
Annize

Release 6.0.6477

Author name not set

30.07.2025

Inhaltsverzeichnis

1	Lizenz	3
2	Up-to-date?	5
3	Abhängigkeiten	7
4	Benutzerhandbuch	9
5	Kommandozeilenschnittstellen-Referenz	11
5.1	Positional Arguments	11
5.2	Named Arguments	11
5.3	Sub-commands	12
6	API-Referenz	13
6.1	annize namespace	13
	Python-Modulindex	131
	Stichwortverzeichnis	133

TODO jooh

KAPITEL 1

Lizenz

Annize wird unter den Bedingungen der AGPL 3 Lizenz verteilt. Das betrifft ebenso alle Dateien ohne Lizenzkopf (Nicht Quelldateien, wie Grafiken), außer sie sind ausdrücklich als Drittinhalt gekennzeichnet. Lesen Sie die Sektion ‚Abhängigkeiten‘ bezüglich enthaltener Drittinhalte.

KAPITEL 2

Up-to-date?

Lesen Sie diesen Text von einer anderen Quelle als der Homepage? Falls Sie unsicher sind ob Ihr Paket aktuell ist, sollten Sie die Homepage besuchen und das prüfen. Sie lesen derzeit die Dokumentation für Version 6.0.6477.

Abhängigkeiten

Annize nutzt einige Teile von Drittanbietern.



Benötigt: **Python 3.13**



Benötigt: **Python package hallyd** ~= 0.90



Benötigt: **Python package klovve[graphical]** ~= 1.6



Benötigt: **Python package lxml** ~= 6.0



Benötigt: **Python package pycountry** ~= 24.6



Empfohlen: **GNU/Linux**



Enthalten: **background image** (Lizenz: <https://creativecommons.org/licenses/by-sa/3.0>; von [hier](#))



Enthalten: **font ‚Inconsolata‘** (for websites; by Raph Levien; Lizenz: OFL; von [hier](#))



Enthalten: **font ,Khula‘** (for websites; by Erin McLaughlin; Lizenz: OFL; [von hier](#))



Enthalten: **font ,Symbola‘** (for logo symbol; Lizenz: free for use; [von hier](#))



Enthalten: **icon set ,Oxygen‘** (some icons; [von hier](#))



Enthalten: **third-party project logos** ([von hier](#))

KAPITEL 4

Benutzerhandbuch

TODO zz

Kommandozeilenschnittstellen-Referenz

```
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: ''

6.1 annize namespace

6.1.1 Subpackages

annize.asset package

Submodules

annize.asset.data module

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

Parameter

culture (*str*)

Rückgabetyp

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`

Parameter

- **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)

Parameter

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

Rückgabetyp

[Color](#)

property html_color_spec: str

scalehue(*, brightness, saturation=None)

Parameter

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

Rückgabetyp

[Color](#)

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

Parameter

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

Rückgabetyp

[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
        Parameter
            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)
            Parameter
                txt (str)
            Rückgabetyt
                str

```

annize.data.version module

```

class annize.data.version.AbstractVersionPatternSegment
    Bases: ABC
        property segment_names: list[str]
        abstractmethod regexp_string()
            Rückgabetyt
                str
        abstractmethod segments_tuples_to_text(segments_tuples)
            Parameter
                segments_tuples (list[Tuple[str, str]])
            Rückgabetyt
                str
        _name_anon(namep)
            Parameter
                namep (str)
            Rückgabetyt
                None
        _str_to_val(txt)
            Parameter
                txt (str)

```

```

    Rückgabotyp
    object
    _abc_impl = <_abc._abc_data object>

class annize.data.version.NumericVersionPatternSegment(*, partname=None)
    Bases: AbstractVersionPatternSegment
    Parameter
        partname (str | None)
    property partname: str | None
    property segment_names: list[str]
    regexp_string()

    Rückgabotyp
    str
    segments_tuples_to_text(segments_tuples)

    Parameter
        segments_tuples (list[Tuple[str, str]])
    Rückgabotyp
    str
    _name_anon(namep)
    _str_to_val(txt)

    Parameter
        txt (str)
    Rückgabotyp
    object
    _abc_impl = <_abc._abc_data object>

class annize.data.version.SeparatorVersionPatternSegment(*, text)
    Bases: AbstractVersionPatternSegment
    Parameter
        text (str)
    regexp_string()

    Rückgabotyp
    str
    segments_tuples_to_text(segments_tuples)

    Parameter
        segments_tuples (list[Tuple[str, str]])
    Rückgabotyp
    str
    _abc_impl = <_abc._abc_data object>
```

```

class annize.data.version.OptionalVersionPatternSegment(*, segments)
    Bases: AbstractVersionPatternSegment
        Parameter
            segments (Iterable[AbstractVersionPatternSegment])
        property segment_names
        regexp_string()
        Rückgabetyt
            str
        segments_tuples_to_text(segments_tuples)
        Parameter
            segments_tuples (list[Tuple[str, str]])
        Rückgabetyt
            str
        _name_anon(namep)
        _abc_impl = <_abc._abc_data object>

class annize.data.version.ConcatenatedVersionPatternSegment(*, segments)
    Bases: AbstractVersionPatternSegment
        Parameter
            segments (Iterable[AbstractVersionPatternSegment])
        property segment_names
        regexp_string()
        Rückgabetyt
            str
        segments_tuples_to_text(segments_tuples)
        Parameter
            segments_tuples (list[Tuple[str, str]])
        Rückgabetyt
            str
        _name_anon(namep)
        _abc_impl = <_abc._abc_data object>

class annize.data.version.VersionPattern(*, segments)
    Bases: object
        Parameter
            segments (Iterable[AbstractVersionPatternSegment])
        property segments: list[AbstractVersionPatternSegment]
        property segment_names

```

text_to_segments_tuples(*text*)

Parameter

text (*str*)

Rückgabetyp

list[Tuple[str, str]]

segments_tuples_to_text(*segments_tuples*)

Parameter

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

Rückgabetyp

str

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

Bases: object

Parameter

- **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

Parameter

text (*TrStr*)

property **text**: *TrStr*

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

Bases: object

Parameter

- **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
        Parameter
            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()

    Rückgabetyt
        Changelog

```

annize.features.dependencies namespace

Submodules

annize.features.dependencies.common module

Project dependency handling.

```

class annize.features.dependencies.common.Kind(*, label, importance)
    Bases: object
        Parameter
            • label (TrStr)
            • importance (int)

    property label: TrStr

    property importance: int

class annize.features.dependencies.common.Dependency(*, kind, label, comment, icon, importance=0)
    Bases: object
        Parameter
            • 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*

Parameter

- **origin** (*str*)
- **license** (*str* | *None*)

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

Parameter

dependencies (*list* [*Dependency*])

Rückgabetyp

str

annize.features.dependencies.python module

Python project dependency handling.

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

Bases: *Dependency*

Parameter

version (*str*)

class `annize.features.dependencies.python.PythonPackage(*, name, version, kind, comment)`

Bases: *Dependency*

Parameter

- **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](#)

Parameter

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

Parameter

token (*str*)

property token: str

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

Bases: object

Parameter

- **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.

Parameter

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

Rückgabotyp*None***abstractmethod package**(***, *name*)

Return package items.

Parameter**name** (*str*) – The item name.**Rückgabotyp***Sequence*[*FileSystemContent*] | *None***abstractmethod package_history**(***, *name*, *limit=3*)

Return the package history.

Parameter

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

Rückgabotyp*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.

Parameter

- **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.

Rückgabotyp*Sequence*[*FileSystemContent*]**property description:** *TrStr*

Additional description text.

property package_store: *PackageStore* | *None*

An optional package store.

```

__files_from_package_store()

    Rückgabotyp
        Iterable[FilesystemContent] | None

__store_files_to_package_store()

    Rückgabotyp
        None

__package_store_name()

    Rückgabotyp
        str

```

annize.features.distributables.debian module

Debian (.deb) packages.

```

class annize.features.distributables.debian.Category(*, debian_name, freedesktop_name)
    Bases: object

```

Parameter

- **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

Parameter

- **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
```

```
        Parameter
```

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

```
    property path: str
```

```
    property name: str
```

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

```
        Parameter
```

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

```
        Rückgabety
```

```
        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)

    Parameter
        name (str)

    Rückgabetyt
        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.OtherOSsAndFSsSection`
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.

Parameter

- **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`

Parameter

- **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_pixmaps: str = None,
    pkgpath_usrbin: str = None, config_files: list[str] = None, pkgsize: int = None, result:
    annize.fs.FilesystemContent = None)
```

Bases: [object](#)

Parameter

- **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)
    Parameter
        build (_BuildInfo)
    Rückgabetyt
        FilesystemContent

classmethod _mkpackage_prepareinfos(build, tmpdir)
    Parameter
        • build (_BuildInfo)
        • tmpdir (str | Path)
    Rückgabetyt
        None

classmethod _mkpackage_mkcopyright(build)
    Parameter
        build (_BuildInfo)
    Rückgabetyt
        None

classmethod _mkpackage_mkchangelog(build)
    Parameter
        build (_BuildInfo)
    Rückgabetyt
        None

classmethod _mkpackage_mkexeclinks(build)
    Parameter
        build (_BuildInfo)
    Rückgabetyt
        None

classmethod _mkpackage_mkmenuentries(build)
    Parameter
        build (_BuildInfo)
    Rückgabetyt
        None

```

classmethod `_mkpackage_mkservices(build)`

Parameter

`build` (`_BuildInfo`)

Rückgabotyp

None

classmethod `_mkpackage_determinesize(build)`

Parameter

`build` (`_BuildInfo`)

Rückgabotyp

None

classmethod `_mkpackage_mkprepostcmds(build)`

Parameter

`build` (`_BuildInfo`)

Rückgabotyp

None

classmethod `_mkpackage_mkdebiancontrolfile(build)`

Parameter

`build` (`_BuildInfo`)

Rückgabotyp

None

classmethod `_mkpackage_mkdebianconffilesfile(build)`

Parameter

`build` (`_BuildInfo`)

Rückgabotyp

None

classmethod `_mkpackage_correctbuildsourcepermissions(build)`

Parameter

`build` (`_BuildInfo`)

Rückgabotyp

None

classmethod `_mkpackage_dpkg(build)`

Parameter

`build` (`_BuildInfo`)

Rückgabotyp

None

annize.features.distributables.flatpak module

Flatpaks.

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

Bases: object

Parameter

- **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

Parameter

- **public_url** (*str*)
- **friendly_name_suggestion** (*str* | *None*)

upload(*source*)

Parameter

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*

Parameter

- **public_url** (*str*)
- **friendly_name_suggestion** (*str* | *None*)
- **upload_path** (*str* | *None*)
- **upload_fsentry** (*FileSystemContent* | *None*)

upload(*source*)

Parameter

source (*FileSystemContent*)

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

Bases: *Group*

Parameter

- **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*

Parameter

- **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](#)

Parameter

- **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](#)

Parameter

- **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`

Parameter

- **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`)

Parameter

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

Rückgabotyp

`None`

classmethod `_mkpackage_flatpak_build_init`(`build`)

Parameter

`build` (`_BuildInfo`)

Rückgabotyp

`None`

classmethod `_mkpackage_applysource(build)`

Parameter

`build` (`_BuildInfo`)

Rückgabety

None

classmethod `_mkpackage_flatpak_build_finish(build)`

Parameter

`build` (`_BuildInfo`)

Rückgabety

None

classmethod `_mkpackage_share(build)`

Parameter

`build` (`_BuildInfo`)

Rückgabety

None

classmethod `_mkpackage_flatpak_build_export(build)`

Parameter

`build` (`_BuildInfo`)

Rückgabety

None

classmethod `_mkpackage(build)`

Parameter

`build` (`_BuildInfo`)

Rückgabety

`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`

Parameter

- `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`

Parameter

- **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`

Parameter

- **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`

Parameter

- `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)`

Parameter

`build` (`_BuildInfo`)

Rückgabotyp

`FileSystemContent`

classmethod `_mkpackage_prepareinfos(build, tmpdir)`

Parameter

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

Rückgabotyp

`None`

classmethod `_mkpackage_setuppyconf_prepare(build)`

Parameter

`build` (`_BuildInfo`)

Rückgabotyp

`None`

classmethod `_mkpackage_setuppyconf_install_requires(build)`

Parameter

`build` (`_BuildInfo`)

Rückgabotyp

`None`

classmethod `_mkpackage_setuppyconf_classifiers(build)`

Parameter

`build` (`_BuildInfo`)

Rückgabotyp

`None`

classmethod `_mkpackage_mkexeclinks(build)`

Parameter

`build` (`_BuildInfo`)

Rückgabotyp

`None`

classmethod `_mkpackage_mksetuppyconf(build)`

Parameter

`build` (`_BuildInfo`)

Rückgabotyp

`None`

classmethod `_mkpackage_bdist_wheel(build)`

Parameter

`build` (`_BuildInfo`)

Rückgabotyp

None

classmethod `_mkpackage_mkmanifestin(build)`**Parameter**`build` (`_BuildInfo`)**Rückgabotyp**

None

annize.features.distributables.tar module

Tarballs.

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

- **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*)

Parameter

output_spec ([OutputSpec](#))

Rückgabotyp

bool

abstractmethod formatname()

Returns the Sphinx format name.

Rückgabotyp

str

prepare_generate(*geninfo*)

Parameter

geninfo (*documentationsphinx.Document.GenerateInfo*)

Rückgabotyp

None

postproc(*preresult*)

Parameter

preresult ([FilesystemContent](#))

Rückgabotyp

[FilesystemContent](#)

multilanguage_frame(*document*)

Parameter

document (*documentationsphinx.Document*)

Rückgabotyp

[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, back-
    ground_image=None)
```

Bases: [HtmlOutputSpec](#)

Parameter

- **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*)

Parameter

preresult (*str* | *Path*)

Rückgabotyp

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

Parameter

- **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

Parameter

- **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)
    Parameter
        geninfo (GenerateInfo)
    Rückgabety
        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
    Parameter
        documents (Iterable[Document])

```

```
__get_inner_generateinfo(innerdoc)
```

Parameter

- **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](#)

Parameter

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

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

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

```
abstractmethod generate_sources(rgeninfo)
```

Parameter

rgeninfo ([ApiReferenceGenerateInfo](#))

Rückgabetyp

[str](#)

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

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

Bases: [Document](#)

An api reference.

Parameter

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

```
available_cultures()
```

```
__get_refgeninfo(geninfo)
```

Parameter

geninfo ([GenerateInfo](#))

```
_generate_sources(geninfo)
```

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

```
class annize.features.documentation.sphinx.common.ArgparseCommandLineInterfaceDocument(*,
                                                                 par-
                                                                 ser_factory,
                                                                 pro-
                                                                 gram_name,
                                                                 hea-
                                                                 ding,
                                                                 sour-
                                                                 ce_directory,
                                                                 cul-
                                                                 tu-
                                                                 res,
                                                                 **kwargs)
```

Bases: [Document](#)

Parameter

- **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](#)

Parameter

- **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.

Parameter

```
variants (list[RstDocumentVariant])
```

```
available_cultures()
```

```
__get_variant(culture)
```

Parameter

culture (*Culture*)

Rückgabebetyp

RstDocumentVariant | None

```
_generate_sources(geninfo)
```

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

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

Bases: *Document*

Parameter

- **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.

Parameter

- **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(*,
    _doxy-
    ge-
    nopts=None,
    ex-
    tract_all=True,
    ex-
    tract_private=False,
    ex-
    tract_package=False,
    ex-
    tract_static=False,
    ex-
    tract_local=False,
    ex-
    tract_local_package=False,
    ex-
    tract_anon=False,
    ex-
    tract_priv=False,
    fi-
    le_patterns=None,
    in-
    li-
    ne_inheritance=True,
    in-
    her-
    it_docs=True,
    hi-
    de_undoc_
    hi-
    de_undoc_
    ex-
    clu-
    de_pattern
    pre-
    de-
    fi-
    ned=None)
```

Bases: [*ApiReferenceLanguage*](#)

Language support for api references powered by Doxygen.

Parameter

- **_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_nsplaces** (*bool*)
- **extract_priv_virtual** (*bool*)
- **file_patterns** (*str*)
- **inline_inherited_memb** (*bool*)
- **inherit_docs** (*bool*)
- **hide_undoc_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*)

Parameter

dirf (*str*)

Rückgabotyp

list[str]

__scanjsfile(*f*)

Parameter

f (*str*)

Rückgabotyp

str

generate_sources(*rgeninfo*)

_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.

Parameter

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

__patch_property_types_in_docstrings (*pdir*)

Parameter

pdir (*str*)

Rückgabetyp

None

generate_sources (*rgeninfo*)

_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.

Parameter

- **text** ([TrStr](#) | *str*)
- **variant** (*str*)
- **sub** (*bool*)
- **anchor** (*str* | *None*)

Rückgabetyp

str

Submodules

annize.features.documentation.common module

Documentation.

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

Bases: object

Parameter

- **file** ([FilesystemContent](#))
- **entry_path** (*str*)

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

property **entry_path**: *str*

_set_entry_path(*entry_path*)

Parameter

entry_path (*str*)

Rückgabotyp

None

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

Bases: [DocumentGenerateResult](#)

Parameter

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

entry_path_for_language(*language*)

Parameter

language (*str*)

Rückgabotyp

str

property **culture_names**: *Sequence[str]*

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

Bases: ABC

abstractmethod available_cultures()

Rückgabotyp

Sequence[Culture]

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

Parameter

culture ([Culture](#))

Rückgabotyp

[DocumentGenerateResult](#)

abstractmethod generate_all_cultures(*output_spec*)

Rückgabotyp

[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.

Parameter

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`

Parameter

- **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)
```

Parameter

 rootpath (str)

Rückgabetyt

 ContextManager[Path, bool | None]

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

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

Bases: [Endpoint](#)

Parameter

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

```
    access_filesystem(rootpath)
```

Parameter

 rootpath (str)

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

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

Bases: object

Parameter

- 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](#)

Parameter

- 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)

    Parameter
        path (str | None)

    Rückgabetyt
        str

exec(cmdline)

    Parameter
        cmdline (str)

    Rückgabetyt
        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](#).

Parameter

- **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.

Parameter

- **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](#).

Parameter

- **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*)

Parameter

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

Rückgabetyp

`bool`

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

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

Parameter

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

Rückgabetyp

`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.

Parameter

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.

Parameter

- **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](#).

Parameter

- **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).

Parameter

- **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()

Parameter

exclude (*Exclude*)

Rückgabotyp

annize.fs.Path.TTransferFilter

class annize.features.files.common.**ProjectDirectory**

Bases: *FilesystemContent*

The Annize project root directory.

Parameter

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. /.

