上海交通大學

SHANGHAI JIAO TONG UNIVERSITY

学士学位论文

THESIS OF BACHELOR

论文题目：基于 WebKit 的可编程命令行爬虫系统

学生姓名: 张越

学生学号: 5080369025

专　　业: 信息安全

指导教师: 陈凯

学院(系): 信息安全工程学院

上海交通大学
毕业设计（论文）学术诚信声明

本人郑重声明：所呈交的毕业设计（论文），是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

作者签名：

日期：　　年　　月　　日

上海交通大学
毕业设计（论文）版权使用授权书

本毕业设计（论文）作者同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权上海交通大学可以将本毕业设计（论文）的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本毕业设计（论文）。

保密□，在___年解密后适用本授权书。

本论文属于

不保密□。

（请在以上方框内打“√”）

作者签名：　　指导教师签名：

日期：　　年　　月　　日　　日期：　　年　　月　　日

基于 WebKit 的可编程命令行爬虫系统

摘要

网络爬虫是在网络上自动下载网络资源的程序。它是搜索引擎的重要组成部分，也是搜集和分析网页信息的重要手段。

当前，以Ajax技术为核心的Web 2.0应用的大规模流行，给网络爬虫带来了巨大的挑战。第一，由于JavaScript在网页中的大量使用，网页上的资源不再全部以静态的形式展现，很多信息是通过JavaScript驱动异步请求/响应，直接从服务器端动态加载的,因此爬虫必须具备解析JavaScript的功能才能获得这些动态信息。第二，许多Web2.0应用都有登录机制，爬虫必须进行模拟用户登录才能获得访问资源的权限。这些挑战颠覆了传统的基于HTTP请求/响应的爬虫机制。

针对Web 2.0应用,我设计并实现了基于WebKit的可编程命令行爬虫系统。我为该系统设计了一套编程语言和解释器,从而允许终端用户使用非常简单直观的语言编写爬虫脚本。该系统是基于WebKit浏览器内核进行开发，支持DOM操作和JavaScript解析，可以很好地支持Ajax。本文最后，我使用该系统针对一些主流的Web2.0社区进行了测试，验证了该系统的有效性，并提出了改进方案。

关键词： Ajax ， 网络爬虫 ， WebKit ， 可编程爬虫

PROGRAMMABLE COMMANDLINE WEB CRAWLER SYSTEM BASED ON WEBKIT

ABSTRACT

Web crawler is a program that automatically downloads web resources from the Internet. It is a critical part of search engines as well as an important approach to collect Internet resources.

At present, along with the massive popularity of Web 2.0 applications build on Ajax, Web crawlers are facing great challenges .First, with the mass employment of JavaScript in websites, resources on web pages are no longer statically presented .Instead, many of them are acquired and created real-time through asynchronous request/response from servers driven by JavaScript. In order to collect dynamic information in these Ajax Websites, Crawlers must be able to parse JavaScript. Second, many Web 2.0 applications have a login system which require crawlers to simulate user login in order to be granted access to resources. These challenges all shatter the traditional Web crawling approach based on HTTP protocol.

For this Web 2.0 Applications I designed and implemented a programmable command-line web crawler system based on WebKit. I designed a script language and an interpreter for this system which enables users to develop Web crawlers using very simple language .The system is based on browser core so that it can manipulate the DOM and parse JavaScript thus supporting Ajax. At the end of this paper, I tested the system by crawling some popular Web 2.0 communities to prove its validity. Then I proposed ways to improve this system.

Key words: Ajax, Web crawler, WebKit, programmable Web crawler

目录

第一章 绪论

1.1 课题研究背景及意义

随着网络技术的飞速发展，互联网已经发生巨大的变化。自从进入Web 2.0时代以来，网络信息不再是由少数网页编辑人员所定制，而是由用户共同创造，每个人都是信息源。换句话说，用户不再仅仅是互联网的浏览者，也是网页内容的创造者。这种强调交互性的模式使得网页信息更加丰富多彩，增加了信息发布的实时性。

促成Web 2.0革命的关键技术是Ajax，即异步 JavaScript 和 XML（Asynchronous JavaScript and XML）。AJAX通过使用JavaScript和DOM操作以及异步服务器通信技术，大大地提高了用户与网站服务器的交互性。网页不再以静态的方式存储在服务器端，而是通过用户的请求直接从数据库里获取，并动态地生成新的网页内容返回给用户。同时，由于采用异步通信的模式，用户发送请求数据和服务器返回数据时都无需刷新整个页面，大大地提高了浏览效率，增强了用户体验。

目前，主流的社交网站、BBS、门户网站几乎都采用了Ajax技术，如新浪微博，天涯，Facebook，Twitter等。每天大量的动态信息都会在这些网站发布、传播。设计网络爬虫抓取这些动态信息是有意义和必要的，一方面，这些动态信息占据了所有网络信息很大比重，而这些信息本身很多是有意义和价值的；另一方面，可以提高搜索引擎的覆盖率和准确度，以满足用户的需求;此外,这些采集来的实时动态信息可以作为未来研究和分析的基础数据，例如针对微博的研究分析、社交网络的研究等。

传统的网络爬虫工作原理是向待采集的网页发送HTTP的POST/GET请求，获得网页的HTML源文件，再分析处理该源文件的内容。这种基于协议驱动的爬虫显然无法抓取Ajax网页的信息，原因是：其一，服务器端的HTML源文件并不会显示页面上的动态内容，或者说我们在浏览器上看到的内容和HTML源文件里包含的内容是完全不一样的,究其原因在于HTML源文件里嵌入了大量JavaScript源码，浏览器通过解析这些源码动态的从服务器端数据库获得数据，构造新的DOM树并呈现给用户。其二，传统爬虫以URL作为网页状态的标识，而对于Ajax网页来说，同一URL的页面，却可以呈现不同的内容，要想完全获取网页信息，必须遍历同一URL页面的所有不同状态。

由此可见，传统爬虫无法采集动态信息迫使需要一种能够有效的抓取Ajax网页的方法，分析JavaScript代码并抓取页面中的异步传输内容成为当前网络爬虫技术的研究课题。

设计针对Ajax网站的爬虫，必须要支持JavaScript的解析和DOM操作。目前的方法主要由以下几种：

a)通过编程模拟浏览器

b)基于JavaScript 引擎+ 浏览器内核

c)基于浏览器的方法

其中，基于浏览器的方法又分为浏览器插件和网络自动化框架这两种方法。这些方法的爬虫架构如下图所示：

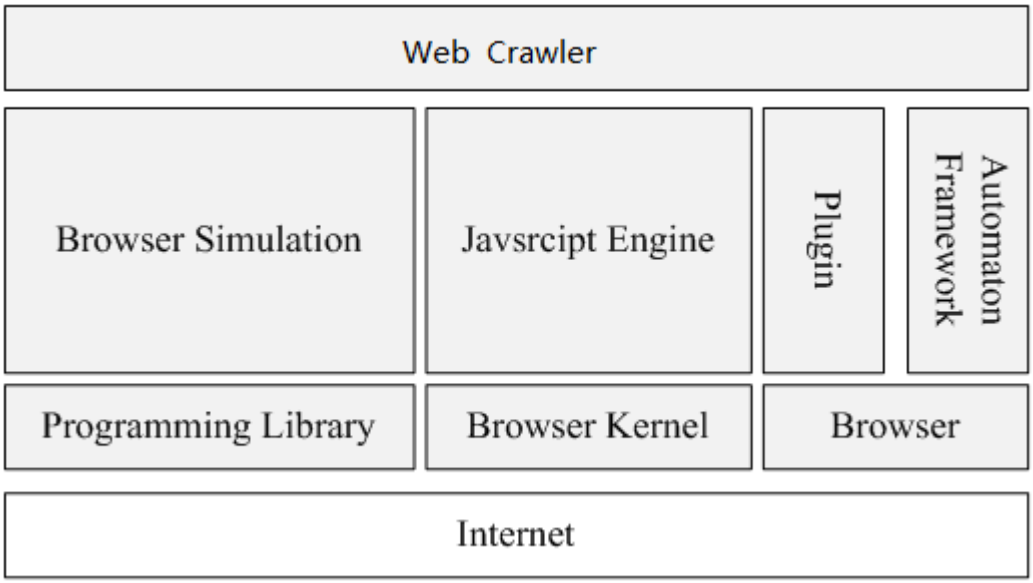

图 1-1 几种网络爬虫设计方法的系统架构示意图

每种方法都各有优缺点，例如基于浏览器的方法编程方便，但速度较慢，而且占用大量资源；编程模拟浏览器虽然效率最高但开发起来十分繁琐。因此我希望实现一种能结合上述方法优点即占资源较少，易于开发同时具有较高效率的爬虫。

使用 JavaScript 引擎和 浏览器内核的组合是一个很好的选择，一方面，浏览器内核具备了浏览器一些基本功能，如 HTML 解析、DOM 树生成，配合 JavaScript 引擎可以实现 JavaScript 的解析，这较之于从零开始开发一款模拟浏览器来说方便很多；另一方面，相对于基于整个浏览器的架构，基于浏览器内核的框架可以使用更少的资源，同时以更高的效率爬取网页。但是，这套框架仍有不足之处，即用户在编程时，仍需调用大量 WebKit 的 API，对于 DOM 操作和 HTML 的分析也需手动完成，对于终端用户来说，这一编程工作还是比较复杂的。

因此，在 JavaScript 引擎 和 浏览器内核的框架之上，我希望设计一套语言，并通过解释器与框架进行协作形成一套可编程命令行爬虫系统。该系统使得终端用户可以使用简单而通用的语言设计和操作爬虫，同时由于该系统基于浏览器内核和 JavaScript 引擎，可以高效地抓取 AJAX 网站的动态信息。

1.2 课题研究的内容和文本结构

本课题根据AJAX技术在网络的应用，研究了针对Ajax网站的网络爬虫实现方法，比较了各种实现方法的优劣，并设计一套用户可编程的爬虫系统。系统定义了一套编程框架，设计了一种专属的语言，用户通过使用该语言进行编程来实现爬虫的设计和操纵。底层采用PhantomJS的框架，实现对JavaScript的解析和DOM操作，是抓取Ajax动态网页的核心部分。中间使用Python脚本语言开发了一个解释器，用于解析用户编写的脚本程序并与PhantomJS框架进行交互协作。

本文各章节的组织如下：

第一章论述了课题的研究背景以及设计该爬虫系统的目的和意义

第二章介绍了网络爬虫相关的知识，Ajax相关知识以及Ajax对网络爬虫造成的挑战，WebKit和浏览器相关知识，以及本论文涉及到的相关技术。讨论和比较了针对Ajax网站的网络爬虫的几种设计方法。

第三章介绍了PhantomJS框架。并以新浪微博为试验对象，分析如何使用PhantomJS框架设计网络爬虫。阐述了基于WebKit可编程命令行爬虫的设计方法和思路，详细说明了系统的功能、系统的架构、编程语言的设计、编译器的设计以及各模块间的交互过程

第四章对该系统进行测试，验证和总结。

第二章 相关研究工作

2.1 网络爬虫简介

2.1.1 网络爬虫工作原理

网络爬虫（Web Crawler 或 Web Spider， Web Bot）是一个自动抓取网页的程序，它为搜索引擎从万维网上不断地下载网页，是搜索引擎的重要组成部分。传统爬虫从一个或若干初始网页的 URL 开始，获得初始网页上的 URL，在抓取网页的过程中，不断从当前页面上抽取新的 URL 放入队列,直到满足系统的一定停止条件。 其工作原理如下图：

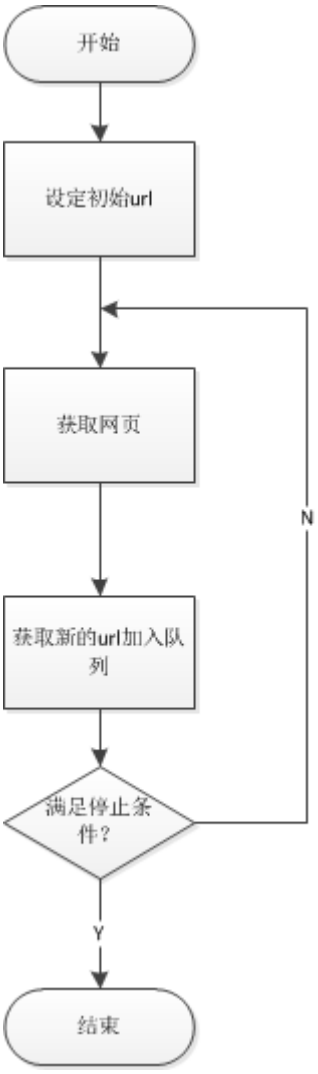

图 2-1 网络爬虫工作流程原理

网络爬虫一般由爬行模块、URL 处理模块、页面分析模块和数据库组成。

爬行模块决定网络爬虫使用何种爬行策略抓取网页,URL 处理模块对 URL 队列进行处理，包括 URL 去重、加入新的 URL、URL 排序等。页面分析模块负责页面内容的处理，数据库用于存放抓取下来的数据。

