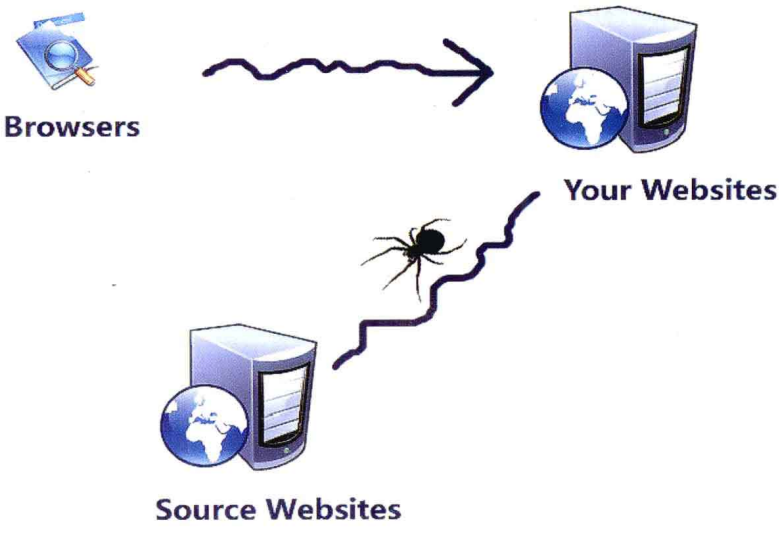
国内**第一本**专门讲解**网络爬虫**开发的书籍
介绍如何应用**云计算**架构开发分布式爬虫



罗 刚 王振东 编著

自己动手写网络爬虫

- 猎兔搜索工程师多年项目经验总结
- 深入介绍Web数据挖掘实现过程
- 光盘中提供了高效的代码解决方案
- 案例均使用流行的Java语言编写

CD-ROM

清华大学出版社

自己动手写网络爬虫

罗　刚　王振东　编著

清华大学出版社
北　京

内 容 简 介

本书介绍了网络爬虫开发中的关键问题与 Java 实现。主要包括从互联网获取信息与提取信息和对 Web 信息挖掘等内容。本书在介绍基本原理的同时注重辅以具体代码实现来帮助读者加深理解，书中部分代码甚至可以直接使用。

本书适用于有 Java 程序设计基础的开发人员。同时也可以作为计算机相关专业本科生或研究生的参考教材。

图书在版编目(CIP)数据

自己动手写网络爬虫/罗刚，王振东编著. —北京：清华大学出版社，2010.10
ISBN 978-7-302-23647-4

Ⅰ. ①自… Ⅱ. ①罗… ②王… Ⅲ. ①计算机网络—程序设计 Ⅳ. ①TP393.09

中国版本图书馆 CIP 数据核字(2010)第 160147 号

责任编辑：张　瑜　杨作梅
装帧设计：杨玉兰
责任校对：周剑云
责任印制：杨　艳
出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编**：100084
社　　总　　机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质　量　反　馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：北京市人民文学印刷厂
装 订 者：三河市兴旺装订有限公司
经　　销：全国新华书店
开　　本：185×260　**印　张**：22.25　**字　数**：535 千字
附光盘 1 张
版　　次：2010 年 10 月第 1 版　　**印　　次**：2010 年 10 月第1 次印刷
印　　数：1～4000
定　　价：43.00 元

产品编号：037230-01

前　言

当你在网上冲浪时，你是否知道还有一类特殊的网络用户也在互联网上默默地工作着，它们就是网络爬虫。这些网络爬虫按照设计者预定的方式，在网络中穿梭，同时自动收集有用的信息，进行分类和整理，将整理结果提供给用户，以方便用户查找他们感兴趣的内容。由于网络爬虫的实用性，引起了很多程序员，特别是 Web 程序员的兴趣。

但是大多数网络爬虫的开发原理与技巧在专业的公司内部都秘而不宣，至今仍然缺少理论与实践相结合的专门介绍网络爬虫的书籍。本书将弥补这个问题，尝试理论与实践相结合，深入透彻地讲解网络爬虫的原理，并且辅以相关代码作为参考。本书相关的代码在附带光盘中可以找到。

