PyJen Documentation

Release 0.0.9dev

Kevin S. Phillips

March 25, 2015

Contents

CHAPTER 1

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.easy_connect("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.easy_connect("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.easy_connect("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.easy_connect("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 pyjen

1.2.1 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**(*data_io_controller*, *jenkins_master*)
Bases: pyjen.view.View

Interface to a view which displays all jobs managed by this Jenkins instance

Instances of this class are typically instantiated directly or indirectly through create()

Parameters

- **data_io_controller** (DataRequester) – IO interface to the Jenkins API
- **jenkins_master** (Jenkins) – Reference to Jenkins master interface

type = 'hudson.model.AllView'

pyjen.plugins.artifactdeployer module Primitives for operating on properties of the 'artifact deployer' publishing plugin

class pyjen.plugins.artifactdeployer.**ArtifactDeployer**(*node*)
Bases: pyjen.utils.plugin_base.PluginBase

Interface to the Jenkins 'artifact deployer' publishing plugin

Parameters **node** (ElementTree.Element) – XML node defining the settings for a this plugin

entries
Gets the list of deployment options associated with this plugin

Returns list of configuration options for each set of artifacts managed by this instance

Return type list of ArtifactDeployerEntry objects

type = 'org.jenkinsci.plugins.artifactdeployer.ArtifactDeployerPublisher'

class pyjen.plugins.artifactdeployer.**ArtifactDeployerEntry**(*node*)
Bases: pyjen.utils.plugin_base.PluginBase

Interface to a single configuration of artifacts to be deployed by an Artifact Deployer instance

Parameters **node** (ElementTree.Element) – XML node defining the settings for a this plugin

remote
Gets the remote location where these artifacts are to be published

Return type str

type = 'org.jenkinsci.plugins.artifactdeployer.ArtifactDeployerEntry'

pyjen.plugins.buildblocker module Interfaces for interacting with Build Blockers job property plugin

class pyjen.plugins.buildblocker.**BuildBlockerProperty**(*node*)
Bases: pyjen.utils.plugin_base.PluginBase

Wrapper for Build Blocker job properties

Parameters **node** – ElementTree node initialized with the XML from the Jenkins job

blockers
Gets the list of search criteria for blocking jobs

Returns list of search criteria for blocking jobs

Return type list

disable()
Disables this set of build blockers

enable()
Enables this set of build blockers

is_enabled
Checks to see whether this blockers property is currently enabled

Returns True if these blocking jobs are enabled, False if not

Return type str

type = 'hudson.plugins.buildblocker.BuildBlockerProperty'

pyjen.plugins.conditionalbuilder module Primitives for operating on Jenkins job builder of type 'Conditional Builder'

class pyjen.plugins.conditionalbuilder.**ConditionalBuilder**(*node*)
Bases: pyjen.utils.plugin_base.PluginBase

Jenkins job builder plugin capable of conditionally executing a build operation

Parameters **node** (ElementTree.Element) – XML node defining the settings for a this plugin

builders
Gets a list of the build operators that will be executed if the conditions on this builder are satisfied

Returns list of build operators

Return type list of PyJen plugins that support the Jenkins builder operations

type = 'org.jenkinsci.plugins.conditionalbuildstep.ConditionalBuilder'

pyjen.plugins.flexiblepublish module Primitives for operating on job publishers of type 'Flexible Publisher'

class pyjen.plugins.flexiblepublish.**ConditionalPublisher**(*node*)
Bases: pyjen.utils.plugin_base.PluginBase

Interface to a single 'conditional' publisher contained within the flexible publish plugin

Parameters **node** (ElementTree.Element) – XML node defining the settings for a this plugin

publisher
Retrieves the action to be performed when the conditions of this publisher are met

Returns list of PyJen objects which control each conditional action to be performed

Return type list of PyJen objects, typically one or more plugins supported by the Flexible Publish plugin Return None if an publisher plugin not currently supported by PyJen is being used

type = 'org.jenkins__ci.plugins.flexible__publish.ConditionalPublisher'

class pyjen.plugins.flexiblepublish.**FlexiblePublisher**(*node*)
Bases: pyjen.utils.plugin_base.PluginBase

Publisher plugin enabling conditional execution of post-build steps in a Jenkins job

Parameters **node** (ElementTree.Element) – XML node defining the settings for a this plugin

actions
Gets the list of publishers associated with this instance of the flexible publisher

Returns list of publishers associated with this instance of the flexible publisher

Return type list of Flexible Publish publishers such as ConditionalPublisher

type = 'org.jenkins__ci.plugins.flexible__publish.FlexiblePublisher'

pyjen.plugins.freestylejob module Primitives that manage Jenkins job of type 'Freestyle'

class pyjen.plugins.freestylejob.**FreestyleJob**(*controller*, *jenkins_master*)
Bases: pyjen.job.Job

Jenkins job of type 'freestyle'

To instantiate an instance of this class using auto-generated configuration parameters, see the easy_connect() method

Parameters

- **data_io_controller** (DataRequester) – class capable of handling common HTTP IO requests sent by this object to the Jenkins REST API
- **jenkins_master** (Jenkins) – Reference to Jenkins object associated with the master instance managing this job

custom_workspace

Returns custom workspace associated with this job

Return type str

scm
Gets the object that manages the source code management configuration for a job

Returns One of several possible plugin objects which exposes the relevant set of properties supported by a given source code management tool.