2.1.2 网络爬虫分类

（1）通用型网络爬虫

通用型网络爬虫工作原理如图 2-1 所示，不断地获取 URL，抓取网页，但是对于 URL 指向的网页不作分析，这种爬虫关注的是爬取效率和覆盖度，而不关心抓取的网页内容。这

种不带目的性的网络爬虫称为通用型网络爬虫。

（2）聚焦型网络爬虫

有时候我们并不像抓取所有的网页，而是只想抓取跟某一主题相关的网页，这种带有目的性的网络爬虫称为聚焦型网络爬虫或是主题相关的网络爬虫。实现该爬虫需要使用一定的网页分析算法过滤与主题无关的 URL，保留有用的 URL 并将其放入待抓取队列。在抓取这些链接指向的网页后，获取新的 URL，再分析过滤这些 URL、加入队列，不断循环直到停止条件。重要的是，所有被爬虫抓取的网页将会被分析、过滤，并建立索引，以方便之后的查询和检索；这一过程所得到的分析结果还可能对以后的抓取过程给出反馈和指导。

（3）分布式网络爬虫

分布式网络爬虫包含多个爬虫，每个爬虫需要完成的一定的爬行任务，它们从互联网上下载网页，并把网页保存在本地，从中抽取 URL 并沿着这些 URL 的指向继续爬行。分布式爬虫可以实现并行爬取、分割任务和容错。并行爬虫需要分割下载任务，爬虫会将自己抽取的 URL 发送给其他爬虫，或者每个爬虫只抓取某个 IP 段的站点信息。这些爬虫可能分布在同一个局域网之中，或者分散在不同的地理位置。

分布式网络爬虫的整体设计重点应该在于爬虫如何进行通信。目前分布式网络爬虫按通信方式不同分布式网路爬虫可以分为主从模式、自治模式与混合模式三种。主从模式是由一个控制节点统一管理各爬虫节点；自治模式则是每个爬虫节点有独立的控制模块，但是没有统一的协调者，因此需要相互间通信；混合模式则是二者的混合。

2.1.3 网络爬虫搜索策略

网络爬虫的爬行策略分为三种:广度优先策略、深度优先策略和最佳优先策略。

（1）广度优先策略

广度优先策略是指网络爬虫在采集网页的过程中，首先抓取最先获得的链接。即从某个起始地址开始，获得该地址的所有 URL 链接，按照先进先出的方式进行循环爬行，直到满足一定的停止条件。这种搜索策略的覆盖率很广，采集的网页比较多。广度优先策略可以由下图表示，数字表示爬行顺序，由上到下表示遍历深度。

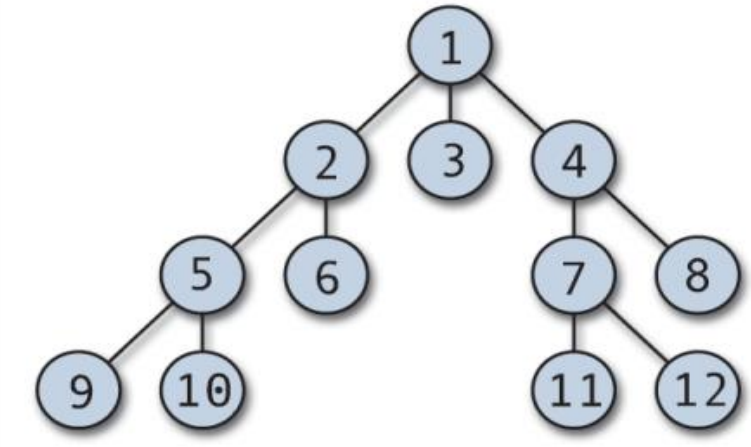

图 2-2 广度优先策略示意图

为覆盖尽可能多的网页，通常使用广度优先搜索方法，避免爬虫陷入问题，倒树型的爬行顺序接近于人们浏览的习惯，具有较好的层次感。

（2）深度优先策略

深度优先策略是指网络爬虫在起始页面开始，一个链接接着一个链接的找下去，处理完一条线路后再转入下一个起始页，继续抽取链接，这样可以对同一个内容的所有页面集中进行处理，避免完整内容被分开采集。用这种采集方式的爬行路径顺序如下图。

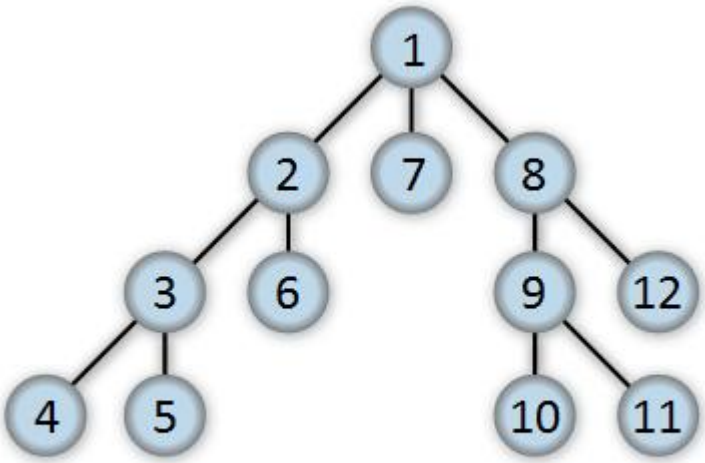

图 2-3 深度优先策略示意图

（3）最佳优先策略

最佳优先策略是按照一定的网页分析算法，预测候选链接地址和目标网页的相似度或者主题相关性，并选取最合适的若干网页进行采集。它只访问经过网页分析算法过滤后的网页。这种策略存在的问题是爬虫在采集路径上会有很多相关的网页被忽略，因为最佳优先策略是一种局部最优的搜索策略，因此需要将此策略结合具体的应用进行改进，跳出局部最优点。解决局部最优点的算法中比较著名的有 FishSearch 算法、遗传算法等。

2.1.4 协议驱动爬虫与事件驱动爬虫

通常网络爬虫获得服务器数据主要依赖于 HTTP 协议，即超文本传送协议（hypertext transport protocol）。该协议是一种无连接、无状态的面向对象协议，具有灵活简单的特点，满足简单数据传输的需要，是 WWW 标准的通信协议。HTTP 协议利用信息头描述事物的处理，HTTP 协议的方法描述了所要执行的动作，常用的 HTTP 协议方法如下:

GET:取回执行的资源。

HEAD:请求服务器传送指定的资源信息，如大小、最后更改时间等。

POST:用于向服务器发送数据，通常是发送 HTTP FORM 内容信息，服务器在得到信息后进行相应的处理。

采用这种方式可以得到网页中的静态信息，但是由 Ajax 技术所生成的动态信息是不能看到的。

对于异步传输的页面，用户的鼠标点击、敲键盘等事件将导致其内容的刷新，通过触发这些事件的方法驱动爬虫，称这种方法为事件驱动。

2.2 页面分析方法

网页中的信息一般以 HTML 文档的形式保存，要在页面中获得自己所需的信息，就需要对 HTML 源码进行分析。以新浪微博为例，页面上显示的一条微博与其在 HTML 文档中的源码如下：

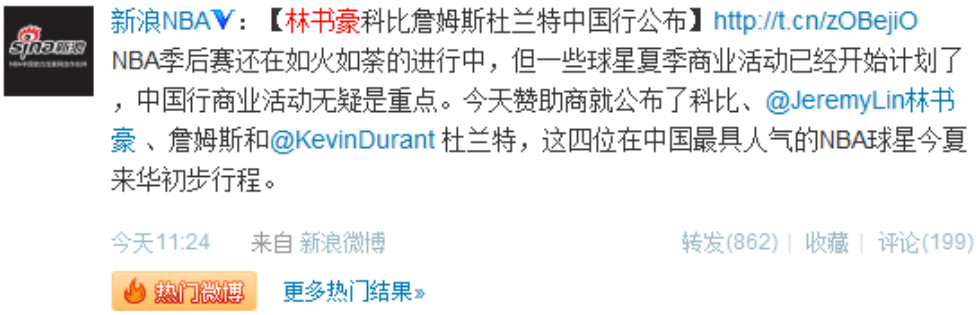

图 2-4 一条微博信息

```
<p node-type="feed_list_content">
    <a usercard="新浪NBA" usercardUid="1736329970" onclick="GB_SUDA._S_uaTrack('tblog_search_v4','weibo_feed_name_h');"
href="http://weibo.com/nbachannel" target="_blank" title="新浪NBA" nick-name="新浪NBA">新浪NBA<img
src="http://img.t.sinajs.cn/t4/style/images/common/transparent.gif?version=201205031835" alt="" class="approve_co" width="11" height="10"/></a>: <em>【<span
style="color:red;">林书豪</span>科比詹姆斯杜兰特中国行公布】<a title="http://sports.sina.com.cn/k/2012-05-26/11126076243.shtml" href="http://t.cn/zOBejiO"
target="_blank" mt="url" >http://t.cn/zOBejiO</a> NBA季后赛还在如火如荼的进行中，但一些球星夏季商业活动已经开始计划了，中国行商业活动无疑是重点。今天赞助商就公布了科比、<a
usercard="JeremyLin林书豪"href="http://weibo.com/n/JeremyLin%E6%9E%97%E4%B9%A6%E8%B1%AA" target="_blank">@JeremyLin林书豪</a> 、詹姆斯和<a
usercard="KevinDurant"href="http://weibo.com/n/KevinDurant" target="_blank">@KevinDurant</a> 杜兰特，这四位在中国最具人气的NBA球星今夏来华初步行程。</em>
</p>
```

图 2-5 微博对应的 HTML 源码

要想用爬虫获得该微博的信息，只需在网页中提取<p>标签的内容，并分析里面的内容。一般来说，页面分析的方法分为基于正则式的方法和基于文档结构的方法。

2.2.1 基于正则式的方法

正则表达式是指一个用来描述或者匹配一系列符合某个句法规则的字符串的单个字符串。目前很多编程语言都支持正则表达式。在使用正则式进行页面分析时，页面信息一般以字符串的形式保存，用户首先通过构造正则式来表示需要抓取的信息，然后通过对页面信息进行匹配，找到页面中所需的字符或字符串。

以 Python 为例，下表列出了 Python 常用的正则表达式：

表达式	含义
.	匹配除换行符外任意字符
\w	匹配字母或数字或下划线或汉字
\s	匹配空格
\d	匹配数字
\b	匹配单词开始或结束
$	匹配结束符
*	0 次或多次出现
+	1 次或多次出现
?	贪婪模式
[a-z]	匹配 a 到 z 之间的任意字母
[^a]	匹配除 a 以外任意字符
{n}	重复 n 次
{n,m}	重复 n 到 m 次

表 2-1 常用的正则表达式

例如，要获取网页源码中<title>“某标题”</title>这部分内容，可以编写正则式<title>（[\s\S]*?）</title>进行匹配，其中[\s\S]匹配任意字符。通过对整个网页的 HTML 源码进行遍历和匹配，就可以找到所需的内容。

采用正则式的方法，十分灵活，只要对网页源码熟悉就一定可以精确匹配到需要的内容。缺点在于，正则式的构造方法的好坏直接影响了网页分析的效率，而且在构造正则式前，必须对网页源码有比较深入地分析。

2.2.2 基于文档结构的方法

该方法首先将网页的 HTML 源码进行解析，生成 DOM 树，该信息作为一个标签元素将会成为 DOM 树上的节点，通过 DOM 操作找到该节点，从而可获得相应信息。DOM 的相关知识会在 2.4.2 节进行介绍。

2.3 Ajax 与 Web2.0

2.3.1 Web2.0

Web2.0 是相对 Web1.0 的新的一类互联网应用的统称。Web1.0 的主要特点在于用户通过浏览器获取信息。Web2.0 则更注重用户的交互作用，用户既是网站内容的浏览者，也是网站内容的制造者。在模式上由单纯的“读”向“写”以及“共同建设”发展；由用户被动地接收互联网信息向主动创造互联网信息发展，从而更加人性化。典型的 Web 2.0 应用有微博、博客、社交网络、P2P 等。

Web 2.0 具有以下特点：

（1）共同建设

Web1.0 里，互联网内容是由少数编辑人员定制的，比如各门户网站；而在 Web2.0 里，每个人都是内容的供稿者。

（2）以人为本

在互联网的新时代，信息是由每个人贡献出来的，互联网信息源由每个用户共同组成。Web2.0 的灵魂是人。

（3）可读可写

在 Web1.0 里，互联网是“阅读式互联网”，而 Web2.0 是“可写可读互联网”。网页内容可以由用户创造、定制。

项目	web 1.0	web 2.0
页面风格	结构复杂，页面繁冗	页面简洁，风格流畅
个性化程度	垂直化、大众化	个性化突出自我品牌
用户体验程度	低参与度、被动接受	高参与度、互动接受
通讯程度	信息闭塞知识程度低	信息灵通知识程度高
感性程度	追求物质性价值	追求精神性价值
功能性	实用追求功能性利益	体验追求情感性利益

