OpenStack基础公共库 - Olso

EasyStack 郭长波

2016.07.15

大纲

- **Olso介绍**
- Olso关键组件分析
- Newton版本工作

Olso介绍

- 官方名称

OpenStack Common Libraries

- 项目使命

To produce a set of python libraries containing code shared by OpenStack projects. The APIs provided by these libraries should be high quality , stable, consistent, documented and generally applicable.

- 项目历史

2011-07 olso-incubator	2013-04 olso messaging
2012-01 olso.config	....
2012 -12 olso.db	2016-06 Farewell olso-incubator

Olso介绍

- Olso team
  - Generallist Code Reviewers
    对Python熟悉，提供建设性输入，Review所有Olso项目
  - Specialist API Maintainers
    应对Olso库过多，每个库有一个或多个专员维护者
    单个项目的core reviewer

- Project Liaisons
  - 人选
    每个下游项目一个协调人，在项目中活跃，熟悉项目特有的需求，
    不必是Core reviewer或者PTL
  - 职责
    帮助对应项目使用olso patch，参与讨论API改变关注
    [Olso] 标签的邮件， 参加Olso meeting

Olso介绍

1 automaton
2 cliff
3 debtcollector
4 futurist
5 openstack-cookiecutter
6 osprofiler
7 oslo.cache
8 oslo.concurrency
9 oslo.context
10 oslo.config
11 oslo-cookiecutter
12 oslo.db
13 oslo.i18n
14 oslo.log
15 oslo.messaging
16 oslo.middleware
17 oslo.policy
18 oslo.privsep
19 oslo.reports
20 oslo.rootwrap
21 oslo.serialization
22 oslo.service
23 oslosphinx
24 oslotest
25 oslo.utils
26 oslo.versionedobjects
27 oslo.version
28 oslo.vmware
29 pylockfile
30 hacking
31 pbr
32 pyCADF
33 stevedore
34 taskflow
35 tooz

Olso介绍

- Oslo库发布过程

  - oslo-incubator时代

  copy代码到各个项目openstack/common目录

  - oslo.* 时代

  每周由PTL提交commit 到releases代码库

  Requirements更新库最低版本号

  下游项目requirements更新，使用新功能

- 参与Oslo开发

  - IRC

  Chanel: #openstack-oslo

  - Weekly Meeting

  Time: Monday 1600 UTC

  Chanel: #openstack-meeting-alt

大纲

- Olso介绍
- **Olso关键组件分析**
- Newton版本工作

Olso.config

- 用途

分析命令行或文件里的配置选项

- 常用类型

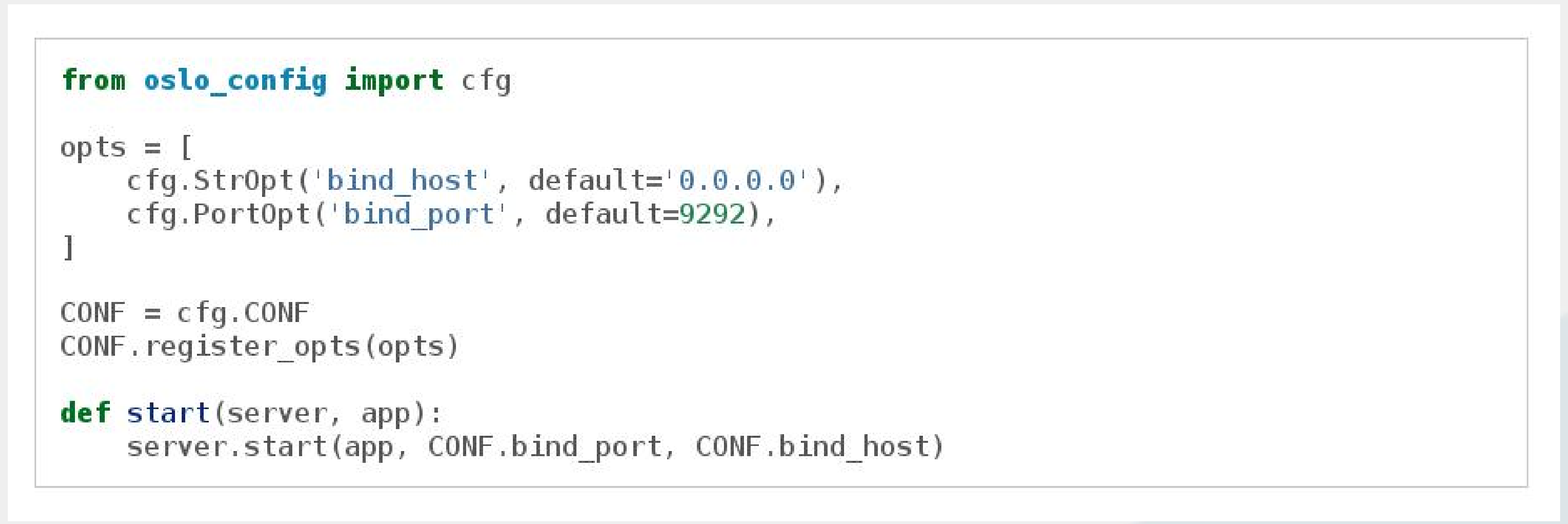
 - StrOpt , BoolOpt , IntOpt , PortOpt ,

