.data.version), 15
 access_filesystem() (annize.features.files.transfer.common.Endpoint
 method), 55
 access_filesystem() (annize.features.files.transfer.common.FsEndpoint
 method), 56
 access_filesystem() (annize.features.files.transfer.ssh.Endpoint
 method), 56
 add_answer() (annize.user_feedback.static.StaticUserFeedbackController
 method), 127
 add_change() (annize.project.file_formats.FileFormat.Marshaler
 method), 111
 add_change() (annize.project.file_formats.xml.Marshaler
 method), 114
 add_change_handler() (annize.project.Node method),
 104
 add_element() (annize.project.file_formats.xml.Marshaler
 method), 114
 add_element_attr() (annize.project.file_formats.xml.Marshaler
 method), 114
 add_element_tree() (annize.project.file_formats.xml.Marshaler
 method), 114
 add_object() (annize.flow.run_context.RunContext
 method), 80
 add_object() (in module annize.flow.run_context), 82
 add_translation_provider() (in module annize.i18n), 92
 add_translations() (annize.features.i18n.common._ProjectDefinedTranslationProvider
 method), 66

additional_info() (*annize.features.licensing.License*
 method), 75
 AdministrationUtilitiesSection (*in module an-*
 nize.features.distributables.debian), 27
 AFLv3 (*in module annize.features.licensing*), 75
 AGPLv3 (*in module annize.features.licensing*), 75
 all() (*annize.project.inspector.Inspector.ArgumentMatchings*
 method), 122
 allows_multiple_args (*an-*
 nize.object.parameter_info.ListParameterInfo
 property), 101
 allows_multiple_args (*an-*
 nize.object.parameter_info.ParameterInfo
 property), 101
 allows_multiple_args (*an-*
 nize.project.inspector.Inspector.ArgumentMatchings
 property), 122
 annize
 module, 13
 annize.annize_cli
 module, 128
 annize.asset
 module, 13
 annize.asset.data
 module, 13
 annize.asset.project_info
 module, 13
 annize.data
 module, 13
 annize.data.color
 module, 13
 annize.data.container
 module, 14
 annize.data.unique
 module, 14
 annize.data.version
 module, 15
 annize.features
 module, 18
 annize.features.authors
 module, 72
 annize.features.base
 module, 72
 annize.features.changelog
 module, 18
 annize.features.changelog.common
 module, 18
 annize.features.dependencies
 module, 19
 annize.features.dependencies.common
 module, 19
 annize.features.dependencies.python
 module, 20
 annize.features.distributables
 module, 21
 annize.features.distributables.common
 module, 21
 annize.features.distributables.debian
 module, 23
 annize.features.distributables.flatpak
 module, 34
 annize.features.distributables.python_wheel
 module, 39
 annize.features.distributables.store
 module, 21
 annize.features.distributables.store.pypi
 module, 21
 annize.features.distributables.tar
 module, 42
 annize.features.documentation
 module, 43
 annize.features.documentation.common
 module, 53
 annize.features.documentation.sphinx
 module, 43
 annize.features.documentation.sphinx.common
 module, 46
 annize.features.documentation.sphinx.cpp
 module, 50
 annize.features.documentation.sphinx.doxygen_compat
 module, 50
 annize.features.documentation.sphinx.javascript
 module, 52
 annize.features.documentation.sphinx.output
 module, 43
 annize.features.documentation.sphinx.output.common
 module, 43
 annize.features.documentation.sphinx.output.html
 module, 44
 annize.features.documentation.sphinx.output.pdf
 module, 45
 annize.features.documentation.sphinx.output.plaintext
 module, 45
 annize.features.documentation.sphinx.python
 module, 52
 annize.features.documentation.sphinx.rst
 module, 53
 annize.features.files
 module, 55
 annize.features.files.common
 module, 57
 annize.features.files.transfer
 module, 55
 annize.features.files.transfer.common
 module, 55
 annize.features.files.transfer.ssh
 module, 56
 annize.features.homepage

module, 60	module, 78
annize.features.homepage.common	annize.flow.run_context
module, 63	module, 78
annize.features.homepage.sections	annize.flow.runner
module, 60	module, 83
annize.features.homepage.sections.about	annize.fs
module, 60	module, 85
annize.features.homepage.sections.changelog	annize.fs.ext
module, 61	module, 90
annize.features.homepage.sections.documentation	annize.i18n
module, 61	module, 91
annize.features.homepage.sections.download	annize.object
module, 61	module, 99
annize.features.homepage.sections.gallery	annize.object.controller
module, 62	module, 99
annize.features.homepage.sections.imprint	annize.object.parameter_info
module, 62	module, 100
annize.features.homepage.sections.license	annize.project
module, 63	module, 102
annize.features.i18n	annize.project.feature_loader
module, 66	module, 120
annize.features.i18n.common	annize.project.file_formats
module, 66	module, 111
annize.features.i18n.gettext	annize.project.file_formats.xml
module, 67	module, 112
annize.features.injections	annize.project.inspector
module, 68	module, 121
annize.features.injections.common	annize.project.loader
module, 68	module, 123
annize.features.injections.python	annize.project.materializer
module, 69	module, 115
annize.features.licensing	annize.project.materializer.behaviors
module, 75	module, 115
annize.features.media_galleries	annize.project.materializer.behaviors.argument
module, 76	module, 116
annize.features.task	annize.project.materializer.behaviors.basket
module, 77	module, 117
annize.features.testing	annize.project.materializer.behaviors.block
module, 69	module, 117
annize.features.testing.common	annize.project.materializer.behaviors.feature_unavailable
module, 69	module, 117
annize.features.testing.pylint	annize.project.materializer.behaviors.reference
module, 69	module, 118
annize.features.testing.pytest	annize.project.materializer.core
module, 70	module, 118
annize.features.version	annize.project.materializer.preprocessors
module, 77	module, 120
annize.features.version_control	annize.ui
module, 70	module, 124
annize.features.version_control.common	annize.ui.apps
module, 70	module, 124
annize.features.version_control.git	annize.user_feedback
module, 71	module, 125
annize.flow	annize.user_feedback.static

- module, 127
- annize_config_rootpath (in module *annize.project.Project* property), 102
- ANNIZE_CONFIG_ROOTPATH_NAME (in module *annize.flow.run_context.RunContext* attribute), 78
- annize_user_interaction_culture (in module *annize.i18n*), 98
- Apache2 (in module *annize.features.licensing*), 75
- ApiReferenceDocument (class in module *annize.features.documentation.sphinx.common*), 48
- ApiReferenceLanguage (class in module *annize.features.documentation.sphinx.common*), 47
- ApiReferenceLanguage.ApiReferenceGenerateInfo (class in module *annize.features.documentation.sphinx.common*), 48
- app() (in module *annize.ui*), 124
- append_child() (in module *annize.project.Node* method), 105
- append_rst() (in module *annize.features.homepage.common.Homepage* method), 64
- append_to (in module *annize.project.ArgumentNode* property), 108
- ApplicationsAccessibilityCategory (in module *annize.features.distributables.debian*), 24
- ApplicationsAmateurradioCategory (in module *annize.features.distributables.debian*), 24
- ApplicationsDatamanagementCategory (in module *annize.features.distributables.debian*), 24
- ApplicationsEditorsCategory (in module *annize.features.distributables.debian*), 24
- ApplicationsEducationCategory (in module *annize.features.distributables.debian*), 24
- ApplicationsEmulatorsCategory (in module *annize.features.distributables.debian*), 24
- ApplicationsFilemanagementCategory (in module *annize.features.distributables.debian*), 24
- ApplicationsGraphicsCategory (in module *annize.features.distributables.debian*), 24
- ApplicationsMobiledevicesCategory (in module *annize.features.distributables.debian*), 24
- ApplicationsNetworkCategory (in module *annize.features.distributables.debian*), 24
- ApplicationsNetworkCommunicationCategory (in module *annize.features.distributables.debian*), 24
- ApplicationsNetworkFiletransferCategory (in module *annize.features.distributables.debian*), 24
- ApplicationsNetworkMonitoringCategory (in module *annize.features.distributables.debian*), 24
- ApplicationsNetworkWebbrowsingCategory (in module *annize.features.distributables.debian*), 24
- ApplicationsNetworkWebnewsCategory (in module *annize.features.distributables.debian*), 24
- ApplicationsOfficeCategory (in module *annize.features.distributables.debian*), 24
- ApplicationsProgrammingCategory (in module *annize.features.distributables.debian*), 24
- ApplicationsProjectmanagementCategory (in module *annize.features.distributables.debian*), 24
- ApplicationsScienceAstronomyCategory (in module *annize.features.distributables.debian*), 25
- ApplicationsScienceBiologyCategory (in module *annize.features.distributables.debian*), 25
- ApplicationsScienceCategory (in module *annize.features.distributables.debian*), 24
- ApplicationsScienceChemistryCategory (in module *annize.features.distributables.debian*), 25
- ApplicationsScienceDataanalysisCategory (in module *annize.features.distributables.debian*), 25
- ApplicationsScienceElectronicsCategory (in module *annize.features.distributables.debian*), 25
- ApplicationsScienceEngineeringCategory (in module *annize.features.distributables.debian*), 25
- ApplicationsScienceGeoscienceCategory (in module *annize.features.distributables.debian*), 25
- ApplicationsScienceMathematicsCategory (in module *annize.features.distributables.debian*), 25
- ApplicationsScienceMedicineCategory (in module *annize.features.distributables.debian*), 25
- ApplicationsSciencePhysicsCategory (in module *annize.features.distributables.debian*), 25
- ApplicationsScienceSocialCategory (in module *annize.features.distributables.debian*), 25
- ApplicationsShellsCategory (in module *annize.features.distributables.debian*), 25
- ApplicationsSoundsCategory (in module *annize.features.distributables.debian*), 25
- ApplicationsSystemAdministrationCategory (in module *annize.features.distributables.debian*), 25
- ApplicationsSystemCategory (in module *annize.features.distributables.debian*), 25
- ApplicationsSystemHardwareCategory (in module *annize.features.distributables.debian*), 25
- ApplicationsSystemLanguageenvironmentCategory (in module *annize.features.distributables.debian*), 25
- ApplicationsSystemMonitoringCategory (in module *annize.features.distributables.debian*), 25