表 2-2 Web1.0 与 Web2.0 的比较

2.3.2 Ajax 技术简介

Ajax 是异步 JavaScript 和 XML 技术的缩写。最早应用这种技术的网站无从考察，但是

最早提出这个概念的是 Garrett，他在文章 AJAX:A New Approach to Web Application 中首次使用了 Ajax 这种称呼。此技术改变了传统的客户端与服务器的工作方式，使用户在浏览 Web 网页的同时，无须等待数据刷新所带来的白屏界面，而是继续执行其他操作，所有的数据处理都在后台进行。组成这种技术的要素有 JavaScript 脚本语言、CSS 样式表、XMLHttpRequest 数据交换技术和 DOM 文档对象(或者 XMLDOM 文档对象)。

Ajax 主要实现的就是局部刷新和异步读取数据。要实现局部刷新，就必须控制浏览器的行为和内容，这正是 JavaScript 的特色。JavaScript 是 Ajax 技术的主要开发语言，其主要功能是驻留在客户端，使浏览器与用户的交互成为可能，这样网页就不再是简单的静态网页了。JavaScript 语一言属于解释型语言，它在运行前不需要编译，脚本解释器会逐行解释客户端中的 JavaScript 语一言，因为是逐行解释，所以在应用对象时必须已经对它进行过声明，否则系统会提示找不到对象。

Ajax 技术在实际应用中并不仅仅是这四种元素，但这是最主要的部分，而且并不一定要全部应用这些技术才算是 AJAX。AJAX 的主要作用是异步获取后台数据和局部刷新技术。下面描述了为 Ajax 网页的一个交互过程。

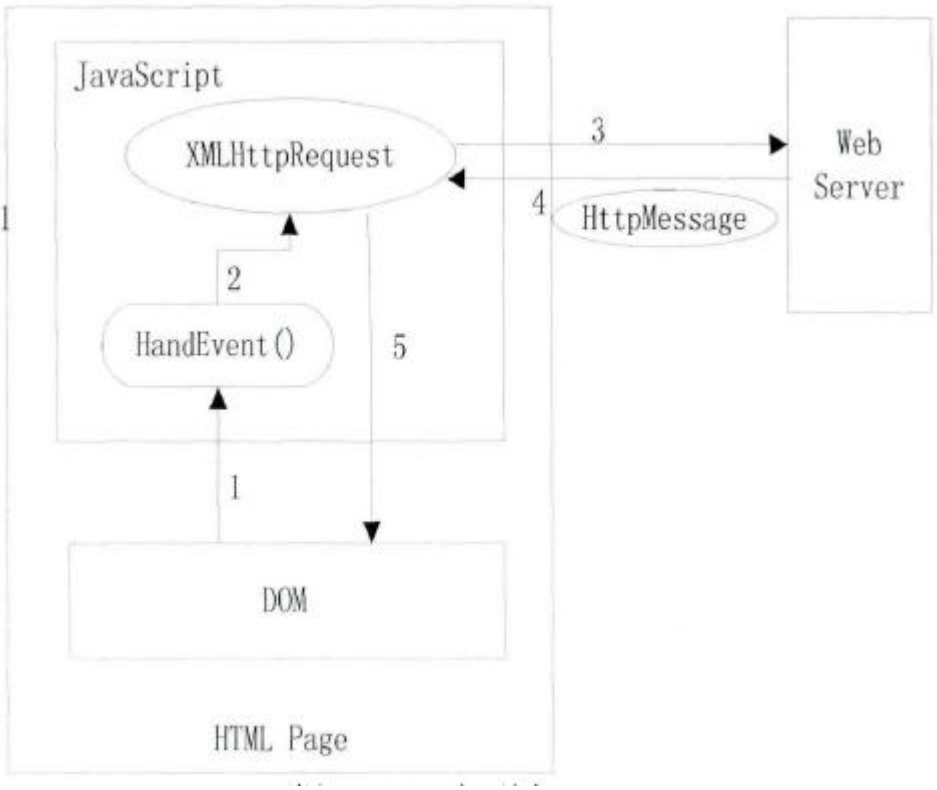

图 2-6 Ajax 工作原理

该交互图的各阶段工作如下:

(1)当对 HTML 网页上的某个元素执行一定的操作后，会激活页面的 JavaScript 函数。

(2)在 HandEvent 这个函数中，会创建一个 XMLHttPRequest 对象的实例(该例子是基于 IE 浏览器介绍的)。

(3)XMLHttpRequest 对象包含了 XML 信息和 HTML 网页一些状态信息，将这些信息发给了服务器。

(4)发送请求之后，XMLHttPRequest 对象就会监听 Web 服务器返回的的信息。

(5)接收到返回信息之后，浏览器会执行这些函数，往往会通过网页的 DOM 结构显示出

新的信息。

应用了 AJAX 技术后，一方面服务器不需要存储大量的静态网页，而是在用户进行一定的操作时返回，节约了存储空间，提高了灵活性;另一方面，大量 AJAX 技术的应用使得浏览器与服务器进行交互时，用户仍可以对浏览的页面进行操作，增强了用户的使用体验。

2.3.3 Ajax 对网络爬虫带来的挑战

应用 Ajax 技术给网络爬虫带来的困难可归结为以下几点:

（1）动态信息的采集

网页中显示的动态信息在该网页源文件的 HTML 中无法查看到，如何获取网页上显示的动态信息是首要解决的问题。

（2）用户与服务器的交互

Ajax 网页信息通过客户端动作，引起异步交互，网页信息发生多次变化。若要查看该地址下其他信息，需要对网页元素进行操作，引起异步交互，获得动态信息。

（3）Ajax 网页中的触发事件

每个 Ajax 网页都存在多个触发元素，例如:点击网页中的“下一页”元素可以查看下一页信息，通过鼠标的操作完成状态的变化等。若要采集该站点的所有动态信息，需要对 Ajax 网页的这些元素及相应的动作进行管理。

（4）状态的标识

在得到一个 Ajax 网页后，对该网页的某些元素进行点击等操作，网页信息会发生改变。这样在同一个 URL 下，Ajax 网页的信息会多次改变，使用 URL 作为信息标识的传统情况就不再适用于 Ajax 网页。

2.3.4 JavaScript 在网页中的使用

JavaScript 在网页中的大量使用，是 Ajax 技术的重要特征，也使得传统爬虫难以提取隐藏在这些动态脚本内的信息。由 JavaScript 脚本的灵活性和复杂性，网页中使用 JavaScript 的方式和目的也不尽相同。

一般来说，JavaScript 脚本描述了网页的动态信息，例如客户端事件的响应和处理，如 onclick 事件的响应;如何与服务器端进行交互动态下载信息；如何动态构建 DOM 等等。同时，一些重要的链接、页面内容也是隐藏于 JS 脚本中。因此，如何解析 JS 脚本，获取动态信息成为网络爬虫设计的重要课题。

JavaScript 应用到网页的方式也有很多种，常见的方式分为:外部引用、脚本标签和内联引用方式。下面对这三种方式解释如下:

（1）JavaScript 外部文件

网页源代码可以直接引用 JavaScript 外部文件，该文件的扩展名通常是.Js。在这类外部文件中，通常只包含 JavaScript 脚本文件，而不包含<script>等网页标签。浏览器在执行这些文件时，自动加载并缓存其中的代码。以这种方式调用 JavaScript 代码实现简单并且易于修改。网页中嵌入方式通常如下所示:

```
<script src="../js/execute.js"></script>
```

（2）脚本标签方式

脚本标签嵌入 JavaScript 代码的方式往往应用于该脚本不需要重用的情况，该脚本在源文件中很容易识别，都是出现在<script></script>中。这种方式的一般模式如下:

```
<script type="text/JavaScript">
//脚本代码
<script>
```

（3）内联引用方式

内联引用是通过 HTML 标签中的事件属性实现的。通过 onclick 等 HTML 事件属性调用

JavaScript。这种方式是目前 Ajax 网页中应用较多的一种，也是本系统所要重点解决的问题。如下面例子:在该标签中有对网页元素的触发动作以及脚本函数。

〈input type=“button” value=“submit” onclick=“do something();” /〉

当对该元素进行点击动作后，系统会自动的执行这些 JavaScript 代码。

2.4 浏览器工作原理

2.4.1 浏览器简介

浏览器浏览器是提供 Web 服务的一种客户端软件程序。它向网络服务器发送各种请求，并对从服务器返回的超文本信息和各种多媒体数据格式进行解释、显示和播放。

浏览器的主要组件包括：

用户界面： 包括地址栏、后退/前进按钮、书签目录等，也就是你所看到的除了用来显示你所请求页面的主窗口之外的其他部分

浏览器引擎： 用来查询及操作渲染引擎的接口

渲染引擎： 用来显示请求的内容，例如，如果请求内容为 HTML，它负责解析 HTML 及 css，并将解析后的结果显示出来

网络： 用来完成网络调用，例如 http 请求，它具有平台无关的接口，可以在不同平台上工作

UI 后端： 用来绘制类似组合选择框及对话框等基本组件，具有不特定于某个平台的通用接口，底层使用操作系统的用户接口

JS 解释器： 用来解释执行 JS 代码

数据存储： 属于持久层，浏览器需要在硬盘中保存类似 cookie 的各种数据，HTML5 定义了 web database 技术，这是一种轻量级完整的客户端存储技术。

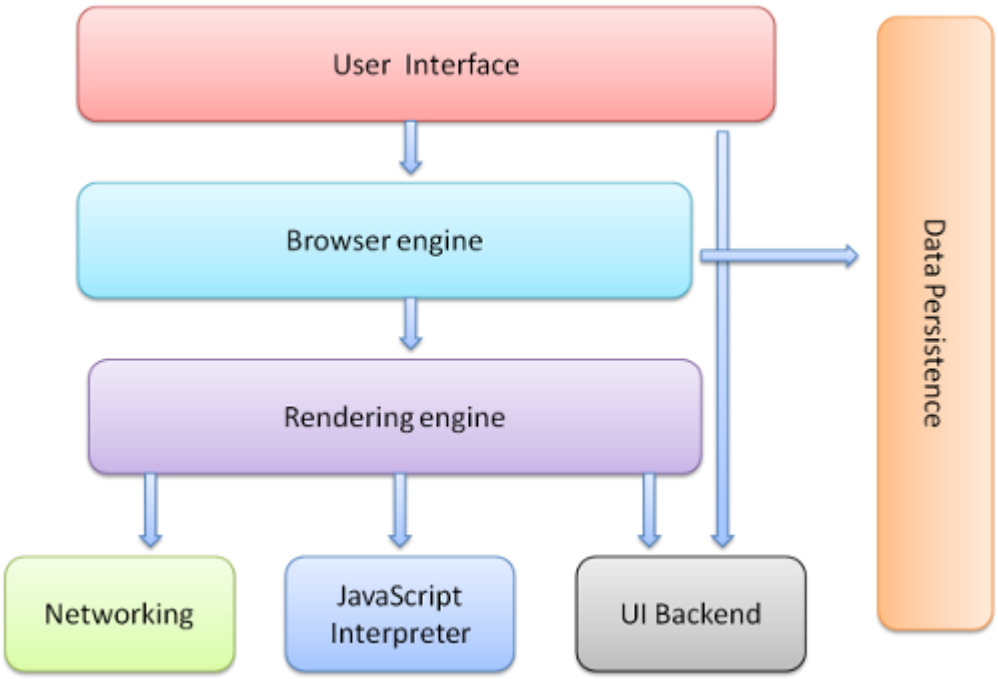

图 2-7 浏览器架构

2.4.2 DOM

DOM（Document Object Model）即文档对象模型。是一种面向对象的方式，对文档进行描述和修改的通用方法。它可以独立于平台和语言，针对任意文档的内容和结构进行访问

和修改。DOM 是表示 HTML 和 XML 的常用方法，它将 HTML 文档抽象成对象模型，定义了文档中有哪些对象，这些对象的关系，对象的属性和修改对象的方法。通常认为 DOM 将页面上的数据组建成树形结构，这样使得页面可以动态的变化，比如插入元素，修改元素，修改元素属性等。当 DOM 树构建完成后，不同的对象节点处于不同的层次，通过根节点向下遍历或者从某一节点利用父子关系进行推导，即可定位指定节点，并进行操作。

常见的 DOM 对象包括文档、元素、节点、属性、文本节点等。

文档（Document）：文档可以看作 DOM 树的根。提供了访问和操作 HTML 文档所有节点的接口的方法。

节点（Node）：节点是一般类型，它涉及一个文档中存在的所有对象。

元素（Element）：元素是 DOM 的原子节点，这些元素在文档中的布局形成了 DOM 树的结构。元素间可以存在父子关系，元素具有属性。

属性（Attribute）：属性不属于文档树的一部分，只是描述元素的基本属性。

文本节点（Text Node）：文本节点处理文档中的文本。

DOM 和标签基本是一一对应的关系，例如，如下的标签：

```
<HTML>
 <body>
    <p>
     “我是文本”
    </p>
    <div><img src=” example.png” /></div>
 </body>
</HTML>
```