Parameter

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](#)

Parameter

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](#)

Parameter

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](#)

Parameter

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

__generate_packagelist(*info*)

Parameter
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*)

Parameter
isize (*int*)

Rückgabety
str

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

Parameter
filepath (*str*)

Rückgabety
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](#)

Parameter
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](#)

```

    Parameter
        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

Parameter

- 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

Parameter

- 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

Parameter

- **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

Parameter

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

property rst_text: [str](#)

property media_files: [list](#)[[FilesystemContent](#)]

append_rst([rst_text](#))

Rückgabotyp

[None](#)

attach_media_file([file](#))

Parameter

file ([FilesystemContent](#))

Rückgabotyp

[None](#)

property head: [TrStr](#)

property sort_index: [int](#)

pre_process_generate([info](#))

Parameter

info ([_PrePostProcGenerateInfo](#))

Rückgabotyp

[None](#)

abstractmethod generate_content([info](#))

Parameter

info ([_GenerateInfo](#))

Rückgabotyp

[Content](#)

`post_process_generate(info)`

Parameter

`info` (`_PrePostProcGenerateInfo`)

Rückgabetyp

None

`_abc_impl = <_abc._abc_data object>`

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

Bases: `object`

Parameter

- `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)`

Parameter

`section` (`HomepageSection`)

Rückgabetyp

None

`generate()`

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

Parameter

- `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)`

Parameter

- `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](#)

Parameter

- **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](#)

Parameter

- **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.

Parameter

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

```
add_translations(string_name, variants)
```

Parameter

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

Rückgabotyp

None

_ProjectDefinedTranslationProvider__translations_for_string_name(*string_name*)**Parameter****string_name** (*str*)**Rückgabotyp**

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()`.**Parameter**

- **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()`.**Parameter**

- **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.

Parameter**cultures** (*list*[[Culture](#)])`annize.features.i18n.common.project_cultures()`Return a list of the current Annize project's target cultures. See also [ProjectCultures](#).**Rückgabotyp***Sequence*[[Culture](#)]

annize.features.i18n.gettext module

gettext-based internationalization.

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

Bases: object

Parameter

po_directory (*str* | *Path*)

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

Bases: object

Parameter

- **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

Parameter

- **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)
```

Parameter

destination (*Path*)

_abc_impl = <_abc._abc_data object>

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

Bases: *Injection*

Parameter

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

property content: *FileSystemContent*

```
inject(destination)
```

Parameter

destination (*Path*)

_abc_impl = <_abc._abc_data object>


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

Bases: object

Parameter

- **injection** ([Injection](#))
- **destination** ([Path](#))

annize.features.injections.python module

Python injections.

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

Bases: [Injection](#)

Parameter

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

inject(*destination*)

Parameter

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](#)

Parameter

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*

Parameter

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*

Parameter

- **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**()

Rückgabotyp

str

abstractmethod **get_revision_list**()

Rückgabotyp

list[str]

abstractmethod **get_commit_message**(*revision*)

Parameter

revision (*str*)

Rückgabotyp

str

abstractmethod `get_commit_time(revision)`

Parameter

revision (*str*)

Rückgabety

datetime

abstractmethod `get_revision_number(revision)`

Parameter

revision (*str*)

Rückgabety

int

`_abc_impl = <_abc._abc_data object>`

class `annize.features.version_control.common.BuildVersion(*, base_version, vcs)`

Bases: *Version*

Parameter

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

`__effversion()`

property `segments_tuples`

property `text`

`annize.features.version_control.common.default_version_control_system()`

Rückgabety

VersionControlSystem | *None*

annize.features.version_control.git module

git-based version control.

class `annize.features.version_control.git.VersionControlSystem(*, path)`

Bases: *VersionControlSystem*

Parameter

path (*str* | *None*)

property `path`: *Path*

`__call_git(cmdline)`

Parameter

cmdline (*list[str]*)

Rückgabety

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](#)

Parameter

- **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.

Parameter

- **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

Parameter

- **full_name** ([TrStr](#))
- **email** ([str](#) | [None](#))

```
property full_name: TrStr
```

```
property email: str | None
```

```
annize.features.authors.project_authors()
```

Rückgabetyp

[list](#)[[Author](#)]

```
annize.features.authors.join_authors(authors)
```

Parameter

authors ([list](#)[[Author](#)])

Rückgabetyp

[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

Parameter

- **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*

Parameter

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

```
class annize.features.base.DateTime(*, iso)
```

Bases: datetime

Parameter

iso (*str*)

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

Bases: object

Parameter

- **from_string** (*str*)
- **split_by** (*str*)

- **keywords** (*list[str]*)

property keywords: *list[str]*

class `annize.features.base.Keyword`(*text*)

Bases: *Keywords*

Parameter

text (*str*)

`annize.features.base.project_keywords()`

Rückgabetyp
Keywords

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

Bases: *Basket*

Parameter

items (*list[object]*)

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

Bases: *Basket*

Parameter

objects (*list[object]*)

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

Parameter

none_on_undefined (*bool*)

Rückgabetyp
Color

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

Parameter

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

Rückgabetyp
Any

`annize.features.base.project_name()`

Rückgabetyp
str

`annize.features.base.pretty_project_name()`

Rückgabetyp
TrStr

`annize.features.base.summary()`

Rückgabetyp
TrStr

`annize.features.base.long_description()`

Rückgabotyp
`TrStr`

`annize.features.base.homepage_url()`

Rückgabotyp
`str`

`annize.features.base.imprint()`

Rückgabotyp
`TrStr`

`annize.features.base.project_directory()`

Rückgabotyp
`Path`

`annize.features.licensing` module

Project licensing information.

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

Bases: object

Parameter

- **name** (`TrStr`)
- **text** (`TrStr` | `None`)

property name: `TrStr`

property text: `TrStr` | `None`

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

Parameter

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

Rückgabotyp
`Any`

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

Parameter

name (`str`)

Rückgabotyp

`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()`

Rückgabetyp
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

Parameter

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

```
class Item(file, description, mediatype)
```

Bases: object

Parameter

- **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*)

Parameter

itemfile (*FilesystemContent*)

Rückgabotyp

TrStr

annize.features.task module

Tasks.

```
class annize.features.task.Task(* , innertasks, is_advanced=False)
```

Bases: object

Parameter

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

property **is_advanced**: bool

annize.features.version module

Project versioning.

```
class annize.features.version.Line(*, version)
```

Bases: object

Parameter

version ([Version](#))

property **version**: [Version](#)

```
class annize.features.version.Version(*, text, pattern, **segment_values)
```

Bases: [Version](#)

Parameter

- **text** ([str](#) / [None](#))
- **pattern** ([VersionPattern](#) / [None](#))

```
annize.features.version.default_version_pattern()
```

Rückgabetyp

[VersionPattern](#)

```
annize.features.version.project_versions()
```

Rückgabetyp

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.

Parameter

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

Rückgabotyp

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\(\)](#).

Parameter

- **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.

Rückgabotyp

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\(\)](#).

Parameter

obj (*Any*) – The object.

Rückgabotyp

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\(\)](#).

Parameter

obj (*Any*) – The object.

Rückgabotyp

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.

Parameter

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

Rückgabotyp

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.

Parameter

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

Rückgabotyp

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.

Parameter

obj (*Any*) – The object to add.

Rückgabotyp

str

is_friendly_name(*name*)

Returns whether the given name is a friendly one.

Parameter

name (*str*) – The name to check.

Rückgabotyp

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).

Parameter**obj** (*Any*) – The object to check.**Rückgabotyp**

bool

mark_object_as_toplevel(*obj*)Mark an object as a toplevel one. See [is_toplevel_object\(\)](#).**Parameter****obj** (*Any*) – The object.**Rückgabotyp**

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\(\)](#).**Parameter**

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

Rückgabotyp*Any***set_object_metadata**(*obj, key, value=None*)

Store a piece of metadata for a given object.

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

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

Rückgabotyp

None

__put_object(*name, obj*)**Parameter**

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

Rückgabotyp

None

__object_raw_name(*obj*)**Parameter****obj** (*Any*)**Rückgabotyp**

str

`__object_metadata_dict(obj)`

Parameter

`obj` (*Any*)

Rückgabetyp

`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.

Rückgabetyp

`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()`).

Parameter

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

Rückgabetyp

Any

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

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

Parameter

`obj` (*Any*)

Rückgabetyp

`list[str]`

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

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

Parameter

`obj` (*Any*)

Rückgabetyp

str

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

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

Parameter

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

Rückgabotyp

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()`).

Parameter

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

Rückgabotyp

list[T]

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

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

Parameter

`obj` (`Any`)

Rückgabotyp

str

`annize.flow.run_context.is_friendly_name(name)`

Same as `RunContext.is_friendly_name()` on the `_current_run` context (`current()`).

Parameter

`name` (`str`)

Rückgabotyp

bool

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

Same as `RunContext.is_toplevel_object()` on the `_current_run` context (`current()`).

Parameter

`obj` (`Any`)

Rückgabotyp

bool

`annize.flow.run_context.object_metadata(obj, key, default=None)`

Same as `RunContext.object_metadata()` on the `_current_run` context (`current()`).

Parameter

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

Rückgabotyp

Any

`annize.flow.run_context.set_object_metadata(obj, key, value=None)`

Same as `RunContext.set_object_metadata()` on the `_current_run` context (`current()`).

Parameter

- `obj` (`Any`)
- `key` (`str`)

- **value** (*Any*)

Rückgabotyp

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.

Parameter

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

run_runner()

Rückgabotyp

None

_dispatch(*func*)

__do_run(*project*)

Parameter

project ([Project](#))

abstractmethod show_task_chooser()

Rückgabotyp

None

abstractmethod show_task_execution()

Rückgabotyp

None

abstractmethod show_task_execution_success()

Rückgabotyp

None

get_tasks()

Rückgabotyp

list[str]

__set_tasks(*tasks*)

Parameter

tasks (*list[str]*)


```

    Rückgabotyp
    None
get_selected_task()

    Rückgabotyp
    str
set_selected_task(task_name)

    Parameter
    task_name (str)

    Rückgabotyp
    None
is_finished()

    Rückgabotyp
    bool
get_success_state()

    Rückgabotyp
    Tuple[bool, str]
__set_success_state(success, message)

    Parameter
    • success (bool)
    • message (str)

    Rückgabotyp
    None
wait_finished()

    Rückgabotyp
    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.

Parameter

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).

Rückgabotyp

`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()`.

Parameter

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`).

Rückgabotyp

`Path`

path()

Return itself (in order to implement `FilesystemContent`).

Rückgabotyp

`Path`

children()

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

Rückgabotyp

Sequence[`Path`]

ctime()

Return the ctime for this path.

Rückgabotyp

datetime

mtime()

Return the mtime for this path.

Rückgabotyp*datetime***write_file(*data*)**

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

Parameter

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

Rückgabotyp*None***remove(*, *missing_ok=True*)**

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

Parameter

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

Rückgabotyp*None***file_size()**

Return the file size in bytes.

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

Return a temporary clone of the content at this path.

Parameter

- **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.

Rückgabotyp*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.

Parameter

- **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.

Rückgabotyp

[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.

Parameter

- **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.

Rückgabotyp

[Path](#)

class TransferFilters

Bases: `object`

class And(**inner_filters*)

Bases: `object`

Parameter

inner_filters (*Path.TTransferFilter*)

class _TransferHelper

Bases: `object`

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

Parameter

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

Rückgabotyp

[Path](#)

static transfer_action_copy(*source*, *destination*)

Parameter

- **source** ([Path](#))
- **destination** ([Path](#))

Rückgabotyp

None

static transfer_action_move(*source, destination*)**Parameter**

- **source** ([Path](#))
- **destination** ([Path](#))

Rückgabotyp

None

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

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

Rückgabotyp

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.

Parameter

- **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*.

Rückgabotyp[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](#).

Parameter

- **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.

Rückgabotyp[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.

Parameter

- **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.

Rückgabotyp

`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.

Parameter

- **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.

Parameter

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

```
_TStaticContent = str | bytes
```

```
_TContent
```

```
alias of str | bytes | Callable[[], 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).

Parameter

- **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 directo-  
ry 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.

Parameter

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

Rückgabotyp

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`.

Parameter

- **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.

Parameter

- **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()`.

Parameter

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

Rückgabetyp

`None`

`annize.i18n.translation_providers()`

Return all translation providers.

See also `add_translation_provider()`.

Rückgabetyp

`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.

Parameter

- **string_name** (`str`) – The string name.
- **culture** (`Culture` | `str` | `None`) – The culture.

Rückgabetyp

`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()`).

Parameter

string_name (`str`) – The string name.

Rückgabetyp**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).

Parameter**culture** (**Culture** | *str* | *None*) – The culture.**Rückgabetyp***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.

Parameter**culture** (**Culture**) – The culture.**Rückgabetyp***str* | *None***format**(**args*, ***kwargs*)

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

Parameter

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

Rückgabetyp**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()*.

Parameter**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.

Parameter**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.

Parameter

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

Rückgabotyp

`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`.

Parameter

text (`TrStr` / `str`) – The text.

Rückgabotyp

`TrStr`

```
class annize.i18n._FixedTrStr(text)
```

Bases: `TrStr`

Parameter

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.

Parameter

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()`.

Parameter

- **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).

Parameter

- **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](#).

Rückgabetypp

[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.

```

    Rückgabetypp
        Iterable[Culture]
__best_system_locale()

    Rückgabetypp
        str
__current_system_locale_setup()

    Rückgabetypp
        _TSystemLocaleSetup
__set_system_locale_setup()

    Parameter
        system_locale_setup(_TSystemLocaleSetup)

    Rückgabetypp
        None
__set_env__var(value)

    Parameter
        • key(str)
        • value(str | None)

    Rückgabetypp
        None
class _TSystemLocaleSetup(LC_ALL: str | None, LANGUAGE: str | None)
    Bases: object
    Parameter
        • 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().
    Parameter
        • 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\(\)](#).

Parameter

- **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).

Parameter

culture ([Culture](#) / *str* / None) – The culture spec.

Rückgabetyp

[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.

Rückgabetyp

[Culture](#)

`annize.i18n._annize_user_interaction_culture()`

Rückgabetyp

[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".

Parameter

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

Rückgabetyp

[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.

Parameter

- **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*)

Parameter

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`

Parameter

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

parameter_name: `str`

explicit_only: `bool` | `None` = `None`

with_updates(*oconfig*)

Parameter

oconfig (`ParameterConfig`)

Rückgabetyt

`ParameterConfig`

```
static create_object(call_type, args, kwargs)
```

Parameter

- **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)
```

Parameter

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

Rückgabetyp

[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)
```

Parameter

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

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

Bases: [TypeError](#)

Parameter

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](#)

Parameter

- **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)
    Parameter
        obj (object)
    Rückgabety
        bool
matches_type(type_)
    Parameter
        type_ (type)
    Rückgabety
        bool
matches_inner_type(type_)
    Parameter
        type_ (type)
    Rückgabety
        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)
    Parameter
        s (str)
    Rückgabety
        Any
class annize.object.parameter_info.ListParameterInfo(name, is_optional, innertypeinfo)
    Bases: ParameterInfo
    Parameter
        • 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
    Parameter
        • name (str)

```

- **is_optional** (*bool*)

property resolved_type

matches_object (*obj*)

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

Parameter

as_optional (*bool*)

Rückgabetyp

[ParameterInfo](#)

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

Parameter

for_callable (*Callable*)

Rückgabetyp

`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\(\)](#).

Parameter

- **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.

Parameter

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

Rückgabetyp

[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.

Parameter

- **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!

Rückgabetyp

[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\(\)](#).

Parameter

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.

Parameter

- **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.

Parameter

- **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.

Parameter

- **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.

Parameter

- **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\(\)`](#).

Parameter

- **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.

Parameter

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

__changed__helpers(*event*)

Parameter

event ([ChangeEvent](#))

_changed__child_added(*child_node*, *child_position*)

Parameter

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

_changed__child_removed(*child_node*, *child_position*)

Parameter

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

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

Parameter

- **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.

Parameter

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

Rückgabety

None

append_child(*node*)

Append a new child node.

Parameter

node ([Node](#)) – The node to append.

Rückgabotyp

None

remove_child(*node*)

Remove a child node.

If that node is not a child node, it raises a `ValueError`.**Parameter****node** (`Node`) – The node to remove.**Rückgabotyp**

None

abstractmethod classmethod **_allowed_child_types**()

Return a list of node types that this node type allows to have as child nodes.

Rückgabotyp`Iterable[type[Node]]`**clone**(*with_children=True*, *with_marshallers=False*)**Parameter**

- **with_children** (`bool`)
- **with_marshallers** (`bool`)

Rückgabotyp`Node`**description**(***, *with_children=True*, *multiline=True*)**Parameter**

- **with_children** (`bool`)
- **multiline** (`bool`)

Rückgabotyp`str`**__description**(*indent*, *with_children*, *multiline*)**Parameter**

- **indent** (`int`)
- **with_children** (`bool`)
- **multiline** (`bool`)

Rückgabotyp`str`**abstractmethod** **_str_helper**()**Rückgabotyp**`Iterable[str]`**_abc_impl** = `<_abc._abc_data object>`

class annize.project.ProjectNodeBases: [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.

Parameter

- **i** – The position.
- **node** – The node to insert.

__changed_handler(*event*)**Parameter****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.

Parameter

- **since** (*int*) – The timestamp where to start with returning changes (inclusive).
- **until** (*int*) – The timestamp where to stop with return changes (non-inclusive).

Rückgabotyplist[[ChangeEvent](#)]**__compacted_changelist**()**Parameter****events** (list[[ChangeEvent](#)])**Rückgabotyp**list[[ChangeEvent](#) | None]**undo_changes**(*since*)

Undo all changes that happened to the project since a given point in time.

Parameter**since** (*int*) – The timestamp where to start with undoing changes (inclusive).**Rückgabotyp**

None

static load(*path*)**Parameter****path** (*str* | *Path*)

Rückgabotyp[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, marshaler)`

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.