- ApplicationsSystemPackagemanagementCategory (in module *annize.features.distributables.debian*), 25
- ApplicationsSystemSecurityCategory (in module *annize.features.distributables.debian*), 26
- ApplicationsTerminalEmulatorsCategory (in module *annize.features.distributables.debian*), 26
- ApplicationsTextCategory (in module *annize.features.distributables.debian*), 26
- ApplicationsTvandradiocategory (in module *annize.features.distributables.debian*), 26
- ApplicationsVideoCategory (in module *annize.features.distributables.debian*), 26
- ApplicationsViewersCategory (in module *annize.features.distributables.debian*), 26
- ApplicationsWebdevelopmentCategory (in module *annize.features.distributables.debian*), 26
- architecture (*annize.features.distributables.debian.Package._BuildInfo* attribute), 32
- arg_name (*annize.project.ArgumentNode* property), 108
- argname (*annize.project.inspector.Inspector.ArgumentMatchings.ArgumentMatching* property), 121
- ArgparseCommandLineInterfaceDocument (class in *annize.features.documentation.sphinx.common*), 48
- ArgumentBehavior (class in *annize.project.materializer.behaviors.argument*), 116
- ArgumentNode (class in *annize.project*), 108
- Artistic1 (in module *annize.features.licensing*), 75
- Artwork (class in *annize.features.dependencies.common*), 20
- AssociateArgumentNodeBehavior (class in *annize.project.materializer.behaviors.argument*), 116
- attach_media_file() (*annize.features.homepage.common.HomepageSection.Content* method), 64
- attr_name (*annize.project.file_formats.xml.Marshaler.XmlParser* attribute), 114
- ATTRIBUTE_COMMAND_END (*annize.project.file_formats.xml._XmlParser* attribute), 113
- ATTRIBUTE_COMMAND_START (*annize.project.file_formats.xml._XmlParser* attribute), 113
- author (*annize.features.distributables.debian.Package._BuildInfo* attribute), 32
- author (*annize.features.distributables.python_wheel.Package._BuildInfo* attribute), 41
- Author (class in *annize.features.authors*), 72
- authors (*annize.features.documentation.sphinx.common.Document* property), 47
- authorstring (*annize.features.distributables.debian.Package._BuildInfo* attribute), 32
- available_cultures() (*annize.features.documentation.common.Document* method), 54
- available_cultures() (*annize.features.documentation.sphinx.common.AboutProjectDocument* method), 50
- available_cultures() (*annize.features.documentation.sphinx.common.ApiReferenceDocument* method), 48
- available_cultures() (*annize.features.documentation.sphinx.common.ArgparseCommandLineInterfaceDocument* method), 49
- available_cultures() (*annize.features.documentation.sphinx.common.ReadmeDocument* method), 50
- available_cultures() (*annize.features.documentation.sphinx.common.RstDocument* method), 49
- ## B
- background_image (*annize.features.documentation.sphinx.output.html.HtmlOutputSpec* property), 45
- BadStructureError, 111
- Basket (class in *annize.data.container*), 14
- Basket (class in *annize.features.base*), 73
- BasketBehavior (class in *annize.project.materializer.behaviors.basket*), 117
- Behavior (class in *annize.project.materializer.behaviors*), 115
- BLOCK (*annize.project.BlockNodeWithScope.Scope* attribute), 110
- BlockBehavior (class in *annize.project.materializer.behaviors.block*), 117
- BlockNode (class in *annize.project*), 109
- BlockNodeWithScope (class in *annize.project*), 110
- BlockNodeWithScope.Scope (class in *annize.project*), 110
- blue (*annize.data.color.Color* property), 13
- brand_color() (in module *annize.features.base*), 74
- BrandColor (class in *annize.features.base*), 73
- BSD2clause (in module *annize.features.licensing*), 75
- BSD3clause (in module *annize.features.licensing*), 75
- BuildVersion (class in *annize.features.version_control.common*), 71
- ByVersionControlSystemCommitMessagesChangelog (class in *annize.features.changelog.common*), 18

C

category (*annize.features.distributables.debian.MenuEntry* property), 23
category (*annize.features.distributables.flatpak.MenuEntry* property), 34
Category (class in *annize.features.distributables.debian*), 23
Cc0v1 (in module *annize.features.licensing*), 75
CcBy3 (in module *annize.features.licensing*), 75
CcByNc3 (in module *annize.features.licensing*), 75
CcByNcNd3 (in module *annize.features.licensing*), 75
CcByNcSa3 (in module *annize.features.licensing*), 76
CcByNd3 (in module *annize.features.licensing*), 76
CcBySa3 (in module *annize.features.licensing*), 76
Changelog (class in *annize.features.changelog.common*), 18
child_node (*annize.project.Node.ChildrenListChangeEvent* property), 103
child_position (*annize.project.Node.ChildrenListChangeEvent* property), 103
children (*annize.project.Node* property), 105
children() (*annize.fs.Path* method), 86
ChildrenNotMaterializableError, 120
choice_dialog() (*annize.user_feedback.NullUserFeedbackController* method), 126
choice_dialog() (*annize.user_feedback.static.StaticUserFeedbackController* method), 127
choice_dialog() (*annize.user_feedback.UserFeedbackController* method), 125
choice_dialog() (in module *annize.user_feedback*), 127
clone() (*annize.project.Node* method), 105
Color (class in *annize.data.color*), 13
command (*annize.features.distributables.debian.MenuEntry* property), 23
command (*annize.features.distributables.flatpak.FlatpakImageBuildInfo* attribute), 38
command (*annize.features.distributables.flatpak.MenuEntry* property), 34
Commands (class in *annize.annize_cli*), 128
comment (*annize.features.dependencies.common.Dependency* property), 19
CommonVersionPattern (class in *annize.features.version*), 78
CommunicationProgramsSection (in module *annize.features.distributables.debian*), 27
CompositeDocument (class in *annize.features.documentation.sphinx.common*), 47
ConcatenatedVersionPatternSegment (class in *annize.data.version*), 17
confdir (*annize.features.documentation.sphinx.common.Document.GeneratedAttribute* attribute), 46
config_files (*annize.features.distributables.debian.Package._BuildInfo* attribute), 32
confglines (*annize.features.documentation.sphinx.common.Document.GeneratedAttribute* attribute), 46
configvalues (*annize.features.documentation.sphinx.common.Document.GeneratedAttribute* attribute), 46
Connection (class in *annize.features.distributables.store.pypi*), 21
construct_from_string() (*annize.object.parameter_info.ParameterInfo* method), 101
content (*annize.features.injections.common.FileSystemContentInjection* property), 68
content() (in module *annize.fs*), 89
copy_to() (*annize.fs.Path* method), 87
CppApiReferenceLanguage (class in *annize.features.documentation.sphinx.cpp*), 50
create_new() (*annize.project.Project* static method), 102
create_object() (*annize.object.controller._CreateObjectHelper* static method), 99
create_object() (in module *annize.object.controller*), 100
ctime() (*annize.fs.Path* method), 86
culture (*annize.features.documentation.sphinx.common.Document.GeneratedAttribute* attribute), 46
culture (*annize.features.documentation.sphinx.common.RstDocumentVariable* property), 49
culture (*annize.features.homepage.common.HomepageSection._GeneratedAttribute* attribute), 63
Culture (class in *annize.features.i18n.common*), 67
Culture (class in *annize.i18n*), 95
Culture._TSystemLocaleSetup (class in *annize.i18n*), 97
culture_by_spec() (in module *annize.i18n*), 97
culture_list() (*annize.i18n.Culture* method), 96
culture_names (*annize.features.documentation.common.DocumentGeneratedAttribute* property), 54
cultures (*annize.features.homepage.common.HomepageSection* property), 65
current() (in module *annize.flow.run_context*), 81
current_culture() (in module *annize.i18n*), 98
custom_arg (*annize.features.homepage.common.HomepageSection._GeneratedAttribute* attribute), 63
custom_arg (*annize.features.homepage.common.HomepageSection._PrePageSectionAttribute* attribute), 64

