Projects API

Release 15.4.1.17

CONTACT Software

Oct 25, 2018

Contents

1 Project **1**

2 Task **2**
2.1 Copy 2

3 Microsoft Project Interface **3**
3.1 Importing XML 3
3.2 Exporting XML 4
3.3 Example 4

Python Module Index **8**

Index **9**

CHAPTER 1

Project

This module provides the business logic of a project.

class cs.pcs.projects.**Project**(***values*)

DefaultCalendarProfileName
Returns the name of the default calendar profile to be used by creating a project.

Can be overwritten by customizations in order to preset the name depending on location, project category or whatever needed.

:return: string

default_roles = ['Projektmitglied']
List of project roles.

Project roles must be defined in the project role catalog.

The values of the identifier of the project role must be entered.

Each project role that is defined, can be used as a default role in a derived class similar to the following lines:

```
from cs.pcs.projects import Project

class MyProject(Project):

    default_roles = ['Projektmitglied', 'Projektassistent']
```

CHAPTER 2

Task

2.1 Copy

To ensure an appropriate performance, when copying task structures, the standard operations *Create* or *Copy* are not used, for example, when creating a project from a template or copying a task group. In these cases, the user exits are not executed and the customization is limited.

Note: To ensure an appropriate performance, when copying task structures, the user exits are not executed and the customization is limited.

For further details please refer to your system consultant.

CHAPTER 3

Microsoft Project Interface

The interface between PCS projects and Microsoft Project plans is based on MSP's own XML format. For further information regarding the XML schema please refer to: https://msdn.microsoft.com/en-us/library/bb428843.aspx

3.1 Importing XML

Import or update PCS projects via MSP's XML schema.

3.1.1 XmlMergeImport

class cs.pcs.msp.imports.**XmlMergeImport**

Following class members are exposed for customization purposes:

PROJECT_MAPPING

Dictionary with a mapping between MSP project fields (as keys) and PCS project attributes (as values). Values can either be single strings, thus the targeted attribute names, or tuples containing a callback class method name and then the targeted attribute names. Callback method signature:

```
def msp_field_x_to_pcs_attr_x(self, msp_project, msp_attr, pcs_
↪project_attrs, pcs_attr,
                              pcs_obj_or_cls):
```

PROJECT_DEFAULTS

Dictionary with a mapping between PCS project attributes (as keys) and default values. The default value is only used when no value gets transferred via the PROJECT_MAPPING attribute.

PROJECT_ATTR_ORDER

The order in which the project attributes are being displayed in the import preview dialog.

TASK_MAPPING

Dictionary with a mapping between MSP task fields (as keys) and PCS task attributes (as values). Values can either be single strings, thus the targeted attribute names, or tuples containing a callback class method name and then the targeted attribute names. Callback method signature:

```
def msp_field_x_to_pcs_attr_x(self, msp_task, msp_attr, pcs_task_
↪attrs, pcs_attr,
                              pcs_obj_or_cls):
```

TASK_DEFAULTS

Dictionary with a mapping between PCS task attributes (as keys) and default values. The default value is only used when no value gets transferred via the TASK_MAPPING attribute.

TASK_ATTR_ORDER

The order in which the task attributes are being displayed in the import preview dialog.

TASK_REFERENCE_MAPPING
Dictionary with a mapping between MSP task fields (as keys) and PCS task reference objects. The dictionary values are callback class method names. Callback method signature:

```
def msp_field_x_to_pcs_object_x(self, msp_task, msp_attr, pcs_task):
```

3.2 Exporting XML

Export PCS projects via MSP's XML schema.

3.2.1 XmlExport

class cs.pcs.msp.exports.**XmlExport**
Following class members are exposed for customization purposes:

PROJECT_MAPPING
Dictionary with a mapping between PCS project attributes (as keys) and MSP project fields (as values). Values can either be single strings, thus the targeted field names, or tuples containing a callback class method name and then the targeted field names. Callback method signature:

```
def pcs_attr_x_to_msp_field_x(self, pcs_project, pcs_attr, msp_
↪project, msp_attr):
```

PROJECT_DEFAULTS
Dictionary with a mapping between MSP project fields (as keys) and default values. The default value is only used when no value gets transferred via the PROJECT_MAPPING attribute.