Parameter

- **path** (*str* | *Path*)
- **marshaler** (*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)
```

Parameter

- **obj** (*Any*)
- **maxlen** (*int*)

Rückgabety

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

Parameter

feature_name (str)

exception annize.project.BadStructureError(message)

Bases: ValueError

Parameter

message (str)

exception annize.project.MaterializerError(message)

Bases: TypeError

Parameter

message (str)

exception annize.project.ParserError(message)

Bases: ValueError

Parsing error like bad input xml.

Parameter

message (str)

exception annize.project.UnresolvableReferenceError(reference_key)

Bases: [MaterializerError](#)

Parameter

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*)

Parameter

change (*TODO*)

Rückgabetyp

None

_abc_impl = <_abc._abc_data object>

classmethod parse_file(*path*)

Read the given file and return a project file node for it.

Parameter

path (*str* | *Path*) – The file to parse.

Rückgabetyp

[FileNode](#)

_abc_impl = <_abc._abc_data object>

annize.project.file_formats.**register_file_format**(*format_name*)

Parameter

format_name (*str*)

annize.project.file_formats.**get_format**(*format_name*)

Parameter

format_name (*str*)

Rückgabetyp

[FileFormat](#) | None

annize.project.file_formats.**parse**(*path*)

Parameter

path (*str* | *Path*)

Rückgabetyp

[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.
        Parameter
            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 marshaler
        in_file(fpath, marshaler)
            Parameter
                fpath (str | Path)
        in_node(xnode)
        static _Context__nodeshort(xnode, maxlen=100)
            Parameter
                xnode (Element)
    class _TagParts(name, namespace='')
        Bases: object
        Parameter
            namespace (str)
    ATTRIBUTE_COMMAND_START = '~'
    ATTRIBUTE_COMMAND_END = '~'
    classmethod escape_attribute_string(txt)
        Parameter
            txt (str)
        Rückgabetyt
            str
    parse_file(fpath)
        Parameter
            fpath (str | Path)
        Rückgabetyt
            FileNode
    classmethod _XmlParser__interpret_attribute_string(txt)
        Parameter
            txt (str)
        Rückgabetyt
            Tuple[bool, str]

```

classmethod `_XmlParser__parse_attr`(*key*, *value*)

Parameter

- **key** (*str*)
- **value** (*str*)

Rückgabety

Node

_XmlParser__parse_child(*node*, *xnode*)

Parameter

- **node** (*Node*)
- **xnode** (*Element*)

Rückgabety

Tuple[*Node*, *Element*]

_XmlParser__parse_children(*node*, *xparent*)

Parameter

- **node** (*Node*)
- **xparent** (*Element*)

_XmlParser__parse_tag(*node*, *argname*, *callname*, *feature*, *xnode*)

Parameter

node (*Node*)

class `annize.project.file_formats.xml.Marshaler`

Bases: *Marshaler*

class `XmlDocumentLocation`(*element*: <cyfunction Element at 0x7fd05e236cf0>, *attr_name*: *str* = '')

Bases: *object*

Parameter

- **element** (*Element*)
- **attr_name** (*str*)

element: *Element*

attr_name: *str* = ''

add_change(*change*)

add_element(*node*, *xelem*)

Parameter

- **node** (*Node*)
- **xelem** (*Element*)

add_element_attr(*node*, *xelem*, *attrname*)

Parameter

- **node** (*Node*)

- **xelem** (*Element*)
- **attrname** (*str*)

_abc_impl = <_abc._abc_data object>

add_element_tree(*node, xtree*)

Parameter

- **node** (*Node*)
- **xtree** (*ElementTree*)

save_filenode_to_file(*node*)

Parameter

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

Parameter

- **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*)

Parameter

node (*Node*)

Rückgabotyp

list[Any] | None

erroneous_nodes()

Rückgabotyp

list[Node]

errors_for_node(*node*)

Parameter

node (*Any*)

Rückgabotyp

list[Exception]

```
annize.project.materializer.materialize(project, *, feature_loader=None)
```

Parameter

- **project** ([ProjectNode](#))
- **feature_loader** ([FeatureLoader](#) | *None*)

Rückgabetyp

[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.

Parameter

nodemat ([NodeMaterialization](#)) – The node materialization for the current node.

Rückgabetyp

[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).

Parameter

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.

Parameter

nodemat – The node materialization for the current node.

_abc_impl = <_abc._abc_data object>

class annize.project.materializer.behaviors.argument.**AssociateArgumentNodeBehavior**(*association*)

Bases: *Behavior*

Parameter

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.

Parameter

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.

Parameter

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.

Parameter

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.

Parameter

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.

Parameter

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

Parameter

- **materializer** ([ProjectMaterializer](#))
- **node** ([Node](#))
- **store** (*dict*)

property node: [Node](#)

set_materialized_result(*resultlist*)

set_problems(*problems*)

Parameter

problems (*Iterable[Exception]*)

get_materialized_children_tuples()

get_materialized_children()

Rückgabety

Iterable[Any]

try_get_materialization_for_node(*node*)

Parameter

node ([Node](#))

property has_result

property result

property problems: *list[Exception]*

class annize.project.materializer.core.**ProjectMaterializer**(*node, *, behaviors*)

Bases: object

Parameter

- **node** ([Node](#))
- **behaviors** (*Iterable*[[annize.project.materializer.behaviors.Behavior](#)])

__materialization_for_node(*node*, *store*)

Parameter

- **node** ([Node](#))
- **store** (*dict*)

Rückgabetyp

[NodeMaterialization](#)

__materialize(*node*, *store*)

Parameter

- **node** ([Node](#))
- **store** (*dict*)

Rückgabetyp

None

__materialize_hlp_childobjs(*node*, *store*)

Parameter

- **node** ([Node](#))
- **store** (*dict*)

Rückgabetyp

list[*Tuple*[[Node](#), *list*[*Any*]]]

__get_erroneous_nodes(*materializationstore*, *old_erroneous_nodes*)

get_materialized()

Rückgabetyp

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](#)

Parameter

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*)

Parameter

topnode ([Node](#))

Rückgabotyp[Node](#)

```
annize.project.materializer.preprocessors.normalize_blockscopes(topnode)
```

Parameter**topnode** ([Node](#))**Rückgabotyp**[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)
```

Parameter**name** (*str*)**Rückgabotyp***Any* | *None*

```
abstractmethod get_all_available_feature_names()
```

Rückgabotyp*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)
```

Parameter**package_name** (*str*)**Rückgabotyp***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.

Parameter

feature_loader ([FeatureLoader](#))

class **ArgumentMatchings**(*all_matchings*)

Bases: object

Parameter

all_matchings (*list*[[ArgumentMatching](#)])

class **ArgumentMatching**(*arg_name, nodes, allows_multiple_args*)

Bases: object

Parameter

- **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*)

Parameter

arg_name (*str*)

Rückgabety

[ArgumentMatching](#)[*None*]

all()

Rückgabety

list[[ArgumentMatching](#)]

class **TypeInfo**(*feature, typename, ctype*)

Bases: object

Parameter

- **feature** (*str* | *None*)
- **typename** (*str*)
- **ctype** (*type*)

property feature: *str* | *None*

property typename: *str*

property type: *type*

match_arguments(*node*)

Parameter

node ([Node](#))

Rückgabotyp

[ArgumentMatchings](#)

match_node(*node*)

Parameter

node ([Node](#))

Rückgabotyp

[ArgumentMatching](#) | None

get_all_types()

Rückgabotyp

list[[TypeInfo](#)]

get_types_for_argument(*node*, *argname*)

Parameter

- **node** ([Node](#))
- **argname** (*str*)

Rückgabotyp

list[[TypeInfo](#)]

get_project_node(*node*)

Parameter

node ([Node](#))

Rückgabotyp

[ProjectNode](#) | None

get_node_by_name(*subtree*, *name*)

Parameter

- **subtree** ([ProjectNode](#))
- **name** (*str*)

Rückgabotyp

[Node](#) | None

resolve_reference_node(*node*, *, *deep=True*)

Parameter

- **node** ([Node](#))
- **deep** (*bool*)

Rückgabotyp

[ArgumentNode](#) | None

`__get_node_materialtype(node)`

Parameter

node ([Node](#))

Rückgabety

`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\(\)](#).

Parameter

project_path (`str` | `Path`) – A path to somewhere inside an Annize project.

Rückgabety

[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.

Parameter

project_path (`str` | `Path`) – A path into the Annize project.

Rückgabety

`Path` | `None`

`annize.project.loader.project_root_directory(annize_config_rootpath)`

Parameter

annize_config_rootpath (`str` | `Path`)

Rückgabety

`Path`

annize.ui package

`annize.ui.app(app_name, **kwargs)`

Parameter

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*)**Parameter**

- **message** (*str*)
- **answers** (*list[str]*)
- **config_key** (*str | None*)

Rückgabotyp

int

abstractmethod input_dialog(*question, suggested_answer, config_key*)**Parameter**

- **question** (*str*)
- **suggested_answer** (*str*)
- **config_key** (*str | None*)

Rückgabotyp

str | None

abstractmethod choice_dialog(*question, choices, config_key*)**Parameter**

- **question** (*str*)
- **choices** (*list[str]*)
- **config_key** (*str | None*)

Rückgabotyp

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*)**Parameter****context** ([RunContext](#))

Rückgabotyp

list[Tuple[int, UserFeedbackController]]

annize.user_feedback._controllers_for_context(context)

Parameter**context** (RunContext)**Rückgabotyp**

list[UserFeedbackController]

annize.user_feedback._add_controller_to_context(*, controller, context, priority_index=0)

Parameter

- **controller** (UserFeedbackController)
- **context** (RunContext)
- **priority_index** (int)

Rückgabotyp

None

annize.user_feedback.message_dialog(message, answers, *, config_key=None)

Parameter

- **message** (TrStr | str)
- **answers** (Iterable[TrStr | str])
- **config_key** (str | None)

Rückgabotyp

int

annize.user_feedback.input_dialog(message, *, suggested_answer, config_key=None)

Parameter

- **message** (TrStr | str)
- **suggested_answer** (TrStr | str)
- **config_key** (str | None)

Rückgabotyp

str | None

annize.user_feedback.choice_dialog(message, choices, *, config_key=None)

Parameter

- **message** (TrStr | str)
- **choices** (Iterable[TrStr | str])
- **config_key** (str | None)

Rückgabotyp

int | None

Submodules

annize.user_feedback.static module

class annize.user_feedback.static.StaticUserFeedbackController(*answers*)

Bases: *UserFeedbackController*

Parameter

answers (*dict[str, Any]*)

add_answer(*config_key, value*)

Parameter

- **config_key** (*str*)
- **value** (*Any*)

Rückgabety

None

__get_answer(*config_key*)

Parameter

config_key (*str*)

Rückgabety

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*)

Parameter

only_documentation (*bool*)

Rückgabety

ArgumentParser

class annize.annize_cli.Commands(*project, with_answers_from_json_file, with_answers_from_json_string, with_answer, **_*)

Bases: *object*

Parameter

- **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)
```

Parameter

- `destination` (*dict*)
- `with_answers_from_json_files` (*Iterable*[*str*])

```
classmethod __answers_from_json_strings(destination, with_answers_from_json_strings)
```

Parameter

- `destination` (*dict*)
- `with_answers_from_json_strings` (*Iterable*[*str*])

```
classmethod __answers_from_single_answers(destination, with_answers)
```

Parameter

- `destination` (*dict*)
- `with_answers` (*Iterable*[*Tuple*[*str*, *str*]])

