Xenops Documentation

Release 0.0.1

Maikel Martens

Nov 18, 2017

Contents:

CHAPTER 1

Introduction

Xenops is a simple program to sync data like (customers/products) between different systems.

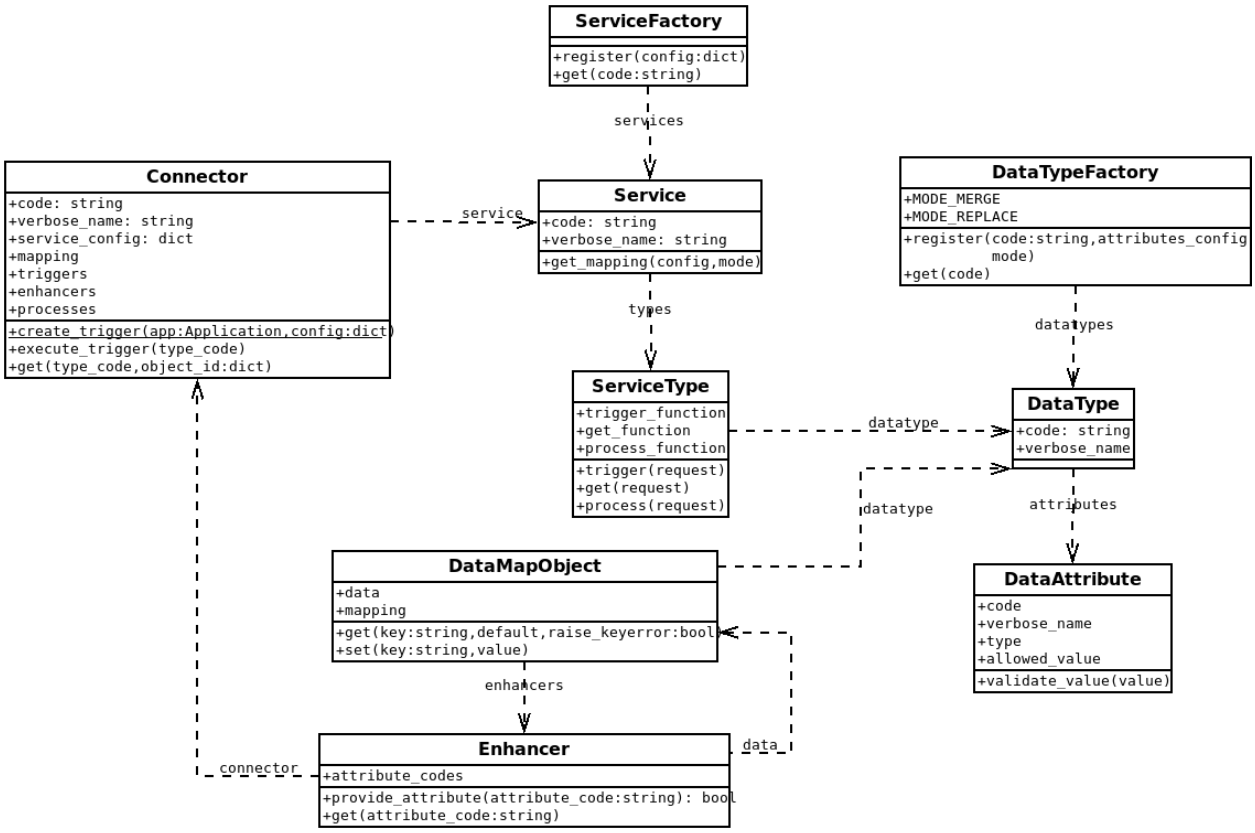
Go to setup project for creating a new project.

Go to setup service for creating a new service for xenops.

1.1 System flow

1. A trigger is run manually or by cron.
2. Service yield new or updated data.
3. Data is wrapped around data mapper object, that can get values based on given mapping.
4. Data mapper object is given to a process service that is registered.

