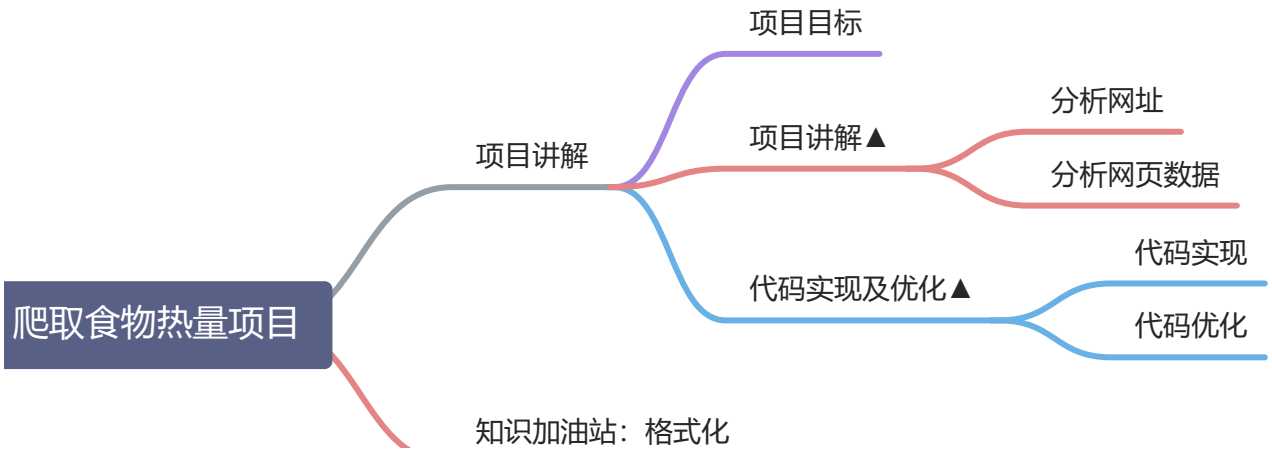
第12课 项目实操：爬取热量

一、课程结构导图



注：▲为重点知识点。

二、项目讲解

2.1 项目目标

爬取目标：用多协程爬取11个常见食物分类里的食物信息（包含食物名、热量、食物详情页面链接），并写入本地文件。

目标网址：http://www.boohee.com/food/ <http://www.boohee.com/food/group/1>

2.2 项目解析

重点讲解分析网址和分析网页数据部分。

2.2.1 分析网址

在寻找网址的规律前，需要先确认需要爬取的数据是否在打开的网页链接上，即查看每一个网址的第0个请求，确认数据在网页上。经过查看，需要爬取的数据都是在网页上，下面开始查看网址的规律。

- **每一类别的网址的规律**

分别查看11个类别的网址，寻找每个类别网址之间的规律，结果如下：

常见食物分类的网址

类别	网址
第1类【谷薯芋、杂豆、主食】	http://www.boohee.com/food/group/1
第2类【蛋类、肉类及制品】	http://www.boohee.com/food/group/2
第3类【奶类及制品】	http://www.boohee.com/food/group/3
第4类【蔬果和菌藻】	http://www.boohee.com/food/group/4
第5类【坚果、大豆及制品】	http://www.boohee.com/food/group/5
第6类【饮料】	http://www.boohee.com/food/group/6
第7类【食用油、油脂及制品】	http://www.boohee.com/food/group/7
第8类【调味品】	http://www.boohee.com/food/group/8
第9类【零食、点心、冷饮】	http://www.boohee.com/food/group/9
第10类【其他】	http://www.boohee.com/food/group/10
第11类【菜肴】	http://www.boohee.com/food/view_menu

by 风变编程

前10个常见食物分类的网址都是：

> http://www.boohee.com/food/group/+ <http://www.boohee.com/food/group/+> 数字

唯独最后一个分类【菜肴】的网址与其它不同：

> http://www.boohee.com/food/view_menu <http://www.boohee.com/food/view_menu>

- **每一类别中各个页面的网址的规律**

该类别中各个网页的网址比这一类别的网址多了【?page=页数】。

【谷薯芋、杂豆、主食】类别各网页的网址

页面	网址
第1页	http://www.boohee.com/food/group/1?page=1
第2页	http://www.boohee.com/food/group/1?page=2
第3页	http://www.boohee.com/food/group/1?page=3
第4页	http://www.boohee.com/food/group/1?page=4
第5页	http://www.boohee.com/food/group/1?page=5
第6页	http://www.boohee.com/food/group/1?page=6
第7页	http://www.boohee.com/food/group/1?page=7
第8页	http://www.boohee.com/food/group/1?page=8
第9页	http://www.boohee.com/food/group/1?page=9
第10页	http://www.boohee.com/food/group/1?page=10