Return type PluginBase

static template_config_xml()
Gets a basic XML configuration template for use when instantiating jobs of this type

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

Return type str

type = 'project'

pyjen.plugins.listview module Primitives that operate on Jenkins views of type 'List'

class pyjen.plugins.listview.**ListView**(*data_io_controller*, *jenkins_master*)
Bases: pyjen.view.View

Class that encapsulates all Jenkins related 'view' information for views of type ListView

Instances of this class are typically instantiated directly or indirectly through pyjen.View.create()

constructor

To instantiate an instance of this class using auto-generated configuration parameters, see the easy_connect() method

Parameters **data_io_controller** (*obj*) – class capable of handling common HTTP IO requests sent by this object to the Jenkins REST API

type = 'hudson.model.ListView'

pyjen.plugins.mavenplugin module Primitives that operate on Jenkins jobs of type 'Maven'

class pyjen.plugins.mavenplugin.**MavenPlugin**(*controller*, *jenkins_master*)
Bases: pyjen.job.Job

Custom Maven job type

Parameters **controller** (DataRequester) – data processing object to manage interaction with Jenkins API

static template_config_xml()
Gets a basic XML configuration template for use when instantiating jobs of this type

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

Return type str

type = 'maven2-moduleset'

pyjen.plugins.myview module Primitives for interacting with Jenkins views of type 'MyView'

class pyjen.plugins.myview.**MyView**(*data_io_controller*, *jenkins_master*)
Bases: pyjen.view.View

Interface to a view associated with a specific user

Instances of this class are typically instantiated directly or indirectly through pyjen.View.create()

To instantiate an instance of this class using auto-generated configuration parameters, see the easy_connect() method

Parameters

- **data_io_controller** (DataRequester) – class capable of handling common HTTP IO requests sent by this object to the Jenkins REST API
- **jenkins_master** (Jenkins) – Reference to Jenkins object associated with the master instance managing this job

type = 'hudson.model.MyView'

pyjen.plugins.nestedview module Primitives for working with Jenkins views of type 'NestedView'

class pyjen.plugins.nestedview.**NestedView**(*controller*, *jenkins_master*)
Bases: pyjen.view.View

Interface to Jenkins views of type "NestedView"

Views of this type contain other views as sub-views

To instantiate an instance of this class using auto-generated configuration parameters, see the easy_connect() method

Parameters

- **controller** (DataRequester) – class capable of handling common HTTP IO requests sent by this object to the Jenkins REST API
- **jenkins_master** (Jenkins) – Reference to Jenkins object associated with the master instance managing this job

all_views
Gets all views contained within this view and it's children, recursively

Returns list of all views contained within this view and it's children, recursively

Return type list

clone_subview(*existing_view*, *new_view_name*)
Creates a clone of an existing view under this nested view

Parameters

- **existing_view** (View) – Instance of a PyJen view to be cloned
- **new_view_name** (*str*) – the new name for the view

Returns reference to new PyJen view object

Return type View

create_view(*view_name*, *view_type*)
Creates a new sub-view within this nested view

Parameters

- **view_name** (*str*) – name of the new sub-view to create
- **view_type** (*str*) – data type for newly generated view

find_view(*view_name*)
Attempts to locate a sub-view under this nested view with the given name

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

Returns Reference to View object for the view with the given name, or None if no view with that name exists

Return type Object derived from View

has_view(*view_name*)
Checks to see whether a view with the given name already exists under this view

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

Returns True if a view with that name already exists, otherwise false

Return type bool

move_view(*existing_view*)
Moves an existing view to a new location

NOTE: The original view object becomes obsolete after executing this operation

Parameters **existing_view** (View) – Instance of a PyJen view to be moved

Returns reference to new, relocated view object

Return type View

type = 'hudson.plugins.nested__view.NestedView'

views
Gets all views contained within this view

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

Returns list of all views contained within this view

Return type list

pyjen.plugins.nullscm module Primitives for operating on SCM properties of Jenkins jobs with no source control configuration

class pyjen.plugins.nullscm.**NullSCM**(*node*)
Bases: pyjen.utils.plugin_base.PluginBase

SCM plugin for Jobs with no source control configurations

Parameters **node** (ElementTree.Element) – XML node defining the settings for a this plugin

type = 'hudson.scm.NullSCM'

pyjen.plugins.paramtrigger module Primitives for operating on Jenkins post-build publisher of type Parameterized Build Trigger

class pyjen.plugins.paramtrigger.**BuildTriggerConfig**(*node*)
Bases: object

job_names
Gets a list of names of jobs triggered by this one

Returns list of job names

Return type list of str

class pyjen.plugins.paramtrigger.**ParameterizedBuildTrigger**(*node*)
Bases: pyjen.utils.plugin_base.PluginBase

SCM plugin for Jobs with no source control configurations

Parameters **node** (ElementTree.Element) – XML node defining the settings for a this plugin

triggers
Gets the list of trigger operations defined for this instance of the plugin

Return type list of BuildTriggerConfig objects

type = 'hudson.plugins.parameterizedtrigger.BuildTrigger'

pyjen.plugins.sectionedview module Primitives for working on Jenkins views of type 'SectionedView'

class pyjen.plugins.sectionedview.**ListViewSection**(*node*)
Bases: pyjen.utils.plugin_base.PluginBase

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

Represents sections of type 'ListView'

Parameters **node** (ElementTree.Element) – XML node defining the settings for a ListView section

include_regex
regular filter for jobs to be shown in this section

Return type str

type = 'hudson.plugins.sectioned_view.ListViewSection'

class pyjen.plugins.sectionedview.**SectionedView**(*controller*, *jenkins_master*)
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

