

# 1.about函数

`about`函数的作用是：放弃请求并返回错误代码  
相当于python中的 `raise` 抛出异常

详细HTTP状态码 [https://seo.juziseo.com/doc/http\\_code/](https://seo.juziseo.com/doc/http_code/)

```
# coding=utf-8
# 导入Flask类
from flask import Flask, abort
# Flask 接收一个参数 name ,
# 导入模块的目录， flask以这个目录为基础，寻找静态文件目录static和模板目录templates

app = Flask( name )
@app.route('/')
def index():
    # 终止视图执行，并返回HTTP状态码
    # abort 函数只能接受标准http代码
    abort(404)
    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)
```

## 2.自定义错误处理视图

使用 `errorhandler` 装饰器，接受一个http状态码为参数

自定义的错误视图不单单作用于`abort`函数抛出的错误，也作用于整个`Flask`应用对应错误码。

自定义错误处理视图接收一个参数，是`Flask`应用的默认报错信息

```
# coding=utf-8
# 导入Flask类
from flask import Flask, abort

# Flask 接收一个参数 name ,
# 导入模块的目录， flask以这个目录为基础，寻找静态文件目录static和模板目录templates
app = Flask( name )

@app.route('/')
def index():
    # 终止视图执行，并返回HTTP状态码
    # abort 函数只能接受标准http代码
    abort(404)
    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)
```

```
def index():
    # 终止视图执行，并返回HTTP状态码
    # abort 函数只能接受标准http代码
    abort(404)
    return 'ok'

# 自定义错误处理视图函数
# 使用 errorhandler 装饰器，接受一个http状态码为参数。
# 自定义的错误视图不单单作用于abort函数抛出的错误，也作用于整个Flask应用对应错误码。
# 自定义错误处理视图接收一个参数，是Flask应用的默认报错信息
@app.errorhandler(404)
def error(e):
    return '哈哈，url错误。 e= %s' % e

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