D

Data (class in *annize.features.base*), 72

DatabasesSection (in module *annize.features.distributables.debian*), 27
DateTime (class in *annize.features.base*), 73
debian_name (*annize.features.distributables.debian.Category* property), 23
DebianInstallerUdebPackagesSection (in module *annize.features.distributables.debian*), 27
DebugPackagesSection (in module *annize.features.distributables.debian*), 27
default_changelog() (in module *annize.features.changelog.common*), 19
default_version_control_system() (in module *annize.features.version_control.common*), 71
default_version_pattern() (in module *annize.features.version*), 77
DefaultFeatureLoader (class in *annize.project.feature_loader*), 121
dependencies (*annize.features.distributables.python_wheel.PackageAttribute* property), 39
dependencies (*annize.features.distributables.python_wheel.PackageAttribute* attribute), 41
dependencies_to_rst_text() (in module *annize.features.dependencies.common*), 20
Dependency (class in *annize.features.dependencies.common*), 19
description (*annize.features.distributables.common.Group* property), 22
description (*annize.features.distributables.debian.PackageAttribute* attribute), 32
description (*annize.features.distributables.flatpak.Group* property), 36
description (*annize.features.distributables.python_wheel.PackageAttribute* attribute), 41
description (*annize.features.media_galleries.Gallery.Item* property), 77
description() (*annize.project.Node* method), 106
destination (*annize.fs.ext.Mount* property), 91
destination_is_parent (*annize.features.files.common.DirectoryPart* property), 59
destination_path (*annize.features.files.common.DirectoryPart* property), 59
DevelopmentSection (in module *annize.features.distributables.debian*), 27
Directory (class in *annize.features.files.common*), 59
DirectoryPart (class in *annize.features.files.common*), 58
do (*annize.project.OnFeatureUnavailableNode* property), 110
do() (*annize.annize_cli.Commands* method), 128
Document (class in *annize.features.documentation.common*), 54
Document (class in *annize.features.documentation.sphinx.common*), 46
Document.GenerateInfo (class in *annize.features.documentation.sphinx.common*), 46
document_root_directory (*annize.features.homepage.common.HomepageSection._PrePostProc* attribute), 64
document_root_url (*annize.features.homepage.common.HomepageSection._GenerateInfo* attribute), 63
document_root_url (*annize.features.homepage.common.HomepageSection._PrePostProc* attribute), 64
document_variant_directory (*annize.features.homepage.common.HomepageSection._GenerateInfo* attribute), 63
document_variant_url (*annize.features.homepage.common.HomepageSection._GenerateInfo* attribute), 63
documentation_source (*annize.features.distributables.debian.Package._BuildInfo* attribute), 32
DocumentationSection (in module *annize.features.distributables.debian*), 27
DocumentGenerateAllCulturesResult (class in *annize.features.documentation.common*), 53
DocumentGenerateResult (class in *annize.features.documentation.common*), 53
does_exclude() (*annize.features.files.common.Exclude* method), 58
does_exclude() (*annize.features.files.common.ExcludeAllBut* method), 58
does_exclude() (*annize.features.version_control.git.ExcludeByGitIgnore* method), 72
DoxygenSupportedApiReferenceLanguage (class in *annize.features.documentation.sphinx.doxygen_compat*), 50
dynamic_file() (in module *annize.fs*), 89
DynamicFile (class in *annize.fs.ext*), 90

E

EditorsSection (in module *annize.features.distributables.debian*), 27
EducationSection (in module *annize.features.distributables.debian*), 27
ElectronicsSection (in module *annize.features.distributables.debian*), 27
element (*annize.project.file_formats.xml.Marshaler.XmlDocumentLocation* attribute), 114
email (*annize.features.authors.Author* property), 72
EmbeddedSoftwareSection (in module *annize.features.distributables.debian*), 27

Endpoint (class in annize.features.files.transfer.common), 55

Endpoint (class in annize.features.files.transfer.ssh), 56

english_lang_name (annize.i18n.Culture property), 96

entries (annize.features.changelog.common.ByVersionControlSystemProperty), 19

entries (annize.features.changelog.common.Changelog property), 18

Entry (class in annize.features.changelog.common), 18

entry_path (annize.features.documentation.common.DocumentGenerateResult property), 53

entry_path (annize.features.documentation.sphinx.common.DocumentGenerateResult attribute), 46

entry_path_for_language() (annize.features.documentation.common.DocumentGenerateResult method), 54

environment (annize.features.distributables.flatpak.FlatpakImage._BuildInfo attribute), 38

EnvironmentVariable (class in annize.features.distributables.flatpak), 35

erroneous_nodes() (annize.project.materializer.MaterializationResult method), 115

errors_for_node() (annize.project.materializer.MaterializationResult method), 115

escape_attribute_string() (annize.project.file_formats.xml._XmlParser class method), 113

Exclude (class in annize.features.files.common), 57

ExcludeAllBut (class in annize.features.files.common), 58

ExcludeByGitIgnores (class in annize.features.version_control.git), 71

excludes (annize.features.files.common.Directory property), 60

excludes (annize.features.files.common.DirectoryPart property), 59

exec() (annize.features.files.transfer.ssh.Endpoint method), 57

executable_links (annize.features.distributables.debian.Package._BuildInfo attribute), 32

executable_links (annize.features.distributables.python_wheel.Package._BuildInfo attribute), 41

ExecutableLink (class in annize.features.distributables.debian), 23

ExecutableLink (class in annize.features.distributables.python_wheel), 39

explicit_only (annize.object.controller._CreateObjectHelper.Parameters attribute), 99

explicit_only() (in module annize.object), 99

extra_dependencies (annize.features.distributables.python_wheel.Package._BuildInfo attribute), 41

extra_name (annize.features.distributables.python_wheel.ExtraDependencies), 39

ExtraDependencies (class in annize.features.distributables.python_wheel), 39

F

failback_cultures() (annize.i18n.Culture property), 96

feature (annize.project.inspector.Inspector.TypeInfo attribute), 109

feature (annize.project.ObjectNode property), 108

feature (annize.project.OnFeatureUnavailableNode property), 110

FeatureLoader (class in annize.project.feature_loader), 120

FeatureUnavailableBehavior (class in annize.project.materializer.behaviors.feature_unavailable), 117

FeatureUnavailableError, 111

file (annize.features.documentation.common.DocumentGenerateResult property), 53

file (annize.features.media_galleries.Gallery.Item property), 77

FILE (annize.project.BlockNodeWithScope.Scope attribute), 110

File (class in annize.features.files.common), 57

file_size() (annize.fs.Path method), 86

FileFormat (class in annize.project.file_formats), 111

FileFormat.Marshaler (class in annize.project.file_formats), 111

filehash() (in module annize.features.homepage.sections.download), 62

FileNode (class in annize.project), 107

files() (annize.features.distributables.common.Group method), 22

files() (annize.features.distributables.flatpak.Group method), 36

Filesystem (class in annize.features.distributables.flatpak), 35

FilesystemContent (class in annize.fs), 85

FilesystemContentInjection (class in annize.features.injections.common), 68

filesystems (annize.features.distributables.flatpak.FlatpakImage._BuildInfo attribute), 38

find_output_generator_for_outputspec() (in module annize.object), 99

[nize.features.documentation.sphinx.output.common GamesBoardCategory](#) (in module [nize.features.distributables.debian](#)), 26
[43](#)
[find_project_annize_config_root_file\(\)](#) (in module [annize.project.loader](#)), 123
[FirstOf](#) (class in [annize.features.base](#)), 74
[FlatpakImage](#) (class in [nize.features.distributables.flatpak](#)), 36
[FlatpakImage._BuildInfo](#) (class in [nize.features.distributables.flatpak](#)), 37
[FlatpakrefFile](#) (class in [nize.features.distributables.flatpak](#)), 36
[FontsSection](#) (in module [nize.features.distributables.debian](#)), 27
[format\(\)](#) ([annize.i18n.TrStr](#) method), 94
[formatname\(\)](#) ([annize.features.documentation.sphinx.output.common GamesToolsCategory](#) method), 43
[formatname\(\)](#) ([annize.features.documentation.sphinx.output.common GamesToysCategory](#) method), 45
[formatname\(\)](#) ([annize.features.documentation.sphinx.output.common GamesPuzzlesCategory](#) method), 45
[formatname\(\)](#) ([annize.features.documentation.sphinx.output.common GamesSection](#) method), 45
[freedesktop_name](#) ([nize.features.distributables.debian.Category](#) property), 23
[fresh_temp_directory\(\)](#) (in module [annize.fs](#)), 89
[FreshTempDirectory](#) (class in [annize.fs.ext](#)), 90
[friendly_filesize\(\)](#) (in module [nize.features.homepage.sections.download](#)), 62
[friendly_join_string_list\(\)](#) (in module [nize.i18n](#)), 98
[friendly_name_suggestion](#) ([nize.features.distributables.flatpak.Repository](#) property), 35
[FromRequirementsFile](#) (class in [nize.features.dependencies.python](#)), 20
[FsEndpoint](#) (class in [nize.features.files.transfer.common](#)), 55
[FsEntry](#) (class in [annize.features.files.common](#)), 57
[full_name](#) ([annize.features.authors.Author](#) property), 72
[full_name](#) ([annize.i18n.Culture](#) property), 96