本书的两位主要作者在搜索引擎领域都有丰富的理论和实践经验。同时，还有多个程序员帮忙开发或编写了代码实现，例如 Java 实现异步 I/O 或对 PDF 文件的处理等。由于作者的日常工作繁忙，做得不够的地方敬请谅解。

作者罗刚在参加编写本书之前，还独立撰写过《自己动手写搜索引擎》一书，但存在讲解不够细致、知识点不够深入等问题。此次与王振东合著本书，相对于上一本书而言，对读者反馈有更高的预期。因为作者相信如下的假设：如果能够与更多的人更好地合作，事情往往能做得更好。

本书从基本的爬虫原理开始讲解，通过介绍优先级队列、宽度优先搜索等内容引领读者入门；之后根据当前风起云涌的云计算热潮，重点讲述了云计算的相关内容及其在爬虫中的应用，以及带偏好的爬虫、信息抽取、链接分析等内容；为了能够让读者更深入地了解爬虫，本书在最后两章还介绍了有关爬虫的数据挖掘的内容。

由于搜索引擎相关领域也正在快速发展中，而且由于篇幅的限制，有些不成熟的内容，没有能够在本书体现，例如有关“暗网”的内容。随着技术的不断发展，我们将在今后的版本中加入这些内容。

本书适合需要具体实现搜索引擎的程序员使用，对于信息检索等相关研究人员也有一定的参考价值，同时猎兔搜索技术团队也已经开发出以本书为基础的专门培训课程和商业软件。目前搜索引擎开发人员仍然很稀缺，作者真诚地希望通过本书把读者带入搜索引擎开发的大门并认识更多的朋友。

感谢开源软件和我们的家人、关心我们的老师和朋友、创业伙伴以及选择猎兔搜索软件的客户多年来的支持。

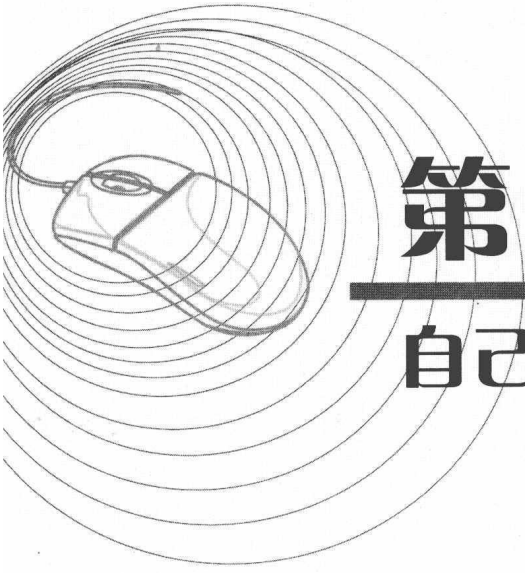
编　者

目　录

第 1 篇　自己动手抓取数据

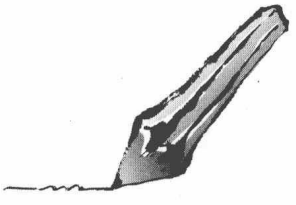
第 2 篇　自己动手抽取 Web 内容

第 3 篇　自己动手挖掘 Web 数据



第1篇

自己动手抓取数据

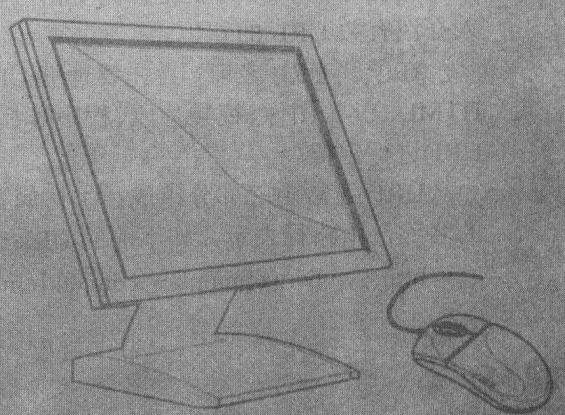


第1章

全面剖析网络爬虫

你知道百度、Google 是如何获取数以亿计的网页并且实时更新的吗？你知道在搜索引擎领域人们常说的 Spider 是什么吗？本章将全面介绍网络爬虫的方方面面。读完之后，你将完全有能力自己写一个网络爬虫，随意抓取互联网上任何感兴趣的东西。

