pyjen

Release 2.0.0

Kevin S. Phillips

May 22, 2022

CONTENTS

CHAPTER

ONE

TABLE OF CONTENTS

1.1 Examples

1.1.1 Display a list of all jobs on the default view

```
from pyjen.jenkins import Jenkins
jk = Jenkins("http://localhost:8080")
vw = jk.default_view
jobs = vw.jobs

for j in jobs:
    print(j.name)
```

1.1.2 Disable all jobs in a view named “My View”

```
from pyjen.jenkins import Jenkins
jk = Jenkins("http://localhost:8080")
vw = jk.find_view("My View")
vw.disable_all_jobs()
```

1.1.3 Get all upstream dependencies of a job named “JobA”

```
from pyjen.jenkins import Jenkins
jen = Jenkins("http://localhost:8080")
jb = jen.find_job("JobA")
upstream = jb.all_upstream_jobs

for u in upstream:
    print u.name
```

1.1.4 Clone all jobs in a view who are named with a 'trunk' identifier for a new branch configuration

```
from pyjen.jenkins import Jenkins
j = Jenkins("http://localhost:8080")
v = j.find_view("trunk_builds")
v.clone_all_jobs("trunk", "branch")
```

1.1.5 Locate a nested subview on a Jenkins instance that uses the NestedView plugin

```
from pyjen.utils.helpers import find_view
v = find_view("http://localhost:8080", ('user', 'pw'), "MySubView")
print(v.name)
```

1.2 Contributors Guide

Developers who are interested in contributing to the PyJen project should start by contacting the project maintainer here. Source for the project can be found on GitHub.

To start working on an improvement for the project, start by forking the project and committing your work there. When you are happy with the changes you have made create a pull request and assign it to the maintainer. Once approved, the changes will be integrated into the next release.

All code is expected to be PEP-8 compliant. This requirement is enforced automatically by our continuous integration builds with the help of PyLint. Pull requests will not be approved when continuous integration builds fail. Further, we ask that all docstrings be compatible with the Sphinx API-doc plugin to facilitate automatic document generation by our scripts and hosting sites. Finally, we encourage contributors to add sufficient unit test coverage for any changes they make using the py.test framework used by this project.

Seeing as how PyJen supports the latest versions of both Python 2 and Python 3, all code contributions must be compatible with both of these versions. Finally, we try our best to ensure the API is compatible with both the LTS and Latest editions of the Jenkins REST API, so care should be taken to make sure contributed code - especially those supporting new Jenkins plugins - is compatible with both of these versions wherever possible.

1.2.1 Development Environment

The project, including all build tools, are expected to work correctly on all major operating systems (Windows, Linux, MacOS) and under all recent versions of Python (2.7, 3.4+). Further, some unit tests require the Docker client tools to be installed as well. See the section on "testing" below for details.

Once you have your host configured correctly, we recommend using a Python virtual environment for all of your development work. Maintainers of the project are currently using virtualenv however any similar virtualization tool should work. Below are the basic steps we recommend contributors follow to set up a compatible build environment:

1. make sure you have the 'virtualenv' tool installed

```
pip install virtualenv
```

2. Next, create a local virtual environment under your working folder for the desired Python version. In most systems this can be done as follows:

```
virtualenv -p python3 ./venv3
```

3. Activate the new virtual environment

```
Linux/Mac: source ./venv3/bin/activate
Windows: .\venv3\bin\activate.bat
```

4. install the required build time dependencies. There are 2 requirements files in the ./tests folder to simplify this process: python2.reqs and python3.reqs. The former defines all pegged revisions of all dependencies needed to build and test the project under a Python 2 runtime. The latter defines a similar set of dependencies for the Python 3 runtime. These can easily be installed using pip as follows:

```
pip install -r ./tests/python3.reqs
```

5. Finally, you should be ready to try performing a test run of the unit tests.

```
tox -r -e py3
```

For more details on different aspects of the project, including more details on how the test framework works, dependency management, as well as some high level architectural information to get you started writing plugins, see the other sections below.

1.2.2 Testing

We endeavor to have as much test coverage of the library and plugin code as possible. To help simplify testing and improve coverage metrics we've adopted several different testing strategies.

Currently, all tests are orchestrated by tox. Under the hood tox runs static code analysis as well as automated unit tests. The tests are further orchestrated using pytest. Further, some tests in the suite require a live Jenkins service to test against. This service is automatically created and configured by the pytest framework. To run these tests you need only install the Docker client for your development system and make sure the service is running before launching tox.

For examples on how to write tests for various parts of the framework and plugins, we encourage you to review the existing tests and find similar ones that you can use as guides. Simply copy a test that performs a similar operation to what you want to test, rename the test and update the implementation to satisfy your new test case.

Test Customizations

Several custom pytest parameters have been added to our test runner to make working with the project easier for contributors. They are as follows:

```
tox -e py3 -- --skip-docker
```

This handy flag allows developers to run the unit tests that do not depend on the Jenkins service that runs under Docker. This can be helpful for contributors who can't or don't want to install and configure Docker on their local machines. Also, tests that require the Docker service tend to be slower than their non-dockerized counterparts, so it can be helpful to run the faster tests as an initial sanity check for new changes being made.

```
tox -e py3 -- --preserve
```

This flag forces the pytest runner to keep the Docker container used for tests running after the test run is complete. This can be handy for debugging purposes allowing developers to examine the contents of the container after the tests have been executed to see what state the service is in, the state of the file system in the container, etc. Also, when this flag

is enabled subsequent runs of the tests will re-use the same container, which can help improve build times when doing local testing.

```
tox -e py3 -- --jenkins-version jenkins:alpine
```

This test option allows us to customize the version of the Jenkins service we test against more easily. By providing the name and tag of the Docker image to use for testing, we can force the test runner to use that exact container for all tests, superseding the default one. Have a bug with a plugin that is only reproducible in v2.150.3, then just run "tox -e py3 –-jenkins-version jenkins:2.150.3-alpine" to try and reproduce it.

1.2.3 Dependency Management

To ensure consistent build results on all platforms and on all continuous integrations servers, we peg all of the build time dependencies needed to build and test the project as specific versions. These pegged revisions are stored in the following files:

- ./tests/python2.reqs - all dependencies needed to build under Python 2.x
- ./tests/python3.reqs - all dependencies needed to build under Python 3.x

To make it easier to manage these requirements files, we have a custom shell script in the root folder that will auto-generate a new set of dependencies for the project based on the package dependencies defined in the project.prop file in the root of the project. The shell script will update the dependency list with all of the latest versions of all dependencies automatically.

1.2.4 Plugins

Just as found in the Jenkins back end implementation, most custom functionality in PyJen will be provided by plugins. PyJen supports a plugin system that essentially mirrors the Jenkins system which allows developers to write their own classes to wrap the REST API for any Jenkins plugin they may like.

Plugins may be packaged independently from the PyJen package or included with the package. Plugins included here are guaranteed to be covered by the same quality metrics and standards as the main library itself, which should improve the confidence users have in them. Standalone plugins packaged separately will be written by third parties and thus may vary greatly in quality and features.

Plugins included directly with the PyJen library are simply Python classes that meet the following criteria:

- the class declarations must be placed in a module under the src/pyjen/plugins subfolder
- the class must derive, directly or indirectly, from the *XMLPlugin* abstract base class

This second requirement forces derived classes to implement specific criteria to implement the required abstract interface. Currently this interface simply has two requirements:

- a static property named 'type' of type `str` containing the character representation of the Jenkins plugin managed by the PyJen plugin
- a constructor compatible with the type of plugin being managed (in most cases, this is a single parameter of type `xml.etree.ElementTree.Element`.)

Beyond that, plugin implementers can then proceed to implement public methods and properties on their plugin class to expose functionality particular to the plugin.

Using Plugins

Any primitive or operation in Jenkins that supports a plugable interface is equally addressable by the associated PyJen interface without further customization by the plugin author. For example, to add support for a new type of 'builder', simply write your plugin class as described above and it will automatically be accessible from the *builders()* property.

This is accomplished by leveraging the metadata embedded in the Jenkins configuration information for each primitive such as a view or a job. The back-end Java plugins supported by Jenkins embed type information in the configuration metadata which maps directly onto PyJen plugin classes. So when you use PyJen to request data from the Jenkins REST API it will automatically look for and load any plugin that the active Jenkins instance may be using without further modification to the PyJen API.

1.3 pyjen

1.3.1 pyjen package

Subpackages

pyjen.plugins package

Submodules

pyjen.plugins.allview module

Class that interact with Jenkins views of type "AllView"

class pyjen.plugins.allview.**AllView**(*api*)

Bases: *pyjen.view.View*

view which displays all jobs managed by this Jenkins instance

Parameters **api** (JenkinsAPI) – Pre-initialized connection to the Jenkins REST API

static get_jenkins_plugin_name()

str: the name of the Jenkins plugin associated with this PyJen plugin

This static method is used by the PyJen plugin API to associate this class with a specific Jenkins plugin, as it is encoded in the config.xml

pyjen.plugins.allview.**PluginClass**

alias of *pyjen.plugins.allview.AllView*

pyjen.plugins.artifactarchiver module

Interface to the Jenkins 'archive artifacts' publishing plugin

class pyjen.plugins.artifactarchiver.**ArtifactArchiverPublisher**(*node*)

Bases: *pyjen.utils.xml_plugin.XMLPlugin*

Interface to the Jenkins 'archive artifacts' publishing plugin

This plugin is a default, built-in plugin which is part of the Jenkins core

Parameters **node** (*xml.etree.ElementTree.Element*) – XML node with the decoded XML data associated with a plugin

property artifact_regex

the regular expression used to locate files to archive

Type str

static get_jenkins_plugin_name()

str: the name of the Jenkins plugin associated with this PyJen plugin

This static method is used by the PyJen plugin API to associate this class with a specific Jenkins plugin, as it is encoded in the config.xml

classmethod instantiate(*file_pattern*)

Factory method for creating a new artifact archiver

Parameters **file_pattern** (*str*) – regular expression matching files to be archived at the end of the build

Returns Reference to the newly created publisher

Return type *ArtifactArchiverPublisher*

pyjen.plugins.artifactarchiver.**PluginClass**

alias of *pyjen.plugins.artifactarchiver.ArtifactArchiverPublisher*

pyjen.plugins.artifactdeployer module

properties of the 'artifact deployer' publishing plugin

class pyjen.plugins.artifactdeployer.**ArtifactDeployer**(*node*)

Bases: *pyjen.utils.xml_plugin.XMLPlugin*

Interface to the Jenkins 'artifact deployer' publishing plugin

https://plugins.jenkins.io/artifactdeployer

Parameters **node** (*xml.etree.ElementTree.Element*) – XML node with the decoded XML data associated with a plugin

add_entry(*new_entry*)

