<https://www.cnblogs.com/kaerxifa/p/11780941.html>

[**【flask】flask项目配置 app.config**](https://www.cnblogs.com/kaerxifa/p/11780941.html)

**【理论】**

　在很多情况下，你需要设置程序的某些行为，这时你就需要使用配置变量。在Flask中，配置变量就是一些大写形式的Python变量，

你也可以称之为配置参数或配置键。使用统一的配置变量可以避免在程序中以硬编码的形式设置程序。

　　在一个项目中，你会用到许多配置，Flask提供的配置，扩展（比如flask-sqlalchemy,flask-mail）提供的配置，还有程序特定的配置。

和平时使用的变量不同，这些配置变量都通过Flask对象的app.config属性作为统一的接口来设置和获取，它指向的Config类实际上是字典的的子类，

所以你可以像操作其它字典一样操作它。

　　当我们创建1个Flask对象(也就是app),就可以通过app.config来查看这个app的所有配置变量

[复制代码](javascript:void(0);)

from flask import Flask

app=Flask(\_\_name\_\_)

print(type(app.config))

<class 'flask.config.Config'>#app.config的类型

print(app.config)#app.config 以键值对的形式 保存app的配置信息

<Config {'ENV': 'production', 'DEBUG': False, 'TESTING': False, 'PROPAGATE\_EXCEPTIONS': None, 'PRESERVE\_CONTEXT\_ON\_EXCEPTION': None, 'SECRET\_KEY': None, 'PERMANENT\_SESSION\_LIFETIME': datetime.timedelta(days=31), 'USE\_X\_SENDFILE': False, 'SERVER\_NAME': None, 'APPLICATION\_ROOT': '/', 'SESSION\_COOKIE\_NAME': 'session', 'SESSION\_COOKIE\_DOMAIN': None, 'SESSION\_COOKIE\_PATH': None, 'SESSION\_COOKIE\_HTTPONLY': True, 'SESSION\_COOKIE\_SECURE': False, 'SESSION\_COOKIE\_SAMESITE': None, 'SESSION\_REFRESH\_EACH\_REQUEST': True, 'MAX\_CONTENT\_LENGTH': None, 'SEND\_FILE\_MAX\_AGE\_DEFAULT': datetime.timedelta(seconds=43200), 'TRAP\_BAD\_REQUEST\_ERRORS': None, 'TRAP\_HTTP\_EXCEPTIONS': False, 'EXPLAIN\_TEMPLATE\_LOADING': False, 'PREFERRED\_URL\_SCHEME': 'http', 'JSON\_AS\_ASCII': True, 'JSON\_SORT\_KEYS': True, 'JSONIFY\_PRETTYPRINT\_REGULAR': False, 'JSONIFY\_MIMETYPE': 'application/json', 'TEMPLATES\_AUTO\_RELOAD': None, 'MAX\_COOKIE\_SIZE': 4093}>

[复制代码](javascript:void(0);)

　　常用配置及含义：

[复制代码](javascript:void(0);)

ENV：测试环境

production：生产环境，development：开发环境

DEBUG：是否开启debug模式

SECRET\_KEY：密钥字符串

JSON\_AS\_ASCII：是否以ascii编码展示响应报文

JSONIFY\_MIMETYPE：响应报文类型

[复制代码](javascript:void(0);)

**值得注意的是：**

　　如果响应报文含中文，JSON\_AS\_ASCII的值为False,JSONIFY\_MIMETYPE值为"application/json;charset=utf-8"，

　　否则响应报文中的中文会显示乱码

JSON\_AS\_ASCII =False

JSONIFY\_MIMETYPE = "application/json;charset=utf-8"

**【实践】**

 　　Flask提供了很多种方式来加载配置。比如，你可以像在字典中添加一个键值对一样来设置一个配置：

app.config['ADMIN\_NAME']='Peter'

**注意：配置的名称必须是全大写形式，小写的变量将不会被读取**

 　　除此之外，还可以通过把配置变量存储在单独的Python脚本、JSON格式的文件或是Python类中，config对象提供了相应的方法来导入配置。

　　 和操作字典一样，读取一个配置就是从config字典里通过将配置的名称作为键读取对应的值：

value=app.config['ADMIN\_NAME']