既然百度、Google 这些搜索引擎巨头已经帮我们抓取了互联网上的大部分信息，为什么还要自己写爬虫呢？因为深入整合信息的需求是广泛存在的。在企业中，爬虫抓取下来的信息可以作为数据仓库多维展现的数据源，也可以作为数据挖掘的来源。甚至有人为了炒股，专门抓取股票信息。既然从美国中情局到普通老百姓都需要，那还等什么，让我们快开始吧。



1.1 抓 取 网 页

网络爬虫的基本操作是抓取网页。那么如何才能随心所欲地获得自己想要的页面？这一节将从 URL 开始讲起，然后告诉大家如何抓取网页，并给出一个使用 Java 语言抓取网页的例子。最后，要讲一讲抓取过程中的一个重要问题：如何处理 HTTP 状态码。

1.1.1 深入理解 URL

抓取网页的过程其实和读者平时使用 IE 浏览器浏览网页的道理是一样的。比如，你打开一个浏览器，输入猎兔搜索网站的地址，如图 1.1 所示。

图 1.1 使用浏览器浏览网页

“打开”网页的过程其实就是浏览器作为一个浏览的“客户端”，向服务器端发送了一次请求，把服务器端的文件“抓”到本地，再进行解释、展现。更进一步，可以通过浏览器端查看“抓取”过来的文件源代码。选择“查看”|“源文件”命令，就会出现从服务器上“抓取”下来的文件的源代码，如图 1.2 所示。

在上面的例子中，我们在浏览器的地址栏中输入的字符串叫做 URL。那么，什么是 URL 呢？直观地讲，URL 就是在浏览器端输入的 http://www.lietu.com 这个字符串。下面我们深入介绍有关 URL 的知识。

在理解 URL 之前，首先要理解 URI 的概念。什么是 URI？Web 上每种可用的资源，如 HTML 文档、图像、视频片段、程序等都由一个通用资源标志符(Universal Resource Identifier，URI)进行定位。