- **controller** (DataRequester) – class capable of handling common HTTP IO requests sent by this object to the Jenkins REST API
- **jenkins_master** (Jenkins) – Reference to Jenkins object associated with the master instance managing this job

sections

Returns a list of sections contained within this view

Return type list of one of the 'SectionedView' section types

type = 'hudson.plugins.sectioned_view.SectionedView'

class pyjen.plugins.sectionedview.**SectionedViewXML**(*xml*)
Bases: pyjen.utils.viewxml.ViewXML

Abstraction for operating on raw config.xml data for a Jenkins view of type 'Sectioned View'

Parameters **xml** (*str*) – XML string describing a sectioned view

sections

Returns a list of all 'section' objects contained in this view

Return type list of section plugins associated with this view

class pyjen.plugins.sectionedview.**TextSection**(*node*)
Bases: pyjen.utils.plugin_base.PluginBase

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

Sections of this type contain simple descriptive text

Parameters **node** (ElementTree.Element) – XML node defining the settings for a ListView section

type = 'hudson.plugins.sectioned_view.TextSection'

pyjen.plugins.statusview module Primitives for operating on Jenkins views of type 'StatusView'

class pyjen.plugins.statusview.**StatusView**(*controller*, *jenkins_master*)
Bases: pyjen.view.View

Interface to Jenkins views of type 'StatusView'

Parameters

- **controller** (DataRequester) – class capable of handling common HTTP IO requests sent by this object to the Jenkins REST API
- **jenkins_master** (Jenkins) – Reference to Jenkins object associated with the master instance managing this job

type = 'hudson.plugins.status_view.StatusView'

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: object

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

Parameters **node** (ElementTree.Element) – XML node defining the settings for a this plugin

depth_option

Returns the current SVN 'depth' options associated with this module

Return 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

ignore_externals
Checks to see whether the 'ignore externals' option is enabled on this job

Returns True if ignore externals is enabled, otherwise False

Return type bool

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

Return type str

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

Return type str

class pyjen.plugins.subversion.**Subversion**(*node*)
Bases: pyjen.utils.plugin_base.PluginBase

Subversion SCM job plugin

Parameters **node** (`ElementTree.Element`) – XML node defining the settings for a this plugin

included_regions
list of patterns reflecting the regions of the SVN repo to include in SCM operations

Return type `list` of `str`

locations
Gets the list of SVN URLs associated with this plugin instance

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

Return type `list` of `ModuleLocation` objects

type = 'hudson.scm.SubversionSCM'

Module contents This sub-package contains modules that manage PyJen plugins which map to Jenkins plugins

For details on how these plugins interact with PyJen and Jenkins see `pyjen.utils.plugin_base`

pyjen.utils package

Submodules

pyjen.utils.datarequester module Primitives for handling direct IO with the Jenkins REST API

class `pyjen.utils.datarequester.`**DataRequester**(*jenkins_url*, *username*, *password*)
Bases: `object`

Abstraction layer encapsulate all IO requests for the Jenkins REST API

Parameters

- **jenkins_url** (*str*) – HTTP URL to use for all subsequent IO operations performed on this object.
- **username** (*str*) – Jenkins user name to use for authentication. May be set to None for anonymous access.
- **password** (*str*) – Password for the given Jenkins user, to use for authentication. May be set to None for anonymous access.

classmethod clear()
Deletes all cached data so subsequent operations will reload from source

WARNING: Make sure to call flush() before clear() if there are potentially unwritten changes in the cache

clone(*new_url=None*)
create a copy of this connection object

Parameters **new_url** (*str*) – optional replacement URL associated with the cloned object credentials will be preserved in the clone

Returns new DataRequester object, with settings cloned from this instance

Return type `DataRequester`

config_xml
Configuration file used to manage the Jenkins entity backed by this object

Return type `str`

credentials
Gets the authentication credentials used for all IO operations on this object

Returns user name and password used for authenticated communication with Jenkins

Return type tuple() of str

flush()
Ensures that any non-synchronized changes cached by this object are uploaded to the remote Jenkins server

get_api_data(*query_params=None*)
Convenience method that retrieves the Jenkins API specific data from the specified URL

Parameters **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 object

get_data(*path=None*)
Convenience method to convert text data loaded from a Jenkins URL to Python data types

Parameters **path** (*str*) – optional extension path to append to the root URL managed by this object when performing the get operation

Returns The results of converting the text data loaded from the Jenkins URL into appropriate Python objects

Return type object

get_headers(*path=None*)
gets the HTTP header attributes 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

Returns dictionary of HTTP header attributes with their associated values

Return type dict

get_text(*path=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

Returns the text loaded from this objects' URL

Return type str

is_dirty
Checks to see if there are any unsynchronized changes pending on this object

Returns True if there are changes cached in this instance that have not yet been flushed to the remote Jenkins server, False otherwise

Return type bool

post(*path=None*, *args=None*)
sends data to or triggers an operation via a Jenkins URL

Parameters

- **path** (*str*) – optional extension path to append to the root URL managed by this object when performing the post operation
- **args** (*dict*) – optional set of data arguments to be sent with the post operation supported keys are as follows:
  - 'headers' - dictionary of HTTP header properties and their associated values
  - 'data' - dictionary of assorted / misc data properties and their values

url

Gets the URL used by all IO operations on this object

Returns the URL used by all IO operations on this object

Return type str

pyjen.utils.helpers module Primitives that perform some common Jenkins operations that span the object heirarchy

The functions and classes defined in this module provide users with some pre-rolled custom scripts that perform some common tasks that leverage a variety of tools and objects provided by the PyJen API to accomplish their tasks (ie: these primitives typically make use of multiple classes offered by the API and thus can't be easily attached or embedded within the public PyJen API)

pyjen.utils.helpers.**find_view**(*jenkins_url*, *credentials*, *view_name*)

Locates a view with a given name recursively across a Jenkins instance

This helper function has knowledge of view plugins that support sub-views and thus recursively searches these sub-views for the requested view

WARNING: This function can be quite slow when executed against a large Jenkins build farm with a large number of views and subviews.