将会被转换为下面的 DOM 树：

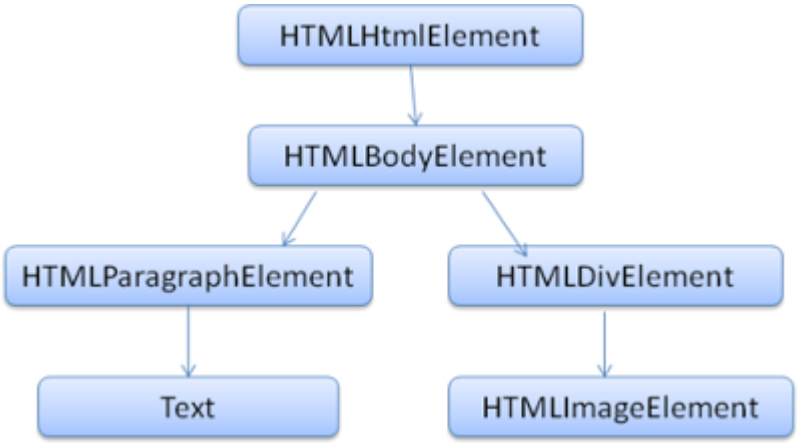

图 2-8 DOM 树

2.4.3 浏览器工作过程

渲染引擎首先通过网络获得所请求文档的内容，然后解析 HTML，并将标签转化为内容树中的 DOM 节点。接着，它解析外部 CSS 文件及 style 标签中的样式信息。这些样式信息以及 HTML 中的可见性指令将被用来构建 render 树。

Render 树由一些包含有颜色和大小等属性的矩形组成，它们将被按照正确的顺序显示到屏幕上。

Render 树构建好了之后，将会执行布局过程，它将确定每个节点在屏幕上的确切坐标。再下一步就是绘制，即遍历 render 树，并使用 UI 后端层绘制每个节点。

值得注意的是，这个过程是逐步完成的，为了更好的用户体验，渲染引擎将会尽可能早的将内容呈现到屏幕上，并不会等到所有的 HTML 都解析完成之后再去构建和布局 render 树。它是解析完一部分内容就显示一部分内容，同时，可能还在通过网络下载其余内容。

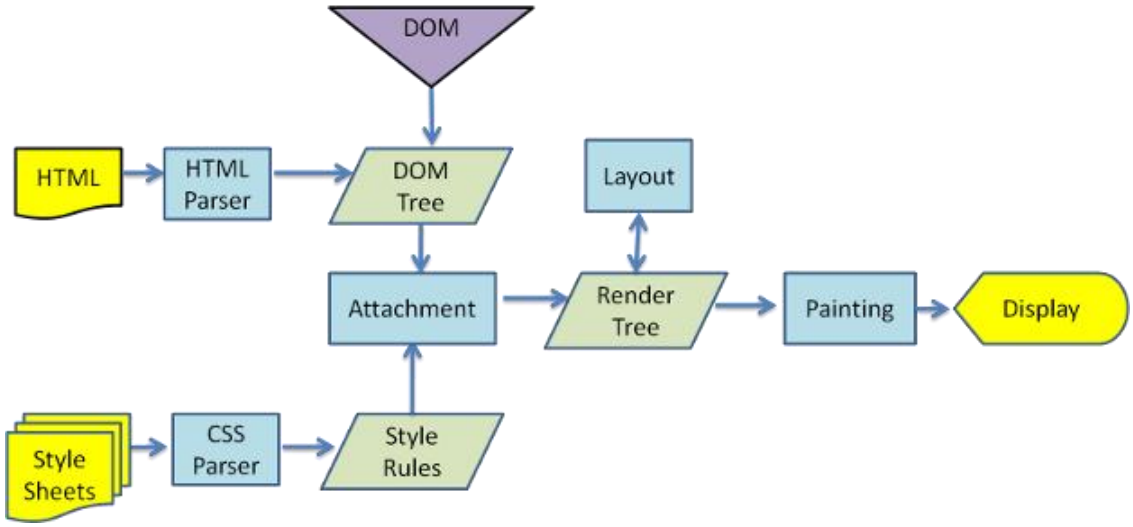

图 2-9 浏览器工作流程

2.4.4 浏览器内核

浏览器内核，又称为排版引擎（Layout Engine）或页面渲染引擎（Rendering Engine），负责对网页语法的解释（如 HTML、JavaScript）并渲染（显示）网页。是浏览器最核心的部分。它决定了浏览器如何显示网页的内容以及页面的格式信息。

不同的浏览器内核对网页编写语法的解释是不同的，因此基于不同浏览器内核的流览器里网页的显示效果也可能不同。主流的浏览器内核有 Trident、Gecko、Presto、WebKit 和 KHTML。

常见的浏览器与其使用的内核如下图所示：

浏览器	浏览器内核
Chrome	WebKit
Internet Explorer	Trident (Win)、 Tasman (Mac)
Konqueror	KHTML (可选装 Gecko)
Maxthon	WebKit，Trident
Mozilla Application Suite	Gecko
Mozilla Firefox	Gecko
Netscape	Gecko
Netscape Browser	Gecko, Trident
Netscape Communicator	Mariner
Netscape Navigator	Mariner
Netscape Navigator 9	Gecko
OmniWeb	WebKit

Opera	Presto
Safari	WebKit
世界之窗浏览器	Trident
QQ 浏览器	Trident
搜狗浏览器	Trident & WebKit

表 2-3　不同浏览器使用的浏览器内核

2.4.5 WebKit

WebKit 是一款开源的浏览器内核。它主要由两个模块构成：WebCore 和 JSCore，WebCore 负责页面的排版及 HTML 解析、DOM 等，是 WebKit 的核心，JSCore 主要负责 JS 的解析。

WebCore 的主要功能有：

- Page：与外框相关的内容（Frame，Page，History，Focus，Window）。
- Loader：加载资源及 Cache。
- HTML：DOM HTML 内容及解析。
- DOM：DOM CORE 内容。
- XML：XML 内容及解析。
- Render：排版功能。
- CSS：DOM CSS 内容。
- Binding：DOM 与 JavaScriptCore 绑定的功能。
- Editing：所有与编辑相关的功能。

JSCore 的主要功能有：

- API：基本 JavaScript 功能。
- Binding：与其他功能绑定的功能，如 DOM、C，JNI。
- DerviedSource：自动产生的代码。
- ForwordHeads：头文件，无实际意义。
- PCRE：Perl-Compatible Regular Expressions（Perl 兼容的规则表达式）。
- KJS：JavaScript 内核。
- WTF：KDE 的 C++模板库。

WebKit 体系结构如下图所示：

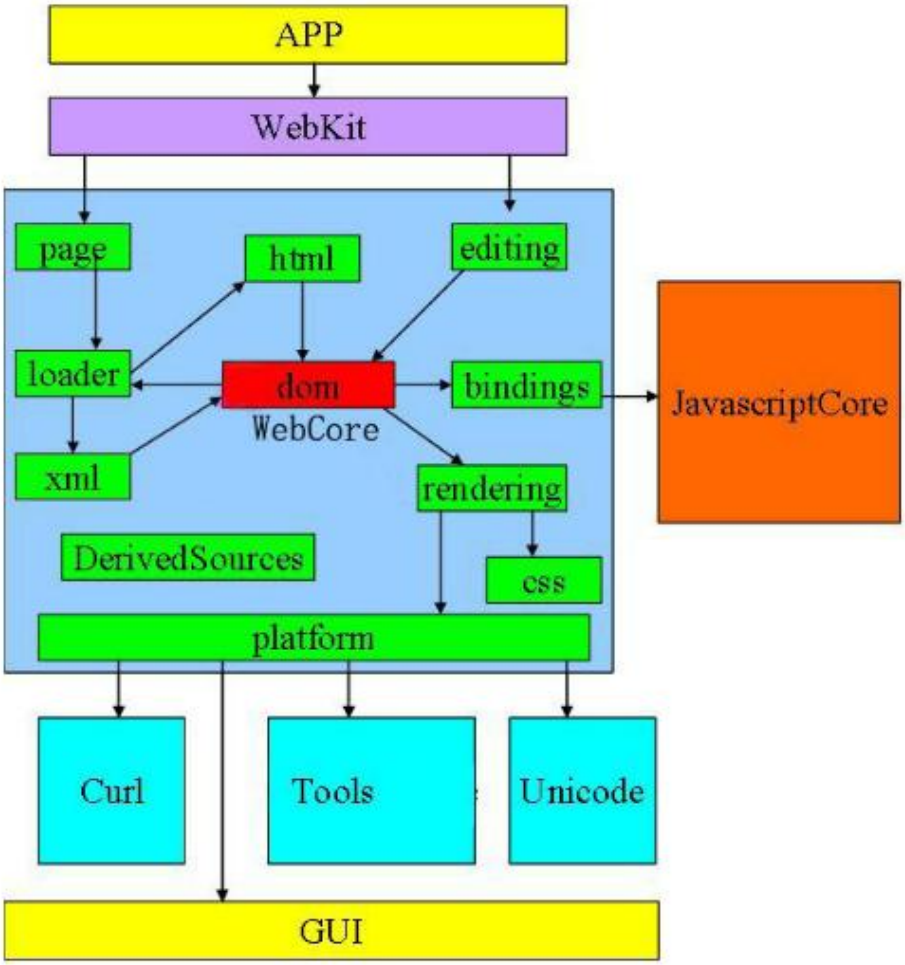

图 2-10 WebKit 的系统架构

WebCore 排版引擎和 JSCore 引擎的前身分别是 KDE 的 KHTML 和 KJS，苹果公司在比较了 Gecko 和 KHTML 后，选择了后者，就因为它拥有清晰的源码结构、极快的渲染速度。同样广为流行的 Android 系统也毫不犹豫地选择了 WebKit。它具备触摸屏、高级图形显示和上网功能，用户能够在手机上查看电子邮件、搜索网址和观看视频节目等。可以看出这是一个非常强大的 Web 应用平台。

2.5 支持 Ajax 的网络爬虫设计方法

2.5.1 基于 Web 自动化框架的方法

使用基于浏览器的方法设计爬虫是目前抓取 Ajax 网站比较流行的方法，浏览器本身具有 HTML 和 JavaScript 解析、页面渲染和 DOM 处理的功能，用户在使用浏览器访问页面时，无需关心隐藏在页面后的 JavaScript 脚本，也无需关心客户端与服务器端的 AJAX 异步交互方式。

WebDriver 就是一款 Web 自动化框架，它可以与大部分主流的浏览器如 IE、Firefox、Chrome 等进行绑定。使用该框架设计爬虫，可以模拟用户行为，例如在输入框输入数据，或是点击按钮，而无需了解这些元素背后的代码和执行过程。

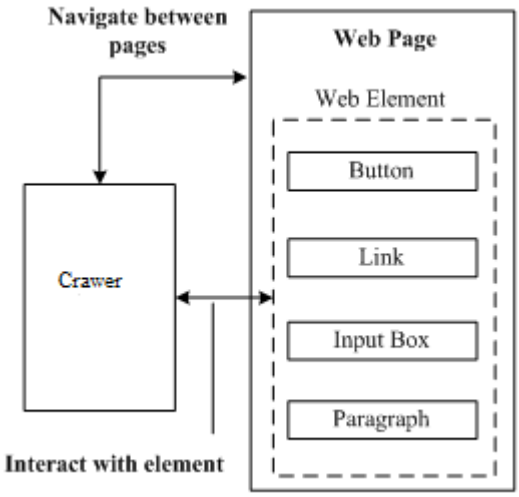

图 2-11 WebDriver 的系统架构

2.5.2 基于浏览器插件的方法

基于浏览器插件的方法与基于 Web 自动化框架的方法类似，都是基于浏览器的，因此无需关心 JS 脚本解析过程，只是直观地操作浏览器。

以 Chickenfoot 为例，它是一款基于 Firefox 的编程系统，它提供了自己的脚本语言，用户可以以此编写脚本操作浏览器。这套系统可以用来做网络自动化、网络爬虫等一系列任务。它的一大优点是，用户在编写脚本时甚至无需查看网页的源码即可对页面元素进行定位和操作，即无需查看网页的 HTML 源文件。

图 2-12 Chickenfoot

2.5.3 模拟浏览器的方法

通过编程模拟浏览器的行为来设计爬虫是一种比较原始方法，该方法通过发送请求，获得服务器端数据后进行解析。该方法需要人工编程来模拟浏览器的一步步的解析工作，包括对 HTML 源码进行解析，生成 DOM，JS 脚本解析等。这些工作十分繁琐，实现起来非常麻烦，优点是该方法开发的爬虫效率最高，占用资源最少。

2.5.4 基于浏览器内核的方法

浏览器内核是浏览器最重要的部分，负责对网页语法的解释（如 HTML、JavaScript）并渲染（显示）网页，它负责解释 Web 页面上所有的代码，包括 HTML 标签、JavaScript 脚本、CSS 样式等。