URI 通常由三部分组成：①访问资源的命名机制；②存放资源的主机名；③资源自身的名称，由路径表示。如下面的 URI：

http://www.webmonkey.com.cn/html/html40/

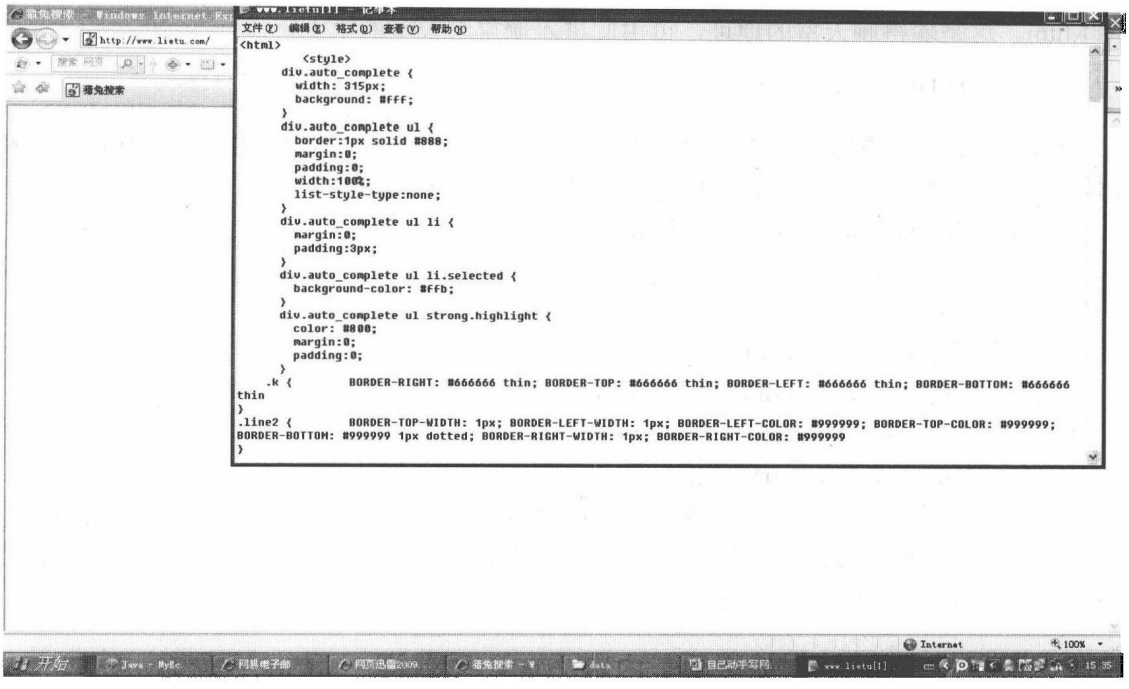


图 1.2　浏览器端源代码

我们可以这样解释它：这是一个可以通过 HTTP 协议访问的资源，位于主机 www.webmonkey.com.cn 上，通过路径“/html/html40”访问。

URL 是 URI 的一个子集。它是 Uniform Resource Locator 的缩写，译为“统一资源定位符”。通俗地说，URL 是 Internet 上描述信息资源的字符串，主要用在各种 WWW 客户程序和服务器程序上，特别是著名的 Mosaic。采用 URL 可以用一种统一的格式来描述各种信息资源，包括文件、服务器的地址和目录等。URL 的格式由三部分组成：

- 第一部分是协议(或称为服务方式)。
- 第二部分是存有该资源的主机 IP 地址(有时也包括端口号)。
- 第三部分是主机资源的具体地址，如目录和文件名等。

第一部分和第二部分用“://”符号隔开，第二部分和第三部分用“/”符号隔开。第一部分和第二部分是不可缺少的，第三部分有时可以省略。

根据 URL 的定义，我们给出了常用的两种 URL 协议的例子，供大家参考。

1. HTTP 协议的 URL 示例

使用超级文本传输协议 HTTP，提供超级文本信息服务的资源。

例：http://www.peopledaily.com.cn/channel/welcome.htm

其计算机域名为 www.peopledaily.com.cn。超级文本文件(文件类型为.html)是在目录/channel 下的 welcome.htm。这是中国人民日报的一台计算机。

例：http://www.rol.cn.net/talk/talk1.htm

其计算机域名为 www.rol.cn.net。超级文本文件(文件类型为.html)是在目录/talk 下的

talk1.htm。这是瑞得聊天室的地址，可由此进入瑞得聊天室的第 1 室。

2. 文件的 URL

用 URL 表示文件时，服务器方式用 file 表示，后面要有主机 IP 地址、文件的存取路径(即目录)和文件名等信息。有时可以省略目录和文件名，但“/”符号不能省略。

例：file://ftp.yoyodyne.com/pub/files/foobar.txt

上面这个 URL 代表存放在主机 ftp.yoyodyne.com 上的 pub/files/目录下的一个文件，文件名是 foobar.txt。

例：file://ftp.yoyodyne.com/pub

代表主机 ftp.yoyodyne.com 上的目录/pub。

例：file://ftp.yoyodyne.com/

代表主机 ftp.yoyodyne.com 的根目录。

爬虫最主要的处理对象就是 URL，它根据 URL 地址取得所需要的文件内容，然后对它进行进一步的处理。因此，准确地理解 URL 对理解网络爬虫至关重要。从下一节开始，我们将详细地讲述如何根据 URL 地址来获得网页内容。

1.1.2 通过指定的 URL 抓取网页内容

上一节详细介绍了 URL 的构成，这一节主要阐述如何根据给定的 URL 来抓取网页。

所谓网页抓取，就是把 URL 地址中指定的网络资源从网络流中读取出来，保存到本地。类似于使用程序模拟 IE 浏览器的功能，把 URL 作为 HTTP 请求的内容发送到服务器端，然后读取服务器端的响应资源。

Java 语言是为网络而生的编程语言，它把网络资源看成是一种文件，它对网络资源的访问和对本地文件的访问一样方便。它把请求和响应封装为流。因此我们可以根据相应内容，获得响应流，之后从流中按字节读取数据。例如，java.net.URL 类可以对相应的 Web 服务器发出请求并且获得响应文档。java.net.URL 类有一个默认的构造函数，使用 URL 地址作为参数，构造 URL 对象：

```
URL pageURL = new URL(path);
```

接着，可以通过获得的 URL 对象来取得网络流，进而像操作本地文件一样来操作网络资源：

```
InputStream stream = pageURL.openStream();
```