```
do(task_name, **_)
```

Parameter

```
task_name (str)
```

```
studio(**_)
```


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, 15
- annize.data.version, 15
- annize.features, 18
- annize.features.authors, 72
- annize.features.base, 73
- 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, 43
- 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, 51
- 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, 46
- annize.features.documentation.sphinx.python, 53
- 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, 56
- 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, 61
- 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, 68
- annize.features.injections, 68
- annize.features.injections.common, 68
- annize.features.injections.python, 69
- annize.features.licensing, 75
- annize.features.media_galleries, 77
- annize.features.task, 77
- annize.features.testing, 69

- annize.features.testing.common, 69
- annize.features.testing.pylint, 70
- annize.features.testing.pytest, 70
- annize.features.version, 78
- 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, 84
- 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, 121
- annize.project.file_formats, 112
- annize.project.file_formats.xml, 112
- annize.project.inspector, 122
- annize.project.loader, 124
- annize.project.materializer, 115
- annize.project.materializer.behaviors, 116
- annize.project.materializer.behaviors.argument, 116
- annize.project.materializer.behaviors.basket, 117
- annize.project.materializer.behaviors.block, 118
- annize.project.materializer.behaviors.feature_unavailable, 118
- annize.project.materializer.behaviors.reference, 118
- annize.project.materializer.core, 119
- annize.project.materializer.preprocessors, 120
- annize.ui, 124
- annize.ui.apps, 124
- annize.user_feedback, 126
- annize.user_feedback.static, 128

Sonderzeichen

<code>_ABOUT_NAME</code>	(Attribut von <code>annize.features.documentation.sphinx.common.ReadmeDocument</code>), 50	(statische Methode von <code>annize.object.controller._CreateObjectHelper</code>), 100
<code>_COMMON_NAMESPACE_POSTFIX</code>	(Attribut von <code>annize.project.feature_loader.DefaultFeatureLoader</code>), 121	<code>_CultureFence</code> (Klasse in <code>annize.i18n</code>), 97
<code>_Context__nodeshort()</code>	(statische Methode von <code>annize.project.file_formats.xml._XmlParser._Context</code>), 113	<code>_FEATURES_NAMESPACE</code> (Attribut von <code>annize.project.feature_loader.DefaultFeatureLoader</code>), 121
<code>_CreateObjectHelper</code>	(Klasse in <code>annize.object.controller</code>), 99	<code>_FixedTrStr</code> (Klasse in <code>annize.i18n</code>), 95
<code>_CreateObjectHelper.ParameterConfig</code>	(Klasse in <code>annize.object.controller</code>), 99	<code>_IS_TOPLEVEL_OBJECT__METADATA_KEY</code> (Attribut von <code>annize.flow.run_context.RunContext</code>), 79
<code>_CreateObjectHelper__convert_kwargs_from_string()</code>	(statische Methode von <code>annize.object.controller._CreateObjectHelper</code>), 100	<code>_ProjectDefinedTranslationProvider</code> (Klasse in <code>annize.features.i18n.common</code>), 66
<code>_CreateObjectHelper__determine_matching_keywords_for_arg()</code>	(statische Methode von <code>annize.object.controller._CreateObjectHelper</code>), 100	<code>_ProjectDefinedTranslationProvider__translations_for_string()</code> (Methode von <code>annize.features.i18n.common._ProjectDefinedTranslationProvider</code>), 67
<code>_CreateObjectHelper__fill_empty_lists()</code>	(statische Methode von <code>annize.object.controller._CreateObjectHelper</code>), 100	<code>_S_CHANGE</code> (Attribut von <code>annize.features.changelog.common.ByVersionControlSystemCommitM</code>), 19
<code>_CreateObjectHelper__fill_unspecified_optionals()</code>	(statische Methode von <code>annize.object.controller._CreateObjectHelper</code>), 100	<code>_S_LABEL</code> (Attribut von <code>annize.features.changelog.common.ByVersionControlSystemCommitM</code>), 19
<code>_CreateObjectHelper__get_parameter_config()</code>	(statische Methode von <code>annize.object.controller._CreateObjectHelper</code>), 100	<code>_TContent</code> (Attribut von <code>annize.fs.ext.DynamicFile</code>), 91
<code>_CreateObjectHelper__put_item_into_kwargs()</code>	(statische Methode von <code>annize.object.controller._CreateObjectHelper</code>), 100	<code>_TStaticContent</code> (Attribut von <code>annize.fs.ext.DynamicFile</code>), 90
<code>_CreateObjectHelper__shift_args_to_kwargs()</code>		<code>_TransferHelper__transfer_piece()</code> (statische Methode von <code>annize.fs.Path._TransferHelper</code>), 89
		<code>_XmlParser</code> (Klasse in <code>annize.project.file_formats.xml</code>), 113
		<code>_XmlParser._Context</code> (Klasse in <code>annize.project.file_formats.xml</code>), 113
		<code>_XmlParser._TagParts</code> (Klasse in <code>annize.project.file_formats.xml</code>), 113
		<code>_XmlParser__interpret_attribute_string()</code> (Klassenmethode von <code>annize.project.file_formats.xml._XmlParser</code>), 113
		<code>_XmlParser__parse_attr()</code> (Klassenmethode von <code>annize.project.file_formats.xml._XmlParser</code>), 113

- 114
 __XmlParser__parse_child() (Methode von annize.project.file_formats.xml._XmlParser), 114
 __XmlParser__parse_children() (Methode von annize.project.file_formats.xml._XmlParser), 114
 __XmlParser__parse_tag() (Methode von annize.project.file_formats.xml._XmlParser), 114
 __answers_from_json_files() (Klassenmethode von annize.annize_cli.Commands), 129
 __answers_from_json_strings() (Klassenmethode von annize.annize_cli.Commands), 129
 __answers_from_single_answers() (Klassenmethode von annize.annize_cli.Commands), 129
 __best_system_locale() (Methode von annize.i18n.Culture), 97
 __call_git() (Methode von annize.features.version_control.git.VersionControlSystem), 71
 __changed_helpers() (Methode von annize.project.Node), 105
 __changed_handler() (Methode von annize.project.ProjectNode), 107
 __cleanup() (Methode von annize.fs.ext.FreshTempDirectory), 90
 __compacted_changelist() (Methode von annize.project.ProjectNode), 107
 __context_catchexceptions() (Methode von annize.project.materializer.behaviors.feature_unavailable.FeatureUnavailableBehavior), 118
 __context_skipnode_featureignorelist() (Methode von annize.project.materializer.behaviors.feature_unavailable.FeatureUnavailableBehavior), 118
 __current_system_locale_setup() (Methode von annize.i18n.Culture), 97
 __description() (Methode von annize.project.Node), 106
 __do_run() (Methode von annize.flow.runner.Runner), 84
 __does_exclude() (Methode von annize.features.files.common.Exclude), 58
 __effversion() (Methode von annize.features.version_control.common.BuildVersion), 71
 __files_from_package_store() (Methode von annize.features.distributables.common.Group), 22
 __find_feature_modules_in_package() (Methode von annize.project.feature_loader.DefaultFeatureLoader), 121
 __generate_geninfo_to_confpy() (Methode von annize.features.documentation.sphinx.common.Document), 47
 __generate_packagelist() (Methode von annize.features.homepage.sections.download.Section), 62
 __generate_pre_post_proc() (Methode von annize.features.homepage.common.Homepage), 65
 __generate_prepare_annizeicons() (Methode von annize.features.documentation.sphinx.common.Document), 47
 __generate_prepare_shortsnippets() (Methode von annize.features.documentation.sphinx.common.Document), 47
 __generate_section() (Methode von annize.features.homepage.common.Homepage), 65
 __generate_set_culture() (Methode von annize.features.documentation.sphinx.common.Document), 47
 __generate_set_misc() (Methode von annize.features.documentation.sphinx.common.Document), 47
 __generate_set_version_and_release() (Methode von annize.features.documentation.sphinx.common.Document), 47
 __get_answer() (Methode von annize.features.documentation.sphinx.common.StaticUserFeedbackController), 128
 __get_erroneous_nodes() (Methode von annize.project.materializer.core.ProjectMaterializer), 120
 __get_inner_generateinfo() (Methode von annize.features.documentation.sphinx.common.CompositeDocument), 47
 __get_node_materialtype() (Methode von annize.project.inspector.Inspector), 123
 __get_refgeninfo() (Methode von annize.features.documentation.sphinx.common.ApiReferenceDocument), 48
 __get_variant() (Methode von annize.features.documentation.sphinx.common.RstDocument), 49
 __info() (Methode von annize.features.documentation.sphinx.doxygen_compat.DoxygenSupport), 52
 __initial_cwd (Attribut von annize.annize_cli.Commands), 129
 __jsfiles() (Methode von annize.features.documentation.sphinx.javascript.JavaScriptApiReference), 52
 __long_str() (Methode von annize.data.unique.UniqueId), 15

__materialization_for_node() (Methode von anni-ze.project.materializer.core.ProjectMaterializer), 120	ze.data.version.OptionalVersionPatternSegment), 17
__materialize() (Methode von anni-ze.project.materializer.core.ProjectMaterializer), 120	_abc_impl (Attribut von anni-ze.data.version.SeparatorVersionPatternSegment), 16
__object_metadata_dict() (Methode von anni-ze.flow.run_context.RunContext), 81	_abc_impl (Attribut von anni-ze.features.distributables.common.PackageStore), 22
__object_raw_name() (Methode von anni-ze.flow.run_context.RunContext), 81	_abc_impl (Attribut von anni-ze.features.documentation.common.Document), 54
__package_store_name() (Methode von anni-ze.features.distributables.common.Group), 23	_abc_impl (Attribut von anni-ze.features.documentation.common.HtmlOutputSpec), 55
__patch_property_types_in_docstrings() (Methode von anni-ze.features.documentation.sphinx.python.Python3ApiReferenceLanguage), 53	_abc_impl (Attribut von anni-ze.features.documentation.common.OutputSpec), 55
__prepare_generate() (Methode von anni-ze.features.documentation.sphinx.doxygen_compat.DoxygenSupportApiReferenceLanguage), 52	_abc_impl (Attribut von anni-ze.features.documentation.common.PdfOutputSpec), 55
__put_object() (Methode von anni-ze.flow.run_context.RunContext), 81	_abc_impl (Attribut von anni-ze.features.documentation.common.PlaintextOutputSpec), 55
__scanjsfile() (Methode von anni-ze.features.documentation.sphinx.javascript.JavaScriptApiReferenceLanguage), 52	_abc_impl (Attribut von anni-ze.features.documentation.sphinx.common.AboutProjectDocument), 50
__set_env__var() (Methode von annize.i18n.Culture), 97	_abc_impl (Attribut von anni-ze.features.documentation.sphinx.common.ApiReferenceDocument), 49
__set_success_state() (Methode von anni-ze.flow.runner.Runner), 85	_abc_impl (Attribut von anni-ze.features.documentation.sphinx.common.ApiReferenceLanguage), 48
__set_system_locale_setup() (Methode von annize.i18n.Culture), 97	_abc_impl (Attribut von anni-ze.features.documentation.sphinx.common.ArgparseCommandLin), 49
__set_tasks() (Methode von anni-ze.flow.runner.Runner), 84	_abc_impl (Attribut von anni-ze.features.documentation.sphinx.common.CompositeDocument), 48
__shorten() (Methode von anni-ze.project.ScalarValueNode), 109	_abc_impl (Attribut von anni-ze.features.documentation.sphinx.common.Document), 47
__store_files_to_package_store() (Methode von annize.features.distributables.common.Group), 23	_abc_impl (Attribut von anni-ze.features.documentation.sphinx.common.ReadmeDocument), 50
__transfer_filter_for_exclude() (Methode von annize.features.files.common.Directory), 60	_abc_impl (Attribut von anni-ze.features.documentation.sphinx.common.RstDocument), 50
__uniqueid_counter (Attribut von anni-ze.data.unique.UniqueId), 15	_abc_impl (Attribut von anni-ze.features.documentation.sphinx.cpp.CppApiReferenceLanguage), 50
__uniqueid_lock (Attribut von anni-ze.data.unique.UniqueId), 15	_abc_impl (Attribut von anni-ze.features.documentation.sphinx.doxygen_compat.DoxygenSuppo), 52
_abc_impl (Attribut von anni-ze.data.version.AbstractVersionPatternSegment), 16	_abc_impl (Attribut von anni-ze.features.documentation.sphinx.common.RstDocument), 50
_abc_impl (Attribut von anni-ze.data.version.ConcatenatedVersionPatternSegment), 17	_abc_impl (Attribut von anni-ze.features.documentation.sphinx.cpp.CppApiReferenceLanguage), 50
_abc_impl (Attribut von anni-ze.data.version.NumericVersionPatternSegment), 16	_abc_impl (Attribut von anni-ze.features.documentation.sphinx.doxygen_compat.DoxygenSuppo), 52
_abc_impl (Attribut von anni-ze.data.version.NumericVersionPatternSegment), 16	_abc_impl (Attribut von anni-ze.features.documentation.sphinx.doxygen_compat.DoxygenSuppo), 52

ze.features.documentation.sphinx.javascript.JavaScriptReferenceLanguage, von anni-
 52 ze.features.i18n.common._ProjectDefinedTranslationProvider),
 _abc_impl (Attribut von anni- 67
 ze.features.documentation.sphinx.output.common.CatalogGenerator), (Attribut von anni-
 44 ze.features.injections.common.FilesystemContentInjection),
 _abc_impl (Attribut von anni- 68
 ze.features.documentation.sphinx.output.html.HtmlCatalogGenerator), (Attribut von anni-
 45 ze.features.injections.common.Injection),
 _abc_impl (Attribut von anni- 68
 ze.features.documentation.sphinx.output.html.HtmlCatalogGenerator), (Attribut von anni-
 45 ze.features.injections.python.ProjectInfoInjection),
 _abc_impl (Attribut von anni- 69
 ze.features.documentation.sphinx.output.pdf.PdfOutputGenerator), (Attribut von anni-
 45 ze.features.testing.common.Test), 69
 _abc_impl (Attribut von anni- _abc_impl (Attribut von anni-
 ze.features.documentation.sphinx.output.plaintext.PlaintextOutputGenerator), 69
 46 _abc_impl (Attribut von anni-
 _abc_impl (Attribut von anni- ze.features.testing.pylint.Test), 70
 ze.features.documentation.sphinx.python.Python3AppReferenceLanguage, (Attribut von anni-
 53 ze.features.testing.pytest.Test), 70
 _abc_impl (Attribut von anni- _abc_impl (Attribut von anni-
 ze.features.files.transfer.common.Endpoint), ze.features.version_control.common.VersionControlSystem),
 56 71
 _abc_impl (Attribut von anni- _abc_impl (Attribut von anni-
 ze.features.files.transfer.common.FsEndpoint), ze.features.version_control.git.VersionControlSystem),
 56 72
 _abc_impl (Attribut von anni- _abc_impl (Attribut von annize.flow.runner.Runner), 85
 ze.features.files.transfer.ssh.Endpoint), 57 _abc_impl (Attribut von anni-
 _abc_impl (Attribut von anni- ze.i18n.GettextTranslationProvider), 93
 ze.features.homepage.common.HomepageSection), _abc_impl (Attribut von annize.i18n.ProvidedTrStr), 94
 65 _abc_impl (Attribut von annize.i18n.TrStr), 94
 _abc_impl (Attribut von anni- _abc_impl (Attribut von anni-
 ze.features.homepage.sections.about.Section), ze.i18n.TranslationProvider), 92
 61 _abc_impl (Attribut von annize.i18n._FixedTrStr), 95
 _abc_impl (Attribut von anni- _abc_impl (Attribut von annize.project.ArgumentNode),
 ze.features.homepage.sections.changelog.Section), 108
 61 _abc_impl (Attribut von annize.project.BlockNode), 110
 _abc_impl (Attribut von anni- _abc_impl (Attribut von anni-
 ze.features.homepage.sections.documentation.Section), ze.project.BlockNodeWithScope), 110
 61 _abc_impl (Attribut von annize.project.FileNode), 108
 _abc_impl (Attribut von anni- _abc_impl (Attribut von annize.project.Node), 106
 ze.features.homepage.sections.download.Section), _abc_impl (Attribut von annize.project.ObjectNode),
 62 109
 _abc_impl (Attribut von anni- _abc_impl (Attribut von anni-
 ze.features.homepage.sections.gallery.Section), ze.project.OnFeatureUnavailableNode), 111
 62 _abc_impl (Attribut von annize.project.ProjectNode),
 _abc_impl (Attribut von anni- 108
 ze.features.homepage.sections.imprint.Section), _abc_impl (Attribut von annize.project.ReferenceNode),
 63 110
 _abc_impl (Attribut von anni- _abc_impl (Attribut von anni-
 ze.features.homepage.sections.license.Section), ze.project.ScalarValueNode), 109
 63 _abc_impl (Attribut von anni-
 _abc_impl (Attribut von anni- ze.project.feature_loader.DefaultFeatureLoader),
 ze.features.i18n.common.String), 67 121

_abc_impl (Attribut von annize.project.feature_loader.FeatureLoader), 121
 _abc_impl (Attribut von annize.project.file_formats.FileFormat), 112
 _abc_impl (Attribut von annize.project.file_formats.FileFormat.Marshaler), 112
 _abc_impl (Attribut von annize.project.file_formats.xml.Marshaler), 115
 _abc_impl (Attribut von annize.project.file_formats.xml.XmlFileFormat), 113
 _abc_impl (Attribut von annize.project.materializer.behaviors.Behavior), 116
 _abc_impl (Attribut von annize.project.materializer.behaviors.argument.ArgumentBehavior), 117
 _abc_impl (Attribut von annize.project.materializer.behaviors.argument.AssociateArgumentBehavior), 117
 _abc_impl (Attribut von annize.project.materializer.behaviors.basket.BasketBehavior), 117
 _abc_impl (Attribut von annize.project.materializer.behaviors.block.BlockBehavior), 118
 _abc_impl (Attribut von annize.project.materializer.behaviors.feature_unavailable.FeatureUnavailableBehavior), 118
 _abc_impl (Attribut von annize.project.materializer.behaviors.reference.ReferenceBehavior), 119
 _abc_impl (Attribut von annize.user_feedback.UserFeedbackController), 126
 _abc_impl (Attribut von annize.user_feedback.static.StaticUserFeedbackController), 128
 _add_controller_to_context() (im Modul annize.user_feedback), 127
 _allowed_child_types() (Klassenmethode von annize.project.BlockNode), 110
 _allowed_child_types() (Klassenmethode von annize.project.FileNode), 108
 _allowed_child_types() (Klassenmethode von annize.project.Node), 106
 _allowed_child_types() (Klassenmethode von annize.project.ObjectNode), 109
 _allowed_child_types() (Klassenmethode von annize.project.ProjectNode), 108
 _allowed_child_types() (Klassenmethode von annize.project.ReferenceNode), 110
 _allowed_child_types() (Klassenmethode von annize.project.ScalarValueNode), 109
 _annize_user_interaction_culture() (im Modul annize.i18n), 98
 _append_section() (Methode von annize.features.homepage.common.Homepage), 65
 _changed__child_added() (Methode von annize.project.Node), 105
 _changed__child_removed() (Methode von annize.project.Node), 105
 _changed__property_changed() (Methode von annize.project.Node), 105
 _controllers_for_context() (im Modul annize.user_feedback), 127
 _controllers_tuples_for_context() (im Modul annize.user_feedback), 126
 _debian_category() (im Modul annize.features.distributables.debian), 24
 _debian_section() (im Modul annize.features.distributables.debian), 27
 _description_for_mediafile() (Methode von annize.features.media_galleries.Gallery), 77
 _dispatch() (Methode von annize.flow.runner.Runner), 84
 _generate_sources() (Methode von annize.features.documentation.sphinx.common.AboutProjectDocument), 50
 _generate_sources() (Methode von annize.features.documentation.sphinx.common.ApiReferenceDocument), 48
 _generate_sources() (Methode von annize.features.documentation.sphinx.common.ArgparseCommandLine), 49
 _generate_sources() (Methode von annize.features.documentation.sphinx.common.CompositeDocument), 48
 _generate_sources() (Methode von annize.features.documentation.sphinx.common.Document), 47
 _generate_sources() (Methode von annize.features.documentation.sphinx.common.RstDocument), 50
 _get_data() (im Modul annize.features.base), 74
 _get_type_info() (im Modul annize.object.parameter_info), 102
 _last_resort_culture (in Modul annize.i18n), 98
 _license() (im Modul annize.features.licensing), 75
 _materialize_hlp_childobjs() (Methode von annize.project.materializer.core.ProjectMaterializer), 120
 _mkpackage() (Klassenmethode von annize.features.distributables.debian.Package), 33

<code>_mkpackage()</code> (Klassenmethode von <code>annize.features.distributables.flatpak.FlatpakImage</code>), 39	<code>senmethode</code> von <code>annize.features.distributables.python_wheel.Package</code>), 42
<code>_mkpackage()</code> (Klassenmethode von <code>annize.features.distributables.python_wheel.Package</code>), 41	<code>_mkpackage_mkmanifestin()</code> (Klassenmethode von <code>annize.features.distributables.python_wheel.Package</code>), 43
<code>_mkpackage_applysource()</code> (Klassenmethode von <code>annize.features.distributables.flatpak.FlatpakImage</code>), 38	<code>_mkpackage_mkmenuentries()</code> (Klassenmethode von <code>annize.features.distributables.debian.Package</code>), 33
<code>_mkpackage_bdist_wheel()</code> (Klassenmethode von <code>annize.features.distributables.python_wheel.Package</code>), 42	<code>_mkpackage_mkprepostcmds()</code> (Klassenmethode von <code>annize.features.distributables.debian.Package</code>), 34