ListOpt , DictOpt , IPOpt , HostnameOpt

- 注意事项

 - 引用已有的配置项值：$name${group.name}

 - 约束条件：choices , required , secret , mutable



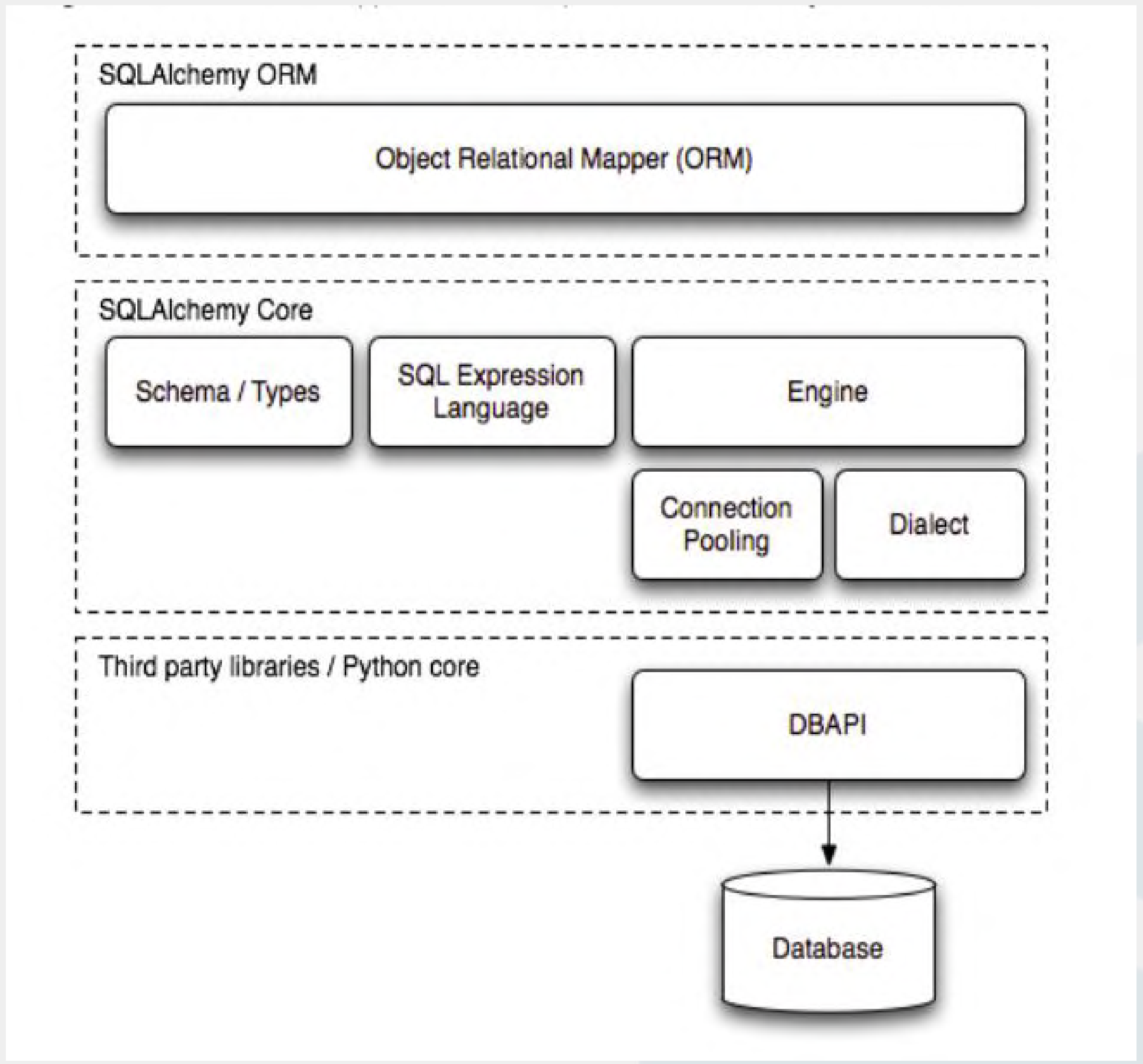
```
from oslo_config import cfg

opts = [
    cfg.StrOpt('bind_host', default='0.0.0.0'),
    cfg.PortOpt('bind_port', default=9292),
]

CONF = cfg.CONF
CONF.register_opts(opts)

def start(server, app):
    server.start(app, CONF.bind_port, CONF.bind_host)
```