　　注意：某些扩展需要读取配置值来完成初始化操作，比如Flask-Mail,因此我们应该尽量将加载配置的操作提前，最好在程序实例app创建后就加载配置。

  　 下面将逐一演示各种配置方式，及其效果

　　 目录结构：



**通过 键值对 配置app.config:**

[复制代码](javascript:void(0);)

from flask import Flask

app =Flask(\_\_name\_\_)

print('配置前',app.config.get('FLASK\_ENV'))

#使用键值对配置app环境变量

app.config['FLASK\_ENV']='ABC'

print('配置后',app.config.get('FLASK\_ENV'))

#------运行结果----------------

配置前 None

配置后 ABC

[复制代码](javascript:void(0);)

**使用update()方法可以一次加载多个值：**

[复制代码](javascript:void(0);)

from flask import Flask

app =Flask(\_\_name\_\_)

print('配置前',app.config.get('FLASK\_ENV'))

#使用 update()方法可以一次加载多个值

app.config.update(

FLASK\_ENV='ABC',

AAAA=123456

)

print('配置后',app.config.get('FLASK\_ENV'))

print('配置后',app.config.get('AAAA'))

#------运行结果----------------

配置前 None

配置后 ABC

配置后 123456

[复制代码](javascript:void(0);)

**通过 json文件 配置app.config:app.config.from\_json**

　　settings.json文件内容如下：

{

"FLASK\_ENV":"json"

}

　　代码示例：

[复制代码](javascript:void(0);)

from flask import Flask

app =Flask(\_\_name\_\_)

import os

basedir = os.path.abspath(os.path.dirname(\_\_file\_\_))

jsondir =os.path.join(basedir,'settings.json')

print('配置前',app.config.get('FLASK\_ENV'))

#使用 json文件 配置app环境变量

app.config.from\_json(jsondir)

print('配置后',app.config.get('FLASK\_ENV'))

#------运行结果----------------

配置前 None

配置后 json

[复制代码](javascript:void(0);)

**通过 字典 配置app.config: app.config.from\_mapping**

　　代码示例：

[复制代码](javascript:void(0);)

from flask import Flask

app =Flask(\_\_name\_\_)

configDict={

'FLASK\_ENV':'dictConfig'

}

print('配置前',app.config.get('FLASK\_ENV'))

#使用 json文件 配置app环境变量

app.config.from\_mapping(configDict)

print('配置后',app.config.get('FLASK\_ENV'))

#------运行结果----------------

配置前 None

配置后 dictConfig

[复制代码](javascript:void(0);)

**通过 python文件 配置app.config:**

　　settings.py文件中内容如下：

　　py文件中的变量名将作为 app.config配置的key，变量值作为app.config[key]的value

FLASK\_ENV='pyConfig'

代码示例：

[复制代码](javascript:void(0);)

from flask import Flask

app =Flask(\_\_name\_\_)

print('配置前',app.config.get('FLASK\_ENV'))

#使用 json文件 配置app环境变量

app.config.from\_pyfile('settings.py')

print('配置后',app.config.get('FLASK\_ENV'))

#------运行结果----------------

配置前 None

配置后 pyConfig

[复制代码](javascript:void(0);)

**通过 配置类 配置app.config:**

　　classConfig.py定义了配置类BaseConfig

　　类中的属性作为 app.config的key，类中的属性值作为app.config[key]的value

class BaseConfig(object):

FLASK\_ENV='objectConfig'

　　代码示例：

[复制代码](javascript:void(0);)

from flask import Flask

app =Flask(\_\_name\_\_)

from demoConfig.classConfig import BaseConfig

objConfig=BaseConfig()

print('配置前',app.config.get('FLASK\_ENV'))

#使用 json文件 配置app环境变量

app.config.from\_object(objConfig)

print('配置后',app.config.get('FLASK\_ENV'))

#------运行结果----------------

配置前 None

配置后 objectConfig

[复制代码](javascript:void(0);)