<code>_mkpackage_correctbuildsourcepermissions()</code> (Klassenmethode von <code>annize.features.distributables.debian.Package</code>), 34	<code>_mkpackage_mkservices()</code> (Klassenmethode von <code>annize.features.distributables.debian.Package</code>), 33
<code>_mkpackage_determinesize()</code> (Klassenmethode von <code>annize.features.distributables.debian.Package</code>), 34	<code>_mkpackage_mksetuppyconf()</code> (Klassenmethode von <code>annize.features.distributables.python_wheel.Package</code>), 42
<code>_mkpackage_dpkg()</code> (Klassenmethode von <code>annize.features.distributables.debian.Package</code>), 34	<code>_mkpackage_prepareinfos()</code> (Klassenmethode von <code>annize.features.distributables.debian.Package</code>), 33
<code>_mkpackage_flatpak_build_export()</code> (Klassenmethode von <code>annize.features.distributables.flatpak.FlatpakImage</code>), 39	<code>_mkpackage_prepareinfos()</code> (Klassenmethode von <code>annize.features.distributables.flatpak.FlatpakImage</code>), 38
<code>_mkpackage_flatpak_build_finish()</code> (Klassenmethode von <code>annize.features.distributables.flatpak.FlatpakImage</code>), 39	<code>_mkpackage_prepareinfos()</code> (Klassenmethode von <code>annize.features.distributables.python_wheel.Package</code>), 42
<code>_mkpackage_flatpak_build_init()</code> (Klassenmethode von <code>annize.features.distributables.flatpak.FlatpakImage</code>), 38	<code>_mkpackage_setuppyconf_classifiers()</code> (Klassenmethode von <code>annize.features.distributables.python_wheel.Package</code>), 42
<code>_mkpackage_mkchangelog()</code> (Klassenmethode von <code>annize.features.distributables.debian.Package</code>), 33	<code>_mkpackage_setuppyconf_install_requires()</code> (Klassenmethode von <code>annize.features.distributables.python_wheel.Package</code>), 42
<code>_mkpackage_mkcopyright()</code> (Klassenmethode von <code>annize.features.distributables.debian.Package</code>), 33	<code>_mkpackage_setuppyconf_prepare()</code> (Klassenmethode von <code>annize.features.distributables.python_wheel.Package</code>), 42
<code>_mkpackage_mkdebianconffilesfile()</code> (Klassenmethode von <code>annize.features.distributables.debian.Package</code>), 34	<code>_mkpackage_share()</code> (Klassenmethode von <code>annize.features.distributables.flatpak.FlatpakImage</code>), 39
<code>_mkpackage_mkdebiancontrolfile()</code> (Klassenmethode von <code>annize.features.distributables.debian.Package</code>), 34	<code>_name_anon()</code> (Methode von <code>annize.data.version.AbstractVersionPatternSegment</code>), 15
<code>_mkpackage_mkexeclinks()</code> (Klassenmethode von <code>annize.features.distributables.debian.Package</code>), 33	<code>_name_anon()</code> (Methode von <code>annize.data.version.ConcatenatedVersionPatternSegment</code>), 17
<code>_mkpackage_mkexeclinks()</code> (Klassenmethode von <code>annize.features.distributables.debian.Package</code>), 33	<code>_name_anon()</code> (Methode von <code>annize.data.version.NumericVersionPatternSegment</code>), 16

_name_anon() (Methode von annize.data.version.OptionalVersionPatternSegment), 17
 _node_clone_link() (im Modul annize.project.materializer), 116
 _path() (Methode von annize.features.distributables.debian.Package), 31
 _path() (Methode von annize.features.distributables.flatpak.FlatpakImage), 37
 _path() (Methode von annize.features.distributables.flatpak.FlatpakrefFile), 37
 _path() (Methode von annize.features.distributables.flatpak.GpgFile), 37
 _path() (Methode von annize.features.distributables.python_wheel.Package), 40
 _path() (Methode von annize.features.distributables.tar.Package), 43
 _path() (Methode von annize.features.documentation.common.GeneratedDocument), 55
 _path() (Methode von annize.features.files.common.Directory), 60
 _path() (Methode von annize.features.files.common.FsEntry), 57
 _path() (Methode von annize.features.files.common.MachineRootDirectory), 60
 _path() (Methode von annize.features.files.common.ProjectDirectory), 60
 _path() (Methode von annize.features.homepage.common.GeneratedHomepage), 66
 _path() (Methode von annize.fs.Path), 86
 _path() (Methode von annize.fs.ext.DynamicFile), 91
 _set_entry_path() (Methode von annize.features.documentation.common.DocumentGenerateResult), 54
 _str_helper() (Methode von annize.project.ArgumentNode), 108
 _str_helper() (Methode von annize.project.BlockNode), 110
 _str_helper() (Methode von annize.project.BlockNodeWithScope), 110
 _str_helper() (Methode von annize.project.FileNode), 108
 _str_helper() (Methode von annize.project.Node), 106
 _str_helper() (Methode von annize.project.ObjectNode), 109
 _str_helper() (Methode von annize.project.OnFeatureUnavailableNode), 111
 _str_helper() (Methode von annize.project.ProjectNode), 108
 _str_helper() (Methode von annize.project.ReferenceNode), 110
 _str_helper() (Methode von annize.project.ScalarValueNode), 109
 _str_to_val() (Methode von annize.data.version.AbstractVersionPatternSegment), 15
 _str_to_val() (Methode von annize.data.version.NumericVersionPatternSegment), 16
 _to_dict() (Methode von annize.features.documentation.sphinx.common.Document.GenerateIndex), 47
 _translate_from_clone() (im Modul annize.project.materializer), 116

A

AboutProjectDocument (Klasse in annize.features.documentation.sphinx.common), 50
 AbstractVersionPatternSegment (Klasse in annize.data.version), 15
 access_filesystem() (Methode von annize.features.files.transfer.common.Endpoint), 56
 access_filesystem() (Methode von annize.features.files.transfer.common.FsEndpoint), 56
 access_filesystem() (Methode von annize.features.files.transfer.ssh.Endpoint), 57
 add_answer() (Methode von annize.user_feedback.static.StaticUserFeedbackController), 128
 add_change() (Methode von annize.project.file_formats.FileFormat.Marshaler), 112
 add_change() (Methode von annize.project.file_formats.xml.Marshaler), 114
 add_change_handler() (Methode von annize.project.Node), 104
 add_element() (Methode von annize.project.file_formats.xml.Marshaler), 114
 add_element_attr() (Methode von annize.project.file_formats.xml.Marshaler), 114
 add_element_tree() (Methode von annize.project.file_formats.xml.Marshaler), 115
 add_object() (im Modul annize.flow.run_context), 83
 add_object() (Methode von annize.flow.run_context.RunContext), 80

<code>add_translation_provider()</code> (in Modul <code>annize.i18n</code>), 93	<code>module</code> , 19
<code>add_translations()</code> (Methode von <code>annize.features.i18n.common._ProjectDefinedTranslations</code>), 66	<code>annize.features.dependencies.common module</code> , 19
<code>additional_info()</code> (Methode von <code>annize.features.licensing.License</code>), 75	<code>annize.features.dependencies.python module</code> , 20
<code>AdministrationUtilitiesSection</code> (in Modul <code>annize.features.distributables.debian</code>), 27	<code>annize.features.distributables module</code> , 21
<code>AFLv3</code> (in Modul <code>annize.features.licensing</code>), 75	<code>annize.features.distributables.common module</code> , 21
<code>AGPLv3</code> (in Modul <code>annize.features.licensing</code>), 75	<code>annize.features.distributables.debian module</code> , 23
<code>all()</code> (Methode von <code>annize.project.inspector.Inspector.ArgumentMatchings</code>), 122	<code>annize.features.distributables.flatpak module</code> , 34
<code>allows_multiple_args</code> (in <code>annize.object.parameter_info.ListParameterInfo</code> property), 101	<code>annize.features.distributables.python_wheel module</code> , 39
<code>allows_multiple_args</code> (in <code>annize.object.parameter_info.ParameterInfo</code> property), 101	<code>annize.features.distributables.store module</code> , 21
<code>allows_multiple_args</code> (in <code>annize.project.inspector.Inspector.ArgumentMatchings</code> property), 122	<code>annize.features.distributables.store.pypi module</code> , 21
<code>annize</code> <code>module</code> , 13	<code>annize.features.distributables.tar module</code> , 43
<code>annize.annize_cli</code> <code>module</code> , 128	<code>annize.features.documentation module</code> , 43
<code>annize.asset</code> <code>module</code> , 13	<code>annize.features.documentation.common module</code> , 53
<code>annize.asset.data</code> <code>module</code> , 13	<code>annize.features.documentation.sphinx module</code> , 43
<code>annize.asset.project_info</code> <code>module</code> , 13	<code>annize.features.documentation.sphinx.common module</code> , 46
<code>annize.data</code> <code>module</code> , 13	<code>annize.features.documentation.sphinx.cpp module</code> , 50
<code>annize.data.color</code> <code>module</code> , 13	<code>annize.features.documentation.sphinx.doxygen_compat module</code> , 51
<code>annize.data.container</code> <code>module</code> , 14	<code>annize.features.documentation.sphinx.javascript module</code> , 52
<code>annize.data.unique</code> <code>module</code> , 15	<code>annize.features.documentation.sphinx.output module</code> , 43
<code>annize.data.version</code> <code>module</code> , 15	<code>annize.features.documentation.sphinx.output.common module</code> , 43
<code>annize.features</code> <code>module</code> , 18	<code>annize.features.documentation.sphinx.output.html module</code> , 44
<code>annize.features.authors</code> <code>module</code> , 72	<code>annize.features.documentation.sphinx.output.pdf module</code> , 45
<code>annize.features.base</code> <code>module</code> , 73	<code>annize.features.documentation.sphinx.output.plaintext module</code> , 46
<code>annize.features.changelog</code> <code>module</code> , 18	<code>annize.features.documentation.sphinx.python module</code> , 53
<code>annize.features.changelog.common</code> <code>module</code> , 18	<code>annize.features.documentation.sphinx.rst module</code> , 53
<code>annize.features.dependencies</code>	<code>annize.features.files module</code> , 55
	<code>annize.features.files.common module</code> , 57
	<code>annize.features.files.transfer</code>

module, 55	module, 70
annize.features.files.transfer.common	annize.features.version_control.common
module, 56	module, 70
annize.features.files.transfer.ssh	annize.features.version_control.git
module, 56	module, 71
annize.features.homepage	annize.flow
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, 84
annize.features.homepage.sections.about	annize.fs
module, 61	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, 121
annize.features.i18n.common	annize.project.file_formats
module, 66	module, 112
annize.features.i18n.gettext	annize.project.file_formats.xml
module, 68	module, 112
annize.features.injections	annize.project.inspector
module, 68	module, 122
annize.features.injections.common	annize.project.loader
module, 68	module, 124
annize.features.injections.python	annize.project.materializer
module, 69	module, 115
annize.features.licensing	annize.project.materializer.behaviors
module, 75	module, 116
annize.features.media_galleries	annize.project.materializer.behaviors.argument
module, 77	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, 118
annize.features.testing.common	annize.project.materializer.behaviors.feature_unavailable
module, 69	module, 118
annize.features.testing.pylint	annize.project.materializer.behaviors.reference
module, 70	module, 118
annize.features.testing.pytest	annize.project.materializer.core
module, 70	module, 119
annize.features.version	annize.project.materializer.preprocessors
module, 78	module, 120
annize.features.version_control	annize.ui

- module, 124
- annize.ui.apps
 - module, 124
- annize.user_feedback
 - module, 126
- annize.user_feedback.static
 - module, 128
- annize_config_rootpath (in *annize.project.Project* property), 102
- ANNIZE_CONFIG_ROOTPATH__NAME (Attribut von *annize.flow.run_context.RunContext*), 79
- annize_user_interaction_culture (in *Modul annize.i18n*), 98
- Apache2 (in *Modul annize.features.licensing*), 75
- ApiReferenceDocument (Klasse in *annize.features.documentation.sphinx.common*), 48
- ApiReferenceLanguage (Klasse in *annize.features.documentation.sphinx.common*), 48
- ApiReferenceLanguage.ApiReferenceGenerateInfo (Klasse in *annize.features.documentation.sphinx.common*), 48
- app() (in *Modul annize.ui*), 124
- append_child() (Methode von *annize.project.Node*), 105
- append_rst() (Methode von *annize.features.homepage.common.HomepageSection*), 64
- append_to (in *annize.project.ArgumentNode* property), 108
- ApplicationsAccessibilityCategory (in *Modul annize.features.distributables.debian*), 24
- ApplicationsAmateurradioCategory (in *Modul annize.features.distributables.debian*), 24
- ApplicationsDatamanagementCategory (in *Modul annize.features.distributables.debian*), 24
- ApplicationsEditorsCategory (in *Modul annize.features.distributables.debian*), 24
- ApplicationsEducationCategory (in *Modul annize.features.distributables.debian*), 24
- ApplicationsEmulatorsCategory (in *Modul annize.features.distributables.debian*), 24
- ApplicationsFilemanagementCategory (in *Modul annize.features.distributables.debian*), 24
- ApplicationsGraphicsCategory (in *Modul annize.features.distributables.debian*), 24
- ApplicationsMobiledevicesCategory (in *Modul annize.features.distributables.debian*), 24
- ApplicationsNetworkCategory (in *Modul annize.features.distributables.debian*), 24
- ApplicationsNetworkCommunicationCategory (in *Modul annize.features.distributables.debian*), 24
- ApplicationsNetworkFiletransferCategory (in *Modul annize.features.distributables.debian*), 24
- ApplicationsNetworkMonitoringCategory (in *Modul annize.features.distributables.debian*), 24
- ApplicationsNetworkWebbrowsingCategory (in *Modul annize.features.distributables.debian*), 25
- ApplicationsNetworkWebnewsCategory (in *Modul annize.features.distributables.debian*), 25
- ApplicationsOfficeCategory (in *Modul annize.features.distributables.debian*), 25
- ApplicationsProgrammingCategory (in *Modul annize.features.distributables.debian*), 25
- ApplicationsProjectmanagementCategory (in *Modul annize.features.distributables.debian*), 25
- ApplicationsScienceAstronomyCategory (in *Modul annize.features.distributables.debian*), 25
- ApplicationsScienceBiologyCategory (in *Modul annize.features.distributables.debian*), 25
- ApplicationsScienceCategory (in *Modul annize.features.distributables.debian*), 25
- ApplicationsScienceChemistryCategory (in *Modul annize.features.distributables.debian*), 25
- ApplicationsScienceDataanalysisCategory (in *Modul annize.features.distributables.debian*), 25
- ApplicationsScienceElectronicsCategory (in *Modul annize.features.distributables.debian*), 25
- ApplicationsScienceEngineeringCategory (in *Modul annize.features.distributables.debian*), 25
- ApplicationsScienceGeoscienceCategory (in *Modul annize.features.distributables.debian*), 25
- ApplicationsScienceMathematicsCategory (in *Modul annize.features.distributables.debian*), 25
- ApplicationsScienceMedicineCategory (in *Modul annize.features.distributables.debian*), 25
- ApplicationsSciencePhysicsCategory (in *Modul annize.features.distributables.debian*), 25
- ApplicationsScienceSocialCategory (in *Modul annize.features.distributables.debian*), 25
- ApplicationsShellsCategory (in *Modul annize.features.distributables.debian*), 25
- ApplicationsSoundsCategory (in *Modul annize.features.distributables.debian*), 25
- ApplicationsSystemAdministrationCategory (in *Modul annize.features.distributables.debian*), 26
- ApplicationsSystemCategory (in *Modul annize.features.distributables.debian*), 26

- ApplicationsSystemHardwareCategory (in Modul *annize.features.distributables.debian*), 26
- ApplicationsSystemLanguageenvironmentCategory (in Modul *annize.features.distributables.debian*), 26
- ApplicationsSystemMonitoringCategory (in Modul *annize.features.distributables.debian*), 26
- ApplicationsSystemPackagemanagementCategory (in Modul *annize.features.distributables.debian*), 26
- ApplicationsSystemSecurityCategory (in Modul *annize.features.distributables.debian*), 26
- ApplicationsTerminalEmulatorsCategory (in Modul *annize.features.distributables.debian*), 26
- ApplicationsTextCategory (in Modul *annize.features.distributables.debian*), 26
- ApplicationsTvandradiocategory (in Modul *annize.features.distributables.debian*), 26
- ApplicationsVideoCategory (in Modul *annize.features.distributables.debian*), 26
- ApplicationsViewersCategory (in Modul *annize.features.distributables.debian*), 26
- ApplicationsWebdevelopmentCategory (in Modul *annize.features.distributables.debian*), 26
- architecture (Attribut von *annize.features.distributables.debian.Package._BuildInfo*), 32
- arg_name (*annize.project.ArgumentNode* property), 108
- argname (*annize.project.inspector.Inspector.ArgumentMatchable* property), 122
- ArgparseCommandLineInterfaceDocument (Klasse in *annize.features.documentation.sphinx.common*), 49
- ArgumentBehavior (Klasse in *annize.project.materializer.behaviors.argument*), 116
- ArgumentNode (Klasse in *annize.project*), 108
- Artistic1 (in Modul *annize.features.licensing*), 76
- Artwork (Klasse in *annize.features.dependencies.common*), 20
- AssociateArgumentNodeBehavior (Klasse in *annize.project.materializer.behaviors.argument*), 117
- attach_media_file() (Methode von *annize.features.homepage.common.HomepageSection*), 64
- attr_name (Attribut von *annize.project.file_formats.xml.Marshaler.XmlDocumentLocation*), 114
- ATTRIBUTE_COMMAND_END (Attribut von *annize.project.file_formats.xml._XmlParser*), 113
- ATTRIBUTE_COMMAND_START (Attribut von *annize.project.file_formats.xml._XmlParser*), 113
- author (Attribut von *annize.features.distributables.debian.Package._BuildInfo*), 32
- author (Attribut von *annize.features.distributables.python_wheel.Package._BuildInfo*), 41
- Author (Klasse in *annize.features.authors*), 72
- authors (*annize.features.documentation.sphinx.common.Document* property), 47
- authorstring (Attribut von *annize.features.distributables.debian.Package._BuildInfo*), 32
- available_cultures() (Methode von *annize.features.documentation.common.Document*), 54
- available_cultures() (Methode von *annize.features.documentation.sphinx.common.AboutProjectDocument*), 50
- available_cultures() (Methode von *annize.features.documentation.sphinx.common.ApiReferenceDocument*), 48
- available_cultures() (Methode von *annize.features.documentation.sphinx.common.ArgparseCommandLineInterfaceDocument*), 49
- available_cultures() (Methode von *annize.features.documentation.sphinx.common.ReadmeDocument*), 50
- available_cultures() (Methode von *annize.features.documentation.sphinx.common.RstDocument*), 49
- ## B
- background_image (Attribut von *annize.features.documentation.sphinx.output.html.HtmlOutputSpec* property), 45
- BadStructureError, 111
- Basket (Klasse in *annize.data.container*), 14
- Basket (Klasse in *annize.features.base*), 74
- BasketBehavior (Klasse in *annize.project.materializer.behaviors.basket*), 117
- Behavior (Klasse in *annize.project.materializer.behaviors*), 116
- BLOCK (Attribut von *annize.project.BlockNodeWithScope.Scope*), 110
- BlockBehavior (Klasse in *annize.project.materializer.behaviors.block*), 118
- BlockNode (Klasse in *annize.project*), 110
- BlockNodeWithScope (Klasse in *annize.project*), 110
- BlockNodeWithScope.Scope (Klasse in *annize.project*), 110
- blue (*annize.data.color.Color* property), 14

brand_color() (im Modul *annize.features.base*), 74

BrandColor (Klasse in *annize.features.base*), 73

BSD2clause (in Modul *annize.features.licensing*), 76

BSD3clause (in Modul *annize.features.licensing*), 76

BuildVersion (Klasse in *annize.features.version_control.common*), 71

ByVersionControlSystemCommitMessagesChangelog (Klasse in *annize.features.changelog.common*), 19

C

category (*annize.features.distributables.debian.MenuEntry* property), 23

category (*annize.features.distributables.flatpak.MenuEntry* property), 35

Category (Klasse in *annize.features.distributables.debian*), 23

Cc0v1 (in Modul *annize.features.licensing*), 76

CcBy3 (in Modul *annize.features.licensing*), 76

CcByNc3 (in Modul *annize.features.licensing*), 76

CcByNcNd3 (in Modul *annize.features.licensing*), 76

CcByNcSa3 (in Modul *annize.features.licensing*), 76

CcByNd3 (in Modul *annize.features.licensing*), 76

CcBySa3 (in Modul *annize.features.licensing*), 76