Parameters

- **jenkins_url** (*str*) – URL of the root Jenkins master
- **credentials** (*tuple*) – 2-tuple containing the user-name and password to authenticate with
- **view_name** (*str*) – name of the view to locate

Returns Reference to the view with the provided name, or None if the view doesn't exist

Return type View

pyjen.utils.jobxml module Abstractions for managing the raw config.xml for a Jenkins job

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

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 **xml** (*str*) – Raw XML character string extracted from a Jenkins job.

XML

Extracts the processed XML for export to a Jenkins job

Returns Raw XML containing any and all customizations applied in previous operations against this object. This character string can be imported into Jenkins to configure a job.

Return type str

assigned_node
Gets the build agent label this job is associated with

Returns the build agent label this job is associated with

Return type str

builders
Gets a list of 0 or more build operations associated with this job

Returns a list of build operations associated with this job

Return type list of builder plugins used by this job

custom_workspace
Gets the local path for the custom workspace associated with this job

Returns the local path for the custom workspace associated with this job

Return type str

disable_custom_workspace()
Disables a jobs use of a custom workspace

If the job is not currently using a custom workspace this method will do nothing

properties
Gets a list of 0 or more Jenkins properties associated with this job

Returns a list of customizable properties associated with this job

Return type list of property plugins supported by this job

publishers
Gets a list of 0 or more post-build publisher objects associated with this job

Returns a list of post-build publishers associated with this job

Return type list of publisher plugins supported by this job

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

Examples: Subversion

Return type PluginBase

pyjen.utils.plugin_base module Declaration for abstract base class to be used by all PyJen plugins

class pyjen.utils.plugin_base.**PluginBase**
Bases: object

Abstract base class common to all PyJen API plugins

All PyJen plugins must derive, directly or indirectly, from this class and implement its abstract interface

Most plugins will derive directly from this class, however plugins that extend the native Jenkins objects like views and jobs must derive from their appropriate base classes instead. Any class that supports such extensions will, themselves, derive from this class, including `View` and `Job`.

type
The Jenkins plugin descriptive name, used when instantiating objects of that type

Some examples from the built-in plugins are:

- "hudson.scm.NullSCM"
- "hudson.scm.SubversionSCM"
- "hudson.model.MyView"

These names can typically be copied verbatim from the XML node in the Jenkins config.xml for the entity that describes the plugin properties. The name should be defined by an XML attribute named "class". Here is an example of the SVN plugin XML

```
<scm class="hudson.scm.SubversionSCM" plugin="subversion@2.3">
```

For plugins that extend Jenkins native objects like views and jobs the plugin name will be defined in the name of the tag itself, like this

```
<hudson.plugins.nested__view.NestedView plugin="nested-view@1.14">
```

TIP: When implementing this property on a concrete class, you will need to declare a static class attribute for the PyJen plugin API to work correctly, something like

```
class MyClass(PluginBase):
    type = "my.name.of.plugin"
```

Returns Jenkins plugin descriptive name, used when instantiating objects of that type

Return type `str`

pyjen.utils.pluginapi module Primitives for interacting with the PyJen plugin API

class `pyjen.utils.pluginapi.`**PluginXML**(*xml_node*)
Bases: `object`

Class used to process XML configuration information associated with Jenkins plugins

Parameters **xml_node** (`xml.etree.ElementTree`) – the XML sub-tree defining the properties of this plugin

get_class_name()
Gets the Java class name of the plugin

Returns the Java class name

Return type `str`

get_module_name()
Gets the name of the plugin

Returns the plugin name

Return type `str`

get_version()
Gets the version of the plugin

Returns the plugin version

Return type str

pyjen.utils.pluginapi.**create_xml_plugin**(*xml_node*)
Instantiates the appropriate XML-compatible PyJen plugin

Parameters **xml_node** (*xml.etree.ElementTree*) – the node of the XML configuration defining the plugin configuration

Returns a pre-initialized plugin of the appropriate type, or None if no supported plugin can be found

Return type PluginBase derived class

pyjen.utils.pluginapi.**find_plugin**(*plugin_type*)
Locates a PyJen plugin of the given type

Parameters **plugin_type** (*str*) – the descriptive type-name for the plugin to find

Returns reference to the plugin class for the specified type, or None if a compatible plugin could not be found

pyjen.utils.pluginapi.**get_job_plugins**()
Returns a list of plugins that extend the default Jenkins Job type

Returns list of plugins that extend the default Jenkins Job type

Return type list of PluginBase derived classes

pyjen.utils.pluginapi.**get_plugin_name**(*xml_node*)
Extracts the name of a plugin from an XML snippet

Parameters **xml_node** (*xml.etree.ElementTree*) – the node of the XML configuration defining the plugin configuration

Returns Name of the plugin this snippet is generated by

Return type str

pyjen.utils.pluginapi.**get_plugins**()
Returns list of classes for all plugins supported by PyJen

Returns list of classes for all PyJen plugins

Return type list of PluginBase derived objects

pyjen.utils.pluginapi.**get_view_plugins**()
Returns a list of plugins that extend the default Jenkins View type

