毕业设计（论文）模版

**西安邮电大学**

**毕业设计（论文）**

题目： 基于Python网络爬虫的天气信息

搜索与预报系统

学院： 通信与信息工程学院

专业： 通信工程

班级： 通工1310

学生姓名： 鲍颖

学号： 03131313

导师姓名： 石薇 职称：

起止时间： 2017年2月 27日 至 2017 年 6 月 17 日

## 毕业设计（论文）声明书

本人所提交的毕业论文《基于Python网络爬虫的天气信息搜索与预报系统》是本人在指导教师指导下独立研究、写作的成果，论文中所引用他人的文献、数据、图件、资料均已明确标注；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注明并表示感谢。

本人完全理解《西安邮电大学本科毕业设计（论文）管理办法》的各项规定并自愿遵守。

本人深知本声明书的法律责任，违规后果由本人承担。

论文作者签名：

日期： 年 月 日

摘□□要

*（“摘要”之间空两格，采用三号字、黑体、居中，与内容空一行）*

在互联网发展的初期，用户主要通过访问一些知名的门户网站的方式去搜集自己想要的数据信息，但是随着Web的急剧发展与扩大，这种传统的获取信息的方式已经变得越来越困难了。因为互联网就像是一个非常庞大的数据库，各种各样的数据和资源以各种形式存储于网络上，在网页展示形式也非常多：文字、图片、Flash动画或者视频等等。用户试图在网页上获取一些信息并点击链接的时候，往往会收到大量外带的数据，例如弹窗推送广告、网页浮动窗口等，这些多余的信息不但容易干扰视线、使用户较难筛选出有效信息，而且某些广告弹窗可能会暗藏一些恶意链接，使用户造成不必要的损失。因此，通过网络爬虫技术来进行高效、快速的数据搜集，将可以给用户提供一个纯净、可靠地获取信息的方式。本课题旨在设计并实现一个可以主动、定向收集天气信息的运行在服务器上的Python网络爬虫程序，使得客户端在去获取信息的时候，避开大量的无效信息的干扰，使用户能够高效、快速的完成信息的获取。

关键词：网络爬虫;客户端-服务器模式;Python

×××××

*采用小四号、宋体、接排*

*小四号、黑体、顶格*

# ABSTRACT

*（采用三号字、Times New Roman字体、加黑、居中、与内容空一行）*

□□In the early days of the Internet, the user mainly through the access to some well-known portal to collect the information they want, but with the rapid development and expansion of the Web, the traditional way to obtain information has become more and more Difficult. Because the Internet is like a very large database, a variety of data and resources in a variety of forms stored in the network, in the form of web display is also very much: text, pictures, Flash animation or video, and so on. When users try to get some information on the page and click on the link, they often receive a lot of data, such as pop-up advertising, page floating window, etc., these redundant information is not only easy to interfere with the line of sight, so that users more difficult to screen out Effective information, and some advertising bounces may hide some malicious links, causing users to cause unnecessary losses. Therefore, through the web crawler technology to carry out efficient and fast data collection, will be able to give users a pure, reliable way to obtain information. The purpose of this project is to design and implement a web crawler program that can be used to collect weather information on the server, so that when users try to get information on the network, avoid a large amount of invalid information interference, so that users can Efficient and fast access to information.*（内容采用小四号Times New Roman字体）*