Adds a new deployer entry to this publisher

Parameters **new_entry** (ArtifactDeployerEntry) – New publisher descriptor entry to be added

property entries

list of deployment options associated with this plugin

Type list (*ArtifactDeployerEntry*)

static get_jenkins_plugin_name()

str: the name of the Jenkins plugin associated with this PyJen plugin

This static method is used by the PyJen plugin API to associate this class with a specific Jenkins plugin, as it is encoded in the config.xml

classmethod instantiate()

Factory method for creating a new artifact deployer

Returns reference to the newly created deployer

Return type *ArtifactDeployer*

class pyjen.plugins.artifactdeployer.**ArtifactDeployerEntry**(*node*)

Bases: *pyjen.utils.xml_plugin.XMLPlugin*

a single artifacts to be deployed by an Artifact Deployer instance

Parameters **node** (*xml.etree.ElementTree.Element*) – XML node with the decoded XML data associated with a plugin

property includes

the path or regular expression describing files to be published

Type str

classmethod instantiate(*include_pattern*, *remote_path*)

Factory method used to instantiate instances of this class

Parameters

- **include_pattern** (*str*) – Path or regular expression of the file(s) to be published
- **remote_path** (*str*) – Path to remote share where files are to be published

Returns instance of the artifact deployer entry

Return type *ArtifactDeployerEntry*

property remote

the remote location where these artifacts are to be published

Type str

pyjen.plugins.artifactdeployer.**PluginClass**

alias of *pyjen.plugins.artifactdeployer.ArtifactDeployer*

pyjen.plugins.buildblocker module

Interfaces for interacting with Build Blockers job property plugin

class pyjen.plugins.buildblocker.**BuildBlockerProperty**(*node*)

Bases: *pyjen.utils.xml_plugin.XMLPlugin*

Wrapper for Build Blocker job properties

https://wiki.jenkins-ci.org/display/JENKINS/Build+Blocker+Plugin

Parameters **node** (*xml.etree.ElementTree.Element*) – XML node with the decoded XML data associated with a plugin

LEVEL_TYPES = ('GLOBAL', 'NODE')

QUEUE_SCAN_TYPES = ('DISABLED', 'ALL', 'BUILDABLE')

property blockers

list of search criteria for blocking jobs

Type list (str)

disable()

Disables this set of build blockers

enable()

Enables this set of build blockers

static get_jenkins_plugin_name()

str: the name of the Jenkins plugin associated with this PyJen plugin

This static method is used by the PyJen plugin API to associate this class with a specific Jenkins plugin, as it is encoded in the config.xml

classmethod instantiate(*patterns*)

Factory method used to instantiate an instance of this plugin

Parameters **patterns** (*list*, *str*) – One or more names or regular expressions for jobs that block the execution of this one.

Returns reference to the newly instantiated object

Return type *BuildBlockerProperty*

property is_enabled

True if these blocking jobs are enabled, False if not

Type bool

property level

the scope of the blocked job settings. One of BuildBlockerProperty.LEVEL_TYPES

Type str

property queue_scan

checks to see whether build blocking scans the build queue or not. One of BuildBlockerProperty.QUEUE_SCAN_TYPES.

Type str

pyjen.plugins.buildblocker.**PluginClass**

alias of *pyjen.plugins.buildblocker.BuildBlockerProperty*

pyjen.plugins.buildtriggerpublisher module

Interface to the Jenkins 'build trigger' publishing plugin

class pyjen.plugins.buildtriggerpublisher.**BuildTriggerPublisher**(*node*)

Bases: *pyjen.utils.xml_plugin.XMLPlugin*

Interface to the Jenkins 'build trigger' publishing plugin

This plugin is a default, built-in plugin which is part of the Jenkins core

Parameters **node** (*xml.etree.ElementTree.Element*) – XML node with the decoded XML data associated with a plugin

static get_jenkins_plugin_name()

str: the name of the Jenkins plugin associated with this PyJen plugin

This static method is used by the PyJen plugin API to associate this class with a specific Jenkins plugin, as it is encoded in the config.xml

classmethod instantiate(*project_names*)

Factory method for creating a new build trigger

The default trigger will run when the parent job is successful

Parameters **project_names** (list of str) – List of 1 or more names of jobs to trigger

Returns reference to the newly instantiated object

Return type *BuildTriggerPublisher*

property job_names

names of 0 or more downstream jobs managed by this config

Type list (str)

pyjen.plugins.buildtriggerpublisher.**PluginClass**

alias of *pyjen.plugins.buildtriggerpublisher.BuildTriggerPublisher*

pyjen.plugins.conditionalbuilder module

Primitives for operating on Jenkins job builder of type 'Conditional Builder'

class pyjen.plugins.conditionalbuilder.**ConditionalBuilder**(*node*)

Bases: *pyjen.utils.xml_plugin.XMLPlugin*

Jenkins job builder plugin

capable of conditionally executing a build operation

https://wiki.jenkins-ci.org/display/JENKINS/Conditional+BuildStep+Plugin

Parameters **node** (*xml.etree.ElementTree.Element*) – XML node with the decoded XML data associated with a plugin

property builder

PyJen plugin describing the build step associated with this condition

Type *XMLPlugin*

property condition

PyJen plugin describing the conditions for this build step

Type *XMLPlugin*

static get_jenkins_plugin_name()

str: the name of the Jenkins plugin associated with this PyJen plugin

This static method is used by the PyJen plugin API to associate this class with a specific Jenkins plugin, as it is encoded in the config.xml

classmethod instantiate(*condition*, *builder*)

Factory method for creating a new conditional build step

Parameters

- **condition** (XMLPlugin) – Condition to be applied to the build step. The build step will only be executed if the terms defined by this condition evaluate to True
- **builder** (XMLPlugin) – Nested job build step to be executed conditionally May be any PyJen plugin that defines / manages a Job build step

Returns newly created conditional build step

Return type *ConditionalBuilder*

pyjen.plugins.conditionalbuilder.**PluginClass**

alias of *pyjen.plugins.conditionalbuilder.ConditionalBuilder*

pyjen.plugins.flexiblepublish module

Primitives for operating on job publishers of type 'Flexible Publisher'

class pyjen.plugins.flexiblepublish.**ConditionalAction**(*node*)

Bases: *pyjen.utils.xml_plugin.XMLPlugin*

conditional action associated with a flexible publisher

Contains 1 or more publishers to be run if a certain set of conditions is met.

Parameters **node** (*xml.etree.ElementTree.Element*) – XML node with the decoded XML data associated with a plugin

classmethod **instantiate**(*condition*, *actions*)

Factory method for creating a new instances of this class

Parameters

- **condition** (XMLPlugin) – Flexible publish build condition pre-configured to control this publish operation
- **actions** (list of *XMLPlugin*) – List of 1 or more "build stage" plugins that you would like to use in the publish phase of a Jenkins job

Returns reference to the newly instantiated object

Return type *ConditionalAction*

property **publishers**

publishers to run should the conditions associated with this action are met

Type list (*XMLPlugin*)

class pyjen.plugins.flexiblepublish.**FlexiblePublisher**(*node*)

Bases: *pyjen.utils.xml_plugin.XMLPlugin*

Job plugin enabling conditional execution of post-build steps

https://plugins.jenkins.io/flexible-publish

Parameters **node** (*xml.etree.ElementTree.Element*) – XML node with the decoded XML data associated with a plugin

property **actions**

list of conditional actions associated with this publisher

Type list (*ConditionalAction*)

static get_jenkins_plugin_name()

str: the name of the Jenkins plugin associated with this PyJen plugin

This static method is used by the PyJen plugin API to associate this class with a specific Jenkins plugin, as it is encoded in the config.xml

classmethod instantiate(*actions*)

Factory method for creating a new instances of this class

Parameters **actions** (list of *ConditionalAction*) – list of conditional actions to perform under this publisher

Returns reference to the newly instantiated object

Return type *FlexiblePublisher*

pyjen.plugins.flexiblepublish.**PluginClass**

alias of *pyjen.plugins.flexiblepublish.FlexiblePublisher*

pyjen.plugins.folderjob module

Primitives that manage Jenkins job of type 'Folder'

class pyjen.plugins.folderjob.**FolderJob**(*api*)

Bases: *pyjen.job.Job*

Jenkins job of type 'folder'

Parameters **api** (JenkinsAPI) – Pre-initialized connection to the Jenkins REST API

create_job(*job_name*, *job_class*)

Creates a new job on the Jenkins dashboard

Parameters

- **job_name** (*str*) – The name for this new job. This name should be unique, different from any other jobs currently managed by the Jenkins instance
- **job_class** – PyJen plugin class associated with the type of job to be created

Returns An object to manage the newly created job

Return type *Job*

find_job(*job_name*)

Searches all jobs managed by this Jenkins instance for a specific job

Parameters **job_name** (*str*) – the name of the job to search for

Returns If a job with the specified name can be found, and object to manage the job will be returned. If no job with the specified name can be found, will return None.

Return type *Job*

static get_jenkins_plugin_name()

str: the name of the Jenkins plugin associated with this PyJen plugin

This static method is used by the PyJen plugin API to associate this class with a specific Jenkins plugin, as it is encoded in the config.xml

property jobs

list of all jobs contained inthis folder

Type list (*Job*)

static template_config_xml()

Returns a basic XML configuration template for use when instantiating jobs of this type

Return type str

pyjen.plugins.folderjob.**PluginClass**

alias of *pyjen.plugins.folderjob.FolderJob*

pyjen.plugins.freestylejob module

Primitives that manage Jenkins job of type 'Freestyle'

class pyjen.plugins.freestylejob.**FreestyleJob**(*api*)

Bases: *pyjen.job.Job*

Jenkins job of type 'freestyle'

Parameters **api** (JenkinsAPI) – Pre-initialized connection to the Jenkins REST API

add_builder(*builder*)

Adds a new build step to this job

Parameters **builder** (XMLPlugin) – PyJen plugin describing the build step to add

add_publisher(*publisher*)

Adds a new job publisher to this job

Parameters **publisher** (XMLPlugin) – job publisher to add

property all_downstream_jobs

list of all jobs that depend on this job, recursively

Includes jobs triggered by this job, and all jobs triggered by those jobs, recursively for all downstream dependencies

Type list (*Job*)

property all_upstream_jobs

list of all jobs that this job depends on, recursively

Includes jobs that trigger this job, and all jobs trigger those jobs, recursively for all upstream dependencies

Type list (*Job*)

property assigned_node

the custom node label restricting which nodes this job can run against

Type str

property assigned_node_enabled

Checks to see if this job has a custom node restriction

Type bool

property builders

PyJen plugins that manage the various 'builders' for this job

Type list (*XMLPlugin*)

property custom_workspace

the custom workspace associated with this job

May return an empty character string if the custom workspace feature is not currently enabled.