Olso.db(1/2)

- 用途
  访问关系型数据库接口
  提供表结构创建，访问记录
  易用接口

- 特点
  支持多种数据库：
  MySQL,PostgreSQL,DB2
  不支持MongoDB
  依赖底层 SqlAchemy

- PyMySQL vs MySQL-python



Olso.db(2/2)

使用示例

```
_BACKEND_MAPPING = {'sqlalchemy': 'nova.db.sqlalchemy.api'}

IMPL = concurrency.TpoolDbapiWrapper(CONF, backend_mapping=_BACKEND_MAPPING)
```

```
def service_get_all(context, disabled=None):
    """Get all services."""
    return IMPL.service_get_all(context, disabled)
```

```
@pick_context_manager_reader
def service_get_all(context, disabled=None):
    query = model_query(context, models.Service)

    if disabled is not None:
        query = query.filter_by(disabled=disabled)

    return query.all()
```

```
class Service(BASE, NovaBase, models.SoftDeleteMixin):
    """Represents a running service on a host."""

    __tablename__ = 'services'
    __table_args__ = (
        schema.UniqueConstraint("host", "topic", "deleted",
                                name="uniq_services0host0topic0deleted"),
        schema.UniqueConstraint("host", "binary", "deleted",
                                name="uniq_services0host0binary0deleted")
        )

    id = Column(Integer, primary_key=True)
    host = Column(String(255))  # , ForeignKey('hosts.id'))
    binary = Column(String(255))
    topic = Column(String(255))
    report_count = Column(Integer, nullable=False, default=0)
    disabled = Column(Boolean, default=False)
    disabled_reason = Column(String(255))
    last_seen_up = Column(DateTime, nullable=True)
    forced_down = Column(Boolean, default=False)
    version = Column(Integer, default=0)

    instance = orm.relationship(
        "Instance",
        backref='services',
        primaryjoin='and_(Service.host == Instance.host,'
                    'Service.binary == "nova-compute",'
                    'Instance.deleted == 0)',
        foreign_keys=host,
    )
```

Olso.messaging(1/2)

