1.请求上下文与应用上下文

Flask从客户端收到请求的时候，视图函数如果要处理请求的话，可能就要访问一些对象。

那么这些对象可以通过参数的形式传递进来，或者在函数中访问外部变量。

这个外部变量要有特定的值才会有意义，也就是上下文

==**例如**==

```
from flask import Flask,  request

app = Flask(__name__)


@app.route('/')
def index():
    user_agent = request.headers.get('User-Agent')
    return 'Your brower is %s' % user_agent


if __name__ == '__main__':
    app.run()

# 这里的request变量就是 请求上下文，
# 也就是当请求被推送之后，request才有意义，接下来才可以用request

# 请求上下文(request context)保存了客户端和服务器交互的数据
# request和session都属于请求上下文对象(request context)。
```

程序中的上下文：
    一个函数通常涉及了外部变量 (或方法)，要正常使用这个函数，就需要先赋值给这些外部变量，这些外部变量值的集合就称为上下文

应用上下文(application context)

current_app和g都属于应用上下文对象。

current_app:表示当前运行程序文件的程序实例。我们可以通过current_app.name打印出当前应用程序实例的名字。

2.current_app对象

current_app表示当前运行程序文件的程序实例。
我们可以通过current_app.name打印出当前应用程序实例的名字。

current_app 对象,它被绑定到当前请求的应用的引用。

如果在程序中需要访问应用，那么需要将应用显式地到处传递应用，
如果使用current_app对象，我们不需要关心创建的应用，
当我们访问current_app对象的时候，实际上是访问请求的应用。

```
# -*- encoding: utf-8 -*-
# 导入Flask类
from flask import Flask, current_app


# 在创建app应用的时候，每个创建的对象名可能都不一样。
# 使用current_app 应用上下文，我们可以不用关心应用怎么创建的。
app = Flask(__name__)
# app_2 = Flask(__name__)
# my_app = Flask(__name__)


class Config(object):
    USERNAME = 'hello'
    PASSWORD = '123456'


app.config.from_object(Config)


@app.route('/')
def index():
    username = current_app.config.get("USERNAME")  # 可以不通过Flask实例名称
```

```
获取配置
    password = current_app.config.get("PASSWORD")  # 获取配置
    return 'username=%s,password=%s' % (username, password)


if __name__ == '__main__':

    # Flask 应用程序实例的方法run启动web服务器
    app.run(debug=True)
```

3.g对象

有时候应用上下文会在必要时被创建和销毁。它不会在线程间移动，并且也不会在不同的请求之间共享。

处理请求时，临时存储的对象，每次请求都会重设这个变量。

--- 比如 数据库连接，请求进来的时候创建连接，在处理完成之后，释放连接。

通过请求钩子deardown_request函数中通过g对象将数据库连接对象传到这个函数中去关闭连接。确保数据库资
源能被释放。

g对象是应用上下文的一种，每一个请求过来都会创建一个g对象。g对象就是一个作用于app应用的全局变量。每
个请求进来g对象都先置为空

在钩子函数与视图函数中变量的传递，可以用g对象做为全局变量去传递。

```
# -*- encoding: utf-8 -*-
# 导入Flask类
from flask import Flask, g

app = Flask(__name__)


@app.before_request
def before_request_():
    g.s = 'g对象传过来的参数'
    print('before_request执行')
```

```
@app.route('/')
def index():
    print('index 执行')
    # 在视图函数中通过g对象获取参数
    print(g.get('s'))
    return 'ok'


if __name__ == '__main__':

    # Flask 应用程序实例的方法run启动web服务器
    app.run(debug=True)
```