链滴

FLask 灵活管理多种场景下的配置

作者：YYJeffrey

原文链接：https://ld246.com/article/1576571300733

来源网站：链滴

Flask的配置文件引入方式又多种，并且它能应对多种不同场景下的不同配置。

Flask引入配置的方式

Flask引入配置的方式有很多种，我比较常用的是以对象或者文件的形式引入

1.字典形式引入

如果需要由业务逻辑修改配置可以用字典方式

```
app.config['DEBUG'] = True
```

2.文件形式引入

通过from_pyfile()方法，可以从文件引入配置

```
# 引入配置
app.config.from_pyfile("文件名")

# config.py
DEBUG = True
```

3.对象形式引入

通过from_object()方法，可以从对象引入配置

```
# 引入配置
app.config.from_object("python类")
```

```
# config.py
class Config(object):
    DEBUG = False
```

4.环境变量形式引入

app.config.from_json("JSON文件名")

5.JSON形式形式引入

app.config.from_envvar("环境变量名称")

6.字典函数形式引入

app.config.from_mapping({'DEBUG': True})

多种环境下配置

我们往往不仅仅只需要一套配置，开发时需要一套配置，测试需要一套，生产需要一套，下面我通过用对象引入的方式演示多种环境下的配置编写。

```
# config.py
import os

class BaseConfig:
    # 基础配置
    DEBUG = False
    HOST = "127.0.0.1"
    PORT = "5000"

    # 系统配置
    SECRET_KEY = os.getenv("SECRET_KEY", "set a secret key")
    PROJECT_ROOT = os.path.abspath(os.path.dirname(__file__))

    # 数据库配置
    SQLALCHEMY_TRACK_MODIFICATIONS = False


class DevConfig(BaseConfig):
    # 基础配置
    DEBUG = True
    SQLALCHEMY_ECHO = True
    SQLALCHEMY_TRACK_MODIFICATIONS = True
    HOST = "0.0.0.0"

    # 数据库配置
    SQLALCHEMY_DATABASE_URI = "sqlite:///" + os.path.join(PROJECT_ROOT, 'data-dev.db')


class TestingConfig(DevConfig):
    # 基础配置
    TESTING = True
```

```
    WTF_CSTF_ENABLED = False

    # 数据库配置
    SQLALCHEMY_DATABASE_URI = "sqlite:///:memory:"


class ProdConfig(BaseConfig):
    # 基础配置
    DEBUG = False
    SQLALCHEMY_ECHO = False
    HOST = "0.0.0.0"
    PORT = "5000"

    # 数据库配置
    SQLALCHEMY_DATABASE_URI = "sqlite:///" + os.path.join(PROJECT_ROOT, 'data.db')


config_choice = {
    "default": BaseConfig,
    "dev": DevConfig,
    "test": TestingConfig,
    "prod": ProdConfig
}
```

之后你可以根据不同的环境在app下调用配置

```
config_env = os.getenv('FLASK_CONFIG') or 'prod'
app.config.from_object(config_choice.get(config_env))
```