- 用途

  为组件提供RPC和notification 功能

- 支持多种driver

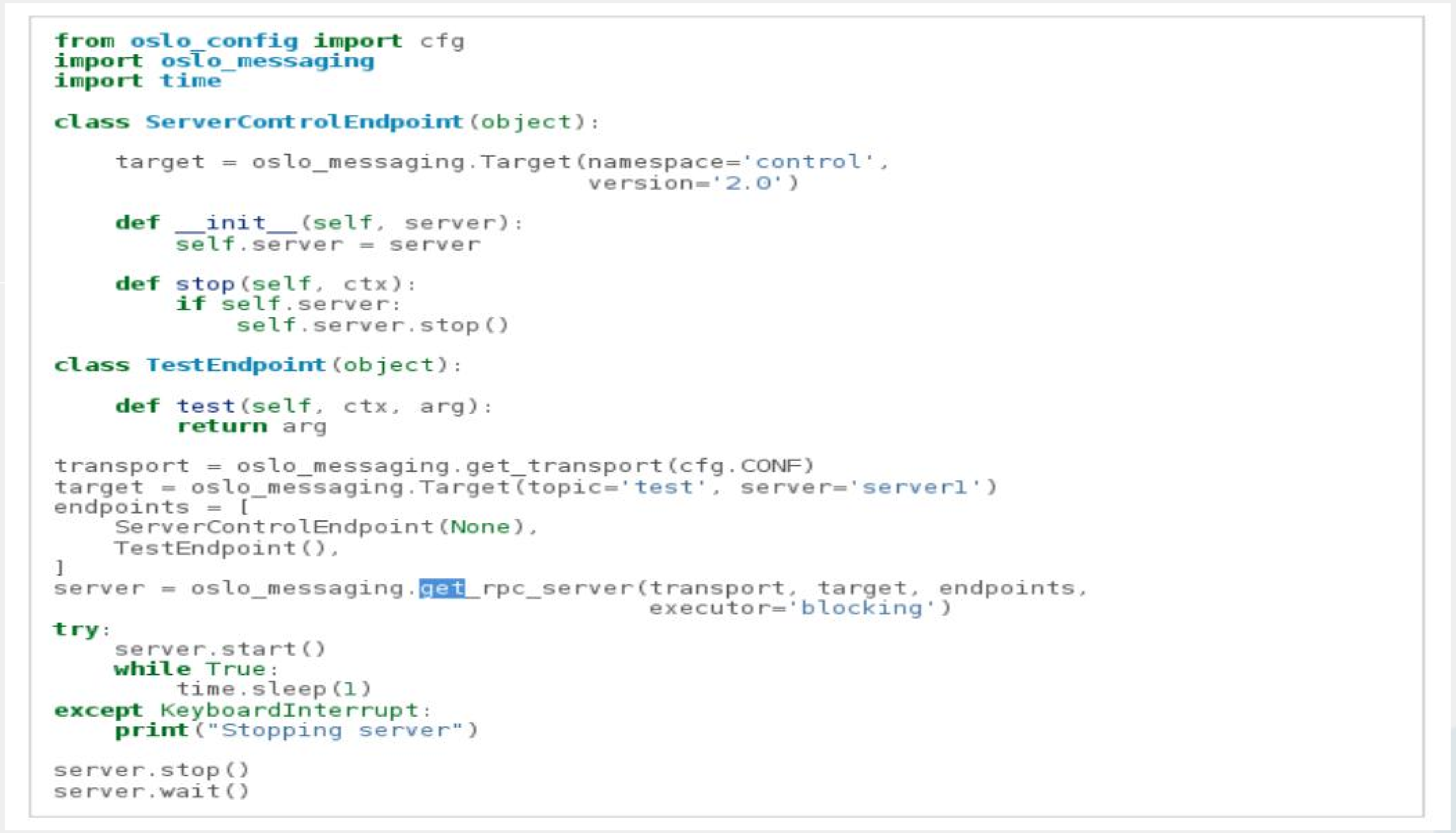
  amqp
  fake
  Kafka
  rabbit(kombu)
  zmq

- 概念

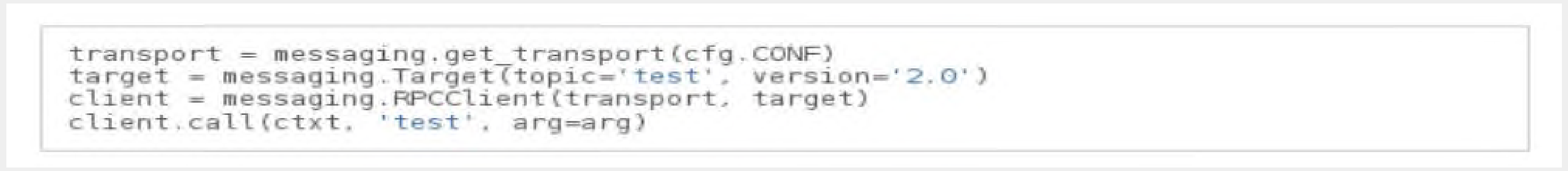
  transport
  executors
  target
  server
  RPC Client

Olso.messaging(2/2)

