Python项目工程化

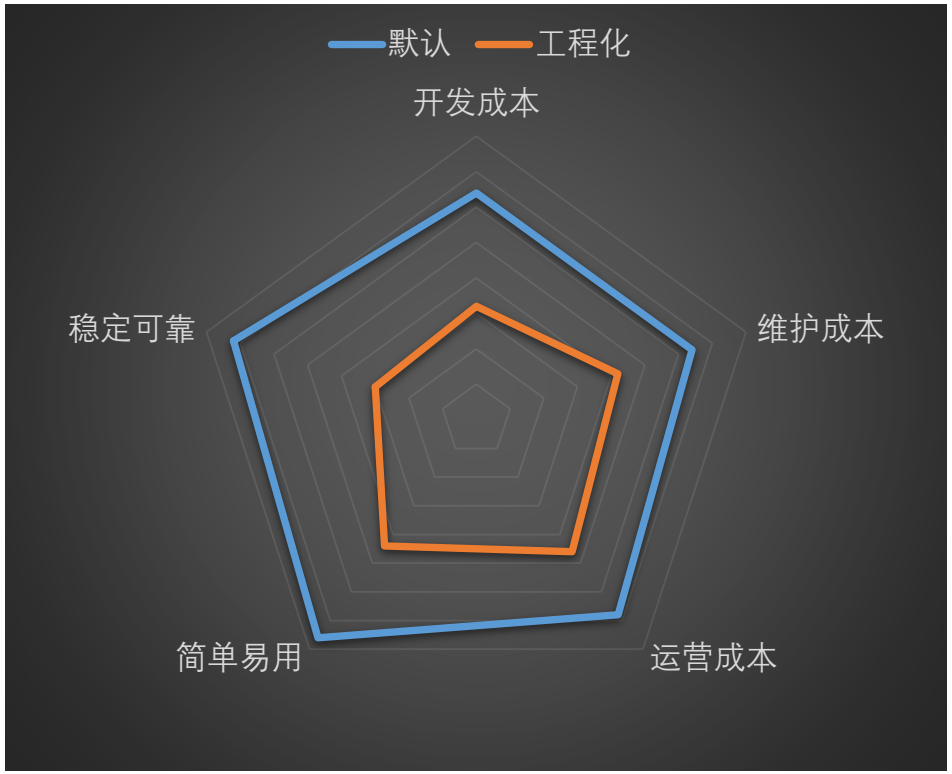
——携程信安是如何做的

2018/04 曾凡伟

agenda

- 工程目标
- 研发环境
- 依赖服务
- 公共组件
- 项目结构
- 日志告警
- 持续集成

工程目标



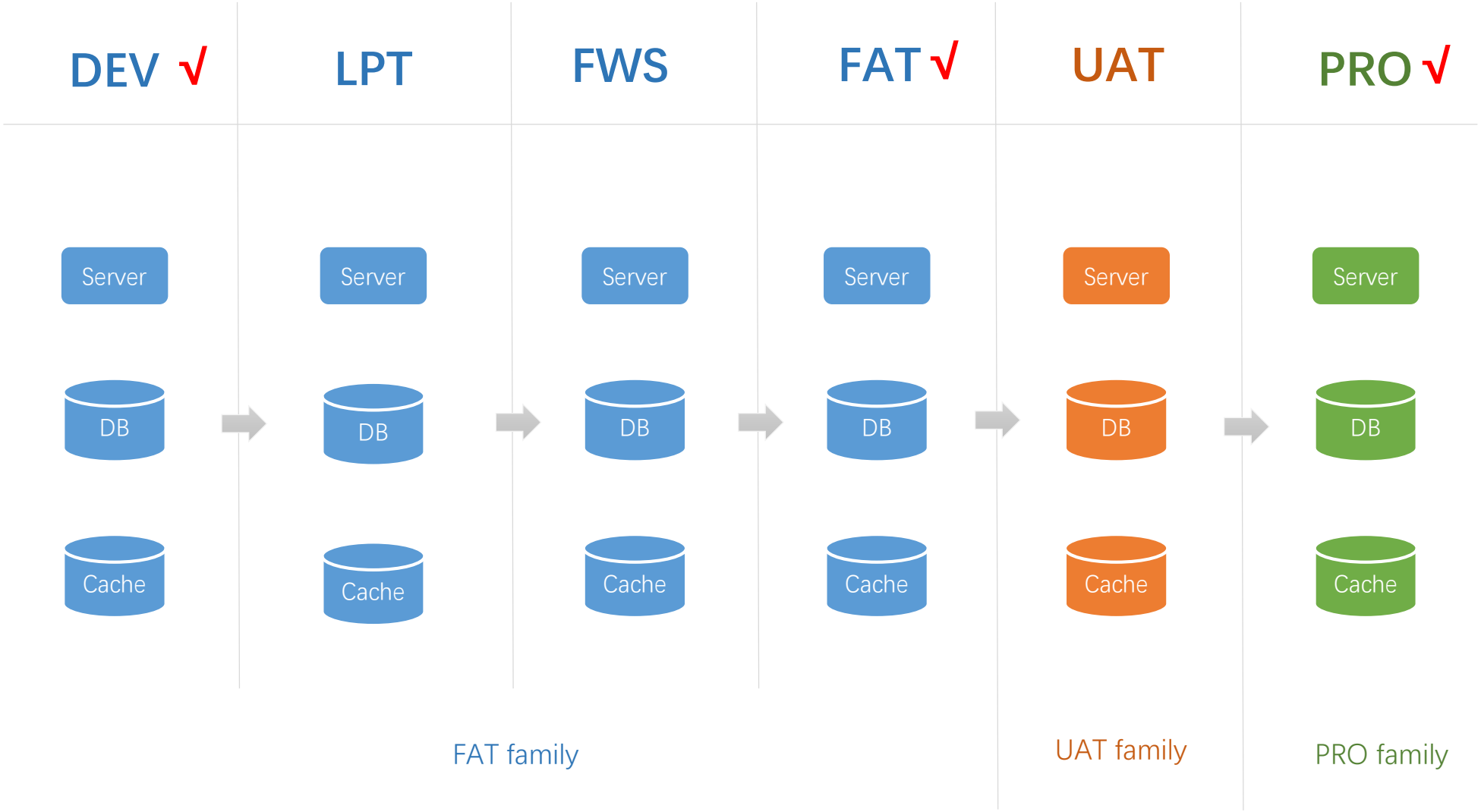
问题

开发思维不同，看待问题角度有别，提供的方案不一
项目混乱，各种八仙过海，成本大，质量没保障

方案

“车同轨，书同文，行同伦” ——《礼记·中庸》