Changelog (Klasse in *annize.features.changelog.common*), 19

child_node (*annize.project.Node.ChildrenListChangeEvent* property), 103

child_position (*annize.project.Node.ChildrenListChangeEvent* property), 103

children (*annize.project.Node* property), 105

children() (Methode von *annize.fs.Path*), 86

ChildrenNotMaterializableError, 120

choice_dialog() (im Modul *annize.user_feedback*), 127

choice_dialog() (Methode von *annize.user_feedback.NullUserFeedbackController*), 126

choice_dialog() (Methode von *annize.user_feedback.static.StaticUserFeedbackController*), 128

choice_dialog() (Methode von *annize.user_feedback.UserFeedbackController*), 126

clone() (Methode von *annize.project.Node*), 106

Color (Klasse in *annize.data.color*), 13

command (*annize.features.distributables.debian.MenuEntry* property), 23

command (*annize.features.distributables.flatpak.MenuEntry* property), 35

command (Attribut von *annize.features.distributables.flatpak.FlatpakImage.BuildInfo*), 38

Commands (Klasse in *annize.annize_cli*), 128

comment (*annize.features.dependencies.common.Dependency* property), 20

CommonVersionPattern (Klasse in *annize.features.version*), 78

CommunicationProgramsSection (in Modul *annize.features.distributables.debian*), 27

CompositeDocument (Klasse in *annize.features.documentation.sphinx.common*), 47

ConcatenatedVersionPatternSegment (Klasse in *annize.data.version*), 17

confdir (Attribut von *annize.features.documentation.sphinx.common.Document.GenerateInfo*), 47

config_files (Attribut von *annize.features.distributables.debian.Package._BuildInfo*), 33

confglines (Attribut von *annize.features.documentation.sphinx.common.Document.GenerateInfo*), 47

configvalues (Attribut von *annize.features.documentation.sphinx.common.Document.GenerateInfo*), 47

Connection (Klasse in *annize.features.distributables.store.pypi*), 21

construct_from_string() (Methode von *annize.object.parameter_info.ParameterInfo*), 101

content (*annize.features.injections.common.FilesystemContentInjection* property), 68

content() (im Modul *annize.fs*), 89

copy_to() (Methode von *annize.fs.Path*), 87

CppApiReferenceLanguage (Klasse in *annize.features.documentation.sphinx.cpp*), 50

create_new() (statische Methode von *annize.project.Project*), 103

create_object() (im Modul *annize.object.controller*), 100

create_object() (statische Methode von *annize.object.controller._CreateObjectHelper*), 99

ctime() (Methode von *annize.fs.Path*), 86

culture (*annize.features.documentation.sphinx.common.RstDocumentVariables* property), 49

culture (Attribut von *annize.features.documentation.sphinx.common.Document.GenerateInfo*), 47

culture (Attribut von *annize.features.homepage.common.HomepageSection._GenerateInfo*), 63

Culture (Klasse in *annize.features.i18n.common*), 67

Culture (Klasse in *annize.i18n*), 95

Culture._TSystemLocaleSetup (Klasse in *annize*), 67

`ze.i18n`), 97
`culture_by_spec()` (im Modul `annize.i18n`), 98
`culture_list()` (Methode von `annize.i18n.Culture`), 96
`culture_names` (Attribut von `annize.features.documentation.common.DocumentGenerateAllCulturesResult`), 54
`cultures` (`annize.features.homepage.common.HomepageSection`), 65
`current()` (im Modul `annize.flow.run_context`), 82
`current_culture()` (im Modul `annize.i18n`), 98
`custom_arg` (Attribut von `annize.features.homepage.common.HomepageSection._PrePostProcGenerateInfo`), 63
`custom_arg` (Attribut von `annize.features.homepage.common.HomepageSection._PrePostProcGenerateInfo`), 64
D
`Data` (Klasse in `annize.features.base`), 73
`DatabasesSection` (in Modul `annize.features.distributables.debian`), 27
`DateTime` (Klasse in `annize.features.base`), 73
`debian_name` (`annize.features.distributables.debian.Category`), 23
`DebianInstallerUdebPackagesSection` (in Modul `annize.features.distributables.debian`), 27
`DebugPackagesSection` (in Modul `annize.features.distributables.debian`), 27
`default_changelog()` (im Modul `annize.features.changelog.common`), 19
`default_version_control_system()` (im Modul `annize.features.version_control.common`), 71
`default_version_pattern()` (im Modul `annize.features.version`), 78
`DefaultFeatureLoader` (Klasse in `annize.project.feature_loader`), 121
`dependencies` (`annize.features.distributables.python_wheel.Exclude`), 40
`dependencies` (Attribut von `annize.features.distributables.python_wheel.Package._BuildInfo`), 41
`dependencies_to_rst_text()` (im Modul `annize.features.dependencies.common`), 20
`Dependency` (Klasse in `annize.features.dependencies.common`), 19
`description` (`annize.features.distributables.common.Group`), 22
`description` (`annize.features.distributables.flatpak.Group`), 36
`description` (`annize.features.media_galleries.Gallery.Item`), 77
`description` (Attribut von `annize.features.distributables.debian.Package._BuildInfo`), 32
`description` (Attribut von `annize.features.distributables.python_wheel.Package._BuildInfo`), 41
`description()` (Methode von `annize.project.Node`), 106
`destination_is_parent` (Attribut von `annize.features.files.common.DirectoryPart`), 59
`destination_path` (Attribut von `annize.features.files.common.DirectoryPart`), 59
`DevelopmentSection` (in Modul `annize.features.distributables.debian`), 27
`Directory` (Klasse in `annize.features.files.common`), 59
`DirectoryPart` (Klasse in `annize.features.files.common`), 59
`do` (`annize.project.OnFeatureUnavailableNode`), 111
`do()` (Methode von `annize.annize_cli.Commands`), 129
`Document` (Klasse in `annize.features.documentation.common`), 54
`Document` (Klasse in `annize.features.documentation.sphinx.common`), 46
`Document.GenerateInfo` (Klasse in `annize.features.documentation.sphinx.common`), 46
`document_root_directory` (Attribut von `annize.features.homepage.common.HomepageSection._PrePostProcGenerateInfo`), 64
`document_root_url` (Attribut von `annize.features.homepage.common.HomepageSection._GenerateInfo`), 63
`document_root_url` (Attribut von `annize.features.homepage.common.HomepageSection._PrePostProcGenerateInfo`), 64
`document_root_url` (Attribut von `annize.features.homepage.common.HomepageSection._GenerateInfo`), 63
`document_root_url` (Attribut von `annize.features.homepage.common.HomepageSection._PrePostProcGenerateInfo`), 64
`document_root_url` (Attribut von `annize.features.homepage.common.HomepageSection._GenerateInfo`), 63
`document_root_url` (Attribut von `annize.features.homepage.common.HomepageSection._PrePostProcGenerateInfo`), 64
`document_root_url` (Attribut von `annize.features.homepage.common.HomepageSection._GenerateInfo`), 63
`documentation_source` (Attribut von `annize.features.distributables.debian.Package._BuildInfo`), 32
`DocumentationSection` (in Modul `annize.features.distributables.debian`), 27
`DocumentGenerateAllCulturesResult` (Klasse in `annize.features.documentation.common`), 54
`DocumentGenerateResult` (Klasse in `annize.features.documentation.common`), 53
`does_exclude()` (Methode von `annize.features.files.common.Exclude`), 58
`does_exclude()` (Methode von `annize.features.files.common.Exclude`), 58

`ze.features.files.common.ExcludeAllBut` (Klassenmethode von `annize.project.file_formats.xml._XmlParser`), 58
`does_exclude()` (Methode von `annize.features.version_control.git.ExcludeByGitIgnore`), 72
`DoxygenSupportedApiReferenceLanguage` (Klasse in `annize.features.documentation.sphinx.doxygen_compat`), 51
`dynamic_file()` (im Modul `annize.fs`), 89
`DynamicFile` (Klasse in `annize.fs.ext`), 90
E
`EditorsSection` (in Modul `annize.features.distributables.debian`), 28
`EducationSection` (in Modul `annize.features.distributables.debian`), 28
`ElectronicsSection` (in Modul `annize.features.distributables.debian`), 28
`element` (Attribut von `annize.project.file_formats.xml.Marshaler.XmlDocumentElement`), 114
`email` (`annize.features.authors.Author` property), 72
`EmbeddedSoftwareSection` (in Modul `annize.features.distributables.debian`), 28
`Endpoint` (Klasse in `annize.features.files.transfer.common`), 56
`Endpoint` (Klasse in `annize.features.files.transfer.ssh`), 56
`english_lang_name` (`annize.i18n.Culture` property), 96
`entries` (`annize.features.changelog.common.ByVersionControlSystemCommitMessagesChangelog` property), 19
`entries` (`annize.features.changelog.common.Changelog` property), 19
`Entry` (Klasse in `annize.features.changelog.common`), 18
`entry_path` (`annize.features.documentation.common.DocumentGenerateInfo` property), 54
`entry_path` (Attribut von `annize.features.documentation.sphinx.common.DocumentGenerateInfo`), 47
`entry_path_for_language()` (Methode von `annize.features.documentation.common.DocumentGenerateAllCulturesResult`), 54
`environment` (Attribut von `annize.features.distributables.flatpak.FlatpakImage._BuildInfo`), 38
`EnvironmentVariable` (Klasse in `annize.features.distributables.flatpak`), 35
`erroneous_nodes()` (Methode von `annize.project.materializer.MaterializationResult`), 115
`errors_for_node()` (Methode von `annize.project.materializer.MaterializationResult`), 115
`escape_attribute_string()` (Klassenmethode von `annize.project.file_formats.xml._XmlParser`), 113
`Exclude` (Klasse in `annize.features.files.common`), 58
`ExcludeAllBut` (Klasse in `annize.features.files.common`), 58
`ExcludeByGitIgnores` (Klasse in `annize.features.version_control.git`), 72
`excludes` (`annize.features.files.common.Directory` property), 60
`excludes` (`annize.features.files.common.DirectoryPart` property), 59
`exec()` (Methode von `annize.features.files.transfer.ssh.Endpoint`), 57
`executable_links` (Attribut von `annize.features.distributables.debian.Package._BuildInfo`), 32
`executable_links` (Attribut von `annize.features.distributables.python_wheel.Package._BuildInfo`), 41
`ExecutableLink` (Klasse in `annize.features.distributables.debian`), 23
`ExecutableLink` (Klasse in `annize.features.distributables.python_wheel`), 39
`explicit_only` (Attribut von `annize.object.controller._CreateObjectHelper.ParameterConfig`), 99
`explicit_only()` (im Modul `annize.object`), 99
`extra_dependencies` (Attribut von `annize.features.distributables.python_wheel.Package._BuildInfo`), 41
`extra_name` (`annize.features.distributables.python_wheel.ExtraDependency` property), 40
`ExtraDependencies` (Klasse in `annize.features.distributables.python_wheel`), 40
F
`FAIL` (Attribut von `annize.project.OnFeatureUnavailableNode.Action`), 111
`FAIL` (Attribut von `annize.project.ReferenceNode.OnUnresolvableAction`), 110
`fallback_cultures` (`annize.i18n.Culture` property), 96
`feature` (`annize.project.inspector.Inspector.TypeInfo` property), 122
`feature` (`annize.project.ObjectNode` property), 109
`feature` (`annize.project.OnFeatureUnavailableNode` property), 111
`FeatureLoader` (Klasse in `annize.project.feature_loader`), 121

FeatureUnavailableBehavior (Klasse in `annize.project.materializer.behaviors.feature_unavailable`), 118
FeatureUnavailableError, 111
file (`annize.features.documentation.common.DocumentGenerator` property), 54
file (`annize.features.media_galleries.Gallery.Item` property), 77
FILE (Attribut von `annize.project.BlockNodeWithScope.Scope`), 110
File (Klasse in `annize.features.files.common`), 57
file_size() (Methode von `annize.fs.Path`), 87
FileFormat (Klasse in `annize.project.file_formats`), 112
FileFormat.Marshaller (Klasse in `annize.project.file_formats`), 112
filehash() (im Modul `annize.features.homepage.sections.download`), 62
FileNode (Klasse in `annize.project`), 108
files() (Methode von `annize.features.distributables.common.Group`), 22
files() (Methode von `annize.features.distributables.flatpak.Group`), 36
Filesystem (Klasse in `annize.features.distributables.flatpak`), 35
FilesystemContent (Klasse in `annize.fs`), 85
FilesystemContentInjection (Klasse in `annize.features.injections.common`), 68
filesystems (Attribut von `annize.features.distributables.flatpak.FlatpakImage.BuildInfo`), 38
find_output_generator_for_outputspec() (im Modul `annize.features.documentation.sphinx.output.common`), 43
find_project_annize_config_root_file() (im Modul `annize.project.loader`), 124
FirstOf (Klasse in `annize.features.base`), 74
FlatpakImage (Klasse in `annize.features.distributables.flatpak`), 37
FlatpakImage._BuildInfo (Klasse in `annize.features.distributables.flatpak`), 37
FlatpakrefFile (Klasse in `annize.features.distributables.flatpak`), 36
FontsSection (in Modul `annize.features.distributables.debian`), 28
format() (Methode von `annize.i18n.TrStr`), 94
formatname() (Methode von `annize.features.documentation.sphinx.output.common.OutputGenerator`), 44
formatname() (Methode von `annize.features.documentation.sphinx.output.html.HtmlOutputGenerator`), 45
formatname() (Methode von `annize.features.documentation.sphinx.output.pdf.PdfOutputGenerator`), 45
formatname() (Methode von `annize.features.documentation.sphinx.output.plaintext.PlaintextOutputGenerator`), 46
freedesktop_name (Attribut von `annize.features.distributables.debian.Category` property), 23
fresh_temp_directory() (im Modul `annize.fs`), 89
FreshTempDirectory (Klasse in `annize.fs.ext`), 90
friendly_filesize() (im Modul `annize.features.homepage.sections.download`), 62
friendly_join_string_list() (im Modul `annize.i18n`), 98
friendly_name_suggestion (Attribut von `annize.features.distributables.flatpak.Repository` property), 35
FromRequirementsFile (Klasse in `annize.features.dependencies.python`), 21
FsEndpoint (Klasse in `annize.features.files.transfer.common`), 56
FsEntry (Klasse in `annize.features.files.common`), 57
full_name (`annize.features.authors.Author` property), 72
full_name (`annize.i18n.Culture` property), 96
G
Gallery (Klasse in `annize.features.media_galleries`), 77
Gallery.Item (Klasse in `annize.features.media_galleries`), 77
GamesActionCategory (in Modul `annize.features.distributables.debian`), 26
GamesAdventureCategory (in Modul `annize.features.distributables.debian`), 26
GamesBlocksCategory (in Modul `annize.features.distributables.debian`), 26
GamesBoardCategory (in Modul `annize.features.distributables.debian`), 26
GamesCardCategory (in Modul `annize.features.distributables.debian`), 26
GamesPuzzlesCategory (in Modul `annize.features.distributables.debian`), 26
GamesSection (in Modul `annize.features.distributables.debian`), 28
GamesSimulationCategory (in Modul `annize.features.distributables.debian`), 27
GamesStrategyCategory (in Modul `annize.features.distributables.debian`), 27
GamesToolsCategory (in Modul `annize.features.distributables.debian`), 27
GamesToysCategory (in Modul `annize.features.distributables.debian`), 27

generate() (Methode von annize.features.homepage.common), 66
 ze.features.documentation.common.Document), 54
 generate() (Methode von annize.features.documentation.sphinx.common.Document), 47
 generate() (Methode von annize.features.homepage.common.Homepage), 65
 generate_all_cultures() (Methode von annize.features.documentation.common.Document), 54
 generate_all_cultures() (Methode von annize.features.documentation.sphinx.common.Document), 47
 generate_content() (Methode von annize.features.homepage.common.HomepageSection), 64
 generate_content() (Methode von annize.features.homepage.sections.about.Section), 61
 generate_content() (Methode von annize.features.homepage.sections.changelog.Section), 61
 generate_content() (Methode von annize.features.homepage.sections.documentation.Section), 61
 generate_content() (Methode von annize.features.homepage.sections.download.Section), 62
 generate_content() (Methode von annize.features.homepage.sections.gallery.Section), 62
 generate_content() (Methode von annize.features.homepage.sections.imprint.Section), 63
 generate_content() (Methode von annize.features.homepage.sections.license.Section), 63
 generate_sources() (Methode von annize.features.documentation.sphinx.common.ApiReferenceLanguage), 48
 generate_sources() (Methode von annize.features.documentation.sphinx.doxygen_compat.DoxygenSupportApiReferenceLanguageNodeMaterialization), 52
 generate_sources() (Methode von annize.features.documentation.sphinx.javascript.JavaScriptApiReferenceLanguageInspector), 52
 generate_sources() (Methode von annize.features.documentation.sphinx.python.Python3ApiReferenceLanguageInspector), 53
 GeneratedDocument (Klasse in annize.features.documentation.common), 55
 GeneratedHomepage (Klasse in annize.features.homepage.common), 66
 GenerateMOS (Klasse in annize.features.i18n.gettext), 68
 get_all_available_feature_names() (Methode von annize.project.feature_loader.DefaultFeatureLoader), 121
 get_all_available_feature_names() (Methode von annize.project.feature_loader.FeatureLoader), 121
 get_all_types() (Methode von annize.project.inspector.Inspector), 123
 get_changes() (Methode von annize.project.ProjectNode), 107
 get_commit_message() (Methode von annize.features.version_control.common.VersionControlSystem), 70
 get_commit_message() (Methode von annize.features.version_control.git.VersionControlSystem), 71
 get_commit_time() (Methode von annize.features.version_control.common.VersionControlSystem), 70
 get_commit_time() (Methode von annize.features.version_control.git.VersionControlSystem), 72
 get_current_revision() (Methode von annize.features.version_control.common.VersionControlSystem), 70
 get_current_revision() (Methode von annize.features.version_control.git.VersionControlSystem), 71
 get_format() (im Modul annize.project.file_formats), 112
 get_from_iso_639_1_lang_code() (statische Methode von annize.i18n.Culture), 96
 get_materialized() (Methode von annize.project.materializer.core.ProjectMaterializer), 120
 get_materialized_children() (Methode von annize.project.materializer.core.NodeMaterialization), 119
 get_materialized_children_tuples() (Methode von annize.project.materializer.core.NodeMaterialization), 119
 get_node_by_name() (Methode von annize.project.inspector.Inspector), 123
 get_project_node() (Methode von annize.project.inspector.Inspector), 123
 get_revision_by_name() (Methode von annize.features.version_control.common.VersionControlSystem), 70
 get_revision_list() (Methode von annize.features.version_control.git.VersionControlSystem), 70

[71](#)
[get_revision_number\(\)](#) (Methode von `annize.features.version_control.common.VersionControlSystem`), [71](#)
[71](#)
[get_revision_number\(\)](#) (Methode von `annize.features.version_control.git.VersionControlSystem`), [72](#)
[72](#)
[get_selected_task\(\)](#) (Methode von `annize.flow.runner.Runner`), [85](#)
[get_success_state\(\)](#) (Methode von `annize.flow.runner.Runner`), [85](#)
[get_tasks\(\)](#) (Methode von `annize.flow.runner.Runner`), [84](#)
[get_types_for_argument\(\)](#) (Methode von `annize.project.inspector.Inspector`), [123](#)
[get_variant\(\)](#) (Methode von `annize.i18n.FixedTrStr`), [95](#)
[get_variant\(\)](#) (Methode von `annize.i18n.ProvidedTrStr`), [94](#)
[get_variant\(\)](#) (Methode von `annize.i18n.TrStr`), [94](#)
[GettextTranslationProvider](#) (Klasse in `annize.i18n`), [92](#)
[GettextTranslationProvider._NoneTranslations](#) (Klasse in `annize.i18n`), [93](#)
[GnomeSection](#) (in Modul `annize.features.distributables.debian`), [28](#)
[GnuLinux](#) (Klasse in `annize.features.dependencies.common`), [20](#)
[GnuRSection](#) (in Modul `annize.features.distributables.debian`), [28](#)
[GnustepSection](#) (in Modul `annize.features.distributables.debian`), [28](#)
[GpgFile](#) (Klasse in `annize.features.distributables.flatpak`), [37](#)
[GPLv3](#) (in Modul `annize.features.licensing`), [76](#)
[GraphicsSection](#) (in Modul `annize.features.distributables.debian`), [28](#)
[green](#) (`annize.data.color.Color` property), [14](#)
[Group](#) (Klasse in `annize.features.distributables.common`), [22](#)
[Group](#) (Klasse in `annize.features.distributables.flatpak`), [36](#)
H
[HamRadioSection](#) (in Modul `annize.features.distributables.debian`), [28](#)
[has_result](#) (`annize.project.materializer.core.NodeMaterialization` property), [119](#)
[has_shell_access](#) (`annize.features.files.transfer.ssh.Endpoint` property), [57](#)
[HaskellSection](#) (in Modul `annize.features.distributables.debian`), [28](#)
[head](#) (`annize.features.homepage.common.HomepageSection` property), [64](#)
[heading](#) (`annize.features.documentation.sphinx.common.ApiReferenceLang` property), [48](#)