1.2 Core Class diagram



CHAPTER 2

Project

Project files setup:

2.1 settings.py

```
from xenops.data import converter

CONNECTORS = {
    'pim_live': {
        'service': 'pim',  # Code of service
        'name': 'Live pim',  # optional verbose name
        'config': {  # Config for service like SOAP url and login
            'url': '',
            'username': '',
            'password': '',
        },
        'mapping': {  # Mapping per DataType code
            'product': {
                'type': 'merge', # merge or replace, Default merge
                'attributes': [
                    converter.Attribute('sku', 'ean'),
                ]
            }
        },
        'triggers': [  # Triggers you want to run
            {
                'trigger_code': 'update_product',  # optional, use type code as␣
↪default
                'type': 'product',  # DataType for which the trigger is run.
                'cron': '1 * * * *',  # Cron for when to run the trigger (Not␣
↪supported now).
            }
        ],
```

```
        'enhancers': [
            {
                'type': 'product',
                'attributes': ['price', 'qty'], # default will enhance type with all␣
↪attributes from mapping
            }
        ],
        'processes': [
            {
                'type': 'product',
                'attributes': ['price', 'qty'], # only run this process when given␣
↪attributes are changed (Not support now)
            },
        ]
    },
}

TYPES = {
    'product': {
        'mode': 'merge', # merge or replace, default merge
        'attributes': {
            'price': {
                'allowed_value': r'\d+\.\d{2}'  # Regex for checking value.
            }
        }
    }
}
```

2.2 Command line

```
import sys
import os

os.environ.setdefault("XENOPS_SETTINGS", "settings")

from xenops import execute_from_command_line

execute_from_command_line(sys.argv)
```

CHAPTER 3

Service

Dummy service files:

Note: TODO: Make it possible to register service in project settings and rewrite existing services.

3.1 pim.py

```
from xenops.data import converter


def trigger(request):
    # Based on request yield back data from source like (CSV, SOAP, REST, etc)
    yield {
        'ean': 'ean-123',
        'name': 'Test'
    }

    yield {
        'ean': 'ean-456',
        'name': 'Pim Product'
    }


def get(request):
    # Based on request return data from source like (CSV, SOAP, REST, etc)
    return {
        'price': 14.5
    }


def process(request):
    # process data and export to (CSV, SOAP, REST, etc) and return id
```

```
    return 827

# Register service config (is used by setup.py entry_points)
register = {
    'code': 'pim',
    'verbose_name': 'Pim',
    'type': {
        'product': {
            'mapping': [
                converter.Attribute('sku', 'sku'),
                converter.Attribute('price', 'price'),
            ],
            'trigger': trigger,
            'get': get,
            'process': process,
        }
    }
}
```

```
from setuptools import setup, find_packages

setup(
    name='xenops_service_pim',
    version="1.0",
    description="Xenops Pim service",
    author="Maikel Martens",
    packages=find_packages(),
    include_package_data=True,
    entry_points={
        'xenops.services': [
            'pim = pim:register'
        ]
    }
)
```

CHAPTER 4

API

4.1 Converters

class xenops.data.converter.**BaseConverter**(*attribute*, *service_attribute*)

Base converter

Service attribute can lookup and export nested data like:

```
# Service raw data
{
    'stock': {
        'level': 10
    }
}

# Mapping
service_attribute = 'stock.level'
```

export_attribute(*data_object*)

Convert DataType data to service data

Parameters **data_object** (*xenops.data.DataType*) –

Returns

get_import_value(*keys*, *data*)

Recursive function for getting data from service data dict

Parameters

- **keys** (*str*) –
- **data** (*dict*) –

Returns

import_attribute(*data*)

Convert raw service data to DataType data

Parameters **data** (*dict*) –

Returns

class xenops.data.converter.**Attribute**(*attribute*, *service_attribute*)

Bases: *xenops.data.converter.BaseConverter*

Default attribute converter

class xenops.data.converter.**Mapper**(*attribute*, *service_attribute*, *mapping*, *use_default=False*, *import_default=None*, *export_default=None*)

Bases: *xenops.data.converter.BaseConverter*

Mapper attribute converter can be used to convert values to same base value example for gender:

```
# Mapping for some erp service
{
    'm': 1,
    'f': 2,
}

# Mapping for some e-commerce service
{
    'm': 'Male',
    'f': 'Female',
}
```

export_attribute(*data_object*)

Get data from super and map value with mapping

Parameters **data_object** –

Returns

import_attribute(*data*)

Get data from super and map value with mapping

Parameters **data** (*dict*) –

Returns

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Index