顶层设计，统一方案，降低成本，保障质量

研发环境


DEV √
LPT
FWS
FAT √
UAT
PRO √
Server
DB
Cache
FAT family
UAT family
PRO family

依赖服务

- 公司提供

1. MySQL（DBA提供）
2. Redis（ DBA提供）

- 组内自研

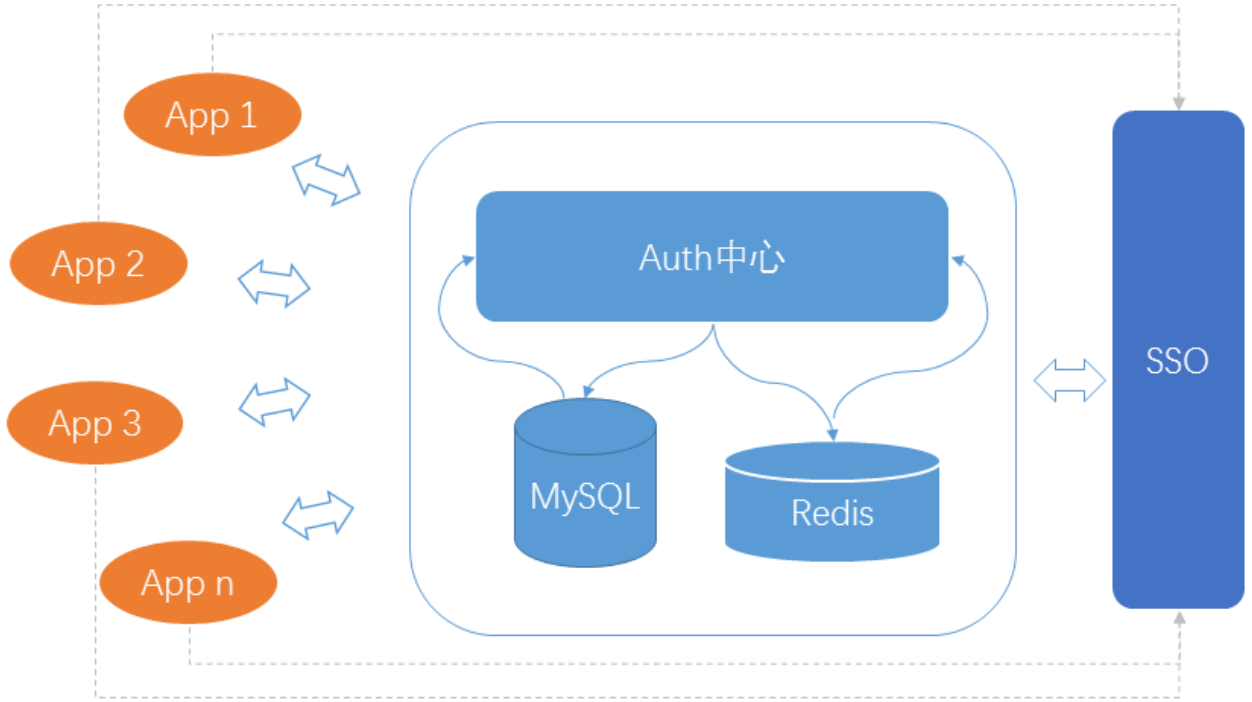
1. 用户认证中心 AuthCenter
2. 授权认证中心 UserCenter