基于浏览器内核的方法相对于基于浏览器的方法而言，优点在于速度快、占用资源少。

2.5.5 各种方法的比较

对于以上四种实现支持 Ajax 爬虫的方法，各自的优缺点比较如下表：

	模拟浏览器	**基于浏览器内核**	**基于浏览器插件**	**基于 Web 自动化框架**
图形界面	无	无	有	有
实施层次	低（需手动处理底层的 HTTP 请求）	中（需手动操作 DOM 和 JS 解析）	高（只需对页面元素进行操作）	高
模拟行为	模拟浏览器	模拟浏览器	模拟用户	模拟用户
远程/本地	本地	本地	本地	远程+本地
是否需查看 HTML	是	是	否	是
性能	快	快	慢	慢
Cookies 处理	手动	手动	自动	自动+手动
内存消耗	低	低	高	高

表 2-4 爬虫实现方法比较

我选择基于浏览器内核的方法开发我的爬虫系统，因为它具有较高的性能、较低的内存消耗，并且支持 DOM 操作和 JavaScript 解析。

第三章 基于 WebKit 的可编程命令行网络爬虫系统实现

3.1 设计目的

由于 Ajax 技术和 JavaScript 脚本在网页中的大量运用，传统的基于协议驱动型的网络爬虫已经很难获取网页中的动态信息。考虑到基于浏览器的方法耗费资源、效率低，而脱离浏览器的方法需要大量编程工作来模拟浏览器的页面解析渲染过程，实现起来十分繁琐。基于 WebKit 开发这套系统，可以保证爬行效率，同时，设计了自己的编程语言，使得无需使用复杂的 WebKit 的 API 进行编程，而是可以像操作浏览器一样，直观、方便的设计自己的爬虫脚本，简化了终端用户的工作，提供了更为人性化的交互方法。

3.2 系统功能

该系统提供一套独立的编程语言，供用户编写爬虫或是网络自动化测试脚本。

该系统采用了 PhantomJS 的框架，支持 JavaScript 解析、DOM 操作、CSS 和 JSON。

该系统提供 DOM 元素的定位方法，支持模拟鼠标点击、输入文本页面跳转等操作，使得用户在设计爬虫时可以模拟人与浏览器的交互过程进行动态信息的采集而无需关心页面背后的 JavaScript 脚本和 Ajax 异步交互过程。

由于可以模拟人与浏览器的交互，可以轻松实现填写表单、登陆等操作，该系统也可以做为无图形界面的网络自动化测试工具。

3.3 系统总体架构

系统底层使用 PhantomJS 的框架,PhantomJS 本身是 WebKit 和 JavaScript 引擎的结合，同时提供了一套基于 JavaScript 的 API，来对 WebKit 和 JavaScript 引擎进行操作。顶层是用户输入模块，用户通过使用该系统自定义的脚本语言编写爬虫脚本，以文本或者命令行的方式进行输入。中间层是一个解释器，负责将用户的脚本翻译成 PhantomJS 支持的脚本语言，并交由 PhantomJS 运行。具体架构如下图：

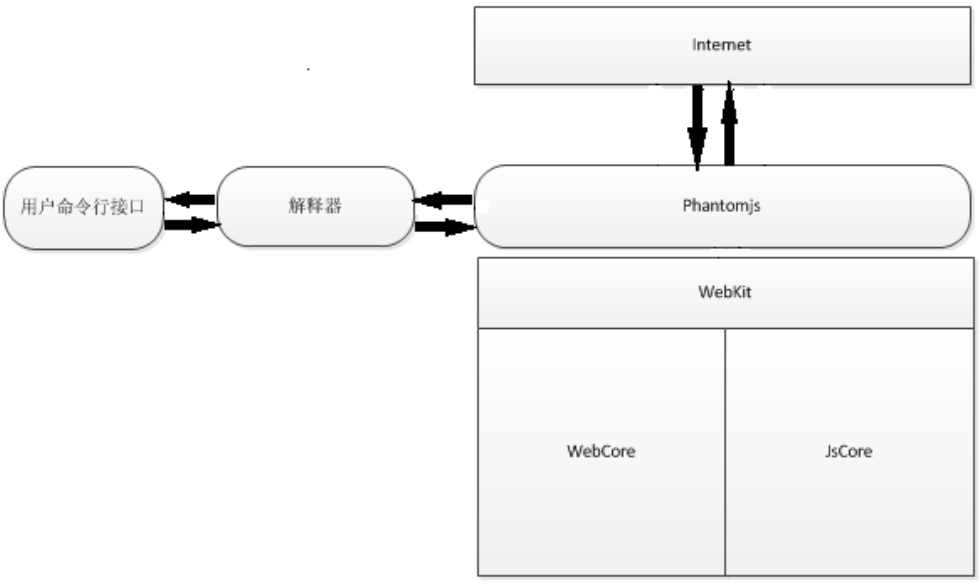

图 3-1 系统架构图

3.4 系统工作流程

系统工作流程如下图所示：

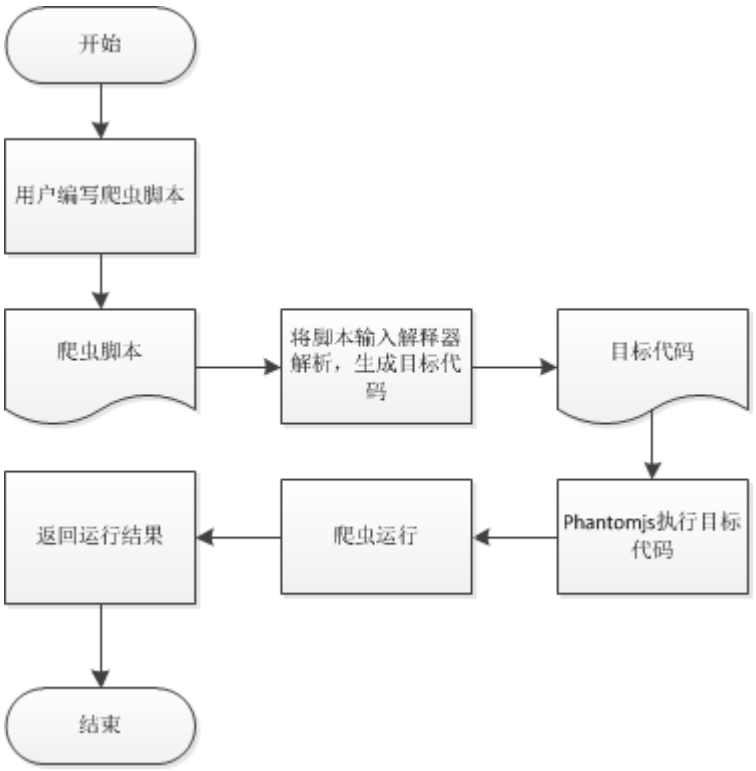

图 3-2 系统的工作流程

3.5 系统详细设计

3.5.1 PhantomJS 简介

该系统底层使用 PhantomJS 的框架，对 WebKit 进行操作。

PhantomJS 是一套基于 Javascript 驱动的命令行 WebKit 引擎，具有轻量级，开发快速，渲染速度较快的特点，提供了一套基于 JavaScript 的应用程序编程接口(API)来操作 WebKit 和 JS 引擎，以实现 DOM 操作、JavaScript 脚本解析及各种网络任务。

PhantomJS 常用的 API 可以分为以下几类：

1) 导入 JS 库
2) 网页导航
3) 解析 JS 脚本
4) 渲染页面
5) 触发事件
6) 上传文件
7) 文件系统操作

通过使用这些 API 结合 JavaScript 语言进行编程，可以实现一系列网络自动化任务，如网页测试，网络爬虫，流量监控等。

3.5.2 PhantomJS 接口的封装

虽然 PhantomJS 提供了很多实用的 API 供我们编写脚本操作 WebKit，但是所需的编程工作还是十分麻烦的。例如，使用 PhantomJS 打开 google 主页，需要编写以下脚本：

```
var page = new WebPage(),
    url = 'http://www.google.com';
page.open(url, function(status) {
    console.log(status);

});
```

图 3-3 PhantomJS 脚本示例

这样的编程工作显得比较繁琐，而我理想中的方法是直接输入 go（'http://www.google.com'），即可实现打开 google 主页。

另外，有些 API 不经封装则无法满足爬虫所需的功能，例如 PhantomJS 提供了 SendEvent（参数一，参数二，参数三）的 API 以模拟事件输入。其中第一个参数指定了事件的类型，如 mousemove 或者 click 等，后两个参数则是坐标。也就是说，该方法可以向网页上某一坐标上的元素发送事件。然而，在设计爬虫的过程中，用户是无法直接知道页面上某元素的坐标的，需要使用 jQuery 通过 DOM 操作定位到元素，再获得元素的相对偏移(offset)以后，才能得到坐标。可见 PhantomJS 的 API 对于爬虫的设计并不友好，编写程序也比较麻烦。

解决方法是一方面我放弃了 PhantomJS 提供的的语言作为终端用户的编程语言，而是自己设计一套简洁直观的语言；另一方面我将 PhantomJS 的 API 进行完善和再次封装，将爬虫所需的功能如打开某个链接，模拟鼠标点击，模拟键盘输入等封装成一个个函数。每个函数负责实现一个功能。当用户输入指令时，只需调用相应的函数即可。

3.5.3 使用 jQuery 操作 DOM

本系统的设计中，DOM 操作主要是 DOM 元素的定位和修改。

支持 DOM 操作是设计抓取 AJAX 网站爬虫的重要前提。网页内容可以看作 DOM 元素的组合。例如新浪微博的登陆界面，登陆按钮、用户名密码的输入框分队对应不同的 DOM 元素，在 HTML 文档中可以找到这些元素对应的源码。如下图：

图 3-4 页面元素与 HTML 源码的对应关系

要定位这些元素，常用的方法是根据该元素的类名（Class Name）、名称（Name）或者是 Id，如上图，如果想要定位登陆按钮，则只需遍历 DOM 树，找到类名为“W_btn_d”的元素即可。

我使用 jQuery 进行 DOM 对象的定位和修改。 jQuery 是一个优秀的 Javascript 框架。它是轻量级的 js 库。兼容 CSS3，还兼容各种浏览器 （IE 6.0+, FF 1.5+, Safari 2.0+, Opera 9.0+）。jQuery 使用户能更方便地处理 HTML documents、events、实现动画效果，并且方便地为网站提供 Ajax 交互。

使用 jQuery 最重要的原因是它的对象选择器十分灵活好用，页面的任何元素都可以简单的使用 jQuery 查询语句找到。

以下是使用 jQuery 进行元素定位的几个例子：

假设页面代码如下：

```
 <div class="class1">
<div id="id1">aa</div>
<div id="id2"><span>bb</span></div>
<div id="id3"><label>cc</label></div>
</div>
```

根据 id 进行定位, 如: $("#id3") ，找到第四行 id = “id3”的对象。

根据标签选择, 如: $("label")，找到第四行标签为 label 的对象

根据样式的类名选择, 如: $(".class1") 找到第一行类名为 class1 的对象

根据父子关系选择，如 $(".class1 > span") 找到类名为 class1 的子节点中标签为 span 的对象。

还有很多的选择方法，在此不一一列举，总之使用 jQuery 定位 DOM 对象，非常的灵活、方便。

在定位好元素以后，即可对这些元素进行操作，实现交互。同样是新浪微博的例子，定位好输入用户名、密码的文本框之后，即可对这两个元素进行赋值，相当于输入用户名密码。使用 jQuery 对元素进行赋值也非常简单。使用如下语句即可：

```
$(.loginname).val(“account”)
$(.pass).val(“password”)
```

之后，需要点击登录按钮进行登录，这需要定位登陆按钮以后，再配合 PhantomJS 的 SendEvent 的方法，进行模拟鼠标点击的操作，即使用如下语句：

```
Offset = $(.W_btn_d).offset();
X= offset.left;
Y =offset.top;
SendEvent(click,X,Y);
```

前三句对按钮进行定位，并获得元素的坐标，第四句向该坐标发送鼠标单击的动作。

3.5.4 JavaScript 解析

JavaScript 脚本的解析，是爬虫抓取 AJAX 网站另一个需要解决的技术问题。PhantomJS 的框架自带了 JS 引擎，可以在下载 HTML 代码的同时，解析代码中的 JS 脚本。但是用户在编程时仍需手动的指定什么时候需要使用 JS 引擎解析 JavaScript 脚本。

PhantomJS 提供的 API 中，使用 Page.evaluate()的函数，可以对页面中的 JavaScript 脚本进行解析,并动态生成 DOM。

3.5.5 语言设计

该语言设计的目的在于能够使用简洁、直观地方式模拟人与浏览器的交互，实现网络爬取任务。该语言包括一些基本指令、简单的赋值语句、条件判断语句、while 循环等。并不完善，但是能满足大部分网络爬虫的设计要求。该语言使用方便、功能丰富、有较好的可扩展性。

由于该语言是针对网络爬虫以及一些网络自动化任务的，因此设计的指令主要是页面导航、内容获取以及模拟用户行为。

语言详细设计如下：

基本指令：

go()

指令功能:页面导航

参数：只接受单参数

指令描述：无

示例： Go（http://www.google.com） //访问谷歌主页

click()

指令功能:模拟鼠标点击页面上某元素

参数：无参数，单参数，双参数

指令描述：该指令会模拟鼠标点击网页某个元素的事件。无参数时会启动自动匹配模式，该模式会在 3.5.7 介绍。单参数时如果是 jQuery 的查询语句，即以“#”或“.”开头的参数，则会在 HTML 源码中定位相应元素；否则启动自动匹配模式。双参数必须为（x,y）坐标的形式。