其它的十个类别的规律同上。

- **食物详情页链接的规律**

每一个食物的详情页都是根目录加上网页源代码上，该食物的链接，比如Easy Fun 营养粉丝(香菇炖鸡)的完整链接就是【http://www.boohee.com <http://www.boohee.com> 】【/shiwu/fda3bd99】。

2.2.2 分析网页数据

打开开发者工具，通过小箭头定位到所要爬取的数据所在的位置，发现食物名和食物详情页面链接都在<h4>标签下的<a>标签里，食物名是属性title的值，链接是属性href的值；热量在和<h4>标签同级的<p>标签中，二者的最小父级标签是 中。可以通过BeautifulSoup库中的find_all()方法获取该最小父级标签，然后分别取出各个食物所需要的三个数据。其它的数据同上。

2.3 代码实现与优化

2.3.1 代码实现

不使用gevent库的情况下，实现项目目标。

- **构造网址**

```
## 构造110个网址

# 构造前10个类别中，各个类别的各10页食物记录的网址
url_1 = 'http://www.boohee.com/food/group/{type}?page={page}'
for x in range(1, 11):
    for y in range(1, 11):
        real_url = url_1.format(type=x, page=y)

# 构造第11个类别的10页食物记录的网址：
url_2 = 'http://www.boohee.com/food/view_menu?page={page}'
for x in range(1,11):
    real_url = url_2.format(page=x)
```

- **定义爬取函数**

```
import requests,bs4
def get_data(url):
    headers = {'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.110 Safari/537.36'}
    #添加请求头
    res = requests.get(url, headers=headers)    #用requests.get获取网页源代码
    bs_res = bs4.BeautifulSoup(res.text, 'html.parser')     #用BeautifulSoup解析网页源代码
    foods = bs_res.find_all('div',class_ = "text-box")      # 提取出最小父级标签
    for food in foods:                              #遍历foods
        food_name = food.find('a')['title']     # 提取<a>元素中属性title的值，即食物名称
        food_url = food.find('a')['href']       # 提取<a>元素中属性href的值，即食物链接
        full_url = 'http://www.boohee.com' + food_url       # 拼接完整链接
        food_calorie = food.find('p').text      # 提取<p>元素中的字符串，即食物热量
```

注：在构造网址的循环中调用爬取函数即可完成爬取。

- **保存文件**

```
import csv
csv_file= open('boohee.csv', 'w', newline='')
writer = csv.writer(csv_file)
```

```
writer.writerow(['食物', '热量', '链接']) # 写入表头

# 写入数据，把该行代码放到爬取的函数里即可把爬取到的数据进行保存
writer.writerow([food_name, food_calorie, full_url])

csv_file.close()    # 放到最后
```

- **完整代码**


```
import requests,bs4,csv
# 写入文件
csv_file= open('boohee.csv', 'w', newline='')
writer = csv.writer(csv_file)
writer.writerow(['食物', '热量', '链接']) # 写入表头

# 定义爬取函数
def get_data(url):
    headers = {'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
    res = requests.get(url, headers=headers)    #用requests.get获取网页源代码
    bs_res = bs4.BeautifulSoup(res.text, 'html.parser')     #用BeautifulSoup解析网页源代码
    foods = bs_res.find_all('div',class_ = "text-box")      # 提取出最小父级标签
    for food in foods:                              #遍历foods
        food_name = food.find('a')['title']     # 提取<a>元素中属性title的值，即食物名称
        food_url = food.find('a')['href']       # 提取<a>元素中属性href的值，即食物链接
        full_url = 'http://www.boohee.com' + food_url       # 拼接完整链接
        food_calorie = food.find('p').text      # 提取<p>元素中的字符串，即食物热量

        # 写入数据，把该行代码放到爬取的函数里即可把爬取到的数据进行保存
        writer.writerow([food_name, food_calorie, full_url])

## 构造110个网址
# 构造前10个类别中，各个类别的各10页食物记录的网址
url_1 = 'http://www.boohee.com/food/group/{type}?page={page}'
for x in range(1, 11):
    for y in range(1, 11):
        real_url = url_1.format(type=x, page=y)
        get_data(real_url)  # 调用爬取函数
# 构造第11个类别的10页食物记录的网址：
url_2 = 'http://www.boohee.com/food/view_menu?page={page}'
for x in range(1,11):
    real_url = url_2.format(page=x)
    get_data(real_url)      # 调用爬取函数