Type str

property custom_workspace_enabled

Checks to see if this job has the custom workspace option enabled

Type bool

property downstream_jobs

list of jobs to be triggered after this job completes

Type list (*Job*)

static get_jenkins_plugin_name()

str: the name of the Jenkins plugin associated with this PyJen plugin

This static method is used by the PyJen plugin API to associate this class with a specific Jenkins plugin, as it is encoded in the config.xml

property publishers

all plugins configured as 'publishers' for this job

Type list (*XMLPlugin*)

property quiet_period

the delay, in seconds, builds of this job wait in the queue before being run. Returns -1 if there is no custom quiet period for this job

Type int

property quiet_period_enabled

Checks to see if a custom quiet period is defined on this job

Type bool

property scm

PyJen plugin describing the source code repository configuration for the job

Type *XMLPlugin*

static template_config_xml()

Returns XML configuration template for instantiating jobs of this type

Return type str

property upstream_jobs

list of upstream dependencies for this job

Type list (*Job*)

class pyjen.plugins.freestylejob.**FreestyleXML**(*api*)

Bases: *pyjen.utils.jobxml.JobXML*

Abstraction around the config.xml for this type of Jenkins job

Parameters **api** (JenkinsAPI) – Rest API for the Jenkins XML configuration managed by this object

add_builder(*builder*)

Adds a new builder node to the build steps section of the job XML

Parameters **builder** (XMLPlugin) – PyJen plugin implementing the new job builder to be added

add_publisher(*new_publisher*)

Adds a new publisher node to the publisher section of the job XML

Parameters **new_publisher** (XMLPlugin) – PyJen plugin which supports the Jenkins publisher API

property assigned_node

the build agent label this job is associated with

Type str

property builders

PyJen plugins that manage the various 'builders' for this job

Type list (*XMLPlugin*)

property custom_workspace

the local path for the custom workspace associated with this job. Returns None if the custom workspace option is not enabled

Type str

disable_assigned_node()

Disables a custom node assignment on this job

disable_custom_workspace()

Disables a jobs use of a custom workspace

disable_quiet_period()

Disables custom quiet period on a job

property publishers

0 or more post-build publishers associated with this job. Each element will be an instance of a compatible PyJen plugin for each publisher. Publishers with no valid PyJen plugin installed will be ignored

Type list (*XMLPlugin*)

property quiet_period

the delay, in seconds, this job waits in queue before running a build

May return None if no custom quiet period is defined. At the time of this writing the default value is 5 seconds however this may change over time.

Type int

property scm

Retrieves the appropriate plugin for the SCM portion of a job

Detects which source code management tool is being used by this job, locates the appropriate plugin for that tool, and returns an instance of the wrapper for that plugin pre-configured with the settings found in the relevant XML subtree.

Returns

One of any number of plugin objects responsible for providing extensions to the source code management portion of a job

Example: *Subversion* Example: *GitSCM*

Return type *XMLPlugin*

pyjen.plugins.freestylejob.**PluginClass**

alias of *pyjen.plugins.freestylejob.FreestyleJob*

pyjen.plugins.gitscm module

SCM properties for jobs which pull sources from a Git repository

class pyjen.plugins.gitscm.**GitSCM**(*node*)

Bases: *pyjen.utils.xml_plugin.XMLPlugin*

SCM properties for jobs which pull sources from a Git repository

Parameters **node** (*xml.etree.ElementTree.Element*) – XML node with the decoded XML data associated with a plugin

static get_jenkins_plugin_name()

str: the name of the Jenkins plugin associated with this PyJen plugin

This static method is used by the PyJen plugin API to associate this class with a specific Jenkins plugin, as it is encoded in the config.xml

classmethod instantiate(*repository_url*)

Factory method for creating a new Git SCM code block

Parameters **repository_url** (*str*) – URI of the repository to check out during the build

Returns reference to the newly instantiated object

Return type *GitSCM*

property url

the repository URL for the git config

Type str

pyjen.plugins.gitscm.**PluginClass**

alias of *pyjen.plugins.gitscm.GitSCM*

pyjen.plugins.listview module

Primitives that operate on Jenkins views of type 'List'

class pyjen.plugins.listview.**ListView**(*api*)

Bases: *pyjen.view.View*

all Jenkins related 'view' information for views of type ListView

Parameters **api** (JenkinsAPI) – Pre-initialized connection to the Jenkins REST API

static get_jenkins_plugin_name()

str: the name of the Jenkins plugin associated with this PyJen plugin

This static method is used by the PyJen plugin API to associate this class with a specific Jenkins plugin, as it is encoded in the config.xml

pyjen.plugins.listview.**PluginClass**

alias of *pyjen.plugins.listview.ListView*

pyjen.plugins.mavenplugin module

Primitives that operate on Jenkins jobs of type 'Maven'

class pyjen.plugins.mavenplugin.**MavenPlugin**(*api*)

Bases: *pyjen.job.Job*

Custom Maven job type

Parameters **api** (JenkinsAPI) – Pre-initialized connection to the Jenkins REST API

static get_jenkins_plugin_name()

str: the name of the Jenkins plugin associated with this PyJen plugin

This static method is used by the PyJen plugin API to associate this class with a specific Jenkins plugin, as it is encoded in the config.xml

static template_config_xml()

Returns a basic XML configuration template for use when instantiating jobs of this type

Return type str

pyjen.plugins.mavenplugin.**PluginClass**

alias of *pyjen.plugins.mavenplugin.MavenPlugin*

pyjen.plugins.multibranch_pipeline module

Primitives that manage Jenkins job of type 'multibranch pipeline'

class pyjen.plugins.multibranch_pipeline.**MultibranchPipelineJob**(*api*)

Bases: *pyjen.job.Job*

Jenkins job of type 'multibranch pipeline'

Parameters **api** (JenkinsAPI) – Pre-initialized connection to the Jenkins REST API

static get_jenkins_plugin_name()

str: the name of the Jenkins plugin associated with this PyJen plugin

This static method is used by the PyJen plugin API to associate this class with a specific Jenkins plugin, as it is encoded in the config.xml

property jobs

all branch jobs managed by this multibranch pipeline

Type list (*PipelineJob*)

static template_config_xml()

Returns config xml data to instantiate a default instance of this job

Return type str

pyjen.plugins.multibranch_pipeline.**PluginClass**

alias of *pyjen.plugins.multibranch_pipeline.MultibranchPipelineJob*

pyjen.plugins.multijob module

Primitives that manage Jenkins job of type 'MultiJob'

class pyjen.plugins.multijob.**MultiJob**(*api*)

Bases: *pyjen.job.Job*

Custom job type provided by the jenkins-multijob-plugin plugin

https://plugins.jenkins.io/jenkins-multijob-plugin

Parameters **api** (JenkinsAPI) – Pre-initialized connection to the Jenkins REST API

static get_jenkins_plugin_name()

str: the name of the Jenkins plugin associated with this PyJen plugin

This static method is used by the PyJen plugin API to associate this class with a specific Jenkins plugin, as it is encoded in the config.xml

static template_config_xml()

Returns XML for a default implementation of this job type

Return type str

pyjen.plugins.multijob.**PluginClass**

alias of *pyjen.plugins.multijob.MultiJob*

pyjen.plugins.myview module

Primitives for interacting with Jenkins views of type 'MyView'

class pyjen.plugins.myview.**MyView**(*api*)

Bases: *pyjen.view.View*

Interface to a view associated with a specific user

Parameters **api** (JenkinsAPI) – Pre-initialized connection to the Jenkins REST API

static get_jenkins_plugin_name()

str: the name of the Jenkins plugin associated with this PyJen plugin

This static method is used by the PyJen plugin API to associate this class with a specific Jenkins plugin, as it is encoded in the config.xml

pyjen.plugins.myview.**PluginClass**

alias of *pyjen.plugins.myview.MyView*

pyjen.plugins.nestedview module

Primitives for working with Jenkins views of type 'NestedView'

class pyjen.plugins.nestedview.**NestedView**(*api*)

Bases: *pyjen.view.View*

all Jenkins related 'view' information for views of type NestedView

Parameters **api** (JenkinsAPI) – Pre-initialized connection to the Jenkins REST API

property all_views

all views contained within this view, recursively

Type list (*View*)

create_view(*view_name*, *view_class*)

Creates a new sub-view within this nested view

Parameters

- **view_name** (*str*) – name of the new sub-view to create
- **view_class** – PyJen plugin class associated with the type of view to be created

Returns reference to the newly created view

Return type *View*

find_all_views(*view_name*)

Attempts to locate a sub-view under this nested view by name, recursively

NOTE: Seeing as how view names need only be unique within a single parent view, there may be multiple nested views with the same name. To reflect this requirement this method will return a list of views nested within this one that have the name given. If the list is empty then there are no matches for the given name anywhere in this view's sub-tree.

Parameters **view_name** (*str*) – the name of the sub-view to locate

Returns 0 or more views with the given name

Return type list (*View*)

find_view(*view_name*)

Attempts to locate a sub-view under this nested view by name

NOTE: Seeing as how view names need only be unique within a single parent view, there may be multiple nested views with the same name. To reflect this requirement this method will return a list of views nested within this one that have the name given. If the list is empty then there are no matches for the given name anywhere in this view's sub-tree.

Parameters **view_name** (*str*) – the name of the sub-view to locate

Returns 0 or more views with the given name

Return type list (*View*)

static get_jenkins_plugin_name()

str: the name of the Jenkins plugin associated with this PyJen plugin

This static method is used by the PyJen plugin API to associate this class with a specific Jenkins plugin, as it is encoded in the config.xml

property views

all views contained within this view, non-recursively

To get a recursive list of all child views and their children use *all_views()*.

