Continuity SDK

Установка и подключение

Необходимо поставить в окружение разработки пакет Continuity SDK и настроить подключение:

```
from klmg_continuity_sdk import Session

session = Session(
    host='https://continuity-demo.k8s.datasapience.ru/api',
    username='rost',
    password='rost'
)
session.health()
```

Создание шаблона

```
from datetime import datetime
from klmg_continuity_sdk.schemas import TEMPLATE

template = TEMPLATE(
    ENTITY_TYPE='template_type',
    DIM_ENTITY_KEY='template_key',
    DIM_ENTITY_TAG=datetime.now().strftime('%a, %d %b %Y %H:%M:%S'),
    DIM_ENTITY_NAME='Template name'
)

session.create(template)
```

Также можно добавить следующие параметры: 1. DIM_ENTITY_DESC (описание шаблона)
2. IS_MAIN_VERSION (true/false)
3. NODES (список узлов в JSON представлении)
4. EDGES (список связей между узлами в JSON редставлении)
5. CONTEXT (дополнительные настройки в JSON представлении)

Пример JSON
Дополнительные настройки

Получение всех шаблонов системы

```
all = session.template.all(entity_type='project')
all
```

Возможные фильтры: 1. entity_type ('project', 'task', 'artefact') 2. owner (user_name) 3. status ('all', 'active', 'completed', 'archived')

Получение количества шаблонов

```
count = session.template.count(entity_type='project')
count
```

Возможные фильтры: 1. entity_type ('project', 'task', 'artefact') 2. owner (user_name) 3. status ('all', 'active', 'completed', 'archived')

Получение шаблона по ключу и версии

```
one = session.template.one(
    key=template.DIM_ENTITY_KEY,
    tag=template.DIM_ENTITY_TAG
)
one
```

Получение всех объектов, созданных по определенному шаблону

```
successors = session.template.successors(
    key=template.DIM_ENTITY_KEY,
    tag=template.DIM_ENTITY_TAG,
    entity_type=template.ENTITY_TYPE
)
successors
```

Создание RUNTIME объекта по шаблону

```
from datetime import datetime
from klmg_continuity_sdk.schemas import RUNTIME

runtime = RUNTIME(
    DIM_ENTITY_ID=template.DIM_ENTITY_ID,
    ENTITY_TYPE='entity_type',
    FCT_ENTITY_KEY='entity_key',
    FCT_ENTITY_TAG=datetime.now().strftime('%a, %d %b %Y %H:%M:%S'),
    FCT_ENTITY_NAME='Entity name'
)
```

```
session.create(runtime)
runtime
```

Получение количества RUNTIME объектов

```
count = session.runtime.count(entity_type=runtime.ENTITY_TYPE)
count
```

Получение всех RUNTIME объектов

```
all = session.runtime.all(entity_type=runtime.ENTITY_TYPE, limit=1)
all
```

Получение RUNTIME объекта по ключу и версии

```
one = session.runtime.one(
    key=runtime.FCT_ENTITY_KEY,
    tag=runtime.FCT_ENTITY_TAG
)
one
```

Получение всех дочерних объектов RUNTIME объекта

```
children = session.runtime.children(
    key=runtime.FCT_ENTITY_KEY,
    tag=runtime.FCT_ENTITY_TAG
)
children
```

Получение всех вышестоящих родителей RUNTIME объекта

```
parents = session.runtime.parents(
    key=runtime.FCT_ENTITY_KEY,
    tag=runtime.FCT_ENTITY_TAG
)
parents
```

Сохранение артефакта

```
artefact_template_key.ENTITY.FCT_ENTITY_DESC = "artefact_value"
artefact_template_key.ENTITY.IS_COMPLETE = True

session.update(task, is_complete=True)
```

Перевод сущности в статус done

```
session.update(task, is_complete=True)
```

Пример настройки handler для сервисной задачи

```
@manager.subscribe(
    key=settings.TEMPLATE_ENTITY_KEY,
    tag=settings.TEMPLATE_ENTITY_TAG,
    entity_type='entity_type',
    interval=60
)
def function_name(entity: RUNTIME):

    artefact: RUNTIME_NODE = None

    for node in entity.NODES:
        if node.ALIAS == 'artefact_template_key':
            artefact = node

    if not artefact:
        return

    # Take all artefact successors keys of project
    args = session.runtime.inputs(
        key=entity.FCT_ENTITY_KEY,
        tag=entity.FCT_ENTITY_TAG
    )

    file_artefact = args['some_file_artefact']

    # File upload
    session.file.upload(
        key=ab_beta_matrix.ENTITY.FCT_ENTITY_KEY,
        tag=ab_beta_matrix.ENTITY.FCT_ENTITY_TAG,
        data=csv,
        filename="file_name.csv"
    )

    # Save artefact value into Continuity
    artefact.ENTITY.FCT_ENTITY_DESC = "file_name.csv"
    artefact.ENTITY.IS_COMPLETE = True

    # Complete task
    session.update(entity, is_complete=True)
```