使用案例



```
from oslo_config import cfg
import oslo_messaging
import time

class ServerControlEndpoint(object):

    target = oslo_messaging.Target(namespace='control',
                                   version='2.0')

    def __init__(self, server):
        self.server = server

    def stop(self, ctx):
        if self.server:
            self.server.stop()

class TestEndpoint(object):

    def test(self, ctx, arg):
        return arg

transport = oslo_messaging.get_transport(cfg.CONF)
target = oslo_messaging.Target(topic='test', server='server1')
endpoints = [
    ServerControlEndpoint(None),
    TestEndpoint(),
]
server = oslo_messaging.get_rpc_server(transport, target, endpoints,
                                       executor='blocking')
try:
    server.start()
    while True:
        time.sleep(1)
except KeyboardInterrupt:
    print("Stopping server")

server.stop()
server.wait()
```



```
transport = messaging.get_transport(cfg.CONF)
target = messaging.Target(topic='test', version='2.0')
client = messaging.RPCClient(transport, target)
client.call(ctxt, 'test', arg=arg)
```

Stevedore (1/2)

- setuptools entry points

－ 支持动态加载可调用Python模块

－ 模块可来自不同package

－ 支持namespace 方式加载

- 应用场景

－Drivers

Single Name, Single Entry Point

－Hooks

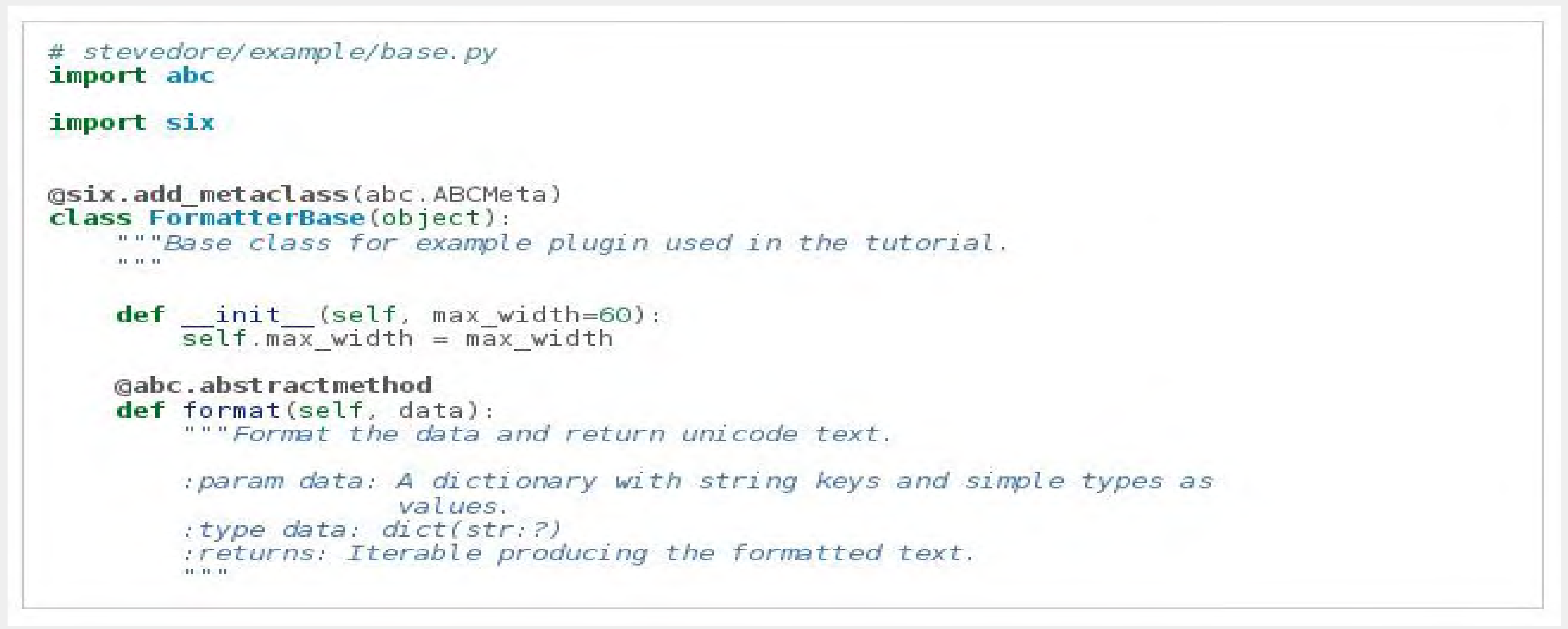
Single Name, Many Entry Points

－Extensions

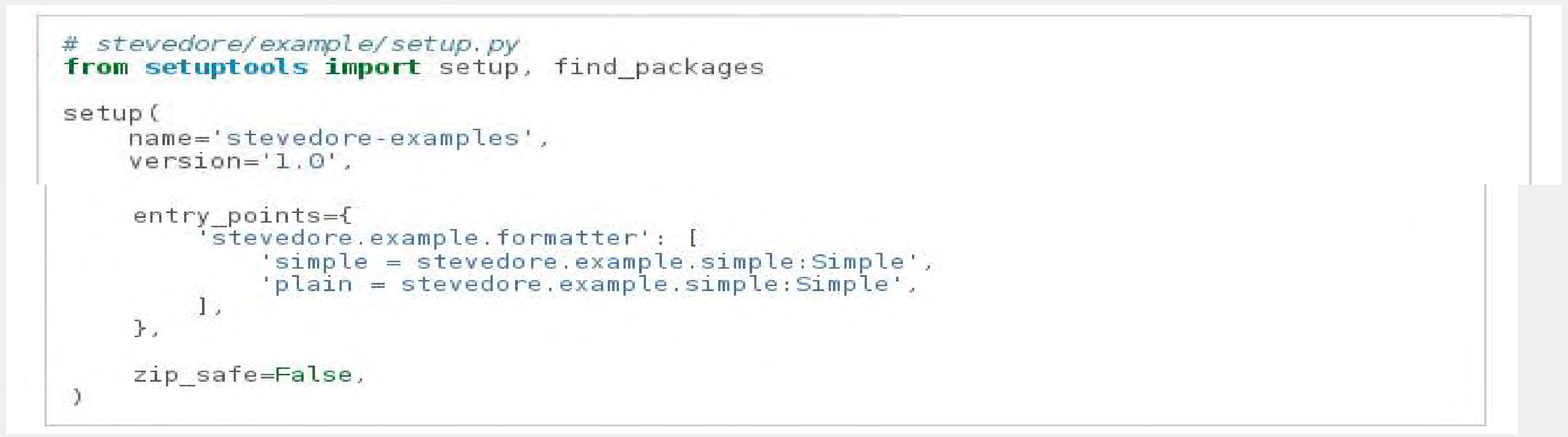
Many Names, Many Entry Points

Stevedore (2/2)