示例 1： click（）#自动匹配页面上某个元素并点击

示例 2： click（google 搜索）//点击页面上显示为“google 搜索的元素”

示例 3： click（$#button）//点击在 HTML 源码中，id =“button”的元素

示例 4： click（$.button）//点击在 HTML 源码中，class =“button”的元素

示例 5： click（625,1000）//点击在页面上坐标为（625,1000）的元素

enter()

指令功能:模拟键盘输入数据

参数：单参数或双参数

指令描述：该指令会模拟键盘向某个输入框输入数据事件。单参数时启动自动匹配模式。双参数时，前一参数是输入的内容，后一参数是待输入按钮的定位方式。

示例 1： enter(password)//自动匹配到某个输入框，并输入数据“password”

示例 2： enter(password，$#inputbox)//找到 id = “inputbox”的元素，并输入“password”

find()

指令功能:元素定位

参数：单参数

指令描述：找到某个元素，一般与赋值语句、click、enter 配合使用

示例：m=find($#login_button)

m.Click()//定位按钮并点击

fetch()
指令功能:抓取当前页面源码
指令描述：抓取结果以 txt 格式保存至当前目录
参数：无
示例：go(“http://www.google.com”)
fetch() //抓取谷歌主页的源码

prtsc()
指令功能:保存当前页面
参数：无参数，单参数，双参数
指令描述：目前只支持将抓取的页面以 pdf 或 png 的格式输出。无参数默认 png 格式保存，单参数选择保存格式，双参数选择保存格式和保存路径。
示例：prtsc(pdf, C:\Users\admin\Desktop\result) //将当前页面以 pdf 格式保存至指定文件夹。

set_windowsize()
指令功能:设置窗口大小
参数：双参数
指令描述：设置虚拟浏览器的窗口大小，默认为 1366*768。
示例：set_windowsize（1366,4000） //将窗口大小设为宽：1366 高：4000

sleep()
指令功能:睡眠
参数：单参数
指令描述：使爬虫暂停一段时间，单位：毫秒。
示例：sleep(1000) //休眠 1000 毫秒

end()
指令功能:结束标识符
参数：无

基本的赋值语句：
示例： content= fetch()

条件判断语句：
示例：

```
If (condition a)
{
   Go（“http://www.google.com.hk”）
  }
else
```

```
{
    Go（“http://www.baidu.com”）
}
```

循环语句：

示例：

```
While(True）
{
click(“下一页”)
}
```

使用该语言实现网络爬虫非常方便，以登陆人人网，获取个人主页信息为例。只需编写以下脚本即可：

```
go("http://www.renren.com")        //访问人人网
enter(account)                     //输入用户名
enter(password)                    //输入密码
click(登录人人网)                   //点击登陆按钮
fetch()                            //抓取内容
end()                              //结束
```

使用基本指令一般可以完成简单网络爬虫的设计，引入了赋值语句、条件判断语句和循环语句是为了处理可能会遇到的复杂任务。事实上，该语言目前还很不成熟，仍有待进一步的设计和完善。

3.5.6 解释器设计

设计完语言之后要做的是为语言和底层的PhantomJS搭建起一座桥梁。因此我需要设计一个解释器将用户自定义的脚本翻译成PhantomJS可以识别的语言。

解释器我选择使用Python开发。Python是一种面向对象、直译式的计算机程序设计语言，也是一种功能强大而完善的通用型语言。这种语言具有非常简洁而清晰的语法特点，适合完成各种高层任务。其良好的通用性使得它能在几乎所有操作系统中运行。

该解释器的最终目的是将用户使用自定义语言编写的脚本程序翻译成 PhantomJS 的脚本，并运行。我将解释器分为输入模块，用户脚本分析模块，中间代码生成模块和执行模块。下面我将一一介绍各模块的作用和实现过程。

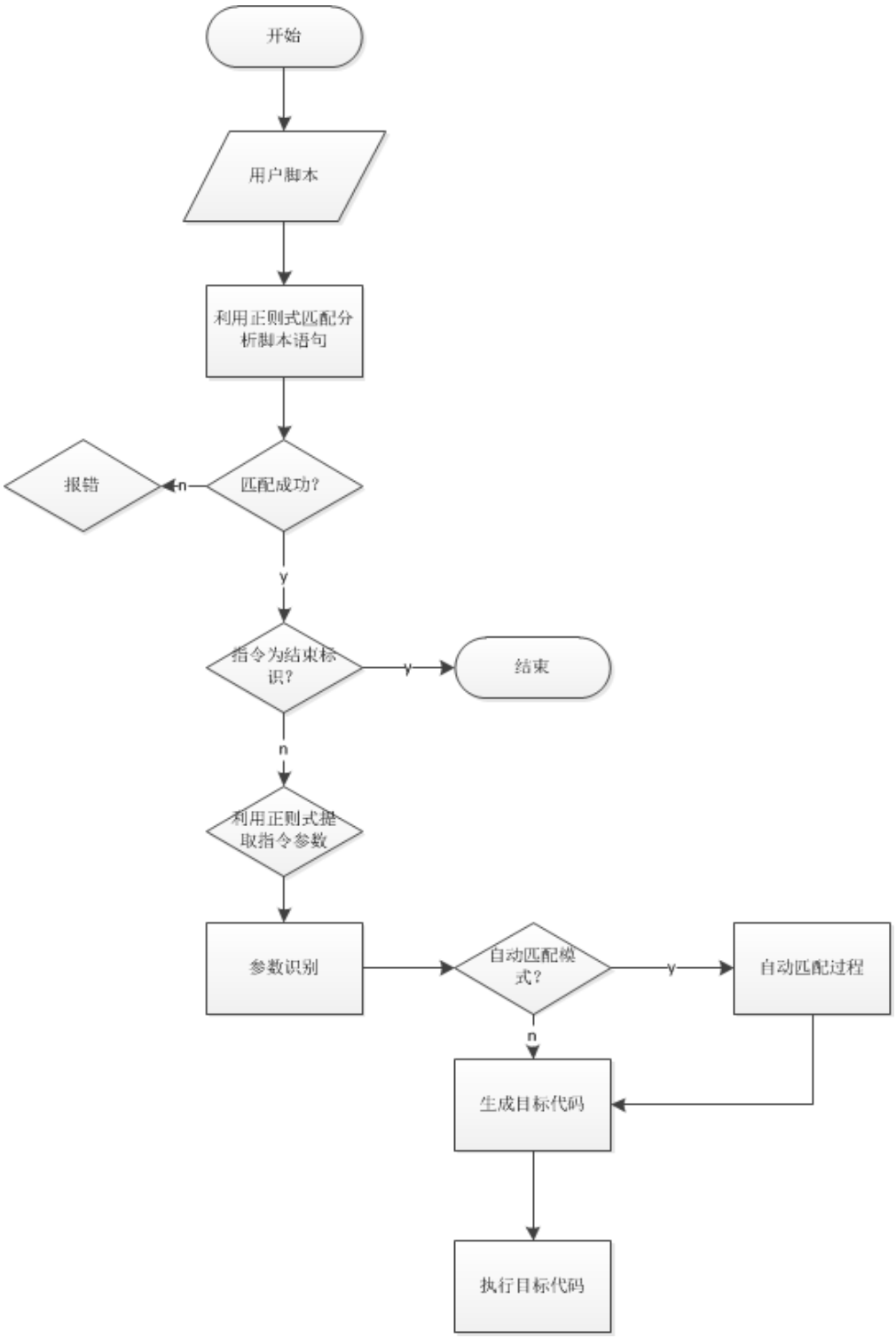

图 3-6 编译器工作流程

输入模块：该模块实现用户如何编写脚本并输入至解释器。目前我实现了两种方法，一是用户将脚本预先写入文本文件，如 mycrawler.txt,将该文件路径作为参数输入至解释器的命令行（解释器是一个 Python 程序，运行时会有命令行等待用户输入），解释器会根据所指定的路径，提取文本中的程序，再进行下一步处理。

第二种方法是直接在命令行中输入爬虫脚本，然后运行。

解释器具有简单的输入检查，当输入的指令格式不对，或者没有给出结束标识（end 指令），均会报错。目前还没有设计图形界面，不过基于命令行输入模块也完全可以满足基本的要求。

用户脚本分析模块：

该模块对用户输入的脚本进行分析，与中间代码生成模块一起构成解释器的核心。由于指令集有限，我使用正则式匹配的方式来识别用户输入的指令，并提取其中的参数。

首先，我将所有用户可能输入的指令以正则式构造出来，例如 Go（）指令，正则式为 go（[\s\S]+？），赋值语句的正则式为[^=]+?=[\s\S]+？，其余指令也都是以正则式进行匹配。

用户输入指令后，会在这些正则式中进行匹配，成功匹配到相应的正则式，则进一步地提取参数、生成目标代码。匹配失败则报错。

提取参数的方法同样采用正则式的方法，每条指令括号内为参数，参数以逗号分割。编译器以参数的内容与数量来判断下一步工作。

目标代码生成模块：

在 3.5.2 中已介绍过我将 PhantomJS 的 API 封装成一块块的代码段，当解释器匹配到某个指令时，只需将相应的 PhantomJS 代码段写入目标代码即可。

执行模块：

该模块将生成的目标代码交由 PhantomJS 执行。我调用了 Python 标准库中的 OS 模块，OS 模块可以实现普遍的操作系统功能 ，而且该模块是跨平台的，在 Linux 或是 Windows 下同样适用。

使用该模块可以实现自动打开 CMD，运行 PhantomJS，并以目标代码作为参数输入。这样就完成了解释器与底层 PhantomJS 框架的交互。

3.5.7 自动匹配模式

目前设计爬虫的方法几乎都是需要用户对网页的 HTML 源码有所熟悉和了解以后，才能对页面上的按钮、输入框等元素进行定位。这对于不熟悉 HTML 源码，或是不了解 DOM 元素定位方法的用户造成了麻烦。

设计自动匹配模式的目的在于使得用户在编写爬虫脚本的时候完全不必关心页面背后的源码，就可以编写网络爬虫程序。

图 3-7 “百度一下”按钮对应的源码

一般情况下 ，需要元素的 id、名称或者类名才能定位该元素，但是这些信息都需要在源码中查看，以百度主页的“百度一下”按钮为例，该按钮的 id 为“su” 类名为 “btn”。

而用户通过浏览器直观看到的只是“百度一下”这个按钮。

使用自动匹配模式，只要输入 click（百度一下）就能自动的定位到该元素，而无需知道该按钮的 HTML 源码。同样，对于输入框，也希望无需知道其 HTML 源码，就能自动定位。

为了实现该功能，我首先分析了这些可触发元素的 HTML 代码的规律。以按钮为例，一般点击按钮就会伴随事件的触发，比如页面跳转、提交表单等，因此在源码中，一般会有“onclick”或者“onmousedown”，另一特征是 value 的值一般是页面上显示的内容。比如 Google 的搜索按钮源码为<input value=″Google 搜索″ name=″btnK″ type=″submit″ onclick=″this.checked=1″> ， 其 中 value=″Google 搜 索 ″ 和 onclick=″this.checked=1″ 。因此，想要定位“Google 搜索”这个按钮，只要找到所有 HTML 源码里有“onclick”或者“onmousedown”的元素，然后在从中找出 value=″Google 搜索″的元素。 所以，用户在使用 Click 指令时，只要给出按钮在页面上显示的名称即可，例如 Click（百度一下），或者是 click（登陆微博）。

Enter 指令同样可以实现自动匹配，参数中只需要说明输入的内容，系统会自动定位到要输入的输入框。如：

enter（account）

enter（password）

系统会自动找到输入用户名和密码的输入框，并输入数据。

实现的方发仍是首先寻找输入框的特征，一般输入框的 HTML 源码具有以下特点：

（1）标签为<input> ，类型为 text 或是 password 或是 account；

（2）包含“maxlength =”字段 ；

（3）包含“autocomplete”字段。

通过以上特征可以获取到页面中的输入框，当使用 enter 指令时，会依次指定输入框进行输入。

目前自动匹配模式并不十分完善，还需要更复杂的逻辑才能跟精确地定位到页面元素。实际测试中，一些主流的社交网站的登陆，和搜索引擎都可以实现自动匹配，但是仍有很多网站匹配失败，该部分仍有待改进。

第四章 测试结果与分析

本章节我将使用该系统对一些主流的 Ajax 应用，如搜索引擎、社交网络、微博、BBS 进行测试。

为方便测试，命令行界面会动态显示当前的 url 信息，如果爬虫脚本里有 prtsc 指令，会将渲染后的网页文件下载到本地保存。

测试一：使用百度搜索“上海交大相关信息”，并抓取每一页的搜索结果

爬虫脚本：

```
set_windowsize(1360,2000)
go("http://www.baidu.com")
enter(上海交大)
click(百度一下)
click($.n:contains(下一页))
prtsc()
end()
```