Returns list of plugins that extend the default Jenkins View type

Return type list of PluginBase derived classes

pyjen.utils.pluginapi.**init_extension_plugin**(*dataio*, *jenkins_master*)
Instantiates a plugin that extends one of the Jenkins native objects such as a view or job

Parameters

- **dataio** – Jenkins REST API interface, initialized with the connection parameters of the new object
- **jenkins_master** – Instance of the Jenkins master object that manages this entity

Returns PyJen plugin pre-initialized with the source data, or None if no compatible plugin could be found

Return type PluginBase derived object

pyjen.utils.user_params module Interfaces for parsing user defined configuration parameters

class pyjen.utils.user_params.**JenkinsConfigParser**(*defaults=None*, *dict_type=<class 'collections.OrderedDict'>*, *allow_no_value=False*)

Bases: ConfigParser.ConfigParser

Interface to the PyJen user configuration file

```
Config File Format
==================
[http://jenkins_server_url]
username=MyUserName
password=MyPassword

[http://another_jenkins_url]
username=other_username
password=other_password

#Anonymous access can be defined like this
[http://some_jenkins_url]
username=
password=
```

For more details on the general format of the config file see these links: https://wiki.python.org/moin/ConfigParserExamples https://docs.python.org/2/library/configparser.html

get_credentials(*jenkins_url*)

Gets the authentication credentials for a given Jenkins URL

Parameters **jenkins_url** (*str*) – arbitrary URL to the Jenkins REST API to retrieve credentials for URL may point to any arbitrary artifact on the Jenkins REST API. The credentials will be matched based on the section headers in any of the associated config files

Returns username and password for the given URL. Will return None if no credentials found.

Return type tuple()

static **get_default_configfiles**()

Gets a list of potential locations where PyJen config files may be found

Returns list of paths to be searched

Return type list

pyjen.utils.viewxml module Abstractions for managing the raw config.xml for a Jenkins view

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

Bases: object

Wrapper around the config.xml for a Jenkins view

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 **xml** (*str*) – Raw XML character string extracted from a Jenkins job.

XML

Extracts the processed XML for export to a Jenkins job

Returns Raw XML containing any and all customizations applied in previous operations against this object. This character string can be imported into Jenkins to configure a job.

Return type str

rename(*new_name*)
Changes the name of the view

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

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**(*data_io_controller*)
Bases: object

Class that encapsulates information about a single build / run of a Job

Builds are executions of jobs and thus instances of this class are typically generated from the Job class.

See also:

Job

Parameters **data_io_controller** (DataRequester) – class capable of handling common HTTP IO requests sent by this object to the Jenkins REST API

artifact_urls
Gets a list of URLs which can be used to download the published build artifacts for this build

Return type list of str

changeset
Gets the list of SCM changes associated with this build

Returns 0 or more SCM changesets associated with / included in this build. If no changesets are found, returns None

Return type Changeset

console_output
Gets the raw console output for this build as plain text

Returns Raw console output from this build, in plain text format

Return type str

description
Gets the descriptive test associated with this build

Return type str

id
Gets the unique identifier associated with this build

Return type str

is_building
Checks to see whether this build is currently executing

Returns True if the build is executing otherwise False

Return type bool

number
Gets the sequence number of this build

Returns sequentially assigned integer value associated with this build

Return type int

start_time
Gets the time stamp of when this build was started

Returns the date and time at which this build was started

Return type datetime.datetime

status
Gets the status of the build

Returns

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

- "SUCCESS"
- "UNSTABLE"
- "FAILED"

Return type str

pyjen.changeset module

Primitives for interacting with SCM changesets

class pyjen.changeset.**Changeset**(*data*, *controller*)
Bases: object

manages the interpretation of the "changeSet" properties of a Jenkins build

See also:

Build

Parameters

- **data** (*dict*) – Dictionary of data elements typically parsed from the "changeSet" node of a builds source data as provided by the Jenkins REST API. Should have at least the following keys:
  - **'kind'** - string describing the SCM tool associated with this change all changes reported by this object are expected to be stored in the same SCM tool
  - **'items'** - list of 0 or more actual changesets included in the associated build
- **controller** (DataRequester) – object controlling access to Jenkins API

affected_items
gets details of the changes associated with the parent build

Returns list of 0 or more items detailing each change associated with this Changeset

Return type list of ChangesetItem objects

has_changes
Checks whether or not there are any SCM changes

Returns True if changes have been found, False if not

Return type bool

scm_type
Gets the name of the SCM tool associated with this change

Returns Name of the SCM tool associated with this change

Return type str

class pyjen.changeset.**ChangesetItem**(*data*, *controller*)
Bases: object

Encapsulates all info related to a single change in a Changeset

See also:

Changeset

Parameters

- **data** (*dict*) – Dictionary of attributes describing this single changeset
- **controller** (DataRequester) – Interface to the Jenkins API

author

Returns Person who committed this change to the associated SCM

Return type User

message

Returns SCM commit message associated with this change

Return type str

pyjen.exceptions module

All PyJen specific exception declarations

exception pyjen.exceptions.**InvalidJenkinsURLError**(*msg*, *url*)
Bases: pyjen.exceptions.PyJenError

Exception raised when attempting to connect to a URL that doesn't point to a valid Jenkins REST API

constructor

Parameters

- **msg** (*str*) – Descriptive message associated with this exception
- **url** (*str*) – URL in question that does not point to a valid Jenkins REST API

exception pyjen.exceptions.**InvalidParameterError**(*msg*)
Bases: pyjen.exceptions.PyJenError

Exception raised when the caller provides an invalid value as an input parameter to a PyJen method call