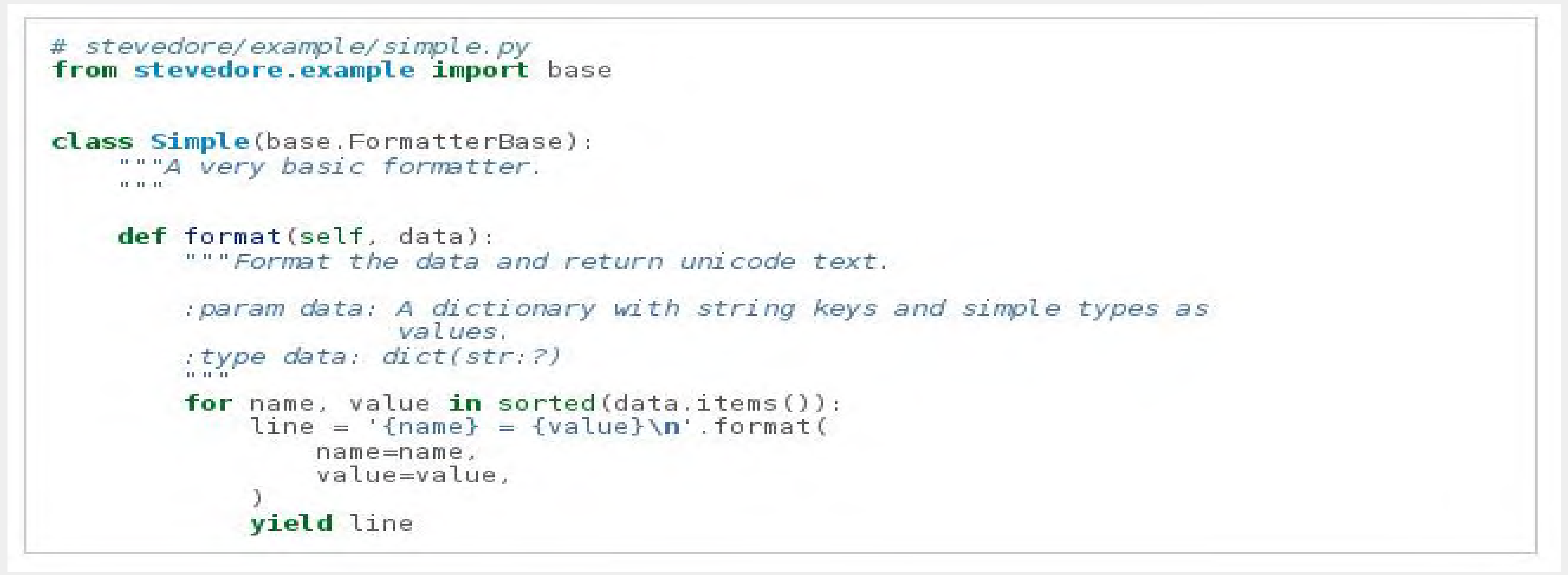
使用案例



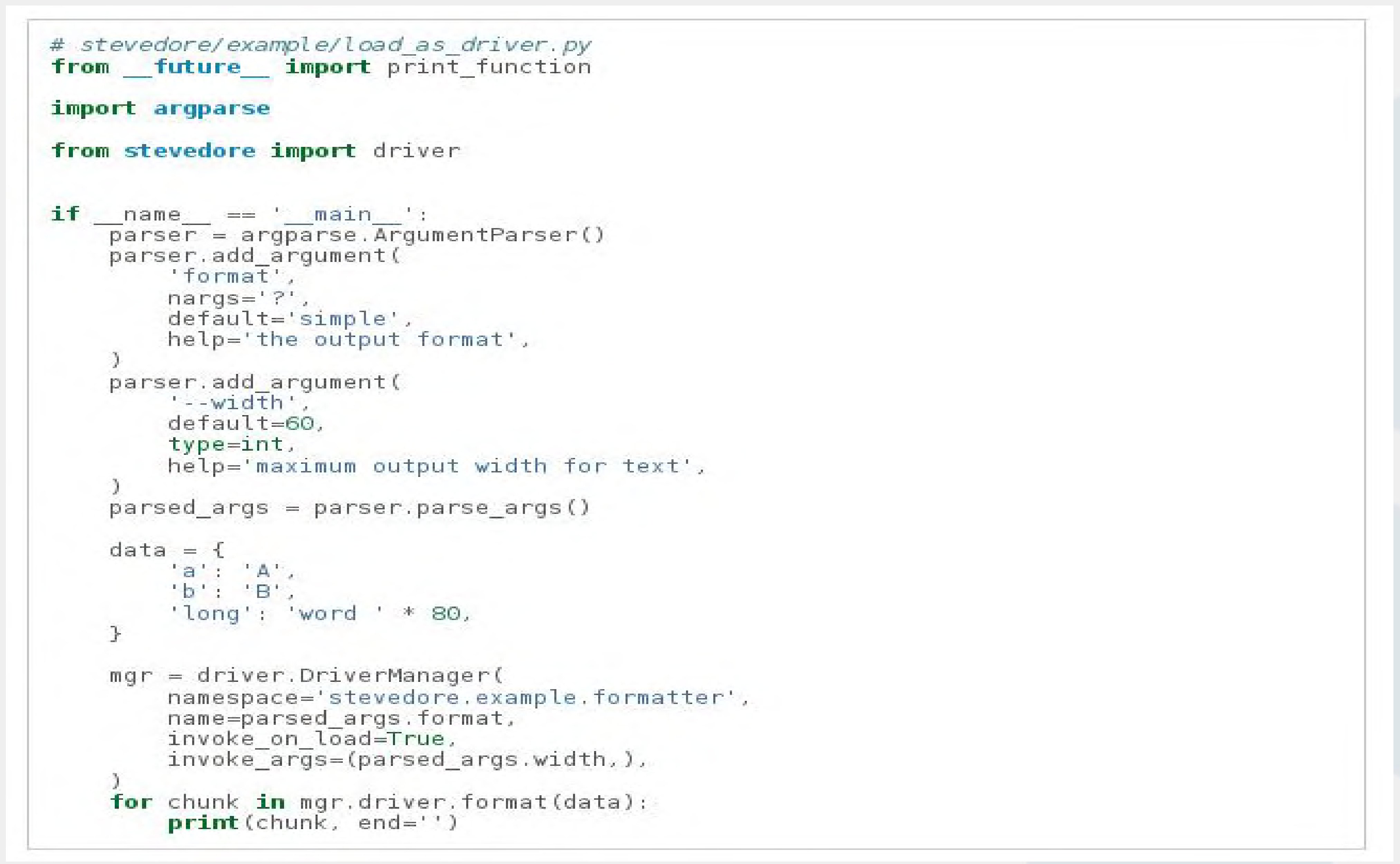
```python
# stevedore/example/base.py
import abc

import six


@six.add_metaclass(abc.ABCMeta)
class FormatterBase(object):
    """Base class for example plugin used in the tutorial.
    """

    def __init__(self, max_width=60):
        self.max_width = max_width

    @abc.abstractmethod
    def format(self, data):
        """Format the data and return unicode text.

        :param data: A dictionary with string keys and simple types as
                     values.
        :type data: dict(str:?)
        :returns: Iterable producing the formatted text.
        """
```



```python
# stevedore/example/setup.py
from setuptools import setup, find_packages

setup(
    name='stevedore-examples',
    version='1.0',

    entry_points={
        'stevedore.example.formatter': [
            'simple = stevedore.example.simple:Simple',
            'plain = stevedore.example.simple:Simple',
        ],
    },

    zip_safe=False,
)
```



```python
# stevedore/example/simple.py
from stevedore.example import base


class Simple(base.FormatterBase):
    """A very basic formatter.
    """

    def format(self, data):
        """Format the data and return unicode text.

        :param data: A dictionary with string keys and simple types as
                     values.
        :type data: dict(str:?)
        """
        for name, value in sorted(data.items()):
            line = '{name} = {value}\n'.format(
                name=name,
                value=value,
            )
            yield line
```



```python
# stevedore/example/load_as_driver.py
from __future__ import print_function

import argparse

from stevedore import driver


if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument(
        'format',
        nargs='?',
        default='simple',
        help='the output format',
    )
    parser.add_argument(
        '--width',
        default=60,
        type=int,
        help='maximum output width for text',
    )
    parsed_args = parser.parse_args()

    data = {
        'a': 'A',
        'b': 'B',
        'long': 'word ' * 80,
    }

    mgr = driver.DriverManager(
        namespace='stevedore.example.formatter',
        name=parsed_args.format,
        invoke_on_load=True,
        invoke_args=(parsed_args.width,),
    )
    for chunk in mgr.driver.format(data):
        print(chunk, end='')
```