Type list (*View*)

pyjen.plugins.nestedview.**PluginClass**

alias of *pyjen.plugins.nestedview.NestedView*

pyjen.plugins.nullscm module

SCM properties of Jenkins jobs with no source control configuration

class pyjen.plugins.nullscm.**NullSCM**(*node*)

Bases: *pyjen.utils.xml_plugin.XMLPlugin*

SCM plugin for Jobs with no source control configurations

Parameters **node** (*xml.etree.ElementTree.Element*) – XML node with the decoded XML data associated with a plugin

static get_jenkins_plugin_name()

str: the name of the Jenkins plugin associated with this PyJen plugin

This static method is used by the PyJen plugin API to associate this class with a specific Jenkins plugin, as it is encoded in the config.xml

classmethod instantiate()

Factory method used to construct instances of this class

Returns instance of this class

Return type *NullSCM*

pyjen.plugins.nullscm.**PluginClass**

alias of *pyjen.plugins.nullscm.NullSCM*

pyjen.plugins.parambuild_string module

String build parameter - plugin for parameterized build plugin

class pyjen.plugins.parambuild_string.**ParameterizedBuildStringParameter**(*node*)

Bases: *pyjen.utils.xml_plugin.XMLPlugin*

String build parameter - plugin for parameterized build plugin

Parameters **node** (*xml.etree.ElementTree.Element*) – XML node with the decoded XML data associated with a plugin

property default_value

the default value for this parameter

Type str

property description

the descriptive text associated with this parameter

Type str

static get_jenkins_plugin_name()

str: the name of the Jenkins plugin associated with this PyJen plugin

This static method is used by the PyJen plugin API to associate this class with a specific Jenkins plugin, as it is encoded in the config.xml

classmethod instantiate(*name*, *default_value*, *description*, *trim*)

Factory method for creating a new string build parameter

Parameters

- **name** (*str*) – Friendly name to give this build parameter
- **default_value** (*str*) – Text to assign this parameter when no user defined value is given
- **description** (*str*) – Descriptive text to show on the Jenkins UI explaining the purpose of this parameter
- **trim** (*bool*) – Indicates whether leading and trailing whitespace should be stripped from values entered into this field at build time

Returns instance of this class

Return type *ParameterizedBuildStringParameter*

property name

Gets the friendly name assigned to this parameter

Return type str

property trim

Checks to see if the value for this parameter should have whitespace stripped from the start and end automatically

Type bool

pyjen.plugins.parambuild_string.**PluginClass**

alias of *pyjen.plugins.parambuild_string.ParameterizedBuildStringParameter*

pyjen.plugins.parameterizedbuild module

Implementation for the parameterized build plugin

class pyjen.plugins.parameterizedbuild.**ParameterizedBuild**(*node*)

Bases: *pyjen.utils.xml_plugin.XMLPlugin*

Plugin which allows custom build parameters to be passed to a job

Parameters **node** (*xml.etree.ElementTree.Element*) – XML node with the decoded XML data associated with a plugin

static get_jenkins_plugin_name()

str: the name of the Jenkins plugin associated with this PyJen plugin

This static method is used by the PyJen plugin API to associate this class with a specific Jenkins plugin, as it is encoded in the config.xml

classmethod instantiate(*params*)

Factory method for this class

Parameters **params** (list of *XMLPlugin*) – List of parameters to add to this build property Each element must be associated with a parameter type supported by the parameterized build plugin

Returns an instance of this class

Return type *ParameterizedBuild*

property parameters

list of the build parameters associated with the job

Type list (*XMLPlugin*)

pyjen.plugins.parameterizedbuild.**PluginClass**

alias of *pyjen.plugins.parameterizedbuild.ParameterizedBuild*

pyjen.plugins.paramtrigger module

Jenkins post-build publisher of type Parameterized Build Trigger

class pyjen.plugins.paramtrigger.**ParameterizedBuildTrigger**(*node*)

Bases: *pyjen.utils.xml_plugin.XMLPlugin*

Custom job publisher that supports triggering other Jenkins jobs which require 1 or more custom build parameters

https://plugins.jenkins.io/parameterized-trigger

Parameters **node** (*xml.etree.ElementTree.Element*) – XML node with the decoded XML data associated with a plugin

static get_jenkins_plugin_name()

str: the name of the Jenkins plugin associated with this PyJen plugin

This static method is used by the PyJen plugin API to associate this class with a specific Jenkins plugin, as it is encoded in the config.xml

classmethod instantiate(*triggers*)

Factory method for creating a new instances of this class

Parameters **triggers** (list of *XMLPlugin*) – List of build trigger configuration objects

Returns instance of this class

Return type *ParameterizedBuildTrigger*

property triggers

list of trigger operations defined for this instance of the plugin

Type list (*BuildTriggerConfig*)

pyjen.plugins.paramtrigger.**PluginClass**

alias of *pyjen.plugins.paramtrigger.ParameterizedBuildTrigger*

pyjen.plugins.paramtrigger_buildtrigger module

Trigger configuration for a parameterized build trigger

class pyjen.plugins.paramtrigger_buildtrigger.**BuildTriggerConfig**(*node*)

Bases: *pyjen.utils.xml_plugin.XMLPlugin*

Contains the configuration options for a single downstream build trigger compatible with the parameterized build trigger plugin

https://plugins.jenkins.io/parameterized-trigger

Parameters **node** (*xml.etree.ElementTree.Element*) – XML node with the decoded XML data associated with a plugin

add_build_param(*param_config*)

Adds a new configuration option for customizing the build parameters passed to the jobs that are triggered by this configuration

Parameters **param_config** (XMLPlugin) – One of several supported plugins which offer unique customizations on how build parameters for the downstream jobs being triggered

property build_params

List of parameter definitions used to configure the build trigger for the downstream jobs associated with this trigger

Each element in the list may be of any number of derived types, each supporting a different type of custom behavior on how the parameters for the downstream job should be created / set.

Type list (*XMLPlugin*)

property condition

The state the current job must be in before the downstream job will be triggered

Type str

static get_jenkins_plugin_name()

str: the name of the Jenkins plugin associated with this PyJen plugin

This static method is used by the PyJen plugin API to associate this class with a specific Jenkins plugin, as it is encoded in the config.xml

classmethod instantiate(*job_names*)

Factory method for creating a new instances of this class

Parameters **job_names** (list of str) – List of the names of 1 or more Jenkins jobs to be triggered by this configuration object

Returns instance of this class

Return type *BuildTriggerConfig*

property job_names

list of the names of downstream jobs to be triggered

Type list (str)

pyjen.plugins.paramtrigger_buildtrigger.**PluginClass**

alias of *pyjen.plugins.paramtrigger_buildtrigger.BuildTriggerConfig*

pyjen.plugins.paramtrigger_currentbuildparams module

Trigger parameter for the Parameterized Trigger plugin

class pyjen.plugins.paramtrigger_currentbuildparams.**CurrentBuildParams**(*node*)

Bases: *pyjen.utils.xml_plugin.XMLPlugin*

Passes the current build parameters along to a parameterized downstream job

Parameters **node** (*xml.etree.ElementTree.Element*) – XML node with the decoded XML data associated with a plugin

static get_jenkins_plugin_name()

str: the name of the Jenkins plugin associated with this PyJen plugin

This static method is used by the PyJen plugin API to associate this class with a specific Jenkins plugin, as it is encoded in the config.xml

classmethod instantiate()

Factory method for creating a new instances of this class

Returns instance of this class

Return type *CurrentBuildParams*

pyjen.plugins.paramtrigger_currentbuildparams.**PluginClass**

alias of *pyjen.plugins.paramtrigger_currentbuildparams.CurrentBuildParams*

pyjen.plugins.pipelinejob module

Primitives that manage Jenkins job of type 'pipeline'

class pyjen.plugins.pipelinejob.**PipelineJob**(*api*)

Bases: *pyjen.job.Job*

Jenkins job of type 'pipeline'

Parameters **api** (JenkinsAPI) – Pre-initialized connection to the Jenkins REST API

static get_jenkins_plugin_name()

str: the name of the Jenkins plugin associated with this PyJen plugin

This static method is used by the PyJen plugin API to associate this class with a specific Jenkins plugin, as it is encoded in the config.xml

property scm

the source code repo where the job config is defined

Type *XMLPlugin*

scm_definition(*scm*, *script_path='Jenkinsfile'*, *lightweight=True*)

Defines the Pipeline groovy script used by this job from files stored in a source code repository

Parameters

- **scm** (XMLPlugin) – PyJen object defining the source code repository to use
- **script_path** (*str*) – Path within the repository where the groovy script to be run is found. Defaults to 'Jenkinsfile' in the root folder

- **lightweight** (*bool*) – Set to True to have the build only check out the Jenkinsfile and no other file from the repository. Set to False to have the build check out the entire repository before running the Jenkinsfile.

property script

the groovy script defining this build

Type str

script_definition(*script*, *sandbox=True*)

Defines the pipeline build using an inline groovy script

Parameters

- **script** (*str*) – Raw Groovy script defining the build process
- **sandbox** (*bool*) – indicates whether the Groovy script can run in the safer 'sandbox' environment.

static template_config_xml()

Returns template XML that can be used to create instances of this class

Return type str

class pyjen.plugins.pipelinejob.**PipelineXML**(*api*)

Bases: *pyjen.utils.jobxml.JobXML*

Object that manages the config.xml for a pipeline job

Parameters **api** (JenkinsAPI) – Rest API for the Jenkins XML configuration managed by this object

property scm

the source code repo where the build script is located

Type *XMLPlugin*

scm_definition(*scm*, *script_path*, *lightweight*)

Defines the Pipeline groovy script used by this job from files stored in a source code repository

Parameters

- **scm** (XMLPlugin) – PyJen object defining the source code repository to use
- **script_path** (*str*) – Path within the repository where the groovy script to be run is found.
- **lightweight** (*bool*) – Set to True to have the build only check out the Jenkinsfile and no other file from the repository. Set to False to have the build check out the entire repository before running the Jenkinsfile.

property script

the groovy script that defines the build process for this job

Type str

script_definition(*script*, *sandbox*)

Defines the pipeline build using an inline groovy script

Parameters

- **script** (*str*) – Raw Groovy script defining the build process
- **sandbox** (*bool*) – indicates whether the Groovy script can run in the safer 'sandbox' environment.

pyjen.plugins.pipelinejob.**PluginClass**

alias of *pyjen.plugins.pipelinejob.PipelineJob*

pyjen.plugins.runcondition_always module

Condition for the run condition plugin that will always produce a true result

class pyjen.plugins.runcondition_always.**AlwaysRun**(*node*)

Bases: *pyjen.utils.xml_plugin.XMLPlugin*

Conditional build step that will always produce a true result

https://plugins.jenkins.io/run-condition

Parameters **node** (*xml.etree.ElementTree.Element*) – XML node with the decoded XML data associated with a plugin

static get_friendly_name()

Returns The user friendly display name for this condition. This typically reflects the text found in the Jenkins UI for the condition

Return type str

static get_jenkins_plugin_name()

str: the name of the Jenkins plugin associated with this PyJen plugin

This static method is used by the PyJen plugin API to associate this class with a specific Jenkins plugin, as it is encoded in the config.xml

classmethod instantiate()

Factory method used to construct an instance of this class

Returns instance of this class

Return type *AlwaysRun*

pyjen.plugins.runcondition_always.**PluginClass**

alias of *pyjen.plugins.runcondition_always.AlwaysRun*

pyjen.plugins.runcondition_and module

Condition for the run condition plugin that performs a logical AND operation on other build conditions

class pyjen.plugins.runcondition_and.**AndCondition**(*node*)

Bases: *pyjen.utils.xml_plugin.XMLPlugin*

Build condition that combines 2 or more other conditions with a logical AND operation. The associated operation using this condition will only run if all of the contained conditions result in a "true" value.

https://plugins.jenkins.io/run-condition

Parameters **node** (*xml.etree.ElementTree.Element*) – XML node with the decoded XML data associated with a plugin

static get_friendly_name()

Returns The user friendly display name for this condition. This typically reflects the text found in the Jenkins UI for the condition

Return type str

static get_jenkins_plugin_name()

str: the name of the Jenkins plugin associated with this PyJen plugin

This static method is used by the PyJen plugin API to associate this class with a specific Jenkins plugin, as it is encoded in the config.xml

Return type str

classmethod instantiate(*terms*)

Creates a new instance of this condition

Parameters **terms** (list of *XMLPlugin*) – List of 2 or more conditions to be combined using a logical AND operation. Each element is expected to be an instance of a PyJen plugin compatible with the Run Condition plugin.