测试结果：

左边为被抓取并保存至本地的文件，右图为文件内容。

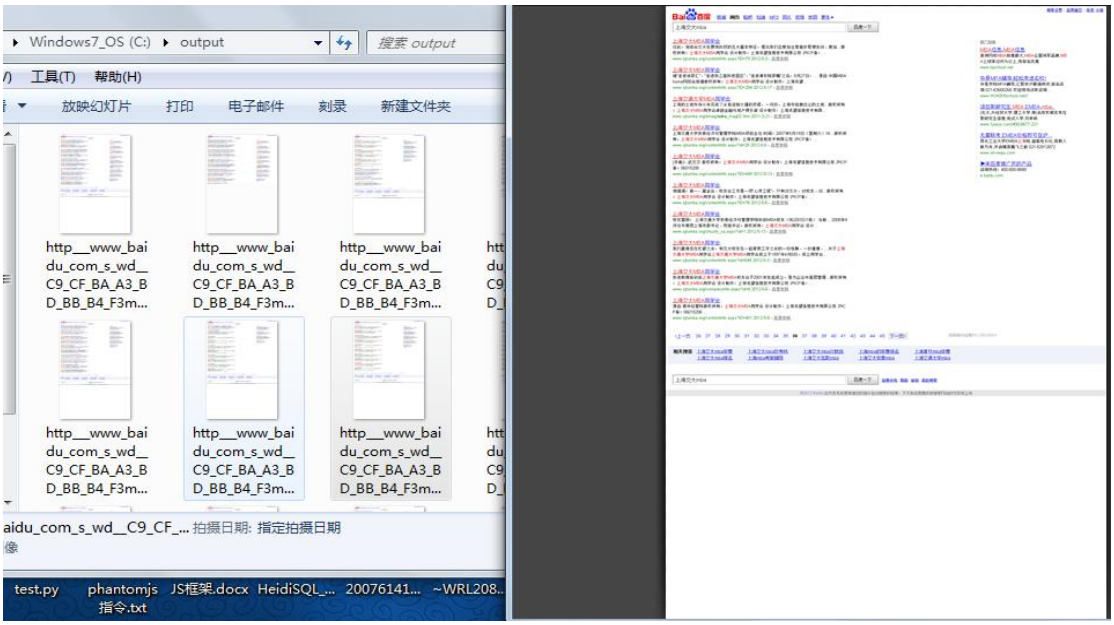

图 4-1 百度测试结果

测试二： 自动登录 LinkedIn，获取个人主页内容

爬虫脚本：

```
go("http://www.linkedin.com")
enter(用户名)
```

enter(密码)
sleep（2000）
click($#btn-login)
prtsc()
end()

测试结果：
如 4-2 所示左边显示爬虫成功登陆 LinkedIn，右边是抓取的结果

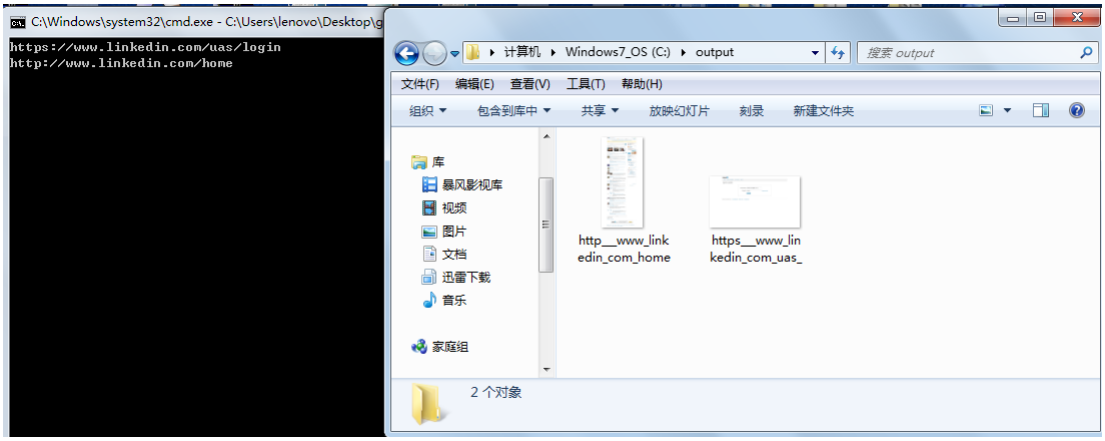

图 4-2 LinkedIn 测试结果

测试三：自动登录新浪微博，使用微博搜索功能，抓取“上海交大”的相关微博
爬虫脚本：
set_windowsize(1360,4000)
go("http://weibo.com/login.php")
enter(用户名,$.name)
enter(密码,$.pass)
click($.W_btn_d)
enter(上海交大,input W_no_outline)
click($.btn)
click($span:contains(下一页))
prtsc()
end()

测试结果：
如图所示：爬虫从初始界面 weibo.com/login.php 成功登陆，进入个人主页 http://weibo.com/u/2658935031?wvr=3.6&lf=reg，然后成功实现微博搜索“上海交大和翻页功能。”

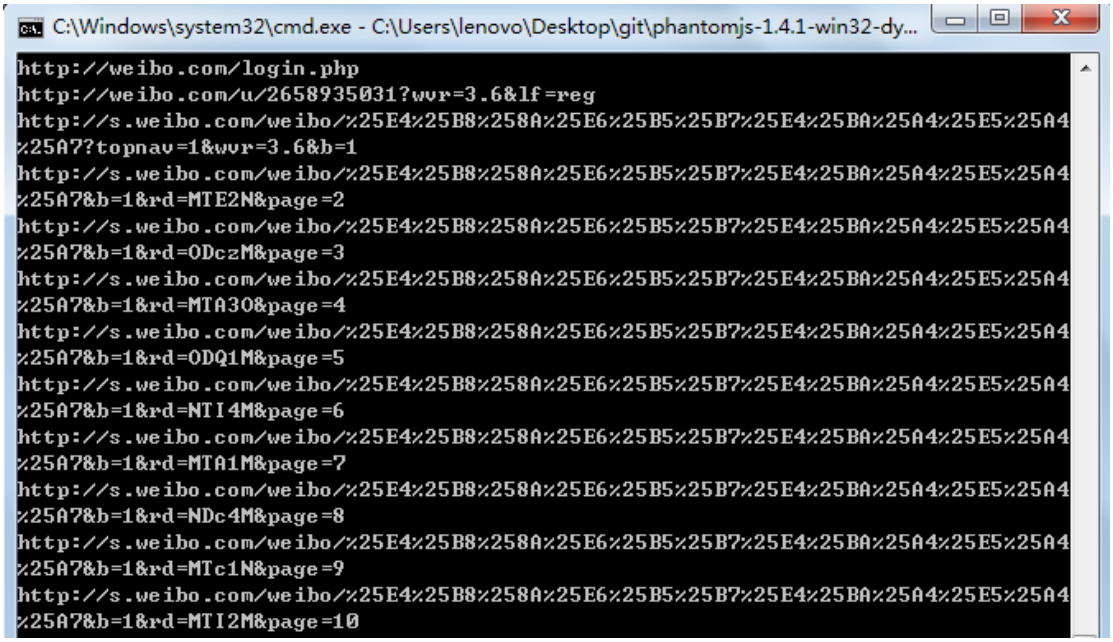

图 4-3 新浪微博测试结果

测试四：登陆上海交通大学电子邮箱

爬虫脚本：

```
go("http://mail.sjtu.edu.cn")
enter(fantasticyue,$#user)
enter(zy5858462,$#pass)
click($input[src*=login-buttom])
prtsc()
end()
```

测试结果：

如图所示，首先进入 jaccount 验证界面，自动输入用户名密码后完成登陆，最后进入邮箱界面 http://mail.sjtu.edu.cn/zimbra/mail。

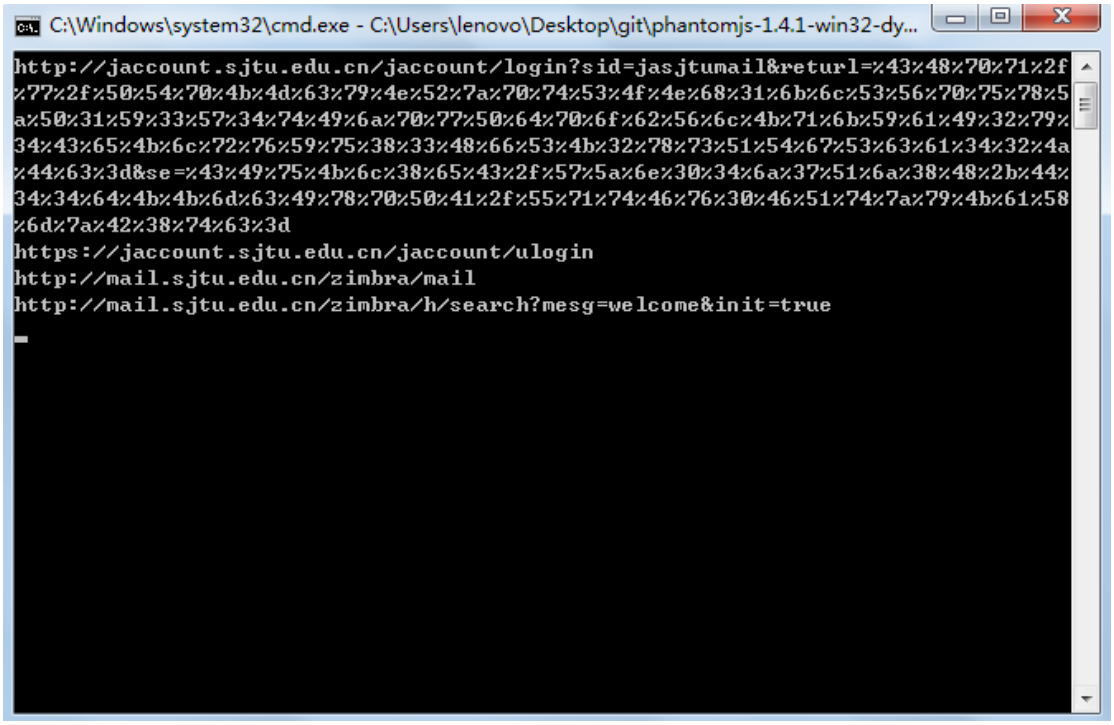

图 4-4 交大邮箱测试结果

测试五：登陆上海交通大学 BBS

爬虫脚本：

```
go("http://bbs.sjtu.edu.cn")
enter(用户名)
enter(密码)
click($a[href*=login])
prtsc()
end()
```

测试结果：

成功登入交大 BBS 主页，bbs.sjtu.edu/fram2.html

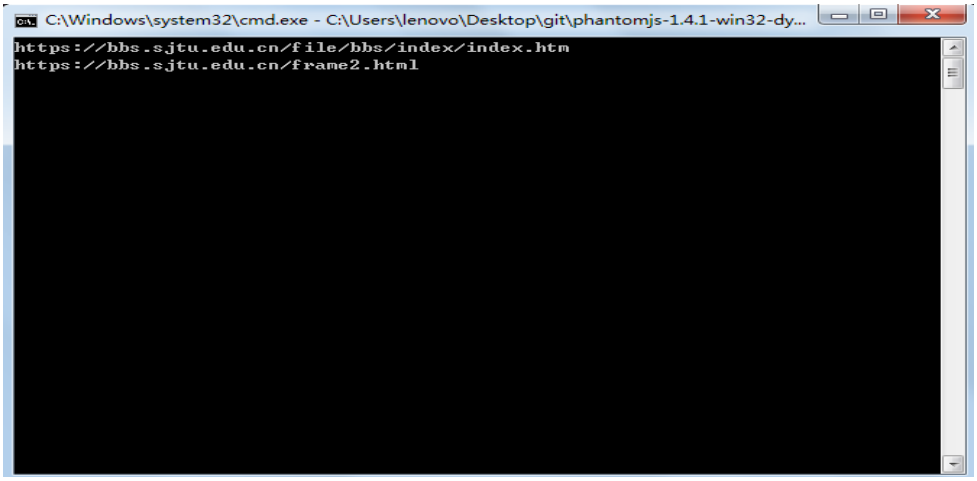

图 4-5 交大 BBS 测试结果

第五章 总结与展望

本次毕设，我所做的工作主要有以下几点：

1) 设计了基于 WebKit 的网络爬虫系统架构
2) 设计了自己的编程语言和对应的解释器，形成一套编程框架
3) 使用该系统，对一些主流 Web2.0 应用进行了测试
4) 研究了 Ajax、浏览器工作原理、浏览器内核等相关知识
5) 分析了现有的网络爬虫开发模式，并进行比较

最主要的工作还是可编程爬虫系统的设计和实现，该系统提供一套编程框架，供用户方便地定制自己的爬虫，功能也十分丰富。但是任存在一些有待改进的地方。一是该系统基于批处理，用户一次性输入指令、系统全部执行完后再返回结果，几乎不存在什么交互性。我的想法是用户没输入一条指令，系统就会执行相应的操作，并有结果的反馈，还可以考虑图形界面的设计，这些都可以大大增强用户与系统间的交互。另一个问题是用户在编写爬虫代码时仍需要查看 HTML 源码，这对于不熟悉网页源码的使用者来说非常麻烦，我设计了自动匹配模式试图解决该问题，希望用户只需只要按照页面元素在页面上显示的名称就可以定位和操作，但是测试下来效果不是很理想，许多网页的页面元素都没办法正常匹配。第三，语言的设计还不够成熟，缺乏语义完整性的审查，指令集也很有限，这部分值得改善的地方还很多。