[heading\(\)](#) (statische Methode von `annize.features.documentation.sphinx.rst.RstGenerator`), [53](#)
[HelpCategory](#) (in Modul `annize.features.distributables.debian`), [27](#)
[homepage](#) (Attribut von `annize.features.distributables.debian.Package._BuildInfo`), [32](#)
[homepage](#) (Attribut von `annize.features.distributables.python_wheel.Package._BuildInfo`), [41](#)
[Homepage](#) (Klasse in `annize.features.homepage.common`), [65](#)
[homepage_url](#) (`annize.features.base.Data` property), [73](#)
[homepage_url\(\)](#) (in Modul `annize.features.base`), [75](#)
[HomepageSection](#) (Klasse in `annize.features.homepage.common`), [63](#)
[HomepageSection._GenerateInfo](#) (Klasse in `annize.features.homepage.common`), [63](#)
[HomepageSection._PrePostProcGenerateInfo](#) (Klasse in `annize.features.homepage.common`), [63](#)
[HomepageSection.Content](#) (Klasse 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](#) (Klasse in `annize.features.documentation.sphinx.output.html`), [45](#)
[HtmlOutputSpec](#) (Klasse in `annize.features.documentation.common`), [55](#)
[HtmlOutputSpec](#) (Klasse in `annize.features.documentation.sphinx.output.html`), [44](#)
[hue](#) (`annize.data.color.Color` property), [14](#)
I
[icon](#) (`annize.features.dependencies.common.Dependency` property), [20](#)
[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), [57](#)
[IMAGE](#) (Attribut von `annize.features.media_galleries.MediaType`), [44](#)

- 77
- importance (annize.features.dependencies.common.Dependency property), 20
- importance (annize.features.dependencies.common.Kind property), 19
- imprint (annize.features.base.Data property), 73
- imprint() (im Modul annize.features.base), 75
- in_file() (Methode von annize.project.file_formats.xml._XmlParser._Context), 113
- in_node() (Methode von annize.project.file_formats.xml._XmlParser._Context), 113
- Included (Klasse in annize.features.dependencies.common), 20
- Inject (Klasse in annize.features.injections.common), 68
- inject() (Methode von annize.features.injections.common.FileSystemContentInjection), 68
- inject() (Methode von annize.features.injections.common.Injection), 68
- inject() (Methode von annize.features.injections.python.ProjectInfoInjection), 69
- Injection (Klasse in annize.features.injections.common), 68
- inner_type_info (annize.object.parameter_info.ListParameterInfo property), 101
- inner_type_info (annize.object.parameter_info.ParameterInfo property), 101
- input_dialog() (im Modul annize.user_feedback), 127
- input_dialog() (Methode von annize.user_feedback.NullUserFeedbackController), 126
- input_dialog() (Methode von annize.user_feedback.static.StaticUserFeedbackController), 128
- input_dialog() (Methode von annize.user_feedback.UserFeedbackController), 126
- insert_child() (Methode von annize.project.Node), 105
- insert_child() (Methode von annize.project.ProjectNode), 107
- Inspector (Klasse in annize.project.inspector), 122
- Inspector.ArgumentMatchings (Klasse in annize.project.inspector), 122
- Inspector.ArgumentMatchings.ArgumentMatching (Klasse in annize.project.inspector), 122
- Inspector.TypeInfo (Klasse in annize.project.inspector), 122
- InternalError, 120
- InterpretersSection (in Modul annize.features.distributables.debian), 28
- IntrospectionSection (in Modul annize.features.distributables.debian), 28
- intstruct (Attribut von annize.features.documentation.sphinx.common.Document.GenerateIntstruct), 46
- is_advanced (annize.features.task.Task property), 77
- is_compatible_for() (Klassenmethode von annize.features.documentation.sphinx.output.common.OutputGenerator), 44
- is_compatible_for() (Klassenmethode von annize.features.documentation.sphinx.output.html.HtmlOutputGenerator), 45
- is_compatible_for() (Klassenmethode von annize.features.documentation.sphinx.output.pdf.PdfOutputGenerator), 45
- is_compatible_for() (Klassenmethode von annize.features.documentation.sphinx.output.plaintext.PlaintextOutputGenerator), 46
- is_constructable_from_string (annize.object.parameter_info.ParameterInfo property), 101
- is_finished() (Methode von annize.flow.runner.Runner), 85
- is_friendly_name() (im Modul annize.flow.run_context), 83
- is_friendly_name() (Methode von annize.flow.run_context.RunContext), 80
- 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), 55
- is_optional (annize.object.parameter_info.ParameterInfo property), 101
- is_toplevel_object() (im Modul annize.flow.run_context), 83
- is_toplevel_object() (Methode von annize.flow.run_context.RunContext), 80
- iso_639_1_language_code (annize.i18n.Culture property), 96
- Item (Klasse in annize.features.changelog.common), 18
- items (annize.features.changelog.common.Entry property), 18
- items (annize.features.media_galleries.Gallery property), 77
- ## J
- JavaScriptApiReferenceLanguage (Klasse in annize-

- `ze.features.documentation.sphinx.javascript`), 52
- JavaScriptSection** (in *Modul* *annize.features.distributables.debian*), 28
- JavaSection** (in *Modul* *annize.features.distributables.debian*), 28
- join_authors()** (im *Modul* *annize.features.authors*), 72
- ## K
- KdeSection** (in *Modul* *annize.features.distributables.debian*), 28
- KernelsSection** (in *Modul* *annize.features.distributables.debian*), 28
- Keyword** (Klasse in *annize.features.base*), 74
- keywords** (*annize.features.base.Keywords* property), 74
- keywords** (Attribut von *annize.features.distributables.python_wheel.Package._BuildInfo*), 41
- Keywords** (Klasse in *annize.features.base*), 73
- kind** (*annize.features.dependencies.common.Dependency* property), 19
- Kind** (Klasse in *annize.features.dependencies.common*), 19
- kitversion** (Attribut von *annize.features.distributables.flatpak.FlatpakImage._BuildInfo*), 38
- ## L
- label** (*annize.features.dependencies.common.Dependency* property), 20
- label** (*annize.features.dependencies.common.Kind* property), 19
- LANGUAGE** (Attribut von *annize.i18n.Culture._TSystemLocaleSetup*), 97
- LanguagePacksSection** (in *Modul* *annize.features.distributables.debian*), 29
- LC_ALL** (Attribut von *annize.i18n.Culture._TSystemLocaleSetup*), 97
- LGPLv3** (in *Modul* *annize.features.licensing*), 76
- LibrariesSection** (in *Modul* *annize.features.distributables.debian*), 29
- LibraryDevelopmentSection** (in *Modul* *annize.features.distributables.debian*), 28
- license** (Attribut von *annize.features.distributables.python_wheel.Package._BuildInfo*), 41
- License** (Klasse in *annize.features.licensing*), 75
- licensename** (Attribut von *annize.features.distributables.debian.Package._BuildInfo*), 32
- lightness** (*annize.data.color.Color* property), 14
- Line** (Klasse in *annize.features.version*), 78
- link_name** (*annize.features.distributables.python_wheel.ExampleNode* property), 39
- LispSection** (in *Modul* *annize.features.distributables.debian*), 29
- ListParameterInfo** (Klasse in *annize.object.parameter_info*), 101
- load()** (statische Methode von *annize.project.Project*), 102
- load()** (statische Methode von *annize.project.ProjectNode*), 107
- load_feature()** (Methode von *annize.project.feature_loader.DefaultFeatureLoader*), 121
- load_feature()** (Methode von *annize.project.feature_loader.FeatureLoader*), 121
- load_project()** (im *Modul* *annize.project.loader*), 124
- LocalRepository** (Klasse in *annize.features.distributables.flatpak*), 35
- locationstring()** (Methode von *annize.features.files.transfer.ssh.Endpoint*), 57
- logo_image** (*annize.features.documentation.sphinx.output.html.HtmlOutput* property), 45
- long_description** (*annize.features.base.Data* property), 73
- long_description** (Attribut von *annize.features.distributables.python_wheel.Package._BuildInfo*), 41
- long_description()** (im *Modul* *annize.features.base*), 74
- long_str** (*annize.data.unique.UniqueId* property), 15
- ## M
- MachineRootDirectory** (Klasse in *annize.features.files.common*), 60
- MailSection** (in *Modul* *annize.features.distributables.debian*), 29
- main()** (im *Modul* *annize.annize_cli*), 128
- main_document** (Attribut von *annize.features.documentation.sphinx.common.Document.Generator*), 46
- mark_object_as_toplevel()** (Methode von *annize.flow.run_context.RunContext*), 81
- marshaler** (*annize.project.file_formats.xml._XmlParser._Context* property), 113
- marshaler** (*annize.project.FileNode* property), 108
- Marshaller** (Klasse in *annize.project.file_formats.xml*), 114
- Marshaller.XmlDocumentLocation** (Klasse in *annize.project.file_formats.xml*), 114
- masterlink** (*annize.features.documentation.sphinx.output.html.HtmlOutput* property), 45
- match_arguments()** (Methode von *annize.project.inspector.Inspector*), 122
- match_node()** (Methode von *annize.project.inspector.Inspector*), 123

<code>matches_inner_type()</code> (Methode von <code>annize.object.parameter_info.ParameterInfo</code>), 101	<code>ze.features.distributables.debian</code>), 29
<code>matches_object()</code> (Methode von <code>annize.object.parameter_info.ParameterInfo</code>), 101	MIT (in Modul <code>annize.features.licensing</code>), 76
<code>matches_object()</code> (Methode von <code>annize.object.parameter_info.UnionParameterInfo</code>), 102	module
<code>matches_type()</code> (Methode von <code>annize.object.parameter_info.ParameterInfo</code>), 101	<code>annize</code> , 13
<code>matching_by_argname()</code> (Methode von <code>annize.project.inspector.Inspector.ArgumentMatchings</code>), 122	<code>annize.annize_cli</code> , 128
<code>MaterializationResult</code> (Klasse in <code>annize.project.materializer</code>), 115	<code>annize.asset</code> , 13
<code>materialize()</code> (im Modul <code>annize.project.materializer</code>), 115	<code>annize.asset.data</code> , 13
<code>MaterializerError</code> , 111	<code>annize.asset.project_info</code> , 13
<code>MathematicsSection</code> (in Modul <code>annize.features.distributables.debian</code>), 29	<code>annize.data</code> , 13
<code>media_files</code> (<code>annize.features.homepage.common.HomepageSection</code> property), 64	<code>annize.data.color</code> , 13
<code>mediatype</code> (<code>annize.features.media_galleries.Gallery.Item</code> property), 77	<code>annize.data.container</code> , 14
<code>MediaType</code> (Klasse in <code>annize.features.media_galleries</code>), 77	<code>annize.data.unique</code> , 15
<code>menu_entries</code> (Attribut von <code>annize.features.distributables.flatpak.FlatpakImage._BuildInfo</code>), 38	<code>annize.data.version</code> , 15
<code>menuentries</code> (Attribut von <code>annize.features.distributables.debian.Package._BuildInfo</code>), 32	<code>annize.features</code> , 18
<code>MenuEntry</code> (Klasse in <code>annize.features.distributables.debian</code>), 23	<code>annize.features.authors</code> , 72
<code>MenuEntry</code> (Klasse in <code>annize.features.distributables.flatpak</code>), 34	<code>annize.features.base</code> , 73
<code>message_dialog()</code> (im Modul <code>annize.user_feedback</code>), 127	<code>annize.features.changelog</code> , 18
<code>message_dialog()</code> (Methode von <code>annize.user_feedback.NullUserFeedbackController</code>), 126	<code>annize.features.changelog.common</code> , 18
<code>message_dialog()</code> (Methode von <code>annize.user_feedback.static.StaticUserFeedbackController</code>), 128	<code>annize.features.dependencies</code> , 19
<code>message_dialog()</code> (Methode von <code>annize.user_feedback.UserFeedbackController</code>), 126	<code>annize.features.dependencies.common</code> , 19
<code>MetaPackagesSection</code> (in Modul <code>annize.features.distributables.debian</code>), 29	<code>annize.features.dependencies.python</code> , 20
<code>method_name</code> (<code>annize.features.distributables.python_wheel.Executable</code> property), 39	<code>annize.features.distributables</code> , 21
<code>MiscellaneousSection</code> (in Modul <code>annize</code>), 46	<code>annize.features.distributables.common</code> , 21
	<code>annize.features.distributables.debian</code> , 23
	<code>annize.features.distributables.flatpak</code> , 34
	<code>annize.features.distributables.python_wheel</code> , 39
	<code>annize.features.distributables.store</code> , 21
	<code>annize.features.distributables.store.pypi</code> , 21
	<code>annize.features.distributables.tar</code> , 43
	<code>annize.features.documentation</code> , 43
	<code>annize.features.documentation.common</code> , 53
	<code>annize.features.documentation.sphinx</code> , 43
	<code>annize.features.documentation.sphinx.common</code> , 46
	<code>annize.features.documentation.sphinx.cpp</code> , 50
	<code>annize.features.documentation.sphinx.doxygen_compat</code> , 51
	<code>annize.features.documentation.sphinx.javascript</code> , 52
	<code>annize.features.documentation.sphinx.output</code> , 43
	<code>annize.features.documentation.sphinx.output.common</code> , 43
	<code>annize.features.documentation.sphinx.output.html</code> , 44
	<code>annize.features.documentation.sphinx.output.pdf</code> , 45
	<code>annize.features.documentation.sphinx.output.plaintext</code> , 46

- annize.features.documentation.sphinx.python, 53
 - 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, 56
 - 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, 61
 - 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, 68
 - annize.features.injections, 68
 - annize.features.injections.common, 68
 - annize.features.injections.python, 69
 - annize.features.licensing, 75
 - annize.features.media_galleries, 77
 - annize.features.task, 77
 - annize.features.testing, 69
 - annize.features.testing.common, 69
 - annize.features.testing.pylint, 70
 - annize.features.testing.pytest, 70
 - annize.features.version, 78
 - 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, 84
 - 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, 121
 - annize.project.file_formats, 112
 - annize.project.file_formats.xml, 112
 - annize.project.inspector, 122
 - annize.project.loader, 124
 - annize.project.materializer, 115
 - annize.project.materializer.behaviors, 116
 - annize.project.materializer.behaviors.argument, 116
 - annize.project.materializer.behaviors.basket, 117
 - annize.project.materializer.behaviors.block, 118
 - annize.project.materializer.behaviors.feature_unavailable, 118
 - annize.project.materializer.behaviors.reference, 118
 - annize.project.materializer.core, 119
 - annize.project.materializer.preprocessors, 120
 - annize.ui, 124
 - annize.ui.apps, 124
 - annize.user_feedback, 126
 - annize.user_feedback.static, 128
 - module_name (annize.features.distributables.python_wheel.ExecutableLink property), 39
 - MonoCliSection (in Modul annize.features.distributables.debian), 27
 - Mount (Klasse in annize.fs.ext), 91
 - move_to() (Methode von annize.fs.Path), 88
 - MPLv11 (in Modul annize.features.licensing), 76
 - MPLv2 (in Modul annize.features.licensing), 76
 - mtime() (Methode von annize.fs.Path), 86
 - multilanguage_frame() (Methode von annize.features.documentation.sphinx.output.common.OutputGenerator), 44
 - multilanguage_frame() (Methode von annize.features.documentation.sphinx.output.html.HtmlOutputGenerator), 45
 - MultipleValuesForSingleArgumentError, 100
- ## N
- name (annize.features.dependencies.python.PythonPackage property), 20
 - name (annize.features.distributables.debian.ExecutableLink property), 24
 - name (annize.features.distributables.debian.MenuEntry property), 23
 - name (annize.features.distributables.debian.Section property), 27
 - name (annize.features.distributables.flatpak.MenuEntry property), 35
 - name (annize.features.licensing.License property), 75

name (*annize.object.parameter_info.ParameterInfo* property), 100
 name (*annize.project.ArgumentNode* property), 108
 name (Attribut von *annize.features.distributables.debian.Package._BuildInfo*), 32
 name (Attribut von *annize.features.distributables.flatpak.FlatpakImage._BuildInfo*), 38
 name (Attribut von *annize.features.distributables.python_wheel.Package._BuildInfo*), 41
 NetworkSection (in Modul *annize.features.distributables.debian*), 29
 new_value (*annize.project.Node.PropertyChangedEvent* property), 104
 NewsgroupsSection (in Modul *annize.features.distributables.debian*), 29
 NoCurrentCultureError, 98
 node (*annize.project.materializer.core.NodeMaterialization* property), 119
 node (*annize.project.Project* property), 102
 Node (Klasse in *annize.project*), 103
 Node.ChangeEvent (Klasse in *annize.project*), 103
 Node.ChildAddedEvent (Klasse in *annize.project*), 103
 Node.ChildRemovedEvent (Klasse in *annize.project*), 104
 Node.ChildrenListChangeEvent (Klasse in *annize.project*), 103
 Node.PropertyChangedEvent (Klasse in *annize.project*), 104
 node_context() (Methode von *annize.project.materializer.behaviors.argument.ArgumentBehavior*), 116
 node_context() (Methode von *annize.project.materializer.behaviors.argument.AssociatedArgumentNodeBehavior*), 117
 node_context() (Methode von *annize.project.materializer.behaviors.basket.BasketBehavior*), 117
 node_context() (Methode von *annize.project.materializer.behaviors.Behavior*), 116
 node_context() (Methode von *annize.project.materializer.behaviors.block.BlockBehavior*), 118
 node_context() (Methode von *annize.project.materializer.behaviors.feature_unavailable.FeatureUnavailableBehavior*), 118
 node_context() (Methode von *annize.project.materializer.behaviors.reference.ReferenceBehavior*), 118
 NodeMaterialization (Klasse in *annize.project.materializer.core*), 119
 nodes (*annize.project.inspector.Inspector.ArgumentMatchings.ArgumentMatchings* property), 122
 normalize_blockscopes() (im Modul *annize.project.materializer.preprocessors*), 121
 NullUserFeedbackController (Klasse in *annize.user_feedback*), 126
 NumericVersionPatternSegment (Klasse in *annize.data.version*), 16
 O
 object_by_name() (im Modul *annize.flow.run_context*), 82
 object_by_name() (Methode von *annize.flow.run_context.RunContext*), 79
 object_metadata() (im Modul *annize.flow.run_context*), 83
 object_metadata() (Methode von *annize.flow.run_context.RunContext*), 81
 object_name() (im Modul *annize.flow.run_context*), 82
 object_name() (Methode von *annize.flow.run_context.RunContext*), 79
 object_names() (im Modul *annize.flow.run_context*), 82
 object_names() (Methode von *annize.flow.run_context.RunContext*), 79
 ObjectNode (Klasse in *annize.project*), 109
 objects_by_type() (im Modul *annize.flow.run_context*), 83
 objects_by_type() (Methode von *annize.flow.run_context.RunContext*), 80
 objects_for_node() (Methode von *annize.project.materializer.MaterializationResult*), 113
 OcamlSection (in Modul *annize.features.distributables.debian*), 29
 old_value (*annize.project.Node.PropertyChangedEvent* property), 104
 OldLibrariesSection (in Modul *annize.features.distributables.debian*), 29
 on_unresolvable (*annize.project.ReferenceNode* property), 110
 OnFeatureUnavailableNode (Klasse in *annize.project*), 110
 OnFeatureUnavailableNode.Action (Klasse in *annize.project*), 111
 OptionalVersionPatternSegment (Klasse in *annize.data.version*), 16
 OtherOSSHardDisksSection (in Modul *annize.features.distributables.debian*), 29
 outdir (Attribut von *annize.features.documentation.sphinx.common.Document.GenerateInplace*), 47
 OutOfContextError, 82

output_spec (annize.features.documentation.sphinx.output.common property), 43
 OutputGenerator (Klasse in annize.features.documentation.sphinx.output.common), 43
 OutputSpec (Klasse in annize.features.documentation.common), 55
P
 Package (Klasse in annize.features.distributables.debian), 30
 Package (Klasse in annize.features.distributables.python_wheel), 40
 Package (Klasse in annize.features.distributables.tar), 43
 package() (Methode von annize.features.distributables.common.PackageStore), 22
 Package._BuildInfo (Klasse in annize.features.distributables.debian), 31
 Package._BuildInfo (Klasse in annize.features.distributables.python_wheel), 40
 package_history() (Methode von annize.features.distributables.common.PackageStore), 22
 package_store (annize.features.distributables.common.Group property), 22
 PackageStore (Klasse in annize.features.distributables.common), 21
 parameter_name (Attribut von annize.object.controller._CreateObjectHelper.ParameterInfo), 99
 ParameterInfo (Klasse in annize.object.parameter_info), 100
 parent (annize.project.Node property), 105