G

[Gallery](#) (class in [annize.features.media_galleries](#)), 76
[Gallery.Item](#) (class in [nize.features.media_galleries](#)), 76
[GamesActionCategory](#) (in module [nize.features.distributables.debian](#)), 26
[GamesAdventureCategory](#) (in module [nize.features.distributables.debian](#)), 26
[GamesBlocksCategory](#) (in module [nize.features.distributables.debian](#)), 26
[GamesBoardCategory](#) (in module [nize.features.distributables.debian](#)), 26
[GamesCardCategory](#) (in module [nize.features.distributables.debian](#)), 26
[GamesPuzzlesCategory](#) (in module [nize.features.distributables.debian](#)), 26
[GamesSection](#) (in module [nize.features.distributables.debian](#)), 27
[GamesSimulationCategory](#) (in module [nize.features.distributables.debian](#)), 26
[GamesStrategyCategory](#) (in module [nize.features.distributables.debian](#)), 26
[GamesToolsCategory](#) (in module [nize.features.distributables.debian](#)), 26
[GamesToysCategory](#) (in module [nize.features.distributables.debian](#)), 26
[generate_all_cultures\(\)](#) ([annize.features.documentation.common.Document](#) method), 54
[generate_all_cultures\(\)](#) ([annize.features.documentation.sphinx.common.Document](#) method), 47
[generate_all_cultures\(\)](#) ([annize.features.documentation.sphinx.common.Homepage](#) method), 65
[generate_all_cultures\(\)](#) ([nize.features.documentation.common.Document](#) method), 54
[generate_all_cultures\(\)](#) ([nize.features.documentation.sphinx.common.Document](#) method), 47
[generate_content\(\)](#) ([nize.features.homepage.common.HomepageSection](#) method), 64
[generate_content\(\)](#) ([nize.features.homepage.sections.about.Section](#) method), 60
[generate_content\(\)](#) ([nize.features.homepage.sections.changelog.Section](#) method), 61
[generate_content\(\)](#) ([nize.features.homepage.sections.documentation.Section](#) method), 61
[generate_content\(\)](#) ([nize.features.homepage.sections.download.Section](#) method), 62
[generate_content\(\)](#) ([nize.features.homepage.sections.gallery.Section](#) method), 62
[generate_content\(\)](#) ([nize.features.homepage.sections.imprint.Section](#) method), 62
[generate_content\(\)](#) ([nize.features.homepage.sections.license.Section](#) method), 63
[generate_sources\(\)](#) ([nize.features.documentation.sphinx.common.ApiReferenceLanguage](#) method), 63

method), 48
generate_sources() (annize.features.documentation.sphinx.doxygen_compiler.NodeGenerator method), 52
generate_sources() (annize.features.documentation.sphinx.javascript.JavascriptProjectReferenceNode method), 52
generate_sources() (annize.features.documentation.sphinx.python.PythonProjectReferenceNode method), 53
GeneratedDocument (class in annize.features.documentation.common), 55
GeneratedHomepage (class in annize.features.homepage.common), 66
GenerateMOS (class in annize.features.i18n.gettext), 67
get_all_available_feature_names() (annize.project.feature_loader.DefaultFeatureLoader method), 121
get_all_available_feature_names() (annize.project.feature_loader.FeatureLoader method), 121
get_all_types() (annize.project.inspector.Inspector method), 122
get_changes() (annize.project.ProjectNode method), 106
get_commit_message() (annize.features.version_control.common.VersionControlSystem method), 70
get_commit_message() (annize.features.version_control.git.VersionControlSystem method), 71
get_commit_time() (annize.features.version_control.common.VersionControlSystem method), 70
get_commit_time() (annize.features.version_control.git.VersionControlSystem method), 71
get_current_revision() (annize.features.version_control.common.VersionControlSystem method), 70
get_current_revision() (annize.features.version_control.git.VersionControlSystem method), 71
get_format() (in module annize.project.file_formats), 112
get_from_iso_639_1_lang_code() (annize.i18n.Culture static method), 95
get_materialized() (annize.project.materializer.core.ProjectMaterializer method), 120
get_materialized_children() (annize.project.materializer.core.NodeMaterializationGroup method), 119
get_materialized_children_tuples() (annize.project.materializer.core.NodeMaterializationGroup method), 119
nize.project.materializer.core.NodeMaterializationGroup (class), 119
get_node_by_name() (annize.project.inspector.Inspector method), 123
get_project_reference() (annize.project.inspector.Inspector method), 123
get_revision_list() (annize.features.version_control.common.VersionControlSystem method), 70
get_revision_list() (annize.features.version_control.git.VersionControlSystem method), 71
get_revision_number() (annize.features.version_control.common.VersionControlSystem method), 70
get_revision_number() (annize.features.version_control.git.VersionControlSystem method), 71
get_selected_task() (annize.flow.runner.Runner method), 84
get_success_state() (annize.flow.runner.Runner method), 84
get_tasks() (annize.flow.runner.Runner method), 84
get_types_for_argument() (annize.project.inspector.Inspector method), 122
get_variant() (annize.i18n._FixedTrStr method), 95
get_variant() (annize.i18n._ProvidedTrStr method), 94
get_variant() (annize.i18n.TrStr method), 93
GettextTranslationProvider (class in annize.i18n), 92
GettextTranslationProvider._NoneTranslations (class in annize.i18n), 92
GnomeSection (in module annize.features.distributables.debian), 28
GnuLinux (class in annize.features.dependencies.common), 20
GnuRSection (in module annize.features.distributables.debian), 28
GnuStepSection (in module annize.features.distributables.debian), 28
GpgFile (class in annize.features.distributables.flatpak), 36
GPLv3 (in module annize.features.licensing), 76
GraphicsSection (in module annize.features.distributables.debian), 28
green (annize.data.color.Color property), 13
Group (class in annize.features.distributables.common), 22
Group (class in annize.features.distributables.flatpak), 35

H

- HamRadioSection (in module *annize.features.distributables.debian*), 28
- has_result (*annize.project.materializer.core.NodeMaterialization* property), 119
- has_shell_access (in module *annize.features.files.transfer.ssh.Endpoint* property), 56
- HaskellSection (in module *annize.features.distributables.debian*), 28
- head (*annize.features.homepage.common.HomepageSection* property), 64
- heading (*annize.features.documentation.sphinx.common.ApiReferenceLanguage.ApiReferenceGenerateInfo* property), 48
- heading() (*annize.features.documentation.sphinx.rst.RstGenerator* static method), 53
- HelpCategory (in module *annize.features.distributables.debian*), 26
- homepage (*annize.features.distributables.debian.Package_BuildInfo* attribute), 32
- homepage (*annize.features.distributables.python_wheel.Package_BuildInfo* attribute), 41
- Homepage (class in *annize.features.homepage.common*), 64
- homepage_url (*annize.features.base.Data* property), 73
- homepage_url() (in module *annize.features.base*), 74
- HomepageSection (class in *annize.features.homepage.common*), 63
- HomepageSection._GenerateInfo (class in *annize.features.homepage.common*), 63
- HomepageSection._PrePostProcGenerateInfo (class in *annize.features.homepage.common*), 63
- HomepageSection.Content (class in *annize.features.homepage.common*), 64
- host (*annize.features.files.transfer.ssh.Endpoint* property), 56
- html_color_spec (*annize.data.color.Color* property), 14
- HtmlOutputGenerator (class in *annize.features.documentation.sphinx.output.html*), 45
- HtmlOutputSpec (class in *annize.features.documentation.common*), 54
- HtmlOutputSpec (class in *annize.features.documentation.sphinx.output.html*), 44
- hue (*annize.data.color.Color* property), 13
- I
- icon (*annize.features.dependencies.common.Dependency* property), 19
- icon (*annize.features.distributables.debian.MenuEntry* property), 23
- icon (*annize.features.distributables.flatpak.MenuEntry* property), 35
- identity_file (*annize.features.files.transfer.ssh.Endpoint* property), 56
- IMAGE (*annize.features.media_galleries.MediaType* attribute), 76
- importance (*annize.features.dependencies.common.Dependency* property), 19
- importance (*annize.features.dependencies.common.Kind* property), 19
- imprint (*annize.features.base.Data* property), 73
- imprint() (in module *annize.features.base*), 74
- in_file() (*annize.project.file_formats.xml.XmlParser._Context* method), 112
- in_node() (*annize.project.file_formats.xml.XmlParser._Context* method), 112
- Included (class in *annize.features.dependencies.common*), 20
- Inject (class in *annize.features.injections.common*), 68
- inject() (*annize.features.injections.common.FilesystemContentInjection* method), 68
- inject() (*annize.features.injections.common.Injection* method), 68
- inject() (*annize.features.injections.python.ProjectInfoInjection* method), 69
- Injection (class in *annize.features.injections.common*), 68
- inner_type_info (in *annize.object.parameter_info.ListParameterInfo* property), 101
- inner_type_info (in *annize.object.parameter_info.ParameterInfo* property), 101
- input_dialog() (*annize.user_feedback.NullUserFeedbackController* method), 126
- input_dialog() (*annize.user_feedback.static.StaticUserFeedbackControl* method), 127
- input_dialog() (*annize.user_feedback.UserFeedbackController* method), 125
- input_dialog() (in module *annize.user_feedback*), 126
- insert_child() (*annize.project.Node* method), 105
- insert_child() (*annize.project.ProjectNode* method), 106
- Inspector (class in *annize.project.inspector*), 121
- Inspector.ArgumentMatchings (class in *annize.project.inspector*), 121
- Inspector.ArgumentMatchings.ArgumentMatching (class in *annize.project.inspector*), 121
- Inspector.TypeInfo (class in *annize.project.inspector*), 122
- InternalError, 120
- InterpretersSection (in module *annize.features.distributables.debian*), 28
- IntrospectionSection (in module *annize.features.distributables.debian*), 28