依赖服务·公司提供

- 为什么使用DBA提供的DB？

1. 一站式的服务，直接使用，无需运维
2. 字段规范检查，强制最佳实践
3. 集中的慢查询监控、告警
4. 统一的数据备份
5. 以及其它各种优势

依赖服务·用户认证中心



注解：
灰色虚线箭头为历史授权方式
蓝色实线箭头为全新授权方式

背景

Portal Console项目中对接的应用，需共享登入登出状态

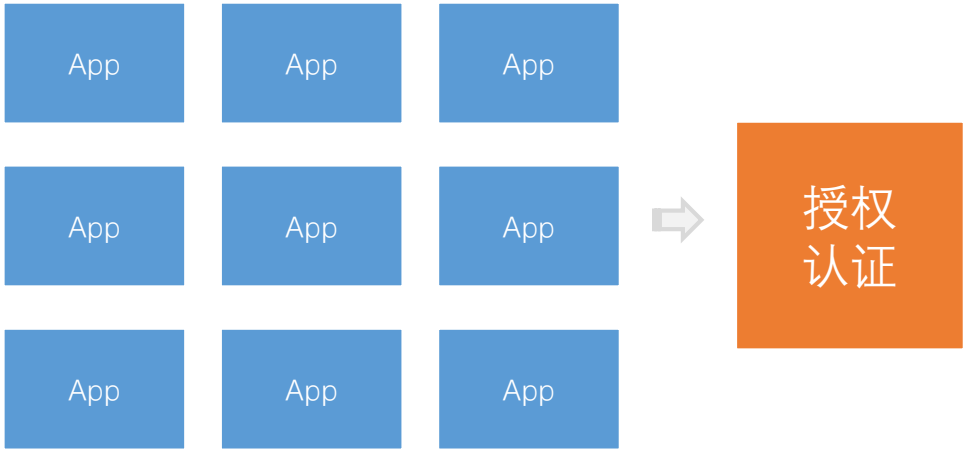
特性

1. 提供对SSO的透明代理
2. 一处登入，处处登入
3. 一处登出，处处登出

依赖服务·授权认证中心

各自为政

统一授权



公共组件

- titan_client

  获取数据库连接串

- framework_foundation

  获取环境信息

- auth_center_agent

  权限认证客户端

公共组件· titan_client

```
MYSQL = {
    "host":  “foo.bar.com",
    "port": 28747,
    "user": "m_user",
    "password":  “xxxx-xxxx",
    "db": "bazDb"
}
```

旧方案

联系DBA索要各环境连接串
硬编码到代码中

```
from titan_client import Titan
MYSQL = Titan.decode( “bazDb_W")
```

√ 更安全、更便捷

新方案

两行代码获取DB连接串

公共组件· framework_foundation

旧方案

使用环境变量

代码没有抽象成package，拷贝复用

```
class Mode(Enum):
    DEV = "DEV"
    FAT = "FAT"
    LPT = "LPT"
    UAT = "UAT"
    PRO = "PRO “


    @staticmethod
    def get(key):
        mode = os.environ[key]
        if mode not in Mode.values():
            raise InvalidModeException("found invalid env value: %s" % mode)
        warnings.warn("Got env variable successfully, %s = %s" % (key, mode))
        return mode

mode = Mode.get("SkyScannerMode")
```

