Pykechain

Pykechain is a python library developed by **KE-works** with the sole purpose to interact with the KE-chain data model, explorer, work breakdown and the scripts environment. Basically, an experienced user in pykechain, can extract any information stored in any project and use it to build scripts that would ultimately produce the desired outputs. Furthermore, he can also automatically create or extend the data model, add part instances, activities and even configure and customize said activities. Below, you will find some video examples of simple scripts ran in a **KE-chain** project , together with the code itself:

This script introduces the user to the one of the most important functionalities of **pykechain**: retrieving and storing values. Property values are accessed, used in computations and the results are then stored in other property values. Iteration can be performed again and again, based on new inputs.

Retrieve and store values

Your browser does not support the HTML5 video element

Using pykechain to perform computations based on property values

```
# Retrieve the project where this script is
ran
project = get_project()

# Retrieve the wheel part model
wheel_model = project.model(name='Wheel')

# Retrieve all the part instances created
based on the wheel model
wheel_parts =
project.parts(model=wheel_model)

# Loop through the list of part instances
for wheel_part in wheel_parts:
    # Retrieve the value of the 'Diameter'
property belonging to the wheel part instance
    wheel_diameter =
wheel_part.property(name='Diameter').value

    # Calculate the circumference based on
the diameter (C = pi*d), rounded to 2
decimals
    circumference = round(wheel_diameter *
math.pi, 2)

    # Store the circumference in the value of
the 'Circumference' property

wheel_part.property(name='Circumference').val
ue = circumference
```

In this script, you can get an idea how new models and new properties can be automatically

Related articles

- KECpkg
- How can I use property validators?
- How can I see when a script has last been executed?
- Concept: Service

created.

Adding model and properties

Your browser does not support the HTML5 video element

Using pykechain to extend the data model

```
# Make the needed imports from pykechain
from pykechain import get_project
from pykechain.enums import PropertyType


# Retrieve the project where this script is
ran
project = get_project()

# Retrieve the bike part model
bike_model = project.model(name='Bicycle')

# Create a new 'Exactly 1' model under
'Bicycle' and call it 'Saddle'
saddle_model =
bike_model.add_model(name='Saddle',
multiplicity='ONE')

# Add some properties to it
saddle_model.add_property(name='Material',
property_type=PropertyType.CHAR_VALUE,
default_value='Nylon')
saddle_model.add_property(name='Saddle tilt',
property_type=PropertyType.FLOAT_VALUE,
                          description='Angle
of saddle compared to ground (degrees)')
```

The third script presents some actions that can be performed on the work breakdown.

Adding a process and activity

Your browser does not support the HTML5 video element

Using pykechain to extend the work breakdown

```
# Make the needed imports from pykechain
from pykechain import get_project
from pykechain.enums import ActivityType


# Retrieve the project where this script is
ran
project = get_project()

# Retrieve the root process of the work
breakdown
workflow_root =
project.activity(name='WORKFLOW_ROOT')

# Create a new process called 'Design saddle'
design_saddle =
workflow_root.create(name='Design saddle',
activity_type=ActivityType.PROCESS)

# Create a new task under it called 'Define
saddle properties'
design_saddle.create(name='Define saddle
properties', activity_type=ActivityType.TASK)
```

Finally, this script will show how easy it is to add widgets to an activity.

Adding a widget to the activity

Your browser does not support the HTML5 video element

Using pykechain to add a widget to an activity

```
# Retrieve the project where this script is
ran
project = get_project()

# Retrieve the 'Define saddle properties'
activity in a variable
define_saddle_properties =
project.activity(name='Define saddle
properties')

# Retrieve the 'Saddle' part model in a
variable
saddle_model = project.model(name='Saddle')

# Retrieve the two properties of 'Saddle'
saddle_material =
saddle_model.property(name='Material')
saddle_tilt =
saddle_model.property(name='Saddle tilt')

# Add a Form widget to the activity with the
two properties as writable inputs
widget_manager =
define_saddle_properties.widgets()
widget_manager.add_propertygrid_widget(part_i
nstance=saddle_model.instance(),

custom_title='Saddle overview',

readable_models=[],

writable_models=[saddle_material,
saddle_tilt])
```

This article is not meant to be a **Pykechain** tutorial, but rather an introduction to the potential this package has in respect to KE-chain. If you desire to follow a Pykechain tutorial, please contact the Service Desk.