Constructor

Parameters **msg** (*str*) – Descriptive message associated with this exception

exception pyjen.exceptions.**InvalidUserParamsError**(*msg*)
Bases: pyjen.exceptions.PyJenError

Exception caused by invalid parameters in the user configuration file

constructor

Parameters **msg** (*str*) – Descriptive message associated with this exception

exception pyjen.exceptions.**JenkinsFlushFailure**(*failed_items*)
Bases: pyjen.exceptions.PyJenError

Exception raised when flushing cached Jenkins data to the remote server fails

failed_items

exception pyjen.exceptions.**NestedViewCreationError**(*msg*)
Bases: pyjen.exceptions.PyJenError

Error when creating a sub-view in the nested-view plugin

Constructor

Parameters **msg** (*str*) – Descriptive message associated with this exception

exception pyjen.exceptions.**NotYetImplementedError**
Bases: pyjen.exceptions.PyJenError

Exception thrown from methods that are not yet implemented

constructor

exception pyjen.exceptions.**PluginNotSupportedError**(*message*, *plugin_name*)
Bases: exceptions.NotImplementedError

Basic extension to the NotImplementedError with details about which plugin was not found

Constructor

Parameters

- **message** (*str*) – description of the error
- **plugin_name** (*str*) – the class name / type of the plugin that was not found

message

plugin_name

exception pyjen.exceptions.**PyJenError**
Bases: exceptions.Exception

Base class for all PyJen related exceptions

pyjen.jenkins module

Primitives for interacting with the main Jenkins dashboard

class pyjen.jenkins.**Jenkins**(*data_io_controller*)
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` - abstraction for a view on the dashboard, allowing jobs to be filtered based on different criteria like job name.
- `Job` - abstraction for a Jenkins job, allowing manipulation of job settings and controlling builds of those jobs
- `Build` - abstraction for an instance of a build of a particular job

Example: finding a job

```
j = Jenkins.easy_connect('http://localhost:8080')
job = j.find_job('My Job')
if job is None:
    print ('no job by that name found')
else:
    print ('job ' + job.name + ' found')
```

Example: find the build number of the last good build of the first job on the default view

```
j = pyjen.Jenkins.easy_connect('http://localhost:8080/')
v = j.get_default_view()
jobs = v.get_jobs()
lgb = jobs[0].get_last_good_build()
print ('last good build of the first job in the default view is ' + lgb.get_build_number())
```

To instantiate an instance of this class using auto-generated configuration parameters, see the `easy_connect()` method

Parameters **data_io_controller** (`DataRequester`) – class capable of handling common HTTP IO requests sent by this object to the Jenkins REST API

all_job_names
Gets list of all jobs found on this server

cancel_shutdown()
Cancels a previous scheduled shutdown sequence

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

create_job(*job_name*, *job_type*)
Creates a new job on this Jenkins instance

Parameters

- **job_name** (*str*) – The name for the job to be created. expected to be universally unique on this instance of Jenkins
- **job_type** (*str*) – descriptive type for the base configuration of this new job for a list of currently supported job types see `job_types()`

create_view(*view_name*, *view_type*)
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_type** (*str*) – type of view to create must match one or more of the available view types supported by this Jenkins instance. See `view_types()` for a list of supported view types.

Returns An object to manage the newly created view

Return type View

default_view
returns a reference to 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.

Returns object that manages the default Jenkins view

Return type View

static easy_connect(*url*, *credentials=None*)
Factory method to simplify creating connections to Jenkins servers

Parameters

- **url** (*str*) – Full URL of the Jenkins instance to connect to. Must be a valid running Jenkins instance.
- **credentials** (*tuple*) – A 2-element tuple with the username and password for authenticating to the URL If omitted, credentials will be loaded from any pyjen config files found on the system If no credentials can be found, anonymous access will be used

Returns Jenkins object, pre-configured with the appropriate credentials and connection parameters for the given URL.

Return type Jenkins

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, otherwise None

Return type Job

find_node(*nodename*)
Locates a Jenkins build agent with the given name on this Jenkins instance

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

Returns reference to Jenkins object that manages this node's information.

Return type Node or None if node not found

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.

Return type User or None if user not found

find_view(*view_name*)
Searches views directly managed by this Jenkins instance for a specific view

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 None

Return type View

flush_cache()
Flushes any pending writes to the remote Jenkins server

WARNING: This method interacts with a new, crude prototype caching system being tested and should not be used in production

get_job(*url*)
Establishes a connection to a Job based on an absolute URL

This method may be a bit less convenient to use in certain situations but it has better performance than `find_job()`

Parameters **url** (*str*) – absolute URL of the job to load

Returns an instance of the appropriate Job subclass for the given job

Return type `Job`

get_node(*url*)
Loads data for a Jenkins build agent based on an absolute URL

This method may be a bit less convenient to use in certain situations but it has better performance than `find_node()`

Parameters **url** (*str*) – absolute URL of the node data to load

Returns A node object allowing interaction with the given node's settings and information

Return type `Node`

get_user(*url*)
Establishes a connection to a Jenkins User based on an absolute URL

This method may be a bit less convenient to use in certain situations but it has better performance than `find_user()`

Parameters **url** (*str*) – absolute URL of the user to load

Returns A user object allowing interaction with the given user's settings and information

Return type `User`

get_view(*url*)
Establishes a connection to a View based on an absolute URL

This method may be a bit less convenient to use in certain situations but it has better performance than `find_view()`

Parameters **url** (*str*) – absolute URL of the view to load

Returns an instance of the appropriate View subclass for the given view

Return type `View`

is_shutting_down
checks to see whether the Jenkins master is in the process of shutting down.