新方案

遵从公司Java系规范

两行代码获取环境配置

```
from framework_foundation import Foundation
env = Foundation.server().get_env()
```

√ 简单、好维护

公共组件·auth_center_agent

```python
def get(self):
    auth = self.check_user('result')
    if auth is True:
        # 授权通过业务逻辑
        pass
    else:
        # 授权失败业务逻辑
        self.render('noauth.html', auth=auth)
```

旧方案

授权代码侵入业务代码

可读性差

```python
from auth_center_agent import authenticated

@acquire.privilege('result')
def get(self):
    # 授权通过业务逻辑
    pass
```

√ 简单直观、好维护

新方案

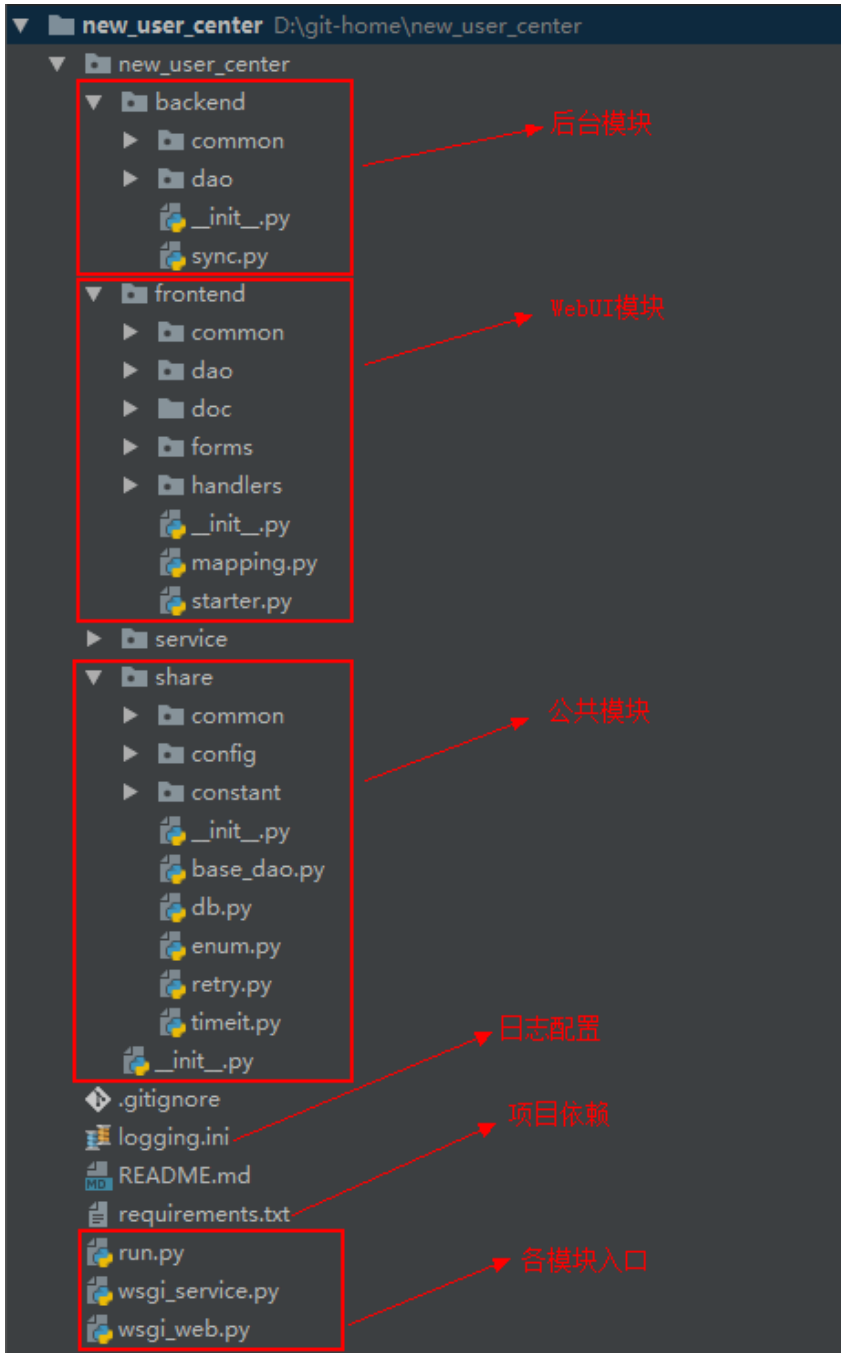
使用切面技术实现授权

项目结构

聚类原则
相关的代码放在一起，方便索引

正交原则
不相关的代码分开放，避免混乱

解耦原则
复用的代码抽离放置，



项目结构

```python
import traceback
import logging.config

import click


@click.group(invoke_without_command=True)
@click.pass_context
def cli(ctx, **kwargs):
    logging.config.fileConfig("logging.ini")
    return ctx