`nize.features.distributables.debian)`, 28
`intstruct` (`annize.features.documentation.sphinx.common.DocumentGenerator`), 28
`attribute`, 46
`is_advanced` (`annize.features.task.Task` property), 77
`is_compatible_for()` (`annize.features.documentation.sphinx.output.common.OutputGenerator` (in module `annize.features.distributables.debian`), 28
`class method`), 43
`is_compatible_for()` (`annize.features.documentation.sphinx.output.html.HtmlOutputGenerator` (in module `annize.features.distributables.debian`), 28
`class method`), 45
`is_compatible_for()` (`annize.features.documentation.sphinx.output.pdf.PdfOutputGenerator` (in module `annize.features.distributables.debian`), 28
`class method`), 45
`is_compatible_for()` (`annize.features.documentation.sphinx.output.plaintext.PlaintextOutputGenerator` (in module `annize.features.distributables.debian`), 28
`class method`), 45
`is_constructable_from_string` (`annize.object.parameter_info.ParameterInfo` property), 101
`is_finished()` (`annize.flow.runner.Runner` method), 84
`is_friendly_name()` (`annize.flow.run_context.RunContext` method), 80
`is_friendly_name()` (in module `annize.flow.run_context`), 83
`is_gui` (`annize.features.distributables.debian.MenuEntry` property), 23
`is_gui` (`annize.features.distributables.flatpak.MenuEntry` property), 35
`is_gui` (`annize.features.distributables.python_wheel.ExecutableLink` property), 39
`is_homepage` (`annize.features.documentation.common.HtmlOutputSpec` property), 54
`is_optional` (`annize.object.parameter_info.ParameterInfo` property), 101
`is_toplevel_object()` (`annize.flow.run_context.RunContext` method), 80
`is_toplevel_object()` (in module `annize.flow.run_context`), 83
`iso_639_1_language_code` (`annize.i18n.Culture` property), 96
`Item` (class in `annize.features.changelog.common`), 18
`items` (`annize.features.changelog.common.Entry` property), 18
`items` (`annize.features.media_galleries.Gallery` property), 77
J
`JavaScriptApiReferenceLanguage` (class in `annize.features.documentation.sphinx.javascript`), 52
`JavascriptSection` (in module `annize.features.distributables.debian`), 28
`JavaSection` (in module `annize.features.distributables.debian`), 28
`join_authors()` (in module `annize.features.authors`), 72
K
`KernelSection` (in module `annize.features.distributables.debian`), 28
`Keyword` (class in `annize.features.base`), 73
`keywords` (`annize.features.base.Keywords` property), 73
`keywords` (`annize.features.distributables.python_wheel.Package._BuildInfo` attribute), 41
`Keywords` (class in `annize.features.base`), 73
`kind` (`annize.features.dependencies.common.Dependency` property), 19
`Kind` (class in `annize.features.dependencies.common`), 19
`kitversion` (`annize.features.distributables.flatpak.FlatpakImage._BuildInfo` attribute), 37
L
`label` (`annize.features.dependencies.common.Dependency` property), 19
`label` (`annize.features.dependencies.common.Kind` property), 19
`LANGUAGE` (`annize.i18n.Culture._TSystemLocaleSetup` attribute), 97
`LanguagePacksSection` (in module `annize.features.distributables.debian`), 28
`LC_ALL` (`annize.i18n.Culture._TSystemLocaleSetup` attribute), 97
`License` (class in `annize.features.licensing`), 76
`LicenseSection` (in module `annize.features.distributables.debian`), 28
`LibraryDevelopmentSection` (in module `annize.features.distributables.debian`), 28
`license` (`annize.features.distributables.python_wheel.Package._BuildInfo` attribute), 41
`License` (class in `annize.features.licensing`), 75
`licensename` (`annize.features.distributables.debian.Package._BuildInfo` attribute), 32
`lightness` (`annize.data.color.Color` property), 13
`Line` (class in `annize.features.version`), 77
`link_name` (`annize.features.distributables.python_wheel.ExecutableLink` property), 39
`LispSection` (in module `annize.features.distributables.debian`), 28
`ListParameterInfo` (class in `annize.object.parameter_info`), 101
`load()` (`annize.project.Project` static method), 102
`load()` (`annize.project.ProjectNode` static method), 107
`load_feature()` (`annize.project.feature_loader.DefaultFeatureLoader` method), 121

`load_feature()` (`annize.project.feature_loader.FeatureLoader` method), 120
`load_project()` (in module `annize.project.loader`), 123
`LocalRepository` (class in `annize.features.distributables.flatpak`), 35
`locationstring()` (`annize.features.files.transfer.ssh.Endpoint` method), 56
`logo_image` (`annize.features.documentation.sphinx.output.html.HtmlOutputSpec` property), 45
`long_description` (`annize.features.base.Data` property), 73
`long_description` (`annize.features.distributables.python_wheel.Package._BuildInfo` attribute), 41
`long_description()` (in module `annize.features.base`), 74
`long_str` (`annize.data.unique.UniqueId` property), 14

M

`MachineRootDirectory` (class in `annize.features.files.common`), 60
`MailSection` (in module `annize.features.distributables.debian`), 28
`main()` (in module `annize.annize_cli`), 128
`main_document` (`annize.features.documentation.sphinx.common.ThemeGenerator` attribute), 46
`mark_object_as_toplevel()` (`annize.flow.run_context.RunContext` method), 80
`marshaler` (`annize.project.file_formats.xml._XmlParser.Context` property), 112
`marshaler` (`annize.project.FileNode` property), 108
`Marshaler` (class in `annize.project.file_formats.xml`), 114
`Marshaler.XmlDocumentLocation` (class in `annize.project.file_formats.xml`), 114
`masterlink` (`annize.features.documentation.sphinx.output.html.HtmlOutputSpec` property), 44
`match_arguments()` (`annize.project.inspector.Inspector` method), 122
`match_node()` (`annize.project.inspector.Inspector` method), 122
`matches_inner_type()` (`annize.object.parameter_info.ParameterInfo` method), 101
`matches_object()` (`annize.object.parameter_info.ParameterInfo` method), 100
`matches_object()` (`annize.object.parameter_info.UnionParameterInfo` method), 101
`matches_type()` (`annize.object.parameter_info.ParameterInfo` method), 100
`matching_by_argname()` (`annize.project.inspector.Inspector.ArgumentMatchings` method), 122
`MaterializationResult` (class in `annize.project.materializer`), 115
`materialize()` (in module `annize.project.materializer`), 115
`MaterializerError`, 111
`MathematicsSection` (in module `annize.features.distributables.debian`), 28
`media_files` (`annize.features.homepage.common.HomepageSection.Content` property), 64
`mediatype` (`annize.features.media_galleries.Gallery.Item` property), 77
`MediaType` (class in `annize.features.media_galleries`), 76
`menu_entries` (`annize.features.distributables.flatpak.FlatpakImage._BuildInfo` attribute), 38
`menuentries` (`annize.features.distributables.debian.Package._BuildInfo` attribute), 32
`MenuEntry` (class in `annize.features.distributables.debian`), 23
`MenuEntry` (class in `annize.features.distributables.flatpak`), 34
`message_dialog()` (`annize.user_feedback.NullUserFeedbackController` method), 126
`message_dialog()` (`annize.user_feedback.static.StaticUserFeedbackController` method), 127
`message_dialog()` (`annize.user_feedback.UserFeedbackController` method), 125
`message_dialog()` (in module `annize.user_feedback`), 126
`MetaPackagesSection` (in module `annize.features.distributables.debian`), 28
`method_name` (`annize.features.distributables.python_wheel.ExecutableLink` property), 39
`MiscellaneousSection` (in module `annize.features.distributables.debian`), 29
`MIT` (in module `annize.features.licensing`), 76
`module`
 `annize`, 13
 `annize.annize_cli`, 128
 `annize.asset`, 13
 `annize.asset.data`, 13
 `annize.asset.project_info`, 13
 `annize.data`, 13
 `annize.data.color`, 13
 `annize.data.container`, 14
 `annize.data.unique`, 14
 `annize.data.version`, 15