 parse() (im Modul annize.project.file_formats), 112
 parse_file() (Klassenmethode von annize.project.file_formats.FileFormat), 112
 parse_file() (Klassenmethode von annize.project.file_formats.xml.XmlFileFormat), 113
 parse_file() (Methode von annize.project.file_formats.xml._XmlParser), 113
 parser() (im Modul annize.annize_cli), 128
 ParserError, 111
 partname (annize.data.version.NumericVersionPatternSegment property), 16
 parts (annize.features.files.common.Directory property), 60
 path (annize.features.distributables.debian.ExecutableLink property), 24
 path (annize.features.distributables.flatpak.FlatpakImage property), 71
 path (annize.fs.ext.FreshTempDirectory property), 90
 path (annize.project.FileNode property), 108
 Path (Klasse in annize.fs), 86
 path() (Methode von annize.fs.FilesystemContent), 86
 path() (Methode von annize.fs.Path), 86
 Path._TransferHelper (Klasse in annize.fs), 88
 Path.TransferFilters (Klasse in annize.fs), 88
 Path.TransferFilters.And (Klasse in annize.fs), 88
 pattern (annize.data.version.Version property), 18
 PdfOutputGenerator (Klasse in annize.features.documentation.sphinx.output.pdf), 45
 PdfOutputSpec (Klasse in annize.features.documentation.common), 55
 PerlSection (in Modul annize.features.distributables.debian), 29
 PhpSection (in Modul annize.features.distributables.debian), 29
 pkgpath_debian (Attribut von annize.features.distributables.debian.Package._BuildInfo), 32
 pkgpath_documentation (Attribut von annize.features.distributables.debian.Package._BuildInfo), 32
 pkgpath_pixmaps (Attribut von annize.features.distributables.debian.Package._BuildInfo), 32
 pkgpath_setuppy (Attribut von annize.features.distributables.python_wheel.Package._BuildInfo), 41
 pkgpath_share (Attribut von annize.features.distributables.flatpak.FlatpakImage._BuildInfo), 38
 pkgpath_share_applications (Attribut von annize.features.distributables.flatpak.FlatpakImage._BuildInfo), 38
 pkgpath_share_icons (Attribut von annize.features.distributables.flatpak.FlatpakImage._BuildInfo), 38
 pkgpath_usrbin (Attribut von annize.features.distributables.debian.Package._BuildInfo), 33
 pkgrootpath (Attribut von annize.features.distributables.debian.Package._BuildInfo), 32
 pkgrootpath (Attribut von annize.features.distributables.flatpak.FlatpakImage._BuildInfo), 38
 pkgrootpath (Attribut von annize.features.distributables.python_wheel.Package._BuildInfo), 41
 pkgsize (Attribut von annize.features.distributables.flatpak.FlatpakImage), 38

`ze.features.distributables.debian.Package._BuildInfo` problems (annize.project.materializer.core.NodeMaterialization property), 119
 33
PlaintextOutputGenerator (Klasse in annize.features.documentation.sphinx.output.plaintext), processonly_long_str (annize.data.unique.UniqueId property), 15
 46
PlaintextOutputSpec (Klasse in annize.features.documentation.common), 55 processonly_short_str (annize.data.unique.UniqueId property), 15
platform (Attribut von annize.features.distributables.flatpak.FlatpakImage._BuildInfo), (Attribut von annize.project.BlockNodeWithScope.Scope), 110
 38
port (annize.features.files.transfer.ssh.Endpoint property), 57
post_process_generate() (Methode von annize.features.homepage.common.HomepageSection), PROJECT (Attribut von annize.project.BlockNodeWithScope.Scope), 110
 64
post_process_generate() (Methode von annize.features.homepage.sections.download.Section), project_authors() (im Modul annize.features.authors), 72
 62
postinst (Attribut von annize.features.distributables.debian.Package._BuildInfo), project_cultures() (im Modul annize.features.i18n.common), 67
 32
postproc() (Methode von annize.features.documentation.sphinx.output.common.DocumentGenerator), project_directory (annize.features.base.Data property), 73
 44
postproc() (Methode von annize.features.documentation.sphinx.output.pdf.PdfOutputGenerator), project_directory() (im Modul annize.features.base), 75
 45
pre_process_generate() (Methode von annize.features.homepage.common.HomepageSection), project_keywords() (im Modul annize.features.base), 74
 64
pre_process_generate() (Methode von annize.features.homepage.sections.documentation.Section), project_licenses() (im Modul annize.features.licensing), 76
 61
pre_process_generate() (Methode von annize.features.homepage.sections.download.Section), project_name (annize.features.base.Data property), 73
 62
pre_process_generate() (Methode von annize.features.homepage.sections.gallery.Section), project_name() (annize.features.documentation.sphinx.common.Document property), 47
 62
prepare() (Methode von annize.flow.run_context.RunContext), 79 project_name() (im Modul annize.features.base), 74
prepare_generate() (Methode von annize.features.documentation.sphinx.output.common.OutputGenerator), project_name__original (annize.features.documentation.sphinx.common.Document property), 47
 44
prepare_generate() (Methode von annize.features.documentation.sphinx.output.html.HtmlOutputGenerator), project_name__original (annize.features.documentation.sphinx.common.Document property), 47
 45
prerm (Attribut von annize.features.distributables.debian.Package._BuildInfo), ProjectNode (Klasse in annize.project), 106
 32
pretty_project_name (annize.features.base.Data property), 73 project_name__original (annize.features.documentation.sphinx.common.Document property), 47
pretty_project_name() (im Modul annize.features.base), 74 ProjectNode (Klasse in annize.project), 106
 Python (Klasse in annize.features.dependencies.python), 20
Python3ApiReferenceLanguage (Klasse in annize.features.documentation.sphinx.python), 53
PythonPackage (Klasse in annize.features.distributables.flatpak.Repository), 35

- `ze.features.dependencies.python`), 20
PythonSection (in Modul `annize.ze.features.distributables.debian`), 29
- ## R
- `readme_pdf()` (im Modul `annize.asset.data`), 13
ReadmeDocument (Klasse in `annize.ze.features.documentation.sphinx.common`), 50
Recommended (Klasse in `annize.ze.features.dependencies.common`), 20
red (`annize.data.color.Color` property), 13
reference_key (`annize.project.ReferenceNode` property), 110
ReferenceBehavior (Klasse in `annize.ze.project.materializer.behaviors.reference`), 118
ReferenceNode (Klasse in `annize.project`), 109
ReferenceNode.OnUnresolvableAction (Klasse in `annize.project`), 109
regex_string() (Methode von `annize.ze.data.version.AbstractVersionPatternSegment`), 15
regex_string() (Methode von `annize.ze.data.version.ConcatenatedVersionPatternSegment`), 17
regex_string() (Methode von `annize.ze.data.version.NumericVersionPatternSegment`), 16
regex_string() (Methode von `annize.ze.data.version.OptionalVersionPatternSegment`), 17
regex_string() (Methode von `annize.ze.data.version.SeparatorVersionPatternSegment`), 16
region_code (`annize.il8n.Culture` property), 96
register_file_format() (im Modul `annize.ze.project.file_formats`), 112
register_output_generator() (im Modul `annize.ze.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() (Methode von `annize.fs.Path`), 87
remove_change_handler() (Methode von `annize.ze.project.Node`), 104
remove_child() (Methode von `annize.project.Node`), 106
Repository (Klasse in `annize.ze.features.distributables.flatpak`), 35
Required (Klasse in `annize.ze.features.dependencies.common`), 20
resolve_appendtonodes() (im Modul `annize.ze.project.materializer.preprocessors`), 120
resolve_reference_node() (Methode von `annize.ze.project.inspector.Inspector`), 123
resolved_type (`annize.ze.object.parameter_info.ParameterInfo` property), 101
resolved_type (`annize.ze.object.parameter_info.UnionParameterInfo` property), 102
result (`annize.ze.project.materializer.core.NodeMaterialization` property), 119
result (Attribut von `annize.ze.features.distributables.debian.Package._BuildInfo`), 33
result (Attribut von `annize.ze.features.distributables.flatpak.FlatpakImage._BuildInfo`), 38
result (Attribut von `annize.ze.features.distributables.python_wheel.Package._BuildInfo`), 41
root (`annize.features.files.common.DirectoryPart` property), 59
root (`annize.features.files.common.FsEntry` property), 57
root_objects (`annize.ze.project.materializer.MaterializationResult` property), 115
rst_text (`annize.features.homepage.common.HomepageSection.Content` property), 64
RstDocument (Klasse in `annize.ze.features.documentation.sphinx.common`), 49
RstDocumentVariant (Klasse in `annize.ze.features.documentation.sphinx.common`), 49
RstGenerator (Klasse in `annize.ze.features.documentation.sphinx.rst`), 53
RubySection (in Modul `annize.ze.features.distributables.debian`), 29
run() (Methode von `annize.ze.features.testing.common.Test`), 69
run() (Methode von `annize.ze.features.testing.common.TestGroup`), 69
run() (Methode von `annize.features.testing.pylint.Test`), 70
run() (Methode von `annize.features.testing.pytest.Test`), 70
run_runner() (Methode von `annize.ze.flow.runner.Runner`), 84
RunContext (Klasse in `annize.flow.run_context`), 78
Runner (Klasse in `annize.flow.runner`), 84
RunTests (Klasse in `annize.features.testing.common`), 69
RustSection (in Modul `annize.ze.features.distributables.debian`), 29

S

- saturation (*annize.data.color.Color* property), 14
- save() (Methode von *annize.project.Project*), 102
- save() (Methode von *annize.project.ProjectNode*), 107
- save_fileno_to_file() (Methode von *annize.project.file_formats.xml.Marshaler*), 115
- ScalarValueNode (Klasse in *annize.project*), 109
- scalehue() (Methode von *annize.data.color.Color*), 14
- ScienceSection (in Modul *annize.features.distributables.debian*), 29
- scope (*annize.project.BlockNodeWithScope* property), 110
- ScreenLockingCategory (in Modul *annize.features.distributables.debian*), 27
- ScreenSavingCategory (in Modul *annize.features.distributables.debian*), 27
- sdk (Attribut von *annize.features.distributables.flatpak.FlatpakImage._BuildInfo*), 38
- section (Attribut von *annize.features.distributables.debian.Package._BuildInfo*), 32
- Section (Klasse in *annize.features.distributables.debian*), 27
- Section (Klasse in *annize.features.homepage.sections.about*), 61
- Section (Klasse in *annize.features.homepage.sections.changelog*), 61
- Section (Klasse in *annize.features.homepage.sections.documentation*), 61
- Section (Klasse in *annize.features.homepage.sections.download*), 61
- Section (Klasse in *annize.features.homepage.sections.gallery*), 62
- Section (Klasse in *annize.features.homepage.sections.imprint*), 62
- Section (Klasse 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), 16
- segment_names (*annize.data.version.OptionalVersionPatternSegment* property), 17
- 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() (Methode von *annize.data.version.AbstractVersionPatternSegment*), 15
- segments_tuples_to_text() (Methode von *annize.data.version.ConcatenatedVersionPatternSegment*), 17
- segments_tuples_to_text() (Methode von *annize.data.version.NumericVersionPatternSegment*), 16
- segments_tuples_to_text() (Methode von *annize.data.version.OptionalVersionPatternSegment*), 17
- segments_tuples_to_text() (Methode von *annize.data.version.SeparatorVersionPatternSegment*), 16
- segments_tuples_to_text() (Methode von *annize.data.version.VersionPattern*), 18
- segments_values (*annize.data.version.Version* property), 18
- SeparatorVersionPatternSegment (Klasse in *annize.data.version*), 16
- ServiceDescription (Klasse in *annize.features.distributables.debian*), 30
- services (Attribut von *annize.features.distributables.debian.Package._BuildInfo*), 32
- set_materialized_result() (Methode von *annize.project.materializer.core.NodeMaterialization*), 119
- set_object_metadata() (im Modul *annize.flow.run_context*), 83
- set_object_metadata() (Methode von *annize.flow.run_context.RunContext*), 81
- set_object_name() (im Modul *annize.flow.run_context*), 82
- set_object_name() (Methode von *annize.flow.run_context.RunContext*), 80
- set_problems() (Methode von *annize.project.materializer.core.NodeMaterialization*), 119
- set_selected_task() (Methode von *annize.flow.runner.Runner*), 85
- setuppy_conf (Attribut von *annize.features.distributables.python_wheel.Package._BuildInfo*),

41
 Share (Klasse in *annize.features.distributables.flatpak*), 35
 shares (Attribut von *annize.features.distributables.flatpak.FlatpakImage._BuildInfo*), 21
 38
 ShellsSection (in Modul *annize.features.distributables.debian*), 29
 short_desc (*annize.features.documentation.sphinx.output.html.HtmlOutputSpec* property), 45
 short_desc (*annize.features.homepage.common.HomepageSummary* (Attribut von *annize.features.distributables.debian.Package._BuildInfo*), property), 65
 short_str (*annize.data.unique.UniqueId* property), 15
 short_title (*annize.features.documentation.sphinx.output.html.HtmlOutputSpec* property), 45
 show_task_chooser() (Methode von *annize.flow.runner.Runner*), 84
 show_task_execution() (Methode von *annize.flow.runner.Runner*), 84
 show_task_execution_success() (Methode von *annize.flow.runner.Runner*), 84
 SimpleProjectHomepage (Klasse in *annize.features.homepage.common*), 66
 SKIP (Attribut von *annize.project.ReferenceNode.OnUnresolvableAction*), 110
 SKIP_BLOCK (Attribut von *annize.project.OnFeatureUnavailableNode.Action*), 111
 SKIP_NODE (Attribut von *annize.project.OnFeatureUnavailableNode.Action*), 111
 sockets (Attribut von *annize.features.distributables.flatpak.FlatpakImage._BuildInfo*), 38
 sort_index (*annize.features.homepage.common.HomepageSection* property), 64
 SoundSection (in Modul *annize.features.distributables.debian*), 29
 source (*annize.features.documentation.sphinx.common.ApiReferenceLanguage.ApiReferenceGenerateInfo* property), 48
 source (*annize.features.documentation.sphinx.common.RstDocumentVariant* property), 49
 source (Attribut von *annize.features.distributables.debian.Package._BuildInfo*), 32
 source (Attribut von *annize.features.distributables.flatpak.FlatpakImage._BuildInfo*), 38
 source (Attribut von *annize.features.distributables.python_wheel.Package._BuildInfo*), 41
 source_path (*annize.features.files.common.DirectoryPart* property), 59
 StaticUserFeedbackController (Klasse in *annize.user_feedback.static*), 128
 store_package() (Methode von *annize.features.distributables.common.PackageStore*), 21
 String (Klasse in *annize.features.i18n.common*), 67
 string_name (*annize.i18n.ProvidedTrStr* property), 94
 studio() (Methode von *annize.annize_cli.Commands*), 128
 summary (*annize.features.base.Data* property), 73
 summary (Attribut von *annize.features.distributables.debian.Package._BuildInfo*), 32
 summary (*annize.features.base.Data* property), 74
 T
 target_node (*annize.project.Node.ChangeEvent* property), 103
 Task (Klasse in *annize.features.task*), 77
 TasksSection (in Modul *annize.features.distributables.debian*), 30
 TCultureSpec (in Modul *annize.i18n*), 93
 temp_clone() (Methode von *annize.fs.Path*), 87
 Test (Klasse in *annize.features.testing.common*), 69
 Test (Klasse in *annize.features.testing.pylint*), 70
 Test (Klasse in *annize.features.testing.pytest*), 70
 TestGroup (Klasse in *annize.features.testing.common*), 69
 TexSection (in Modul *annize.features.distributables.debian*), 30
 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() (Methode von *annize.data.version.VersionPattern*), 17
 TextProcessingSection (in Modul *annize.features.distributables.debian*), 30
 TextSource (Klasse in *annize.features.i18n.gettext*), 68
 theme (*annize.features.documentation.sphinx.output.html.HtmlOutputSpec* property), 45
 time (*annize.features.changelog.common.Entry* property), 19
 title (*annize.features.distributables.common.Group* property), 22
 title (*annize.features.distributables.debian.MenuEntry* property), 23
 title (*annize.features.distributables.flatpak.MenuEntry* property), 35
 title (*annize.features.documentation.sphinx.common.Document* property), 47

title (*annize.features.documentation.sphinx.common.ReadableUniqueId* (Klasse in *annize.data.unique*), 15
property), 50
title (*annize.features.homepage.common.Homepage*
property), 65
title (*annize.features.media_galleries.Gallery* *proper-*
ty), 77
title__original (*annize.features.documentation.sphinx.common.Document*
property), 47
to_trstr() (*im Modul annize.i18n*), 95
token (*annize.features.distributables.store.pypi.Connection*
property), 21
tr() (*im Modul annize.i18n*), 93
tr() (*statische Methode von annize.i18n.TrStr*), 93
tr_if_trstr() (*im Modul annize.i18n*), 95
transfer_action_copy() (*statische Methode von an-*
nize.fs.Path._TransferHelper), 88
transfer_action_move() (*statische Methode von an-*
nize.fs.Path._TransferHelper), 89
transfer_to() (*statische Methode von anni-*
ze.fs.Path._TransferHelper), 88
transformed() (*Methode von annize.data.color.Color*),
14
translate() (*Methode von anni-*
ze.features.i18n.common._ProjectDefinedTranslationProvider),
66
translate() (*Methode von anni-*
ze.i18n.GettextTranslationProvider), 92
translate() (*Methode von anni-*
ze.i18n.TranslationProvider), 92
translate() (*Methode von annize.i18n.TrStr*), 94
translation_providers() (*im Modul annize.i18n*), 93
TranslationProvider (*Klasse in annize.i18n*), 92
TranslationUnavailableError, 99
TrStr (*Klasse in annize.i18n*), 93
TrStrOrStr (*in Modul annize.i18n*), 94
try_get_materialization_for_node()
(*Methode von anni-*
ze.project.materializer.core.NodeMaterialization),
119
TTransferFilter (*Attribut von annize.fs.Path*), 87
type (*annize.project.inspector.Inspector.TypeInfo* *proper-*
ty), 122
type_name (*annize.project.ObjectNode* *property*), 109
type_parameter_infos() (*im Modul anni-*
ze.object.parameter_info), 102
typename (*annize.project.inspector.Inspector.TypeInfo*
property), 122

U
undo_changes() (*Methode von anni-*
ze.project.ProjectNode), 107
UnionParameterInfo (*Klasse in anni-*
ze.object.parameter_info), 101
UnresolvableReferenceError, 111
UnsatisfiableUserFeedbackAttemptError, 126
UnspecifiedCulture (*Klasse in annize.i18n*), 97
UpdatePOs (*Klasse in annize.features.i18n.gettext*), 68
Upload (*Klasse in anni-*
ze.features.distributables.store.pypi), 21
upload (*Klasse in annize.features.files.transfer.common*),
56
upload() (*Methode von anni-*
ze.features.distributables.flatpak.LocalRepository),
36
upload() (*Methode von anni-*
ze.features.distributables.flatpak.Repository),
35
UserFeedbackController (*Klasse in anni-*
ze.user_feedback), 126
username (*annize.features.files.transfer.ssh.Endpoint*
property), 57
UtilitiesSection (*in Modul anni-*
ze.features.distributables.debian), 30

V
value (*annize.project.ScalarValueNode* *property*), 109
version (*annize.features.changelog.common.Entry* *proper-*
ty), 19
version (*annize.features.dependencies.python.PythonPackage*
property), 21
version (*annize.features.documentation.sphinx.common.Document*
property), 47
version (*annize.features.version.Line* *property*), 78
version (*Attribut von anni-*
ze.features.distributables.debian.Package._BuildInfo),
32
version (*Attribut von anni-*
ze.features.distributables.python_wheel.Package._BuildInfo),
41
Version (*Klasse in annize.data.version*), 18
Version (*Klasse in annize.features.version*), 78
VersionControlSystem (*Klasse in anni-*
ze.features.version_control.common), 70
VersionControlSystem (*Klasse in anni-*
ze.features.version_control.git), 71
VersionControlSystemsSection (*in Modul anni-*
ze.features.distributables.debian), 30
VersionPattern (*Klasse in annize.data.version*), 17
VIDEO (*Attribut von anni-*
ze.features.media_galleries.MediaType),
77
VideoSection (*in Modul anni-*
ze.features.distributables.debian), 30
VirtualPackagesSection (*in Modul anni-*
ze.features.distributables.debian), 30

W

`wait_finished()` (Methode von `annize.flow.runner.Runner`), 85

`WebServersSection` (in Modul `annize.features.distributables.debian`), 28

`WebSoftwareSection` (in Modul `annize.features.distributables.debian`), 30

`with_modified()` (Methode von `annize.data.color.Color`), 14

`with_updates()` (Methode von `annize.object.controller._CreateObjectHelper.ParameterConfig`), 99

`write_file()` (Methode von `annize.fs.Path`), 87

X

`XfceSection` (in Modul `annize.features.distributables.debian`), 30

`XmlFileFormat` (Klasse in `annize.project.file_formats.xml`), 112

`XWindowSystemSoftwareSection` (in Modul `annize.features.distributables.debian`), 30

Z

`ZopePloneFrameworkSection` (in Modul `annize.features.distributables.debian`), 30