Returns If the Jenkins master is preparing to shutdown (ie: in quiet down state), return True, otherwise returns False.

Return type `bool`

job_types

Returns a list of Jenkins job types currently supported by this instance of PyJen Elements from this list may be used when creating new jobs on this Jenkins instance, so long as the accompanying job type is supported by the live Jenkins server

Return type list of str

nodes
gets the list of nodes (aka: agents) managed by this Jenkins master

Returns list of 0 or more Node objects managed by this Jenkins master

Return type list of Node objects

prepare_shutdown()
Sends a shutdown signal to the Jenkins master preventing 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

reset_cache()
Resets all cached data

WARNING: Any unwritten changes to the cache will be lost if not flushed previously using the flush_cache() method

WARNING: This method interacts with a new, crude prototype caching system being tested and should not be used in production

version
Gets the version of Jenkins pointed to by this object

Returns Version number of the currently running Jenkins instance

Return type str

view_names
Gets a list of the names of the views managed by this Jenkins instance

Return type list of View objects

view_types

Returns a list of Jenkins view types currently supported by this instance of PyJen Elements from this list may be used when creating new views on this Jenkins instance, so long as the accompanying view type is supported by the live Jenkins server

Return type list of str

views
Gets a list of all views directly managed by the Jenkins dashboard

To retrieve all views managed by this Jenkins instance, including recursing into views that support sub-views, see the all_views() property

Returns list of one or more views defined on this Jenkins instance.

Return type list of View objects

pyjen.job module

Primitives for interacting with Jenkins jobs

class pyjen.job.**Job**(*controller*, *jenkins_master*)
Bases: pyjen.utils.plugin_base.PluginBase

'Abstract' base class used by all job classes, providing functionality common to them all

Parameters

- **controller** (DataRequester) – IO interface which manages interaction with the live Jenkins job
- **jenkins_master** (Jenkins) – Jenkins instance containing this job

all_builds
Gets all recorded builds for this job

Returns all recorded builds for this job

Return type list of Build objects

all_downstream_jobs
Gets the list of all jobs that depend on this job, including all indirect descendants

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

Returns A list of 0 or more jobs which depend on this one

Return type list of Job objects

all_upstream_jobs
Gets the list of all jobs that this job depends on, including all indirect descendants

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

Returns A list of 0 or more jobs this job depend on

Return type list of Job objects

builders
Gets all plugins configured as 'builders' for this job

clone(*new_job_name*)
"Create a new job with the same configuration as this one

Parameters **new_job_name** (*str*) – Name of the new job to be created

config_xml
Gets the raw XML configuration for the job

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

Returns the full XML tree describing this jobs configuration

Return type str

static **create**(*controller*, *jenkins_master*)
Factory method used to instantiate the appropriate job type for a given configuration

Parameters

- **controller** (DataRequester) – IO interface to the Jenkins API. This object is expected to be pre-initialized with the connection parameters for the job to be processed.
- **jenkins_master** (Jenkins) – Jenkins instance containing this job

Returns An instance of the appropriate derived type for the given job

Return type Job

delete()
Deletes this job from the Jenkins dashboard

disable()
Disables this job

Sets the state of this job to disabled so as to prevent the job from being triggered.

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

downstream_jobs
Gets the list of jobs to be triggered after this job completes

Returns A list of 0 or more jobs which depend on this one

Return type `list` of `Job` objects

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

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 Build object for the build with the given numeric identifier If such a build does not exist, returns None

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

Returns a list of 0 or more builds

Return type `list` of `Build` objects

has_been_built
Checks to see whether this job has ever been built or not

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

Return type `bool`

is_disabled
Indicates whether this job is disabled or not

Returns True if the job is disabled, otherwise False

Return type `bool`

last_build
Returns a reference to the most recent build of this job

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

Returns object that provides information and control for the most recent build of this job. If there are no such builds in the build history, this method returns None

Return type Build

last_failed_build
Returns a reference to 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

Returns Most recent build with a status of 'failed' If there are no such builds in the build history, this method returns None

Return type Build

last_good_build
Gets the most recent successful build of this job

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

Returns object that provides information and control for the last build which completed with a status of 'success' If there are no such builds in the build history, this method returns None

Return type Build

last_stable_build
Returns a reference to 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

Returns Most recent build with a status of 'stable' If there are no such builds in the build history, this method returns None

Return type Build

last_unsuccessful_build
Returns a reference to 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

Returns Most recent build with a status of 'unstable' If there are no such builds in the build history, this method returns None

Return type Build

name
Returns the name of the job managed by this object

Returns The name of the job

Return type str

properties
Gets all plugins configured as extra configuration properties for this job

publishers
Gets all plugins configured as 'publishers' for this job

recent_builds
Gets a list of the most recent builds for 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.

Returns a list of the most recent builds for this job

Return type list of Build objects

start_build()
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.

static **supported_types**()
Returns a list of all job types supported by this instance of PyJen

These job types can be used in such methods as `create_job()`, which take as input a job type classifier

Returns list of all job types supported by this instance of PyJen, including those supported by plugins

Return type `str`

static **template_config_xml**(*job_type*)
Generates a generic configuration file for use when creating a new job on the live Jenkins instance

Parameters **job_type** (*str*) – the type descriptor of the job being created For valid values see the `supported_types()` method

Returns XML configuration data for the specified job type

Return type `str`

upstream_jobs
Gets the list of upstream dependencies for this job

Returns A list of 0 or more jobs that this job depends on

Return type `list` of `Job` objects

pyjen.node module

Declarations for the abstraction of a Jenkins build agent

class pyjen.node.**Node**(*data_io_controller*)
Bases: `object`

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

Use this class to manipulate agents managed by a Jenkins master

Instances of this class are typically created using one of the node methods on the Jenkins class, such as `find_node()`

To instantiate an instance of this class using auto-generated configuration parameters, see the `easy_connect()` method

Parameters **data_io_controller** (`DataRequester`) – class capable of handling common HTTP IO requests sent by this object to the Jenkins REST API

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

Returns returns True if there are no active builds on this Node at the moment otherwise returns False

Return type `bool`

is_offline
Checks to see whether this Node is currently offline or not

Returns True if this Node is offline otherwise False

Return type bool

name
Gets the display name of this Node

Returns the name of this Node

Return type str

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" when calling this method, and vice versa.