Returns instance of this class

Return type *AndCondition*

pyjen.plugins.runcondition_and.**PluginClass**

alias of *pyjen.plugins.runcondition_and.AndCondition*

pyjen.plugins.runcondition_never module

Condition for the run condition plugin that always produces a False result

class pyjen.plugins.runcondition_never.**NeverRun**(*node*)

Bases: *pyjen.utils.xml_plugin.XMLPlugin*

Condition for the run condition plugin that always produces a False result

https://plugins.jenkins.io/run-condition

Parameters **node** (*xml.etree.ElementTree.Element*) – XML node with the decoded XML data associated with a plugin

static get_friendly_name()

Returns The user friendly display name for this condition. This typically reflects the text found in the Jenkins UI for the condition

Return type str

static get_jenkins_plugin_name()

str: the name of the Jenkins plugin associated with this PyJen plugin

This static method is used by the PyJen plugin API to associate this class with a specific Jenkins plugin, as it is encoded in the config.xml

classmethod instantiate()

Factory method used to construct an instance of this class

Returns instance of this class

Return type *NeverRun*

pyjen.plugins.runcondition_never.**PluginClass**

alias of *pyjen.plugins.runcondition_never.NeverRun*

pyjen.plugins.runcondition_not module

Condition for the run condition plugin that inverts the logical result of another build condition.

class pyjen.plugins.runcondition_not.**NotCondition**(*node*)

Bases: *pyjen.utils.xml_plugin.XMLPlugin*

Condition for the run condition plugin that inverts the logical result of another build condition.

https://plugins.jenkins.io/run-condition

Parameters **node** (*xml.etree.ElementTree.Element*) – XML node with the decoded XML data associated with a plugin

static get_friendly_name()

Returns The user friendly display name for this condition. This typically reflects the text found in the Jenkins UI for the condition

Return type str

static get_jenkins_plugin_name()

str: the name of the Jenkins plugin associated with this PyJen plugin

This static method is used by the PyJen plugin API to associate this class with a specific Jenkins plugin, as it is encoded in the config.xml

classmethod instantiate(*condition*)

Factory method that constructs an instance of this class

Parameters **condition** (XMLPlugin) – Any PyJen plugin compatible with the Run Condition plugin

Returns instance of this class

Return type *NotCondition*

pyjen.plugins.runcondition_not.**PluginClass**

alias of *pyjen.plugins.runcondition_not.NotCondition*

pyjen.plugins.sectionedview module

Primitives for working on Jenkins views of type 'SectionedView'

pyjen.plugins.sectionedview.**PluginClass**

alias of *pyjen.plugins.sectionedview.SectionedView*

class pyjen.plugins.sectionedview.**SectionedView**(*api*)

Bases: *pyjen.view.View*

Interface to Jenkins views of type "SectionedView"

Views of this type support groupings of jobs into 'sections' which each have their own filters

Parameters **api** (JenkinsAPI) – Pre-initialized connection to the Jenkins REST API

add_section(*section_type*, *name*)

Adds a new section to the sectioned view

Parameters

- **section_type** (*str*) – name of class used to implement the new section to add

- **name** (*str*) – descriptive text to appear in the title area of the section

static get_jenkins_plugin_name()

str: the name of the Jenkins plugin associated with this PyJen plugin

This static method is used by the PyJen plugin API to associate this class with a specific Jenkins plugin, as it is encoded in the config.xml

property sections

list of sections contained within this view

Type list (*XMLPlugin*)

class pyjen.plugins.sectionedview.**SectionedViewXML**(*api*)

Bases: *pyjen.utils.viewxml.ViewXML*

raw config.xml parser for a Jenkins view of type 'Sectioned View'

Parameters **api** (JenkinsAPI) – Rest API for the Jenkins XML configuration managed by this object

add_section(*section_type*, *name*)

Adds a new section to the sectioned view

Parameters

- **section_type** (*str*) – name of class used to implement the new section to add
- **name** (*str*) – descriptive text to appear in the title area of the section

property sections

list of all 'section' objects contained in this view

Type list (*XMLPlugin*)

pyjen.plugins.sectionedview_listsection module

Primitives for controlling list view sub-sections on a sectioned view

This is a plugin supported by the SectionedView plugin

class pyjen.plugins.sectionedview_listsection.**ListViewSection**(*node*)

Bases: *pyjen.utils.xml_plugin.XMLPlugin*

One of several 'section' types defined for a sectioned view

Represents sections of type 'ListView'

Parameters **node** (*xml.etree.ElementTree.Element*) – XML node with the decoded XML data associated with a plugin

static get_jenkins_plugin_name()

str: the name of the Jenkins plugin associated with this PyJen plugin

This static method is used by the PyJen plugin API to associate this class with a specific Jenkins plugin, as it is encoded in the config.xml

property include_regex

regular filter for jobs to be shown in this section

Type str

classmethod instantiate(*section_name*)

Factory method for creating a new Git SCM code block

Parameters **section_name** (*str*) – Text to appear at the top of the section

Returns instance of this class

Return type *ListViewSection*

property name

the title text of this section

Type str

pyjen.plugins.sectionedview_listsection.**PluginClass**

alias of *pyjen.plugins.sectionedview_listsection.ListViewSection*

pyjen.plugins.sectionedview_textsection module

Primitives for controlling plain text view sub-sections on a sectioned view

This is a plugin supported by the SectionedView plugin

pyjen.plugins.sectionedview_textsection.**PluginClass**

alias of *pyjen.plugins.sectionedview_textsection.TextSection*

class pyjen.plugins.sectionedview_textsection.**TextSection**(*node*)

Bases: *pyjen.utils.xml_plugin.XMLPlugin*

One of several 'section' types defined for a sectioned view

Sections of this type contain simple descriptive text

Parameters **node** (*xml.etree.ElementTree.Element*) – XML node with the decoded XML data associated with a plugin

static get_jenkins_plugin_name()

str: the name of the Jenkins plugin associated with this PyJen plugin

This static method is used by the PyJen plugin API to associate this class with a specific Jenkins plugin, as it is encoded in the config.xml

classmethod instantiate(*section_name*)

Factory method for creating a new Git SCM code block

Parameters **section_name** (*str*) – Text to appear at the top of the section

Returns instance of this class

Return type *TextSection*

property name

the title text of this section

Type str

pyjen.plugins.shellbuilder module

Interface to control a basic shell build step job builder plugin

pyjen.plugins.shellbuilder.**PluginClass**

alias of *pyjen.plugins.shellbuilder.ShellBuilder*

class pyjen.plugins.shellbuilder.**ShellBuilder**(*node*)

Bases: *pyjen.utils.xml_plugin.XMLPlugin*

Interface to control a basic shell build step job builder plugin

This plugin is a default, built-in plugin which is part of the Jenkins core

Parameters **node** (*xml.etree.ElementTree.Element*) – XML node with the decoded XML data associated with a plugin

static get_jenkins_plugin_name()

str: the name of the Jenkins plugin associated with this PyJen plugin

This static method is used by the PyJen plugin API to associate this class with a specific Jenkins plugin, as it is encoded in the config.xml

classmethod instantiate(*script*)

Factory method for creating a new shell build step

Parameters **script** (*str*) – shell script to run as part of this build step

Returns instance of this class

Return type *ShellBuilder*

property script

the shell script associated with this builder

Type str

property unstable_return_code

the return code that marks the build as unstable

Type int

pyjen.plugins.statusview module

Primitives for operating on Jenkins views of type 'StatusView'

pyjen.plugins.statusview.**PluginClass**

alias of *pyjen.plugins.statusview.StatusView*

class pyjen.plugins.statusview.**StatusView**(*api*)

Bases: *pyjen.view.View*

Interface to Jenkins views of type 'StatusView'

Parameters **api** (JenkinsAPI) – Pre-initialized connection to the Jenkins REST API

static get_jenkins_plugin_name()

str: the name of the Jenkins plugin associated with this PyJen plugin

This static method is used by the PyJen plugin API to associate this class with a specific Jenkins plugin, as it is encoded in the config.xml

pyjen.plugins.subversion module

Module defining the interfaces for interacting with Subversion properties associated with a *pyjen.job.Job*

class pyjen.plugins.subversion.**ModuleLocation**(*node*)

Bases: *pyjen.utils.xml_plugin.XMLPlugin*

Interface to SCM module declarations in a Subversion property of a job

Parameters **node** (*xml.etree.ElementTree.Element*) – XML node with the decoded XML data associated with a plugin

property depth_option

the current SVN 'depth' options associated with this module

Type str

disable_ignore_externals()

Disables the 'ignore externals' option on this SCM module

enable_ignore_externals()

Enables the 'ignore externals' option on this SCM module

property ignore_externals

True if ignore externals is enabled, otherwise False

Type bool

property local_dir

folder where the source code for this module is checked out to

Type str

property node

the XML node associated with this plugin

Type xml.etree.ElementTree.Element

property url

SVN URL where the source code for this module can be found

Type str

pyjen.plugins.subversion.**PluginClass**

alias of *pyjen.plugins.subversion.Subversion*

class pyjen.plugins.subversion.**Subversion**(*node*)

Bases: *pyjen.utils.xml_plugin.XMLPlugin*

Subversion SCM job plugin

https://wiki.jenkins-ci.org/display/JENKINS/Subversion+Plugin

Parameters **node** (*xml.etree.ElementTree.Element*) – XML node with the decoded XML data associated with a plugin

static get_jenkins_plugin_name()

str: the name of the Jenkins plugin associated with this PyJen plugin

This static method is used by the PyJen plugin API to associate this class with a specific Jenkins plugin, as it is encoded in the config.xml

property included_regions

patterns of the regions of the SVN repo to include in SCM operations

Type list (str)

property locations

set of 0 or more ModuleLocation objects describing the SVN parameters for this module.