在实际的项目中，网络环境比较复杂，因此，只用 java.net 包中的 API 来模拟 IE 客户端的工作，代码量非常大。需要处理 HTTP 返回的状态码，设置 HTTP 代理，处理 HTTPS 协议等工作。为了便于应用程序的开发，实际开发时常常使用 Apache 的 HTTP 客户端开源项目——HttpClient。它完全能够处理 HTTP 连接中的各种问题，使用起来非常方便。只需在项目中引入 HttpClient.jar 包，就可以模拟 IE 来获取网页内容。例如：

```
//创建一个客户端，类似于打开一个浏览器
HttpClient httpclient=new HttpClient();
```

```
//创建一个 get 方法，类似于在浏览器地址栏中输入一个地址
GetMethod getMethod=new GetMethod("http://www.blablabla.com");

//回车，获得响应状态码
int statusCode=httpclient.executeMethod(getMethod);

//查看命中情况，可以获得的东西还有很多，比如 head、cookies 等
System.out.println("response=" + getMethod.getResponseBodyAsString());

//释放
getMethod.releaseConnection();
```

上面的示例代码是使用 HttpClient 进行请求与响应的例子。第一行表示创建一个客户端，相当于打开浏览器。第二行使用 get 方式对 http://www.blablabla.com 进行请求。第三行执行请求，获取响应状态。第四行的 getMethod.getResponseBodyAsString()方法能够以字符串方式获取返回的内容。这也是网页抓取所需要的内容。在这个示例中，只是简单地把返回的内容打印出来，而在实际项目中，通常需要把返回的内容写入本地文件并保存。最后还要关闭网络连接，以免造成资源消耗。

这个例子是用 get 方式来访问 Web 资源。通常，get 请求方式把需要传递给服务器的参数作为 URL 的一部分传递给服务器。但是，HTTP 协议本身对 URL 字符串长度有所限制。因此不能传递过多的参数给服务器。为了避免这种问题，通常情况下，采用 post 方法进行 HTTP 请求，HttpClient 包对 post 方法也有很好的支持。例如：

```
//得到 post 方法
PostMethod PostMethod = new PostMethod("http://www.saybot.com/postme");

//使用数组来传递参数
NameValuePair[] postData = new NameValuePair[2];

//设置参数
postData[0] = new NameValuePair("武器", "枪");
postData[1] = new NameValuePair("什么枪", "神枪");
postMethod.addParameters(postData);

//回车，获得响应状态码
int statusCode=httpclient.executeMethod(getMethod);

//查看命中情况，可以获得的东西还有很多，比如 head、cookies 等
System.out.println("response=" + getMethod.getResponseBodyAsString());

//释放
getMethod.releaseConnection();
```

上面的例子说明了如何使用 post 方法来访问 Web 资源。与 get 方法不同，post 方法可以使用 NameValuePair 来设置参数，因此可以设置“无限”多的参数。而 get 方法采用把参

数写在 URL 里面的方式，由于 URL 有长度限制，因此传递参数的长度会有限制。

有时，我们执行爬虫程序的机器不能直接访问 Web 资源，而是需要通过 HTTP 代理服务器去访问，HttpClient 对代理服务器也有很好的支持。如：

```
//创建 HttpClient 相当于打开一个代理
HttpClient httpClient=new HttpClient();

//设置代理服务器的 IP 地址和端口
httpClient.getHostConfiguration().setProxy("192.168.0.1", 9527);

//告诉 httpClient，使用抢先认证，否则你会收到“你没有资格”的恶果
httpClient.getParams().setAuthenticationPreemptive(true);

//MyProxyCredentialsProvder 返回代理的 credential(username/password)
httpClient.getParams().setParameter(CredentialsProvider.PROVIDER,
new MyProxyCredentialsProvider());

//设置代理服务器的用户名和密码
httpClient.getState().setProxyCredentials(new AuthScope("192.168.0.1",
  AuthScope.ANY_PORT, AuthScope.ANY_REALM),
new UsernamePasswordCredentials("username","password"));
```