@cli.command()
@click.option('-a', '--address', type=str)
@click.option('-p', '--port', type=int)
@click.pass_context
def webui(ctx, address, port):
    from new_user_center.frontend.starter import start
    start(address, port)


if __name__ == "__main__":
    cli()
```

使用click，集中管理所有模块入口
避免手动修改PYTHON_PATH

日志告警.规范

第一步

配置logging.ini

```
[loggers]
keys=root,osgSync,webui,tornado

[logger_root]
level=DEBUG
handlers=screen

[logger_osgSync]
level=DEBUG
handlers=screen,fileOsgSync
qualname=osgSync
propagate=0

[logger_webui]
level=DEBUG
handlers=screen,fileWebui
qualname=webui
propagate=0

[logger_tornado]
level=DEBUG
handlers=screen,fileTornado
qualname=tornado
propagate=0

[handlers]
keys=screen,fileOsgSync,fileWebui,fileTornado
```

第二步

入口处加载配置

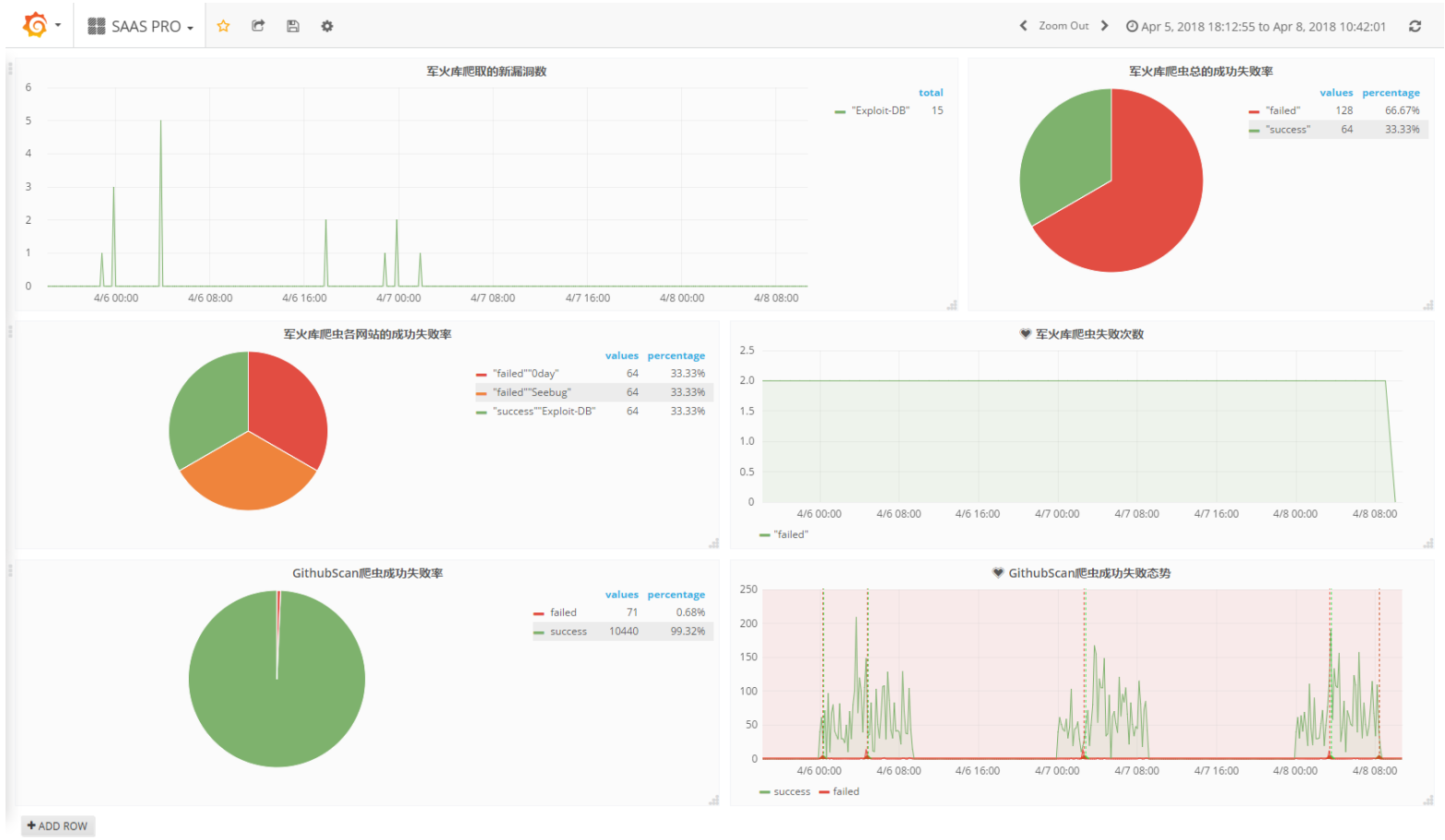
```
logging.config.fileConfig("logging.ini")
```