Parameters **message** (*str*) – optional descriptive message to display on the dashboard 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*) – The maximum 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.user module

Primitives for interacting with Jenkins users

class pyjen.user.**User**(*data_io_controller*)
Bases: object

Interface to all primitives associated with a Jenkins user

Instances of this class are typically created using one of the user methods on the Jenkins class, such as find_user()

Parameters **data_io_controller** (DataRequester) – class capable of handling common HTTP IO requests sent by this object to the Jenkins REST API

description
Gets some descriptive text associated with the user

Returns some descriptive text explaining something about this user. May be None if no description found

Return type str

email
Gets this users' email address as reported by Jenkins

Returns email address of this user

Return type str

full_name
Gets the users full name, typically first and last names separated by a space

Returns this users' full name

Return type str

user_id
Gets the unique identifier for this user

Returns unique identifier for this user

Return type str

pyjen.view module

Primitives for interacting with Jenkins views

class pyjen.view.**View**(*data_io_controller*, *jenkins_master*)
Bases: pyjen.utils.plugin_base.PluginBase

'Abstract' base class used by all view classes, providing functionality common to them all

Parameters

- **data_io_controller** (DataRequester) – IO interface which manages interaction with the live Jenkins view
- **jenkins_master** (Jenkins) – Jenkins instance containing this job

clone(*new_view_name*)
Make a copy of this view with the specified name

Parameters **new_view_name** (*str*) – name of the newly cloned view

Returns reference to the View object that manages the new, cloned view

Return type View

clone_all_jobs(*source_job_name_pattern*, *new_job_substring*)
Batch-clones all jobs contained within this view

Parameters

- **source_job_name_pattern** (*str*) – pattern to use as a substitution rule when generating new names for cloned jobs. Substrings within the existing job names that match this pattern will be replaced by the given substitution string
- **new_job_substring** (*str*) – character string used to generate new job names for the clones of the existing jobs. The substring of an existing job that matches the given regex will be replaced by this new string to create the new job name for it's cloned counterpart.

config_xml
Gets the raw configuration data in XML format

This is an advanced function which allows the caller to manually manipulate the raw configuration settings of the view. Use with caution.

This method allows callers to dynamically update arbitrary properties of this view.

Returns returns the raw XML of the views configuration in a plain text string format

Return type str

static create(*controller*, *jenkins_master*)
Factory method used to instantiate the appropriate view type for a given configuration

Parameters

- **controller** (DataRequester) – IO interface to the Jenkins API. This object is expected to be pre-initialized with the connection parameters for the view to be processed.
- **jenkins_master** (Jenkins) – Jenkins instance containing this job

Returns An instance of the appropriate derived type for the given view

Return type View

delete()
Deletes this view from the dashboard

delete_all_jobs()
Batch method that allows callers to do bulk deletes of all jobs found in this view

disable_all_jobs()
Batch method that allows caller to bulk-disable all jobs found in this view

enable_all_jobs()
Batch method that allows caller to bulk-enable all jobs found in this view

job_count
Gets the number of jobs contained under this view

Returns number of jobs contained under this view

Return type int

job_names
Gets the list of names of all jobs contained within this view

Returns the list of names of all jobs contained within this view

Return type list of str

jobs
Gets a list of 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.

Returns list of 0 or more jobs that are included in this view

Return type list of Job objects

name
Gets the display name for this view

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

Returns the name of the view

Return type str

rename(*new_name*)
Rename this view

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

static supported_types()
Returns a list of all view types supported by this instance of PyJen

These view types can be used in such methods as create_view(), which take as input a view type classifier

Returns list of all view types supported by this instance of PyJen, including those supported by plugins

Return type `list` of `str`

Module contents

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

1.3 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 here.

To start working on an improvement for the project, start by creating a development branch and committing your work there. When you are happy with the changes you have made simply perform a pull request.

We try to keep our code inline with PEP-8 standards, and we do have PyLint support in the project to verifying the content meets this standard. 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 wiith both of these versions wherever possible.

1.3.1 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 plugin they may like.

At the most basic level, PyJen plugins are simply Python classes that meet the following two criteria:

- the class declarations must be placed in a module under the pyjen/plugins subfolder
- the class must derive, directly or indirectly, from te `PluginBase` 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.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.

1.4 Revision History

1.4.1 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

1.4.2 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 2

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. Used in production in at least one major software development company (CARIS), 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 3

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 distributionss / 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 1.5.6 from C:\Users\kevin\Documents\python\pyjen\py3\lib\site-packages (python 3.4)
```

3. Install PyJen directly from PyPI using PIP:

```
# pip install pyjen --pre
```

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.easy_connect("http://localhost:8080")
>>> default_view=jenkins_obj.default_view
>>> print(default_view.name)
```

p

A

B

C

D

E

F

G

H

I

J

T

U

V

W

X