```

```
# 关闭文件
csv_file.close()    # 放到最后
```


2.3.2 代码优化

```
#导入所需的库和模块：
from gevent import monkey
monkey.patch_all()
import gevent,requests, bs4, csv
from gevent.queue import Queue

work = Queue()      # 创建队列对象，并赋值给work。

# 构造前10个类别中，各个类别的各10页食物记录的网址
url_1 = 'http://www.boohee.com/food/group/{type}?page={page}'
for x in range(1, 11):
    for y in range(1, 11):
        real_url = url_1.format(type=x, page=y)
        work.put_nowait(real_url)   # 把网址添加进队列
# 构造第11个类别的10页食物记录的网址：
url_2 = 'http://www.boohee.com/food/view_menu?page={page}'
for x in range(1,11):
    real_url = url_2.format(page=x)
    work.put_nowait(real_url)       # 把网址添加进队列

# 定义爬取函数
def get_data():     # url通过队列传入
    headers = {'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.110 Safari/537.36'}
#添加请求头
    while not work.empty():         # 判断队列是否为空
        url = work.get_nowait()     # 获取队列中的网址

        res = requests.get(url, headers=headers)    #用requests.get获取网页源代码
        bs_res = bs4.BeautifulSoup(res.text, 'html.parser')     #用BeautifulSoup解析
网页源代码
        foods = bs_res.find_all('div',class_ = "text-box")      # 提取出最小父级标签
        for food in foods:                          #遍历foods
            food_name = food.find('a')['title']     # 提取<a>元素中属性title的值，即食
物名称
            food_url = food.find('a')['href']       # 提取<a>元素中属性href的值，即食
```

```
    物链接
            full_url = 'http://www.boohee.com' + food_url       # 拼接完整链接
            food_calorie = food.find('p').text      # 提取<p>元素中的字符串，即食物热
    量

            # 写入数据，把该行代码放到爬取的函数里即可把爬取到的数据进行保存
            writer.writerow([food_name, food_calorie, full_url])
# 写入文件
csv_file= open('boohee.csv', 'w', newline='')
writer = csv.writer(csv_file)
writer.writerow(['食物', '热量', '链接'])

tasks_list = []
for x in range(5):                  # 创建5个协程，相当于创建5个爬虫
    task = gevent.spawn(get_data)   # 创建任务，传入函数名get_data
    tasks_list.append(task)         # 添加任务到列表中
gevent.joinall(tasks_list)          # 执行任务
```

注：调用gevent库实现批量爬取，提高效率。

三、知识加油站：格式化

示例代码：

```
# 以下print()函数都是打印:1 加 1 等于 2
# 使用format
print('{} 加 1 等于 {} '.format(1,2))      # 位置传参，一一对应
print('{0} 加 1 等于 {1} '.format(1,2))    # 给大括号加上偏移量，索引到(1,2)的值
print('{num1} 加 1 等于 {sum} '.format(num1=1, sum=2))     # 大括号加上变量，使用变量传值
print('{num1} 加 1 等于 {sum} '.format(sum=2, num1=1))     # 使用变量传值，不受位置顺序影响

str = '{} 加 1 等于 {} '     # 定义字符串，使用大括号占位，可用于传值
print(str.format(1, 2))     # 给str中的括号分别传值1,2

str = '{num1} 加 1 等于 {sum} '    # 定义字符串，使用大括号占位，大括号加上变量
print(str.format(sum=2, num1=1))    # 通过变量给大括号所在位置传值

# 使用%
print('%d 加 1 等于 %d ' % (1,2))
str = '%d 加 1 等于 %d '     # 定义字符串，使用大括号占位，可用于传值
```

```
17 print(str % (1, 2))
```

代码解析：

1. 第3~6行各行代码、第8~9行代码和第11~12行代码三者都可以用于format字符串传递值。
2. 第15~17行使用%进行格式化也可以使用字符串和格式化拆分成两部分的格式。