Type list (*ModuleLocation*)

Module contents

modules that manage PyJen plugins which map to Jenkins plugins

For details on how these plugins interact with PyJen and Jenkins see *pyjen.utils.xml_plugin.XMLPlugin*

pyjen.utils package

Submodules

pyjen.utils.helpers module

Misc helper methods shared across the library

pyjen.utils.helpers.**create_job**(*api*, *job_name*, *job_class*)

Creates a new job on the Jenkins dashboard

Parameters

- **api** (JenkinsAPI) – Jenkins rest api connection to use for creation of the view`
- **job_name** (*str*) – The name for this new job. This name should be unique, different from any other jobs currently managed by the Jenkins instance
- **job_class** – PyJen plugin class associated with the type of job to be created

pyjen.utils.helpers.**create_view**(*api*, *view_name*, *view_class*)

Creates a new view on the Jenkins dashboard

Parameters

- **api** (JenkinsAPI) – Jenkins rest api connection to use for creation of the view`
- **view_name** (*str*) – The name for this new view. This name should be unique, different from any other views currently managed by the Jenkins instance
- **view_class** – PyJen plugin class associated with the type of view to be created

pyjen.utils.jenkins_api module

Base class for all objects that interact with the Jenkins REST API

class pyjen.utils.jenkins_api.**JenkinsAPI**(*url*, *session*)

Bases: object

Abstraction around the raw Jenkins REST API

Parameters

- **url** (*str*) – URL of the Jenkins API endpoint to manage
- **session** (*requests.Session*) – HTTP session to use for interacting with the Jenkins REST API

clone(*api_url*)

Creates a copy of this instance, for a new endpoint URL

Parameters **api_url** (*str*) – URL for the new REST API endpoint to be managed

Returns reference to the newly created API interface

Return type *JenkinsAPI*

property crumb

unique "crumb" identifier required by all POST operations

Introduced in Jenkins v2

Output from this helper can be used directly in post operations as an HTTP header, something like this:

```
requests.post(... headers=self.crumb)
```

reference: https://wiki.jenkins-ci.org/display/JENKINS/Remote+access+API (see CSRF protection section)

Type dict

get_api_data(*target_url=None*, *query_params=None*)

retrieves the Jenkins API specific data from the specified URL

Parameters

- **target_url** (*str*) – Full URL to the REST API endpoint to be queried. If not provided, data will be loaded from the default 'url' for this object
- **query_params** (*str*) – optional set of query parameters to customize the returned data

Returns The set of Jenkins attributes, converted to Python objects, associated with the given URL.

Return type dict

get_api_xml(*path=None*, *params=None*)

Gets api XML data from a given REST API endpoint

Parameters

- **path** (*str*) – optional extension path to append to the root URL managed by this object when performing the get operation
- **params** (*dict*) – optional query parameters to be passed to the request

Returns parsed XML data

Return type xml.etree.ElementTree.Element

get_text(*path=None*, *params=None*)

gets the raw text data from a Jenkins URL

Parameters

- **path** (*str*) – optional extension path to append to the root URL managed by this object when performing the get operation
- **params** (*dict*) – optional query parameters to be passed to the request

Returns the text loaded from this objects' URL

Return type str

property jenkins_headers

HTTP headers from the main Jenkins dashboard using the REST API

The dashboard headers contain metadata describing the Jenkins instance hosting the REST API, including details such as version number, current UI theme, and others.

Type dict

property jenkins_version

version number of the Jenkins server hosting this REST API, parsed into a tuple of integers

Typically returns a 3 tuple with the major, minor and update digits of the version number

Type tuple

post(*target_url*, *args=None*)

sends data to or triggers an operation via a Jenkins URL

Parameters

- **target_url** (*str*) – Full URL to sent post request to
- **args** (*dict*) – optional set of data arguments to be sent with the post operation.

Returns reference to the response data returned by the post request

Return type requests.Response

property root_url

URL of the main Jenkins dashboard associated with the current object

NOTE: The URL returned by this property is guaranteed to end with a trailing slash character

Type str

property url

the URL for the REST API endpoint managed by this object

NOTE: The URL returned by this property is guaranteed to end with a trailing slash character

Type str

pyjen.utils.jobxml module

Abstractions for managing the raw config.xml for a Jenkins job

class pyjen.utils.jobxml.**JobXML**(*api*)

Bases: object

Wrapper around the config.xml for a Jenkins job

The source xml can be loaded from nearly any URL by appending "/config.xml" to it, as in "http://server/jobs/job1/config.xml"

Parameters **api** (JenkinsAPI) – Rest API for the Jenkins XML configuration managed by this object

add_property(*prop*)

Adds a new job property to the configuration

Parameters **prop** (XMLPlugin) – PyJen plugin associated with the job property to add

property plugin_name

the name of the Jenkins plugin associated with XML definition

Type str

property properties

list of 0 or more Jenkins properties associated with this job

Each element in the list will be a reference to a PyJen plugin which is able to manipulate each job property. Any properties not supported by the PyJen plugins currently installed will be ignored.

Type list (*XMLPlugin*)

update()

Posts all changes made to the object back to Jenkins

property xml

Raw XML representation describing the configuration of this plugin, in plain-text format

Type str

pyjen.utils.plugin_api module

Primitives for interacting with the PyJen plugin subsystem

pyjen.utils.plugin_api.**find_plugin**(*plugin_name*)

Locates the PyJen class associated with a given Jenkins plugin

Parameters **plugin_name** (*str*) – Name of the Jenkins plugin to find the associated

Returns reference to the PyJen plugin class associated with the given Jenkins plugin, if one exists. If one doesn't exist, returns None.

pyjen.utils.plugin_api.**get_all_plugins**()

Returns a list of all PyJen plugins installed on the system

Returns 0 or more PyJen plugins installed on this system

Return type list

pyjen.utils.plugin_api.**instantiate_xml_plugin**(*node*, *parent*)

Instantiates a PyJen XML plugin from an arbitrary XML node

Parameters

- **node** (*xml.etree.ElementTree.Element*) – ElementTree node describing the plugin to be instantiated
- **parent** (XMLPlugin) – PyJen plugin that owns the XML for the plugin being instantiated. Used for managing plugins that support nesting of other plugins

Returns Reference to the instantiated plugin, or None if the associated plugin isn't currently implemented in PyJen yet

Return type *XMLPlugin*

pyjen.utils.viewxml module

Abstractions for managing the raw config.xml for a Jenkins view

class pyjen.utils.viewxml.**ViewXML**(*api*)

Bases: object

Wrapper around the config.xml for a Jenkins view

Loaded from the ./view/config.xml REST API endpoint for any arbitrary Jenkins view.

Parameters **api** (JenkinsAPI) – Rest API for the Jenkins XML configuration managed by this object

property plugin_name

the name of the Jenkins plugin associated with XML definition

Type str

rename(*new_name*)

Changes the name of the view

Parameters **new_name** (*str*) – The new name for the view

update()

Posts all changes made to the object back to Jenkins

property xml

Raw XML representation describing the configuration of this plugin, in plain-text format

Type str

pyjen.utils.xml_plugin module

Primitives common to all PyJen plugins that extend Jenkins config.xml

class pyjen.utils.xml_plugin.**XMLPlugin**(*node*)

Bases: object

Base class for all PyJen plugins that extend Jenkins config.xml

Plugins derived from this base class are always instantiated with a sub-node within a primitive's config.xml. The particulars of these sub-nodes including names, attributes, and children nodes, is unique to each Jenkins plugin. However, all such PyJen plugins must extend this base class to provide those plugin specific behaviors.

All derived classes are expected to implement the following public interfaces:

@classmethod def create(cls, ...):

a factory method that takes 0 or more parameters (anything necessary to construct an instance of the derived class) and returns an instance of the class pre-constructed with relevant XML content. Used when adding an instance of the plugin to an existing Jenkins object.

@staticmethod def get_jenkins_plugin_name():

Helper method that returns a character string representation of the Jenkins plugin name the class is associated with. This is typically the Java class name of the Jenkins plugin the PyJen plugin is meant to work with.

Parameters **node** (*xml.etree.ElementTree.Element*) – XML node with the decoded XML data associated with a plugin

property node

the encoded XML data associated with this plugin

Type xml.etree.ElementTree.Element

property parent

Object that manages an outer XML code block containing the nodes managed by this object.

Type *XMLPlugin*

update()

Updates parent XML tree when one exists

Module contents

Sub-package for common utilities used by the PyJen APIs

Submodules

pyjen.build module

Primitives for interacting with Jenkins builds

class pyjen.build.**Build**(*api*)

Bases: object

information about a single build / run of a *Job*

Parameters **api** (JenkinsAPI) – Pre-initialized connection to the Jenkins REST API

abort()

Aborts this build before it completes

property artifact_urls

list of 0 or more URLs to download published build artifacts

Type list ()

property changeset

Description of 0 or more SCM revisions associated with / included in this build

Type *Changeset*

property console_output

raw console output for this build as plain text

Type str

property description

Gets the descriptive text associated with this build. May be an empty string if no description given.

Type str

property duration

total runtime of the build, in milliseconds. Returns 0 if build hasn't finished

Type int

property estimated_duration

Estimated runtime for a running build, in milliseconds. Estimate is based off average duration of previous builds

Type int

property is_building

True if the build is currently executing otherwise False

Type bool

kill()

Performs hard kill on this build

property number

sequentially assigned numeric ID for the build

Type int

property result

state of the associated job upon completion of this build. Typically one of the following:

- "SUCCESS"
- "UNSTABLE"
- "FAILURE"
- "ABORTED"

Type str

property start_time

time stamp of when this build was started

Type datetime.datetime

property uid

internal, unique identifier associated with this build

Type str

property url

URL of this build

Type str

pyjen.changeset module

Primitives for interacting with SCM changesets

class pyjen.changeset.**Changeset**(*api*, *data*)

Bases: object

Represents a set of SCM revisions associated with a *Build* of a *Job*.

See *changeset()* for details.