大纲

- Olso介绍
- Olso关键组件分析
- Newton版本工作

Olso.policy改进

- 关于oslo.policy
  - 根据policy.json 规则检验API授权
  - 静态读取配置文件policy.json

- 现阶段问题
  - 部署者必须定义所有规则
  - 不能在代码里嵌入默认规则
  - 没有方法获得需要设置哪些规则

- 改进方式
  - 增加注册policy规则机制
  - 添加自动生成policy规则功能

- 参考：

https://review.openstack.org/#/c/309152/
https://review.openstack.org/#/c/309153/

Olso Adoption

- Oslo项目完善

— 完善文档
— 编写文章介绍oslo库
— 清理废弃的功能
—periodic jobs

- 下游项目

— 清除oslo-incubator
— 使用oslo 库
— 提供项目特有需求

Python3支持

- 为什么要支持Python 3

— Python 进化需要软件进化
— 发行版自带Python版本
— 更好的特性支持

- Python 版本支持

— 放弃支持2.6，支持2.7+
— 支持3.4,计划支持3.5

- 工作内容

— 改造外部依赖库，如eventlet
—OpenStack 项目 Python 3 支持
—IRC: #openstack-python3
— 参考https://wiki.openstack.org/wiki/

EasyStack
open cloud computing

THANKS

http://www.easystack.cn