第三步

获取logger，打印日志

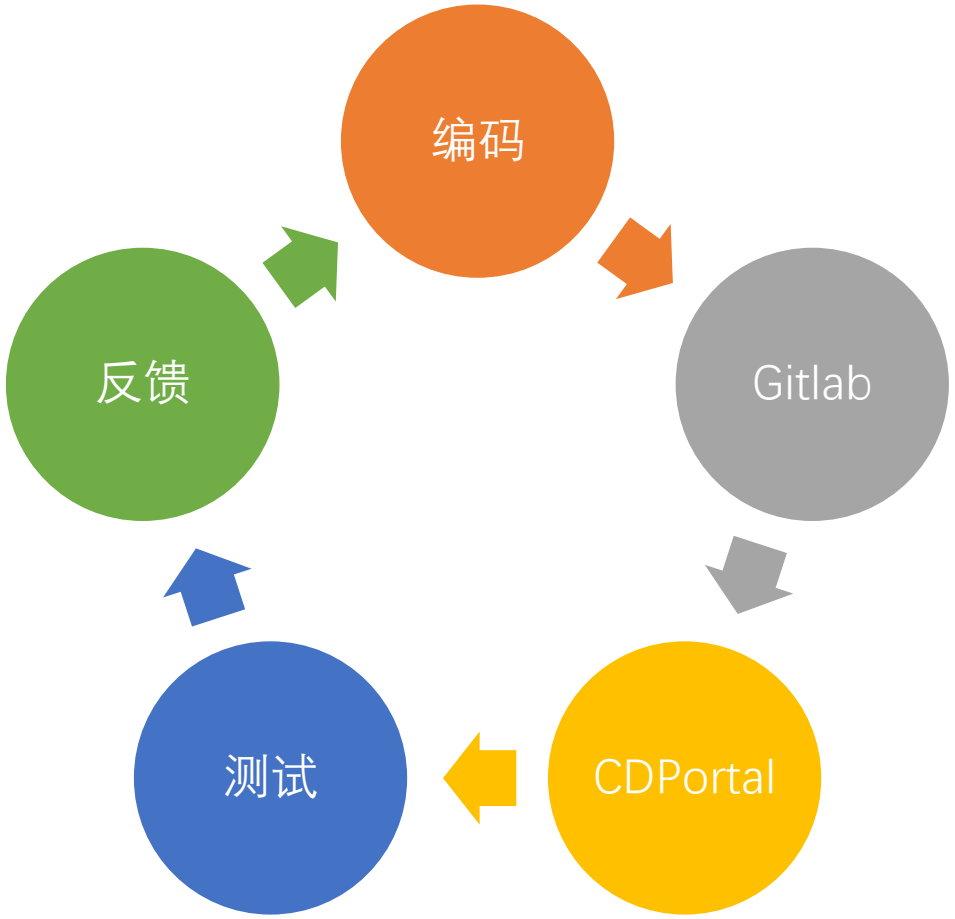
```
logger = logging.getLogger("webui")
logger.info("a info msg")
logger.debug("a debug msg")
logger.error("a error msg")
```

日志告警·Grafana



持续集成

- 避免手工发布
- 支持敏捷开发
- 减少误操作概率



thank you

- Q&A
- 畅所欲言