Parameters

- **api** (JenkinsAPI) – Pre-initialized connection to the Jenkins REST API
- **data** (*dict*) – Dictionary of data elements typically parsed from the "changeSet" node of a builds source data as provided by the Jenkins REST API.

property affected_items

details of the individual commits associated with a build

Type list (*ChangesetItem*)

property has_changes

True if there are SCM changes associated with this changeset, False if not

Type bool

property scm_type

name of the SCM tool associated with this change

Type str

class pyjen.changeset.**ChangesetItem**(*api*, *data*)

Bases: object

details of each SCM revision associated with a given *Changeset*

Parameters **data** (*dict*) – Dictionary of attributes describing a single commit.

property affected_files

list of files modified in this commit

Type list (str)

property author

Person who committed this change to the associated SCM

Type *User*

property commit_id

the SCM ref spec associated with this specific change

Type str

property message

SCM commit message associated with this change

Type str

pyjen.jenkins module

Primitives for interacting with the main Jenkins dashboard

class pyjen.jenkins.**Jenkins**(*url*, *session*)

Bases: object

Python wrapper managing the Jenkins primary dashboard

Generally you should use this class as the primary entry point to the PyJen APIs. Finer grained control of each aspect of the Jenkins dashboard is then provided by the objects exposed by this class including *View*, *Job* and *Build*.

Parameters

- **url** (*str*) – URL of the Jenkins service to connect to
- **session** (*requests.Session*) – pre-configured HTTP session to use for communicating to the Jenkins REST API. This session may be user defined, or built using one of the factory methods associated with this class such as *basic_auth()*.

property all_jobs

all jobs managed by this Jenkins instance, recursively

Unlike the *jobs()* method, this method attempts to expose jobs which are managed by custom jobs created from third party plugins which support nesting jobs under sub-folders / sub-paths. Any job which exposes a custom 'jobs' property.

Type list (*Job*)

classmethod basic_auth(*url*, *credentials=None*, *ssl_cert=None*)

Factory method used to instantiate a connection to a Jenkins server using HTTP basic auth protocol

Parameters

- **url** (*str*) – Full HTTP URL to the main Jenkins dashboard
- **credentials** (*tuple*) – Optional 2-tuple containing the username and password / api key to authenticate with. If not provided, anonymous access will be assumed
- **ssl_cert** (*str*) – Passed directly to the requests library when authenticating to the remote server. Maybe be a boolean indicating whether SSL verification is enabled or disabled, or may be a path to a certificate authority bundle.

Returns instance of this class, preconfigured to connect to the specified Jenkins service using an HTTP basic auth connection

Return type *Jenkins*

property build_queue

interface for managing the Jenkins build queue

Type *Queue*

cancel_shutdown()

Cancels a previous scheduled shutdown sequence

Cancels a shutdown operation initiated by the *prepare_shutdown()* method

property connected

True if API still connected to the service, False if not

Type bool

create_job(*job_name*, *job_class*)

Creates a new job on the Jenkins dashboard

Parameters

- **job_name** (*str*) – The name for this new job. This name should be unique, different from any other jobs currently managed by the Jenkins instance
- **job_class** – PyJen plugin class associated with the type of job to be created

Returns An object to manage the newly created job

Return type *Job*

create_view(*view_name*, *view_class*)

Creates a new view on the Jenkins dashboard

Parameters

- **view_name** (*str*) – The name for this new view. This name should be unique, different from any other views currently managed by the Jenkins instance
- **view_class** – PyJen plugin class associated with the type of view to be created

Returns An object to manage the newly created view

Return type *View*

property default_view

the primary / default Jenkins view

The default view is the one displayed when navigating to the main URL. Typically this will be the "All" view, but this may be customized by the user at runtime.

Type *View*

find_job(*job_name*)

Searches all jobs managed by this Jenkins instance for a specific job

Some plugins allow jobs to be nested within other jobs. To perform a recursive search across all such entities, see *all_jobs()*.

Parameters **job_name** (*str*) – the name of the job to search for

Returns If a job with the specified name can be found, and object to manage the job will be returned, otherwise None

Return type *Job*

find_node(*nodename*)

Locates a Jenkins build agent with the given name

Parameters **nodename** (*str*) – name of node to locate

Returns reference to Jenkins object that manages this node's information, or None if no node with the given name can be found

Return type *Node*

`find_user`(*username*)

Locates a user with the given username on this Jenkins instance

Parameters **username** (*str*) – name of user to locate

Returns reference to Jenkins object that manages this users information, or None if no user with the specified name can be found

Return type *User*

`find_view`(*view_name*)

Searches views for a specific one

Parameters **view_name** (*str*) – the name of the view to search for

Returns If a view with the specified name can be found, an object to manage the view will be returned, otherwise returns None

Return type *View*

`property is_shutting_down`

True if the Jenkins master is scheduled for a shutdown, False if not

Type bool

`property jobs`

all jobs managed by this Jenkins instance

Type list (*Job*)

`property nodes`

list of build agents

Type list (*Node*)

`property plugin_manager`

interface for managing the plugins installed on this Jenkins instance

Type *PluginManager*

`prepare_shutdown`()

Starts a "quiet down" and prevents new builds from executing

Analogous to the "Prepare for Shutdown" link on the Manage Jenkins configuration page

You can cancel a previous requested shutdown using the *cancel_shutdown()* method

`property version`

version of Jenkins service, parsed into a tuple of integers

Type tuple

`property views`

all views directly managed by this Jenkins instance

Type list (*View*)

pyjen.job module

Primitives for interacting with Jenkins jobs

class pyjen.job.**Job**(*api*)

Bases: object

Abstraction for operations common to all job types on Jenkins

Parameters **api** (JenkinsAPI) – Pre-initialized connection to the Jenkins REST API

add_property(*new_property*)

Adds a new job property to the configuration

Parameters **new_property** (XMLPlugin) – Custom job property to be added. May be any PyJen plugin that supports the Jenkins job property plugin API.

property all_builds

all recorded builds for this job

Type list (*Build*)

property build_health

the percentage of good builds from recorded history of this job

This metric is associated with the "weather" icon that can be shown next to jobs in certain views

Type int

clone(*new_job_name*, *disable=True*)

"Create a new job with the same configuration as this one

Parameters

- **new_job_name** (*str*) – Name of the new job to be created
- **disable** (*bool*) – Indicates whether the newly created job should be disabled after creation to prevent builds from accidentally triggering immediately after creation

Returns reference to the newly created job

Return type *Job*

property config_xml

raw XML configuration for the job

Allows callers to manipulate the raw job configuration definition without relying on the PyJen abstractions.

> **Warning:** For advanced use only.

Type str

delete()

Deletes this job from the Jenkins dashboard

disable()

Disables this job to prevent new builds from being executed

Use in conjunction with *enable()* and *is_disabled* to control the state of the job.

enable()

Enables this job

If this jobs current state is disabled, it will be re-enabled after calling this method. If the job is already enabled then this method does nothing.

Enabling a job allows it to be triggered, either automatically via commit hooks / polls or manually through the dashboard.

Use in conjunction with *disable()* and *is_disabled* to control the state of the job

find_build_by_queue_id(*queue_id*)

the build of this job which correlates to a specific queue item

Parameters **queue_id** (*int*) – ID of the build queue item to correlate with. Typically extracted from the *uid()* property.

Returns reference to the build associated with the specified queue id or None if no such build exists

Return type *Build*

get_build_by_number(*build_number*)

Gets a specific build of this job from the build history

Parameters **build_number** (*int*) – Numeric identifier of the build to retrieve Value is typically non-negative

Returns Reference to the build with the given numeric identifier, or None if no such build exists

Return type *Build*

get_builds_in_time_range(*start_time*, *end_time*)

Returns a list of all of the builds for a job that occurred between the specified start and end times

Parameters

- **start_time** (*datetime.datetime*) – starting time index for range of builds to find
- **end_time** (*datetime.datetime*) – ending time index for range of builds to find

Returns list of 0 or more builds of this job that began during the time range provided

Return type list (*Build*)

classmethod get_supported_plugins()

list: list of PyJen plugin classes that derive from this class

property has_been_built

True if the job has been built at least once, otherwise False

Type bool

static instantiate(*json_data*, *rest_api*)

Factory method for finding the appropriate PyJen view object based on data loaded from the Jenkins REST API

Parameters

- **json_data** (*dict*) – data loaded from the Jenkins REST API summarizing the view to be instantiated
- **rest_api** (JenkinsAPI) – PyJen REST API configured for use by the parent container. Will be used to instantiate the PyJen view that is returned.

Returns PyJen job object wrapping the REST API for the given job

Return type *Job*

`property is_disabled`

True if the job is disabled, otherwise False

Type bool

`property is_failing`

True if the latest build of the job is a failure, otherwise False

Type bool

`property is_unstable`

True if the latest build of the job is unstable, otherwise False

Type bool

`property jenkins_plugin_name`

Extracts the name of the Jenkins plugin providing the features associated with this job. May reference any number of third party plugins supported by the Jenkins instance being managed.

The data returned by this helper property is extracted from the config XML that defines this job.

Type str

`property last_build`

the most recent build of this job

Synonymous with the "Last Build" permalink on the jobs' main status page

Type *Build*

`property last_failed_build`

the most recent build of this job with a status of "failed"

Synonymous with the "Last failed build" permalink on the jobs' main status page

Type *Build*

`property last_good_build`

the most recent successful build of this job

Synonymous with the "Last successful build" permalink on the jobs' main status page

Type *Build*

`property last_stable_build`

the most recent build of this job with a status of "stable"

Synonymous with the "Last stable build" permalink on the jobs' main status page

Type *Build*

`property last_unsuccessful_build`

the most recent build of this job with a status of "unstable"

Synonymous with the "Last unsuccessful build" permalink on the jobs' main status page

Type *Build*

property name

the name of the Jenkins job

Type str

property properties

custom properties associated with this job, typically describing features of the job provided by third party plugins

Type list (*XMLPlugin*)

property recent_builds

list of the most recent builds of this job

Rather than returning all data on all available builds, this method only returns the latest 20 or 30 builds. This list is synonymous with the short list provided on the main info page for the job on the dashboard.

Type list (*Build*)

rename(*new_job_name*)

Changes the name of this job

Parameters **new_job_name** (*str*) – new name to assign to this job

start_build(***kwargs*)

Forces a build of this job

Synonymous with a manual trigger. A new instance of the job (ie: a build) will be added to the appropriate build queue where it will be scheduled for execution on the next available agent + executor.

Parameters **kwargs** (*dict*) – 0 or more named arguments to pass as build parameters to the job when triggering the build.

Returns Reference to the Jenkins queue item that tracks the progress of the triggered build prior to the build actually running.

Return type *QueueItem*

pyjen.node module

Declarations for the abstraction of a Jenkins build agent

class pyjen.node.**Node**(*api*)

Bases: object

Wrapper around a Jenkins build agent (aka: Node) configuration

See *find_node()* for more details

Parameters **api** (JenkinsAPI) – Pre-initialized connection to the Jenkins REST API

property is_idle

checks to see whether any executors are in use on this Node or not

Type bool

property is_offline

checks to see whether this Node is currently offline or not

Type bool

property name

the display name of this Node

Type str

property number_of_executors

the number of executors this node provides

Type int

toggle_offline(*message=None*)

Toggles the online status of this Node

If the current state of this Node is "offline" it will be toggled to "online" and vice-versa.