annize.features, 18
 annize.features.authors, 72
 annize.features.base, 72
 annize.features.changelog, 18
 annize.features.changelog.common, 18
 annize.features.dependencies, 19
 annize.features.dependencies.common, 19
 annize.features.dependencies.python, 20
 annize.features.distributables, 21
 annize.features.distributables.common, 21
 annize.features.distributables.debian, 23
 annize.features.distributables.flatpak, 34
 annize.features.distributables.python_wheel, 39
 annize.features.distributables.store, 21
 annize.features.distributables.store.pypi, 21
 annize.features.distributables.tar, 42
 annize.features.documentation, 43
 annize.features.documentation.common, 53
 annize.features.documentation.sphinx, 43
 annize.features.documentation.sphinx.common, 46
 annize.features.documentation.sphinx.cpp, 50
 annize.features.documentation.sphinx.doxygen, 50
 annize.features.documentation.sphinx.javascript, 52
 annize.features.documentation.sphinx.output, 43
 annize.features.documentation.sphinx.output.common, 43
 annize.features.documentation.sphinx.output.html, 44
 annize.features.documentation.sphinx.output.pdf, 45
 annize.features.documentation.sphinx.output.plaintext, 45
 annize.features.documentation.sphinx.python, 52
 annize.features.documentation.sphinx.rst, 53
 annize.features.files, 55
 annize.features.files.common, 57
 annize.features.files.transfer, 55
 annize.features.files.transfer.common, 55
 annize.features.files.transfer.ssh, 56
 annize.features.homepage, 60
 annize.features.homepage.common, 63
 annize.features.homepage.sections, 60
 annize.features.homepage.sections.about, 60
 annize.features.homepage.sections.changelog, 61
 annize.features.homepage.sections.documentation, 61
 annize.features.homepage.sections.download, 61
 annize.features.homepage.sections.gallery, 62
 annize.features.homepage.sections.imprint, 62
 annize.features.homepage.sections.license, 63
 annize.features.i18n, 66
 annize.features.i18n.common, 66
 annize.features.i18n.gettext, 67
 annize.features.injections, 68
 annize.features.injections.common, 68
 annize.features.injections.python, 69
 annize.features.licensing, 75
 annize.features.media_galleries, 76
 annize.features.task, 77
 annize.features.testing, 69
 annize.features.testing.common, 69
 annize.features.testing.pylint, 69
 annize.features.testing.pytest, 70
 annize.features.version, 77
 annize.features.version_control, 70
 annize.features.version_control.common, 70
 annize.features.version_control.git, 71
 annize.flow, 78
 annize.flow.run_context, 78
 annize.flow.runner, 83
 annize.fs, 85
 annize.fs.ext, 90
 annize.i18n, 91
 annize.object, 99
 annize.object.controller, 99
 annize.object.parameter_info, 100
 annize.project, 102
 annize.project.feature_loader, 120
 annize.project.file_formats, 111
 annize.project.file_formats.xml, 112
 annize.project.inspector, 121
 annize.project.loader, 123
 annize.project.materializer, 115
 annize.project.materializer.behaviors, 115
 annize.project.materializer.behaviors.argument, 116
 annize.project.materializer.behaviors.basket, 117
 annize.project.materializer.behaviors.block, 117

annize.project.materializer.behaviors.feature_signatures_section (in module *annize.features.distributables.debian*), 29
 117
 annize.project.materializer.behaviors.reference_culture_error, 98
 118
 annize.project.materializer.core, 118
 annize.project.materializer.preprocessors, 120
 annize.ui, 124
 annize.ui.apps, 124
 annize.user_feedback, 125
 annize.user_feedback.static, 127
 module_name (*annize.features.distributables.python_wheel.NodeChildLink* property), 39
 MonoCliSection (in module *annize.features.distributables.debian*), 27
 Mount (class in *annize.fs.ext*), 90
 move_to() (*annize.fs.Path* method), 87
 MPLv11 (in module *annize.features.licensing*), 76
 MPLv2 (in module *annize.features.licensing*), 76
 mtime() (*annize.fs.Path* method), 86
 multilanguage_frame() (in module *annize.features.documentation.sphinx.output.common_node_processor* method), 44
 multilanguage_frame() (in module *annize.features.documentation.sphinx.output.html.HtmlOutputGenerator* method), 45
 MultipleValuesForSingleArgumentError, 100
N
 name (*annize.features.dependencies.python.PythonPackage* property), 20
 name (*annize.features.distributables.debian.ExecutableLink* property), 23
 name (*annize.features.distributables.debian.MenuEntry* property), 23
 name (*annize.features.distributables.debian.Package._BuildInfo* attribute), 32
 name (*annize.features.distributables.debian.Section* property), 27
 name (*annize.features.distributables.flatpak.FlatpakImage._BuildInfo* attribute), 37
 name (*annize.features.distributables.flatpak.MenuEntry* property), 34
 name (*annize.features.distributables.python_wheel.Package._BuildInfo* attribute), 41
 name (*annize.features.licensing.License* property), 75
 name (*annize.object.parameter_info.ParameterInfo* property), 100
 name (*annize.project.ArgumentNode* property), 108
 NetworkSection (in module *annize.features.distributables.debian*), 29
 new_value (*annize.project.Node.PropertyChangedEvent* property), 104
 NodeMaterialization (class in *annize.project.materializer.core*), 119
 NodeMaterialization (class in *annize.project*), 103
 Node.ChangeEvent (class in *annize.project*), 103
 Node.ChildAddedEvent (class in *annize.project*), 103
 Node.ChildRemovedEvent (class in *annize.project*), 103
 NodeChildLink (class in *annize.project*), 103
 Node.PropertyChangedEvent (class in *annize.project*), 104
 node_context() (*annize.project.materializer.behaviors.argument.Argument* method), 116
 node_context() (*annize.project.materializer.behaviors.argument.Association* method), 116
 node_context() (*annize.project.materializer.behaviors.basket.BasketBehavior* method), 117
 node_context() (*annize.project.materializer.behaviors.Behavior* method), 116
 node_context() (*annize.project.materializer.behaviors.block.BlockBehavior* method), 117
 node_context() (*annize.project.materializer.behaviors.feature_unavailable* method), 118
 node_context() (*annize.project.materializer.behaviors.reference.Reference* method), 118
 NodeMaterialization (class in *annize.project.materializer.core*), 118
 nodes (*annize.project.inspector.Inspector.ArgumentMatchings.ArgumentMatchings* property), 122
 normalize_blockscopes() (in module *annize.project.materializer.preprocessors*), 120
 NullUserFeedbackController (class in *annize.user_feedback*), 126
 NumericVersionPatternSegment (class in *annize.data.version*), 15
O
 object_by_name() (in module *annize.flow.run_context.RunContext* method), 78
 object_by_name() (in module *annize.flow.run_context*), 82
 object_metadata() (in module *annize.flow.run_context.RunContext* method), 80
 object_metadata() (in module *annize.flow.run_context*), 83
 object_name() (*annize.flow.run_context.RunContext* method), 79

object_name() (in module *annize.flow.run_context*), 82
 object_names() (*annize.flow.run_context.RunContext* method), 79
 object_names() (in module *annize.flow.run_context*), 82
 ObjectNode (class in *annize.project*), 108
 objects_by_type() (*annize.flow.run_context.RunContext* method), 79
 objects_by_type() (in module *annize.flow.run_context*), 82
 objects_for_node() (*annize.project.materializer.MaterializationResult* method), 115
 OcamlSection (in module *annize.features.distributables.debian*), 29
 old_value (*annize.project.Node.PropertyChangedEvent* property), 104
 OldLibrariesSection (in module *annize.features.distributables.debian*), 29
 on_unresolvable (*annize.project.ReferenceNode* property), 109
 OnFeatureUnavailableNode (class in *annize.project*), 110
 OnFeatureUnavailableNode.Action (class in *annize.project*), 110
 OptionalVersionPatternSegment (class in *annize.data.version*), 16
 OtherOssAndFssSection (in module *annize.features.distributables.debian*), 29
 outdir (*annize.features.documentation.sphinx.common.Documentation* attribute), 46
 OutOfContextError, 81
 output_spec (*annize.features.documentation.sphinx.output.common.OutputSpec* property), 43
 OutputGenerator (class in *annize.features.documentation.sphinx.output.common*), 43
 OutputSpec (class in *annize.features.documentation.common*), 54
P
 Package (class in *annize.features.distributables.debian*), 30
 Package (class in *annize.features.distributables.python_wheel*), 40
 Package (class in *annize.features.distributables.tar*), 42
 package() (*annize.features.distributables.common.PackageStore* method), 21
 Package._BuildInfo (class in *annize.features.distributables.debian*), 31
 Package._BuildInfo (class in *annize.features.distributables.python_wheel*), 40
 package_history() (*annize.features.distributables.common.PackageStore* method), 22
 package_store (*annize.features.distributables.common.Group* property), 22
 PackageStore (class in *annize.features.distributables.common*), 21
 parameter_name (*annize.object.controller._CreateObjectHelper.ParameterInfo* attribute), 99
 ParameterInfo (class in *annize.object.parameter_info*), 100
 parent (*annize.project.Node* property), 105
 parse() (in module *annize.project.file_formats*), 112
 parse_file() (*annize.project.file_formats.FileFormat* class method), 111
 parse_file() (*annize.project.file_formats.xml.XmlParser* method), 113
 parse_file() (*annize.project.file_formats.xml.XmlFileFormat* class method), 112
 parser() (in module *annize.annize_cli*), 128
 ParserError, 111
 partname (*annize.data.version.NumericVersionPatternSegment* property), 15
 parts (*annize.features.files.common.Directory* property), 60
 path (*annize.features.distributables.debian.ExecutableLink* property), 23
 path (*annize.features.version_control.git.VersionControlSystem* property), 71
 path (*annize.features.version_control.fresh_temp_directory* property), 90
 path (*annize.project.FileNode* property), 107
 Path (class in *annize.fs*), 85
 path() (*annize.fs.Filesystem* method), 85
 path() (*annize.fs.Path* method), 86
 Path._TransferHelper (class in *annize.fs*), 88
 Path.TransferFilters (class in *annize.fs*), 88
 Path.TransferFilters.And (class in *annize.fs*), 88
 pattern (*annize.data.version.Version* property), 18
 PdfOutputGenerator (class in *annize.features.documentation.sphinx.output.pdf*), 45
 PdfOutputSpec (class in *annize.features.documentation.common*), 55
 PerlSection (in module *annize.features.distributables.debian*), 29
 PhpSection (in module *annize.features.distributables.debian*), 29
 pkgpath_debian (*annize.features.distributables.debian.Package._BuildInfo* attribute), 32
 pkgpath_documentation (*annize.features.distributables.debian.Package._BuildInfo* attribute), 32
 pkgpath_pixmaps (*annize.features.distributables.debian.Package._BuildInfo* attribute), 32