TASK_MAPPING
Dictionary with a mapping between PCS task attributes (as keys) and MSP task fields (as values). Values can either be single strings, thus the targeted field names, or tuples containing a callback class method name and then the targeted field names. Callback method signature:

```
def pcs_attr_x_to_msp_field_x(self, pcs_task, pcs_attr, msp_task,␣
↪msp_attr):
```

TASK_DEFAULTS
Dictionary with a mapping between MSP task fields (as keys) and default values. The default value is only used when no value gets transferred via the TASK_MAPPING attribute.

TASK_REFERENCE_MAPPING
Dictionary with a mapping between callback class method names (as keys) and MSP task field names (as values). The task field name can optionally be followed by a colon and a custom MSP column name. Callback method signature:

```
def pcs_object_x_to_msp_field_x(self, pcs_task, msp_task, msp_attr):
```

3.3 Example

```
from cs.pcs.msp.exports import XmlExport
from cs.pcs.msp.imports import XmlMergeImport
from cs.pcs.projects import Project

class MyXmlMergeImport(XmlMergeImport):
    TASK_MAPPING = XmlMergeImport.TASK_MAPPING
    # Now add an import mapping between MSP task field
    # "Text7" and PCS task field "cdbpcs_task.category"
```

```
    TASK_MAPPING["Text7"] = "category"

class MyXmlExport(XmlExport):
    TASK_MAPPING = XmlExport.TASK_MAPPING
    # Now add an export mapping between PCS task field
    # "cdbpcs_task.category" and MSP task field "Text7"
    TASK_MAPPING["category"] = "Text7"

class MyProject(Project):
    # Attach both XML classes to the Project class
    # by overwriting the standard XML classes
    XML_IMPORT_CLASS = MyXmlMergeImport
    XML_EXPORT_CLASS = MyXmlExport
```

p

cs.pcs.msp, 3
cs.pcs.msp.exports, 4
cs.pcs.msp.imports, 3
cs.pcs.projects, 1

C

cs.pcs.msp (module), 3
cs.pcs.msp.exports (module), 4
cs.pcs.msp.imports (module), 3
cs.pcs.projects (module), 1

D

default_roles (cs.pcs.projects.Project attribute), 1
DefaultCalendarProfileName (cs.pcs.projects.Project attribute), 1

P

Project (class in cs.pcs.projects), 1
PROJECT_ATTR_ORDER (cs.pcs.msp.imports.XmlMergeImport attribute), 3
PROJECT_DEFAULTS (cs.pcs.msp.exports.XmlExport attribute), 4
PROJECT_DEFAULTS (cs.pcs.msp.imports.XmlMergeImport attribute), 3
PROJECT_MAPPING (cs.pcs.msp.exports.XmlExport attribute), 4
PROJECT_MAPPING (cs.pcs.msp.imports.XmlMergeImport attribute), 3

T

TASK_ATTR_ORDER (cs.pcs.msp.imports.XmlMergeImport attribute), 3
TASK_DEFAULTS (cs.pcs.msp.exports.XmlExport attribute), 4
TASK_DEFAULTS (cs.pcs.msp.imports.XmlMergeImport attribute), 3
TASK_MAPPING (cs.pcs.msp.exports.XmlExport attribute), 4
TASK_MAPPING (cs.pcs.msp.imports.XmlMergeImport attribute), 3
TASK_REFERENCE_MAPPING (cs.pcs.msp.exports.XmlExport attribute), 4
TASK_REFERENCE_MAPPING (cs.pcs.msp.imports.XmlMergeImport attribute), 3

X

XmlExport (class in cs.pcs.msp.exports), 4
XmlMergeImport (class in cs.pcs.msp.imports), 3