Parameters **message** (*str*) – optional descriptive message explaining the reason this node has been taken offline.

wait_for_idle(*max_timeout=None*)

Blocks execution until this Node enters an idle state

Parameters **max_timeout** (*int*) – Optional amount of time, in seconds, to wait for an idle state. If this value is undefined, this method will block indefinitely.

Returns True if the Node has entered idle state before returning otherwise returns False

Return type bool

pyjen.plugin module

Interface for interacting with Jenkins plugins

class pyjen.plugin.**Plugin**(*plugin_config*)

Bases: object

Abstraction around one Jenkins plugin

Parameters **plugin_config** (*dict*) – Parsed Jenkins API data associated with this plugin. Typically this content is produced by the Jenkins plugin manager API. See *PluginManager* for details.

download(*output_folder*, *overwrite=False*)

Downloads the plugin installation file for this plugin

Parameters

- **output_folder** (*str*) – path where the downloaded plugin file will be stored
- **overwrite** (*bool*) – indicates whether existing plugin files should be overwritten in the target folder

property download_url

URL where the version of this plugin may be downloaded

Type str

property enabled

checks to see if this plugin is enabled or not

Type bool

property info_url

the URL for the website describing the use of this plugin

Type str

property latest_download_url

URL where the latest version of this plugin can be downloaded

Type str

property long_name

the descriptive name of this plugin

Type str

property required_dependencies

metadata describing the plugins this plugin depends on. Nested dictionaries contain the 'shortName' and 'version' fields for use by the caller.

Type list (dict)

property short_name

the abbreviated name of this plugin

Type str

property version

the version of this plugin

Type str

pyjen.plugin_manager module

Interfaces for managing plugins for a particular Jenkins instance

class pyjen.plugin_manager.**PluginManager**(*api*)

Bases: object

Abstraction around Jenkins plugin management interfaces

Supports adding, removing and querying information about Jenkins plugins

Parameters **api** (JenkinsAPI) – Pre-initialized connection to the Jenkins REST API

find_plugin_by_shortname(*short_name*)

Finds an installed plugin based on it's abbreviated name

Parameters **short_name** (*str*) – abbreviated form of the plugin name to locate

Returns reference to the given plugin, or None if no plugin with the specified short name could be found

Return type *Plugin*

install_plugin(*plugin_file*)

Installs a new plugin on the selected Jenkins instance

NOTE: If using this method to batch-install many plugins at once you may want to add a short wait / sleep between calls so as to not overwhelm the target server with upload requests. Ad-hoc tests show that Jenkins will refuse connections if too many uploads are running in parallel.

Parameters **plugin_file** (*str*) – path to the HPI/JPI file to install

property plugins

interfaces describing the plugins installed on the current Jenkins instance

Type list (*Plugin*)

pyjen.queue module

Abstraction around the Jenkins build queue

class pyjen.queue.**Queue**(*api*)

Bases: object

Abstraction around the Jenkins build queue

Parameters **api** (JenkinsAPI) – Pre-initialized connection to the Jenkins REST API

property items

list of scheduled builds waiting in the queue

Type list (*QueueItem*)

pyjen.queue_item module

Abstraction around a scheduled build contained in the Jenkins build queue

class pyjen.queue_item.**QueueItem**(*api*)

Bases: object

Abstraction around a scheduled build contained in the Jenkins build queue

Parameters **api** (JenkinsAPI) – Pre-initialized connection to the Jenkins REST API

property blocked

Is this scheduled build waiting for some other event to complete?

> **Warning:** May return None if this queue item has been invalidated by Jenkins

Type bool

property build

Once this scheduled build leaves the queue, this property returns a reference to the running build. While the item is still queued, this property returns None.

See the *waiting()* property on this object for a way to detect whether a queued item has left the queue or not.

Type *Build*

property buildable

is this queued build able to be built on this build farm?

> **Warning:** May return None if this queue item has been invalidated by Jenkins

Type bool

`cancel()`

Cancels this queued build

`property cancelled`

Has this queued build been cancelled?

> **Warning:** May return None if this queue item has been invalidated by Jenkins

Type bool

`is_valid()`

bool: Checks to make sure the queue item this object manages still exists

Jenkins periodically expires / invalidates queue items server-side. There is no way for us to detect or predict when this will happen. When it does, this client-side queue item object will no longer refer to a valid REST API endpoint. This helper method helps users of the PyJen library check to see if the object still points to a valid queue item.

`property job`

the Jenkins job associated with this scheduled build

> **Warning:** May return None if this queue item has been invalidated by Jenkins

Type *Job*

`property reason`

Descriptive text explaining why this build is still in the queue

> **Warning:** May return None if this queue item has been invalidated by Jenkins

Type str

`property stuck`

Is this scheduled build blocked / unable to build?

> **Warning:** May return None if this queue item has been invalidated by Jenkins

Type bool

`property uid`

unique numeric identifier of this queued build

Guaranteed to return a valid identifier, even when the queue item this object refers to has been invalidated by Jenkins.

Type int

property waiting

Is this queue item still waiting in the queue?

> **Warning:** May return None if this queue item has been invalidated by Jenkins

Type bool

pyjen.user module

Primitives for interacting with Jenkins users

class pyjen.user.**User**(*api*)

Bases: object

Interface to all primitives associated with a Jenkins user

See *author()* and *find_user()* for examples on where this class is used.

Parameters **api** (JenkinsAPI) – Pre-initialized connection to the Jenkins REST API

property description

descriptive text associated with the user. May be an empty string.

Type str

property email

Gets this users' email address as reported by Jenkins. May be None if no email on record for user.

Type str

property full_name

the users first and last names separated by a space

Type str

property user_id

the unique identifier for this user

Type str

pyjen.version module

pyjen.view module

Primitives for interacting with Jenkins views

class pyjen.view.**View**(*api*)

Bases: object

generic Jenkins views providing interfaces common to all view types

Parameters **api** (JenkinsAPI) – Pre-initialized connection to the Jenkins REST API

clone(*new_view_name*)

Make a copy of this view with the specified name

Parameters **new_view_name** (*str*) – name to give the newly created view

Returns reference to the created view, with the same configuration options as this one

Return type *View*

property config_xml

the raw XML configuration for the view

Allows callers to manipulate the raw view configuration file as desired.

> **Warning:** For advanced use only.

Type str

delete()

Deletes this view from the dashboard

delete_all_jobs()

Batch operation that deletes all jobs found in this view

disable_all_jobs()

Batch operation that disables all jobs found in this view

enable_all_jobs()

Batch operation that enables all jobs found in this view

classmethod get_supported_plugins()

list: list of PyJen plugin classes that may be used to instantiate views on this Jenkins instance

static instantiate(*json_data*, *rest_api*)

Factory method for finding the appropriate PyJen view object based on data loaded from the Jenkins REST API

Parameters

- **json_data** (*dict*) – data loaded from the Jenkins REST API summarizing the view to be instantiated
- **rest_api** (JenkinsAPI) – PyJen REST API configured for use by the parent container. Will be used to instantiate the PyJen view that is returned.

Returns PyJen view object wrapping the REST API for the given view

Return type *View*

property jenkins_plugin_name

Extracts the name of the Jenkins plugin associated with this View

The data returned by this helper property is extracted from the config XML that defines this job.

Type str

property jobs

list of 0 or more jobs associated with this view

Views are simply filters to help organize jobs on the Jenkins dashboard. This method returns the set of jobs that meet the requirements of the filter associated with this view.

Type list (*Job*)

property name

the name as it appears in the tabbed view of the main Jenkins dashboard

Type str

rename(*new_view_name*)

Changes the name of this view

Parameters **new_view_name** (*str*) – new name for this view

property view_metrics

Composes a report on the jobs contained within the view

Type dict

Module contents

Abstraction layer for the Jenkins REST API designed to simplify the interaction with the Jenkins web interface from the Python scripting environment.

1.4 Revision History

1.4.1 1.0.0

- First official release
- reworked API to eliminate easy_connect methods
- Updated unit tests to use py.test framework
- Fixed PyLint warnings

1.4.2 0.0.11dev

- Added prototype caching support for REST API endpoints
- Added Travis CI support to builds
- misc cleanup

1.4.3 0.0.10dev

- Added helper method for view metrics calculations
- misc bug fixes

1.4.4 0.0.9dev

- rewrote plugin API to make it easier to use
- overhauled the object interfaces to create parent-child relationships between entities
- added support for online API documentation from ReadTheDocs.org
- numerous improvements to the public API

1.4.5 0.0.1dev - 0.0.8dev

Early revisions of the API from its inception through to numerous changes and improvements

1.5 Frequently Asked Questions

TBD

CHAPTER

TWO

OVERVIEW

PyJen is an extensible, user and developer friendly Python interface to the Jenkins CI tool, wrapping the features exposed by the standard REST API using Pythonic objects and functions. Tested against the latest 2.x and 3.x versions of CPython and the latest trunk and LTS editions of the Jenkins REST API, we endeavor to provide a stable, reliable tool for a variety of users.

With an intuitive and well thought out interface, PyJen offers anyone familiar with the Python programming language an easy way to manage Jenkins dashboards from a simple command prompt. All core primitives of Jenkins, including views, jobs and builds are easily accessible and can be loaded, analyzed and even modified or created via simple Python commands.

Comments, suggestions and bugs may be reported to the project maintainer

CHAPTER

THREE

QUICK START GUIDE

1. First, and most obviously, you must have Python installed on your system. For details specific to your OS we recommend seeing Python's website. We recommend using the latest version of Python 2.x / 3.x for best results.
2. Next, we recommend that you install the pip package manager as described here. If you are using newer editions of Python (3.x), or if you are using certain Linux distributions / packages you likely already have this tool installed. You can confirm this by running the following command:

```
# pip --version
```

which should result in output that looks something like this:

```
pip 8.0.2 from C:\Python34x64\lib\site-packages (python 3.4)
```

3. Install PyJen directly from PyPI using PIP:

```
# pip install pyjen
```

4. import the pyjen module and start scripting! Here is a short example that shows how you can get the name of the default view from a Jenkins instance:

```
>>> from pyjen.jenkins import Jenkins
>>> jenkins_obj = Jenkins("http://localhost:8080", ('username', 'passwd'))
>>> default_view = jenkins_obj.default_view
>>> print(default_view.name)
```

PYTHON MODULE INDEX

p

INDEX

C

D

E

F

G

H

I

J

K

L

M

N

P

Q

R

S

T

U

V

W

X