`nize.features.distributables.debian.Package._BuildInfo` (attribute), 32
`nize.features.homepage.sections.gallery.Section` (method), 62
`pkgpath_setuppy` (attribute), 41
`pkgpath_share` (attribute), 38
`pkgpath_share_applications` (attribute), 38
`pkgpath_share_icons` (attribute), 38
`pkgpath_usrbin` (attribute), 32
`pkgrootpath` (attribute), 32
`pkgrootpath` (attribute), 38
`pkgrootpath` (attribute), 41
`pkgsize` (attribute), 32
`PlaintextOutputGenerator` (class in `annize.features.documentation.sphinx.output.plaintext`), 45
`PlaintextOutputSpec` (class in `annize.features.documentation.common`), 55
`platform` (attribute), 37
`port` (property), 56
`post_process_generate()` (method), 64
`post_process_generate()` (method), 61
`postinst` (attribute), 32
`postproc()` (method), 44
`postproc()` (method), 45
`pre_process_generate()` (method), 64
`pre_process_generate()` (method), 61
`pre_process_generate()` (method), 61
`pre_process_generate()` (method), 61
`pre_process_generate()` (method), 61
`prepare()` (method), 78
`prepare_generate()` (method), 43
`prepare_generate()` (method), 45
`prerm` (attribute), 32
`pretty_project_name` (property), 73
`pretty_project_name()` (method), 74
`problems` (property), 119
`processonly_long_str` (property), 15
`processonly_short_str` (property), 15
`PROJECT` (property), 110
`Project` (class in `annize.project`), 102
`project_authors()` (method), 72
`project_cultures()` (method), 67
`project_directory` (property), 73
`project_directory()` (method), 74
`project_keywords()` (method), 73
`project_licenses()` (method), 76
`project_name` (property), 73
`project_name` (property), 47
`project_name_generator` (class in `annize.features.base`), 74
`project_root_directory()` (method), 124
`project_versions()` (method), 77
`ProjectCultures` (class in `annize.features.i18n.common`), 67
`ProjectDirectory` (class in `annize.features.files.common`), 60
`ProjectInfoInjection` (class in `annize.features.injections.python`), 69
`ProjectMaterializer` (class in `annize.project.materializer.core`), 119
`projectname__original` (property), (an-

nize.features.documentation.sphinx.common.Document (class in *annize.features.documentation.sphinx.common*), 47
ProjectNode (class in *annize.project*), 106
property_name (*annize.project.Node.PropertyChangedEvent* property), 104
ProvidedTrStr (class in *annize.i18n*), 94
public_url (*annize.features.distributables.flatpak.Repository* property), 35
PublicDomain (in module *annize.features.licensing*), 76
Python (class in *annize.features.dependencies.python*), 20
Python3ApiReferenceLanguage (class in *annize.features.documentation.sphinx.python*), 52
PythonPackage (class in *annize.features.dependencies.python*), 20
PythonSection (in module *annize.features.distributables.debian*), 29
R
readme_pdf() (in module *annize.asset.data*), 13
ReadmeDocument (class in *annize.features.documentation.sphinx.common*), 50
Recommended (class in *annize.features.dependencies.common*), 19
red (*annize.data.color.Color* property), 13
reference_key (*annize.project.ReferenceNode* property), 109
ReferenceBehavior (class in *annize.project.materializer.behaviors.reference*), 118
ReferenceNode (class in *annize.project*), 109
ReferenceNode.OnUnresolvableAction (class in *annize.project*), 109
regex_string() (*annize.data.version.AbstractVersionPatternSegment* method), 15
regex_string() (*annize.data.version.ConcatenatedVersionPatternSegment* method), 17
regex_string() (*annize.data.version.NumericVersionPatternSegment* method), 16
regex_string() (*annize.data.version.OptionalVersionPatternSegment* method), 16
regex_string() (*annize.data.version.SeparatorVersionPatternSegment* method), 16
region_code (*annize.i18n.Culture* property), 96
register_file_format() (in module *annize.project.file_formats*), 112
register_output_generator() (in module *annize.features.documentation.sphinx.output.common*), 43
relative_path (*annize.features.files.common.FsEntry* property), 57
release (*annize.features.documentation.sphinx.common.Document* property), 47
remove() (*annize.fs.Path* method), 86
remove_change_handler() (*annize.project.Node* method), 104
remove_child() (*annize.project.Node* method), 105
Repository (class in *annize.features.distributables.flatpak*), 35
Required (class in *annize.features.dependencies.common*), 19
resolve_appendtonodes() (in module *annize.project.materializer.preprocessors*), 120
resolve_reference_node() (*annize.project.inspector.Inspector* method), 123
resolved_type (*annize.object.parameter_info.ParameterInfo* property), 100
resolved_type (*annize.object.parameter_info.UnionParameterInfo* property), 101
result (*annize.features.distributables.debian.Package._BuildInfo* attribute), 32
result (*annize.features.distributables.flatpak.FlatpakImage._BuildInfo* attribute), 38
result (*annize.features.distributables.python_wheel.Package._BuildInfo* attribute), 41
result (*annize.project.materializer.core.NodeMaterialization* property), 119
root (*annize.features.files.common.DirectoryPart* property), 59
root (*annize.features.files.common.FsEntry* property), 57
root_objects (*annize.project.materializer.MaterializationResult* property), 115
rst_text (*annize.features.homepage.common.HomepageSection.Content* property), 64
RstDocument (class in *annize.features.documentation.sphinx.common*), 49
RstDocumentVariant (class in *annize.features.documentation.sphinx.common*), 49
RstGenerator (class in *annize.features.documentation.sphinx.rst*), 53
RubySection (in module *annize.features.distributables.debian*), 29
run() (*annize.features.testing.common.Test* method), 69
run() (*annize.features.testing.common.TestGroup* method), 69

run() (*annize.features.testing.pylint.Test* method), 69
 run() (*annize.features.testing.pytest.Test* method), 70
 run_runner() (*annize.flow.runner.Runner* method), 84
 RunContext (class in *annize.flow.run_context*), 78
 Runner (class in *annize.flow.runner*), 83
 RunTests (class in *annize.features.testing.common*), 69
 RustSection (in module *annize.features.distributables.debian*), 29

S

saturation (*annize.data.color.Color* property), 14
 save() (*annize.project.Project* method), 102
 save() (*annize.project.ProjectNode* method), 106
 save_fileno_to_file() (*annize.project.file_formats.xml.Marshaler* method), 114
 ScalarValueNode (class in *annize.project*), 109
 scalehue() (*annize.data.color.Color* method), 14
 ScienceSection (in module *annize.features.distributables.debian*), 29
 scope (*annize.project.BlockNodeWithScope* property), 110
 ScreenLockingCategory (in module *annize.features.distributables.debian*), 27
 ScreenSavingCategory (in module *annize.features.distributables.debian*), 26
 sdk (*annize.features.distributables.flatpak.FlatpakImage._BuildInfo* attribute), 37
 section (*annize.features.distributables.debian.Package._BuildInfo* attribute), 32
 Section (class in *annize.features.distributables.debian*), 27
 Section (class in *annize.features.homepage.sections.about*), 60
 Section (class in *annize.features.homepage.sections.changelog*), 61
 Section (class in *annize.features.homepage.sections.documentation*), 61
 Section (class in *annize.features.homepage.sections.download*), 61
 Section (class in *annize.features.homepage.sections.gallery*), 62
 Section (class in *annize.features.homepage.sections.imprint*), 62
 Section (class in *annize.features.homepage.sections.license*), 63
 sections (*annize.features.homepage.common.Homepage* property), 65
 segment_names (*annize.data.version.AbstractVersionPatternSegment* property), 15
 segment_names (*annize.data.version.ConcatenatedVersionPatternSegment* property), 17
 segment_names (*annize.data.version.NumericVersionPatternSegment* property), 15
 segment_names (*annize.data.version.OptionalVersionPatternSegment* property), 16
 segment_names (*annize.data.version.VersionPattern* property), 17
 segments (*annize.data.version.VersionPattern* property), 17
 segments_tuples (*annize.data.version.Version* property), 18
 segments_tuples (*annize.features.version_control.common.BuildVersion* property), 71
 segments_tuples_to_text() (*annize.data.version.AbstractVersionPatternSegment* method), 15
 segments_tuples_to_text() (*annize.data.version.ConcatenatedVersionPatternSegment* method), 17
 segments_tuples_to_text() (*annize.data.version.NumericVersionPatternSegment* method), 16
 segments_tuples_to_text() (*annize.data.version.OptionalVersionPatternSegment* method), 16
 segments_tuples_to_text() (*annize.data.version.SeparatorVersionPatternSegment* method), 16
 segments_tuples_to_text() (*annize.data.version.VersionPattern* method), 17
 segments_values (*annize.data.version.Version* property), 18
 SeparatorVersionPatternSegment (class in *annize.data.version*), 16
 ServiceDescription (class in *annize.features.distributables.debian*), 30
 services (*annize.features.distributables.debian.Package._BuildInfo* attribute), 32
 set_materialized_result() (*annize.project.materializer.core.NodeMaterialization* method), 119
 set_object_metadata() (*annize.flow.run_context.RunContext* method), 81
 set_object_metadata() (in module *annize.flow.run_context*), 83
 set_object_name() (*annize.flow.run_context.RunContext* method), 79

