3.5 实践项目－爬取网站的图像文件

3.5.1 项目简介

我们指定一个网站（例如中国天气网站），可以爬取这个网站中的所有图像文件，同时把这些文件保存到程序锁在文件夹的 images 子文件夹中。我们先设计了一个单线程的爬取程序，这个程序会因网站的某个图像下载过程缓慢而效率低下，为了提高爬取的效率我们还设计了一个多线程的爬取程序。在多线程程序中如果一个图像下载缓慢，那么也就是爬取它的那个线程缓慢，不影响别的爬行过程。

3.5.2 单线程爬取图像的程序

```
from bs4 import BeautifulSoup
from bs4 import UnicodeDammit
import urllib.request

def imageSpider(start_url):
    try:
        urls=[]
        req=urllib.request.Request(start_url,headers=headers)
        data=urllib.request.urlopen(req)
        data=data.read()
        dammit=UnicodeDammit(data,["utf-8","gbk"])
        data=dammit.unicode_markup
        soup=BeautifulSoup(data,"lxml")
        images=soup.select("img")
        for image in images:
            try:
                src=image["src"]
                url=urllib.request.urljoin(start_url,src)
                if url not in urls:
                    urls.append(url)
                    print(url)
                    download(url)
            except Exception as err:
                print(err)
    except Exception as err:
        print(err)

def download(url):
    global count
    try:
        count=count+1
        if(url[len(url)-4]=="."):
            ext=url[len(url)-4:]
```

```
                else:
                    ext=""
                req=urllib.request.Request(url,headers=headers)
                data=urllib.request.urlopen(req,timeout=100)
                data=data.read()
                fobj=open("images\\"+str(count)+ext,"wb")
                fobj.write(data)
                fobj.close()
                print("downloaded "+str(count)+ext)
            except Exception as err:
                print(err)

    start_url="http://www.weather.com.cn/weather/101280601.shtml"

    headers = {
        "User-Agent": "Mozilla/5.0 (Windows; U; Windows NT 6.0 x64; en-US; rv:1.9pre) Gecko/2008072421 Minefield/3.0.2pre"}
    count=0
    imageSpider(start_url)
```

在这个单线程的爬取程序中是逐个取下载图像文件的，如果一个文件没有完成下载，后面一个下载任务就必须等待。如果一个文件没有完成下载或者下载出现问题，就直接会影响下一个文件的下载，因此效率低，可靠性低。

3.5.3 多线程爬取图像的程序

```
    from bs4 import BeautifulSoup
    from bs4 import UnicodeDammit
    import urllib.request
    import threading

    def imageSpider(start_url):
        global threads
        global count
        try:
            urls=[]
            req=urllib.request.Request(start_url,headers=headers)
            data=urllib.request.urlopen(req)
            data=data.read()
            dammit=UnicodeDammit(data,["utf-8","gbk"])
            data=dammit.unicode_markup
            soup=BeautifulSoup(data,"lxml")
            images=soup.select("img")
            for image in images:
```

```
                    try:
                        src=image["src"]
                        url=urllib.request.urljoin(start_url,src)
                        if url not in urls:
                            print(url)
                            count=count+1
                            T=threading.Thread(target=download,args=(url,count))
                            T.setDaemon(False)
                            T.start()
                            threads.append(T)
                    except Exception as err:
                        print(err)
            except Exception as err:
                print(err)

    def download(url,count):
        try:

            if(url[len(url)-4]=="."):
                ext=url[len(url)-4:]
            else:
                ext=""
            req=urllib.request.Request(url,headers=headers)
            data=urllib.request.urlopen(req,timeout=100)
            data=data.read()
            fobj=open("images\\"+str(count)+ext,"wb")
            fobj.write(data)
            fobj.close()
            print("downloaded "+str(count)+ext)
        except Exception as err:
            print(err)

    start_url="http://www.weather.com.cn/weather/101280601.shtml"

    #start_url="http://www.sziit.edu.cn"

    headers = {
        "User-Agent": "Mozilla/5.0 (Windows; U; Windows NT 6.0 x64; en-US; rv:1.9pre) Gecko/2008072421 Minefield/3.0.2pre"}
    count=0
    threads=[]

    imageSpider(start_url)
```

```
for t in threads:
    t.join()
print("The End")
```

在这个多线程的爬取程序中下载图像文件的过程是一个线程，因此可以有多个文件在同时下载，而且互不干扰，如果一个文件没有完成下载或者下载出现问题，也不会影响别的文件的下载，因此效率高，可靠性高。