我将来的工作，就是针对以上这些问题，对我的爬虫系统进一步地改进和完善。

参考文献

[1] 杨靖韬,陈会果. 对网络爬虫技术的研究[J]. 科技创业,2010:170-171.
[2] Michael Bolin. End-User Programming for the Web [D]. Massachusetts Institute of Tec hnology : Department of Electrical Engineering and Computer Science, 2005.
[3] Garrett J. Ajax: A New Approach to Web Applications[EB/OL]. (2005-02-18). [2012-04-15].http://www.adaptivepath.com/ideas/essays/archives/000385.php.
[4] 夏冰, 高军, 王腾蛟，等. 一种高效的动态脚本网站有效页面获取方法[J]. 软件学报, 2009, 20(z): 176-183.
[5] 罗兵. 支持 Ajax 的互联网搜索引擎爬虫设计与实现[D]. 杭州: 浙江大学, 2007.
[6] 袁小节. 基于协议驱动与事件驱动的综合聚焦爬虫研究与实现[D]. 长沙: 国防科学技术大学, 2009.
[7] Watir [EB/OL]. [2012-04-16]. http://watir.com/.
[8] Selenium WebDriver [EB/OL]. [2012-5-11].http://seleniumhq.org/projects/webdriver/
[9] 刘金红, 陆余良. 主题网络爬虫研究综述[J].计算机应用研究,2007 ,24 (10) :26229.
[10] 肖卓磊. 基于 Ajax 技术的搜索引擎研究[D].武汉: 武汉理工大学, 2009.
[11] Shah S. Crawling Ajax-driven Web 2.0 Applications [R/OL] (2007-02-14) .[2012-04-15] .http://www.infosecwriters.com/text_resources/pdf/Crawling_AJAX_SShah.pdf.
[12] 郭浩, 陆余良, 刘金红. 一种基于状态转换图的 Ajax 爬行算法[J].计算机应用研究, 2009, 26(11): 4266-4269.
[13] PhantomJS [EB/OL]. [2012-02-19]. http://code.google.com/p/phantomjs/.
[14] Chickenfoot [EB/OL]. [2012-04-01]. http://groups.csail.mit.edu/uid/chickenfoot/quickstart.HTML.
[15] jQuery [EB/OL]. [2012-04-19]. http://jquery.com/.
[16] jQuery 教程 [EB/OL]. [2012-04-19]. http://www.w3school.com.cn/jquery/.
[17] 浏览器工作原理 [EB/OL]. [2012-04-11]. http://www.iefans.net/liulanqi-ruhe-gongzuo-yuanli/
[18]卢亮，张博文.搜索引擎原理、实践与应用[M].北京：电子工业出版社，2007.
[19]谢新洲．网络信息检索技术与案例[M].北京：北京图书馆出版社，2005：29-30.
[20]周立柱，林玲.聚焦爬虫技术研究综述[J].计算机应用，2005，25(9)：1965-1989.
[21]刘玮玮.搜索引擎中主题爬虫的研究与实现田[D].南京：南京理工大学，2006.
[22]杨溥.搜索引擎中爬虫的若干问题研究[D].北京：北京邮电大学，2009，1.

谢辞

本论文的研究工作是在陈凯老师的精心指导和帮助之下才完成的。我对老师给予我的巨大的帮助与支持表示由衷的感谢。陈凯老师都拥有着丰富的科研经验和严谨的治学态度，他们在整个课题进展的过程中对我严格要求，积极鼓励，帮助我解决了各类的问题，也向我展现了一名科研人员所应该具有的能力和素质。

此外，我还要感谢在课题进展过程中给予了我帮助的同学们。我要感谢储曦庆同学，由于我们研究内容都是网络爬虫，在互相交流和帮助中一起学到了很多知识，也互相提出了许多宝贵的意见和建议，帮助我找到了很多灵感和解决问题的思路。在此，我希望能够向以上所有人以及其他帮助过我的人表达我最真诚的谢意。

PROGRAMMABLE COMMANDLINE WEB CRAWLER SYSTEM BASED ON WEBKIT

ABSTRACT

Information explodes with the fast development of Internet. People rely on search engines in order to get information needed. Web crawler is a program that automatically downloads web resources from the Internet. It is a critical part of search engines as well as an important approach to collect Internet resources.

At present, Web 2.0 applications like blogs, BBS and Social Networks are becoming more and more popular among Internet surfers. Through posting real-time messages， uploading pictures and videos, creating homepages or editing webpages , Web 2.0 application users become an important source of information.Howerver, based on Ajax, these Web 2.0 applications present great challenges toward Web crawlers. First, with the mass employment of JavaScript in websites, resources on web pages are no longer statically presented .Instead, many of them are acquired and created real-time through asynchronous request/response from servers driven by JavaScript. In order to collect dynamic information in these Ajax Websites, Crawlers must be able to parse JavaScript. Second, many Web 2.0 applications have a login system which require crawlers to simulate user login in order to be granted access to resources. These challenges all shatter the traditional Web crawling approach based on HTTP protocol.

In order to tackle with these problems, a Web crawler must have following functions.

- Javascript interpretation
- DOM event handling and dispatching
- Dynamic DOM content extraction

There are still many approaches to build Web crawlers that can solve the problem of Ajax,and most of them are based on browser simulation . As shown in Figure 1. There are following types of design choices to build Web crawlers:

- Programming to simulate browser manually
- Browser Kernel
- Browser Plugin
- Browser Automaton Framework

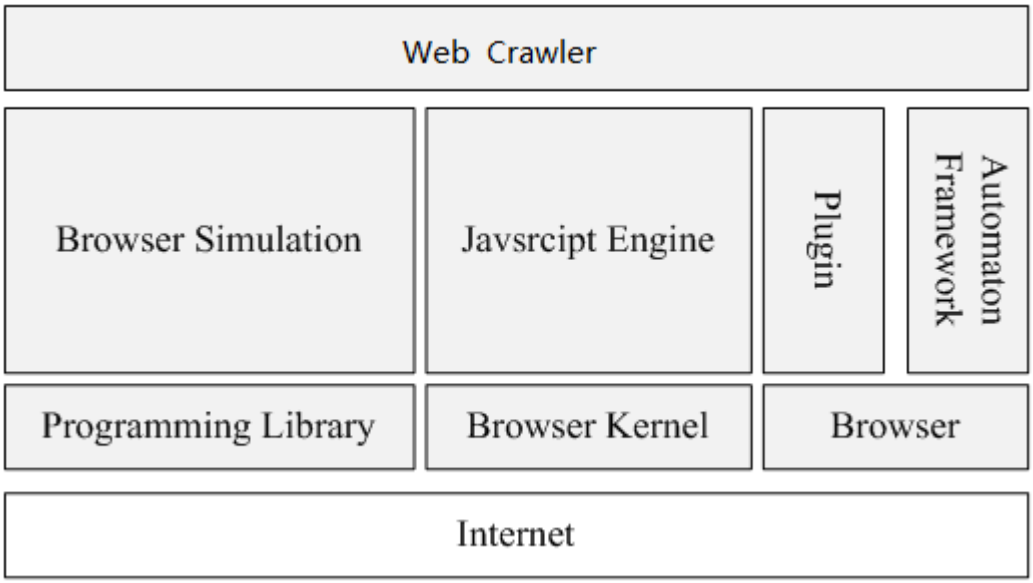

Figure 1, Architecture of Web crawlers using browser simulation

Each approach has its pros and cons. Programming from scratch to simulate browser is too complex for to implement since it requires HTML parsing, JavaScript interpretation and DOM handling manually. The advantage for this method is its high-efficiency and low memory consumption.
Approaches based on real browsers however, relieve developer from “low level” programming, instead they only have to control browsers to finish all the crawling tasks. Here, everything on web page is regarded as “Web Element” from end-user programmer. Crawler can simulate the human being to interact with browser, no longer to simulate the browser itself. So it will change the crawler behavior. It will no longer notice about the JavaScript behind the buttons. The authentication process will simply be filling in the input boxes on the login page then click a login button like a real human user.

Building Web crawlers based on real browsers also has distinct disadvantages .It consumes too much memory resources and its efficiency is low. Therefore, I chose the other way, building my Web crawler on browser kernel instead of a real browser. I believe it’s a more promising way because its relatively high efficiency and it also has the ability to handle HTML parsing , JavaScript and DOM.

I designed and implemented a programmable command-line web crawler system based on WebKit. WebKit is an open source web browser engine. Many mainstream browsers are build on WebKit such as chrome and Safari. Webkit is consisted of two component, Webcore and JScore. Webcore handles HTML parsing , DOM tree construction and page rendering while JScore interprets JavaScript codes embedded within HTML source codes.

In order to manipulate WebKit, I use the framework of PhantomJS. PhantomJS

(www.phantomjs.org) is a headless WebKit scriptable with JavaScript or CoffeeScript. It supports many W3C stands like DOM handling, CSS selector, JSON and Canvas. I also designed a script language and an interpreter for this system which enables users to develop Web crawlers using very simple language .Figure 2 illustrates the architecture of the system.

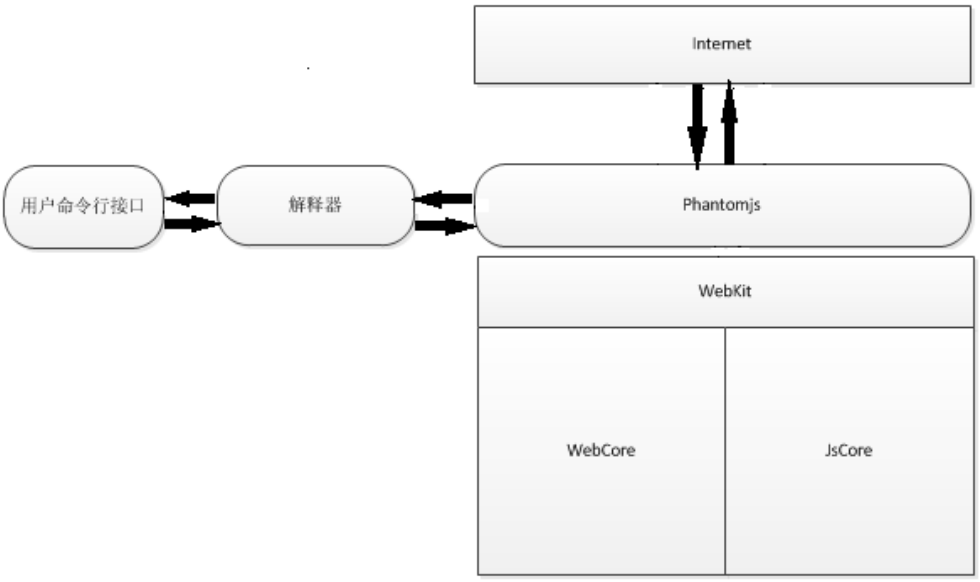

Figure 2, Architecture of programmable web crawler system based on WebKit

User first write scripts using the language I designed to design Web crawlers. Then the interpreter would analyze the script , including syntax and semantic analysis in order to recognize each instruction and to extract the parameters, variables and operators within the instructions. Once the script is analyzed, it would be translated into a target code which would be executed by PhantomJS.The target code will drive PhantomJS to perform HTTP request/response, DOM handling and JavaScript interpretation which is the core process of crawling.

The language is simple and easy to use ,by writing script like go(“url”) to navigate through pages or like enter(text) to simulate keyboard input or like click(button) to simulate a mouse click, users would feel like controlling a actual browser when they are programming. They wouldn’t have to care about how HTTP request/response are handled, how JavaScript are interpreted or how DOM are dynamically constructed. Therefore users have only to care about which element on the webpage to be operated and to perform which kind of operation .

For example, when user wants to search “sjtu” with Google and to get the search results ,user first navigate to homepage of Google by using “go” instruction .Then he use enter(sjtu, $#lst-ib) to input “sjtu” into the search box. The second parameter tells the crawler where to enter these keywords. Finally he use click($#btnG) to click

“Google search” button. Since its necessary to locate where a page element is in order to manipulate it, element orientation is an very important port of my work. I use jQuery to locate page elements since it offers a powerful set of tools to match and locate elements in the document conveniently.

At the end of this paper, I tested the system by crawling some popular Web 2.0 communities to prove its validity. The system still needs to be improved in several aspects. First , the system is based on batch processing which means there is little interaction between user and system. I am thinking of improving it by making the system react and respond to the user instantly after the user put in each instruction. Second , users still needs HTML source codes inspection to know how to locate the element. I am designing a pattern that users, in order to match or locate a web page element , only need to know the name of the web page element that are shown on the web pages rather than to manually inspect HTML. This pattern is still immature and not applied to many pages. The future work is to improve this pattern and to implement a better way of interaction between user and the system.