set_object_name() (in module annize.flow.run_context), 82
 set_problems() (annize.project.materializer.core.NodeMaterialization method), 21
 set_selected_task() (annize.flow.runner.Runner method), 84
 setuppy_conf (annize.features.distributables.python_wheels.BuildInfo attribute), 41
 Share (class in annize.features.distributables.flatpak), 35
 shares (annize.features.distributables.flatpak.FlatpakImage.BuildInfo attribute), 38
 ShellsSection (in module annize.features.distributables.debian), 29
 short_desc (annize.features.documentation.sphinx.output.html.HtmlOutputSpec property), 44
 short_desc (annize.features.homepage.common.Homepage property), 65
 short_str (annize.data.unique.UniqueId property), 15
 short_title (annize.features.documentation.sphinx.output.html.HtmlOutputSpec property), 44
 show_task_choser() (annize.flow.runner.Runner method), 84
 show_task_execution() (annize.flow.runner.Runner method), 84
 show_task_execution_success() (annize.flow.runner.Runner method), 84
 SimpleProjectHomepage (class in annize.features.homepage.common), 65
 SKIP (annize.project.ReferenceNode.OnUnresolvableAction attribute), 109
 SKIP_BLOCK (annize.project.OnFeatureUnavailableNode.Action attribute), 110
 SKIP_NODE (annize.project.OnFeatureUnavailableNode.Action attribute), 110
 sockets (annize.features.distributables.flatpak.FlatpakImage.BuildInfo attribute), 38
 sort_index (annize.features.homepage.common.Homepage property), 64
 SoundSection (in module annize.features.distributables.debian), 29
 source (annize.features.distributables.debian.Package._BuildInfo attribute), 32
 source (annize.features.distributables.flatpak.FlatpakImage._BuildInfo attribute), 37
 source (annize.features.distributables.python_wheel.Package._BuildInfo attribute), 41
 source (annize.features.documentation.sphinx.common.ApiReference property), 48
 source (annize.features.documentation.sphinx.common.RstDocumentVariant property), 49
 source_path (annize.features.files.common.DirectoryPart property), 59
 StaticUserFeedbackController (class in annize.user_feedback.static), 127
 store_package() (annize.features.distributables.common.PackageStore String (class in annize.features.i18n.common), 67
 string_name (annize.i18n.ProvidedTrStr property), 94
 studio() (annize.annize_cli.Commands method), 128
 Summary (annize.features.base.Data property), 73
 summary (annize.features.distributables.debian.Package._BuildInfo attribute), 32
 summary (in module annize.features.base), 74
 T
 target_node (annize.project.Node.ChangeEvent property), 13
 Task (class in annize.features.task), 77
 TasksSection (in module annize.features.distributables.debian), 29
 TCultureSpec (in module annize.i18n), 93
 temp_clone() (annize.fs.Path method), 87
 Test (class in annize.features.testing.common), 69
 Test (class in annize.features.testing.pylint), 69
 Test (class in annize.features.testing.pytest), 70
 TestGroup (class in annize.features.testing.common), 69
 TexSection (in module annize.features.distributables.debian), 29
 text (annize.data.version.Version property), 18
 text (annize.features.changelog.common.Item property), 18
 text (annize.features.licensing.License property), 75
 text (annize.features.version_control.common.BuildVersion property), 71
 text_to_segments_tuples() (annize.data.version.VersionPattern method), 17
 TextProcessingSection (in module annize.features.distributables.debian), 29
 TextSource (class in annize.features.i18n.gettext), 68
 theme (annize.features.documentation.sphinx.output.html.HtmlOutputSpec property), 44
 time (annize.features.changelog.common.Entry property), 18
 title (annize.features.distributables.common.Group property), 22
 title (annize.features.distributables.debian.MenuEntry property), 23
 title (annize.features.distributables.flatpak.MenuEntry property), 23
 title (annize.features.distributables.flatpak.MenuEntry property), 34
 title (annize.features.documentation.sphinx.common.Document property), 47
 title (annize.features.documentation.sphinx.common.ReadmeDocument property), 50
 title (annize.features.homepage.common.Homepage property), 65

- [title](#) ([annize.features.media_galleries.Gallery](#) property), 77
[title__original](#) ([annize.features.documentation.sphinx.common.Document](#) property), 47
[to_trstr\(\)](#) (in module [annize.i18n](#)), 94
[token](#) ([annize.features.distributables.store.pypi.Connection](#) property), 21
[tr\(\)](#) ([annize.i18n.TrStr](#) static method), 93
[tr\(\)](#) (in module [annize.i18n](#)), 93
[tr_if_trstr\(\)](#) (in module [annize.i18n](#)), 94
[transfer_action_copy\(\)](#) ([annize.fs.Path._TransferHelper](#) static method), 88
[transfer_action_move\(\)](#) ([annize.fs.Path._TransferHelper](#) static method), 88
[transfer_to\(\)](#) ([annize.fs.Path._TransferHelper](#) static method), 88
[transformed\(\)](#) ([annize.data.color.Color](#) method), 14
[translate\(\)](#) ([annize.features.i18n.common._ProjectDefinedTranslationProvider](#) method), 66
[translate\(\)](#) ([annize.i18n.GettextTranslationProvider](#) method), 92
[translate\(\)](#) ([annize.i18n.TranslationProvider](#) method), 92
[translate\(\)](#) ([annize.i18n.TrStr](#) method), 93
[translation_providers\(\)](#) (in module [annize.i18n](#)), 93
[TranslationProvider](#) (class in [annize.i18n](#)), 91
[TranslationUnavailableError](#), 98
[TrStr](#) (class in [annize.i18n](#)), 93
[TrStrOrStr](#) (in module [annize.i18n](#)), 94
[try_get_materialization_for_node\(\)](#) ([annize.project.materializer.core.NodeMaterialization](#) method), 119
[TTransferFilter](#) ([annize.fs.Path](#) attribute), 87
[type](#) ([annize.project.inspector.Inspector.TypeInfo](#) property), 122
[type_name](#) ([annize.project.ObjectNode](#) property), 108
[type_parameter_infos\(\)](#) (in module [annize.object.parameter_info](#)), 101
[typename](#) ([annize.project.inspector.Inspector.TypeInfo](#) property), 122
- ## U
- [undo_changes\(\)](#) ([annize.project.ProjectNode](#) method), 107
[UnionParameterInfo](#) (class in [annize.object.parameter_info](#)), 101
[UniqueId](#) (class in [annize.data.unique](#)), 14
[UnresolvableReferenceError](#), 111
[UnsatisfiableUserFeedbackAttemptError](#), 126
[UnspecifiedCulture](#) (class in [annize.i18n](#)), 97
- [UpdatePOs](#) (class in [annize.features.i18n.gettext](#)), 67
[Upload](#) (class in [annize.features.distributables.store.pypi](#)), 21
[upload](#) (class in [annize.features.files.transfer.common](#)), 56
[upload\(\)](#) ([annize.features.distributables.flatpak.LocalRepository](#) method), 35
[upload\(\)](#) ([annize.features.distributables.flatpak.Repository](#) method), 35
[UserFeedbackController](#) (class in [annize.user_feedback](#)), 125
[username](#) ([annize.features.files.transfer.ssh.Endpoint](#) property), 56
[UtilitiesSection](#) (in module [annize.features.distributables.debian](#)), 29
- ## V
- [value](#) ([annize.project.ScalarValueNode](#) property), 109
[version](#) ([annize.features.changelog.common.Entry](#) property), 18
[version](#) ([annize.features.dependencies.python.PythonPackage](#) property), 20
[version](#) ([annize.features.distributables.debian.Package._BuildInfo](#) attribute), 32
[version](#) ([annize.features.distributables.python_wheel.Package._BuildInfo](#) attribute), 41
[version](#) ([annize.features.documentation.sphinx.common.Document](#) property), 47
[version](#) ([annize.features.version.Line](#) property), 77
[Version](#) (class in [annize.data.version](#)), 17
[Version](#) (class in [annize.features.version](#)), 77
[VersionControlSystem](#) (class in [annize.features.version_control.common](#)), 70
[VersionControlSystem](#) (class in [annize.features.version_control.git](#)), 71
[VersionControlSystemsSection](#) (in module [annize.features.distributables.debian](#)), 29
[VersionPattern](#) (class in [annize.data.version](#)), 17
[VIDEO](#) ([annize.features.media_galleries.MediaType](#) attribute), 76
[VideoSection](#) (in module [annize.features.distributables.debian](#)), 29
[VirtualPackagesSection](#) (in module [annize.features.distributables.debian](#)), 30
- ## W
- [wait_finished\(\)](#) ([annize.flow.runner.Runner](#) method), 85
[WebServersSection](#) (in module [annize.features.distributables.debian](#)), 28
[WebSoftwareSection](#) (in module [annize.features.distributables.debian](#)), 30
[with_modified\(\)](#) ([annize.data.color.Color](#) method), 14

`with_updates()` (*annize.object.controller._CreateObjectHelper.ParameterConfig*
method), [99](#)

`write_file()` (*annize.fs.Path* *method*), [86](#)

X

`XfceSection` (*in module* *annize.features.distributables.debian*), [30](#)

`XmlFileFormat` (*class in* *annize.project.file_formats.xml*), [112](#)

`XWindowSystemSoftwareSection` (*in module* *annize.features.distributables.debian*), [30](#)

Z

`ZopePloneFrameworkSection` (*in module* *annize.features.distributables.debian*), [30](#)