上面的例子详细解释了如何使用 HttpClient 设置代理服务器。如果你所在的局域网访问 Web 资源需要代理服务器的话，你可以参照上面的代码设置。

这一节，我们介绍了使用 HttpClient 抓取网页的内容，之后，我们将给出一个详细的例子来说明如何获取网页。

1.1.3 Java 网页抓取示例

在这一节中，我们根据之前讲过的内容，写一个实际的网页抓取的例子。这个例子把上一节讲的内容做了一定的总结，代码如下：

```
public class RetrivePage {
    private static HttpClient httpClient = new HttpClient();
    // 设置代理服务器
    static {
        // 设置代理服务器的 IP 地址和端口
        httpClient.getHostConfiguration().setProxy("172.17.18.84", 8080);
    }
    public static boolean downloadPage(String path) throws HttpException,
            IOException {
        InputStream input = null;
        OutputStream output = null;
        // 得到 post 方法
        PostMethod postMethod = new PostMethod(path);
        //设置 post 方法的参数
          NameValuePair[] postData = new NameValuePair[2]; postData[0] = new
          NameValuePair("name","lietu"); postData[1] = new
```

```
            NameValuePair("password","******");
            postMethod.addParameters(postData);
        // 执行，返回状态码
        int statusCode = httpClient.executeMethod(postMethod);
        // 针对状态码进行处理 (简单起见，只处理返回值为 200 的状态码)
        if (statusCode == HttpStatus.SC_OK) {
            input = postMethod.getResponseBodyAsStream();
            //得到文件名
            String filename = path.substring(path.lastIndexOf('/')+1);
            //获得文件输出流
            output = new FileOutputStream(filename);

            //输出到文件
            int tempByte = -1;
            while((tempByte=input.read())>0){
                output.write(tempByte);
            }
            //关闭输入输出流
            if(input!=null){
                input.close();
            }
            if(output!=null){
                output.close();
            }
            return true;
        }
        return false;
    }

    /**
     * 测试代码
     */
    public static void main(String[] args) {
        // 抓取 lietu 首页，输出
        try {
            RetrivePage.downloadPage("http://www.lietu.com/");
        } catch (HttpException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

上面的例子是抓取猎兔搜索主页的示例。它是一个比较简单的网页抓取示例，由于互联网的复杂性，真正的网页抓取程序会考虑非常多的问题。比如，资源名的问题，资源类型的问题，状态码的问题。而其中最重要的就是针对各种返回的状态码的处理。下一节将重点介绍处理状态码的问题。

1.1.4 处理 HTTP 状态码

上一节介绍 HttpClient 访问 Web 资源的时候，涉及 HTTP 状态码。比如下面这条语句：

```
int statusCode=httpClient.executeMethod(getMethod);//回车，获得响应状态码
```

HTTP 状态码表示 HTTP 协议所返回的响应的状态。比如客户端向服务器发送请求，如果成功地获得请求的资源，则返回的状态码为 200，表示响应成功。如果请求的资源不存在，则通常返回 404 错误。

HTTP 状态码通常分为 5 种类型，分别以 1～5 五个数字开头，由 3 位整数组成。1XX 通常用作实验用途。这一节主要介绍 2XX、3XX、4XX、5XX 等常用的几种状态码，如表 1.1 所示。

表 1.1 HTTP 常用状态码

状态代码	代码描述	处理方式
200	请求成功	获得响应的内容，进行处理
201	请求完成，结果是创建了新资源。新创建资源的 URI 可在响应的实体中得到	爬虫中不会遇到
202	请求被接受，但处理尚未完成	阻塞等待
204	服务器端已经实现了请求，但是没有返回新的信息。如果客户是用户代理，则无须为此更新自身的文档视图	丢弃
300	该状态码不被 HTTP/1.0 的应用程序直接使用，只是作为 3XX 类型回应的默认解释。存在多个可用的被请求资源	若程序中能够处理，则进行进一步处理，如果程序中不能处理，则丢弃
301	请求到的资源都会分配一个永久的 URL，这样就可以在将来通过该 URL 来访问此资源	重定向到分配的 URL
302	请求到的资源在一个不同的 URL 处临时保存	重定向到临时的 URL
304	请求的资源未更新	丢弃
400	非法请求	丢弃
401	未授权	丢弃
403	禁止	丢弃
404	没有找到	丢弃
5XX	回应代码以“5”开头的状态码表示服务器端发现自己出现错误，不能继续执行请求	丢弃

当返回的状态码为 5XX 时，表示应用服务器出现错误，采用简单的丢弃处理就可以解决。