**Key words：** Web Crawler;Client - server mode;Python

×××××

*采用小四号、Times New Roman字体、接排*

*小四号、Times New Roman、加黑、顶格*

目录

*（三号、黑体、居中、目录两字空两格、与正文空一行）*

第一章 *引言…………………………………………………………*×

1.1选题背景与意义………………………………………………………………×

1.2网络爬虫技术发展现状**……………………………………………………**×

1.3 C/S架构下的软件开发…………………………………**………………**×

1.4 本文章节安排**………………………………………………………………**×

……………

第二章 *网络爬虫技术…………………………………………………*×

2.1 网络爬虫的基本架构及工作流程**…………………………………………**×

2.2 网络爬虫的搜索策略**…………………………………………**×

2.2.1 深度优先搜索策略**………………………………………………………**×

2.2.2 广度优先搜索策略**………………………………………………………**×

2.2.3 聚焦搜索策略**……………………………………………………………**×

2.3 网络爬虫伪装用户**…………………………………………**×

2.3.1 IP代理**…………………………………………**×

2.3.2 User Agent**……………………………………**×

2.3.3 Cookie**……………………………………**×

2.4 几种开源爬虫框架**……………………………………………**×

2.5 网络爬虫的几种应用**……………………………………………**×

………………

第三章 *基于客户端/服务器模式下的软件开发……………………*×

3.1 客户端/服务器模式介绍**…………………………………………………**×

3.1.1 什么是C/S架构**…………………………………………………………**×

3.1.2 C/S架构与B/S架构的区别**……………………………………………**×

3.2 客户端与服务器的通信过程及原理**………………………………………**×

3.2.1 底层网络协议**…………………………………………**×

3.2.2 I/O多路复用**………………………………………**×

………………

1. *基于网络爬虫的天气搜索工具的设计与实现………………*×

4.1 总体框架介绍

4.2 数据爬取**………………………………………………**×

4.2.1 网站结构的分析**……………………………………………………**×

4.2.2 Spider的设计实现**………………………………………**×

4.2.3 Spider的伪装与使用代理IP**………………………………………**×

4.2.4 使用多线程提升爬虫性能**………………………………………**×

4.2.5 Crontab实现定时数据爬取**………………………………………**×

4.3 数据存储**……………………**×

4.4 服务器程序的设计实现**………………………**×

4.4.1 使用网络套接字与Client通信**………………………**×

4.4.2 Server验证登录用户合法性**………………………**×

4.3.3 后端数据分析与查询**………………………**×

4.3.4 使用IO多路复用提升Server性能**………………………**×

4.5 Client的设计与实现**………………………**×

4.5.1 使用Qt进行GUI开发**………………………**×

4.5.2 用户登录模块设计与实现**………………………**×

4.5.3 数据分析与曲线绘制**………………………**×

4.6 **………………………**×

………………

结束语*（四号、宋体）………………………………………*………×

致谢*（四号、宋体）……………………………………………………*×

参考文献*（四号、宋体）………………………………………*………×

附录*（四号、宋体）……………………………………………*……**…**×

第一章□□引言*（居中、小三号、黑体）*

1.1选题背景与意义*（四号、黑体、顶格）*

正文开始标注页眉，宋体五号居中

□□随着时代的进步与科技的发展，互联网已经在不知不觉间蔓延、扩散到了我们生活空间的每一个角落。近年来被社会谈论的非常热门的一些科技词汇，比如大数据、云计算、物联网等，几乎都和互联网的繁荣发展息息相关。互联网具有的多维性、自由平等性、虚拟交互性、以及最重要的海量性与全球性，改变了生产信息与传播信息的方式，进而改变了人类的工作方式、生活方式、生产方式甚至是思维方式。人们已逐渐习惯通过互联网去获取自己需要的各种信息（浏览新闻网站或者在线观看视频），或者是通过互联网去发布一些自己愿意公开的信息（发微博、写说说），而不是通过曾经的那些传统渠道（报纸、杂志），传统渠道在各方面所占的份额也在逐渐降低，这证明人们对信息的获取方式的态度也在转变。

□□然而这种情景的背后，有我们需要迫切关注的一点，那就是每天都有EB级别的新数据涌入互联网这个庞大的数据库，那么随着数据的不断累积，摆在我们家面前的的就是一个拥有海量数据的、无比庞大的数据库，各种各样的信息全部沉淀、堆积在这个庞大的数据库之中。这就意味着，当用户想要从互联网上接收信息的时候，不一定能够快速、准确的获取到自己真正想要的信息，而且这些信息中或许掺杂了非常多的无用数据，甚至于是包含了恶意链接的、会造成用不必要损失的内容。因此，能够从网络中定向的、准确的、高效的提取用户需要的信息，才可以真正的提升用户的体验。本课题以使用频率较高的天气搜索功能为起点，通过网络爬虫技术与C/S软件架构模型，设计并实现了一套能够高效获取互联网天气数据并分析的的软件，可以精确、快速的获取天气信息，降低用户浪费在数据分辨与筛选上的时间，为用户提供一个纯净、可靠的信息获取来源。

1.2网络爬虫技术发展现状

传统的网络爬虫技术，主要是用来爬取静态的web页面，基于socket的httpclient功能简单，性能强大，特别是在多线程、高并发的情境下，从而被开发者青睐，特别是搜索引擎中，若抓取的是静态页面，httpclient是非常适合这种情景的。但是随着AJAX/web2.0技术的兴起，很多网页都使用这种新技术改写成了动态网页，所以，如何抓取AJAX等动态页面，就成了搜索引擎紧急需要解决的问题。因为AJAX颠覆了传统的纯HTTP请求/响应的协议机制，当遇到AJAX加载的信息、需要JavaScript渲染的时候，httpclient就无法正常工作了。所以若爬虫程序还是按照传统的下载网页、分析文档、提取链接的机制的话，是无法抓取到AJAX页面的有效数据的。AJAX采用的是JavaScript驱动的异步请求/响应机制，传统的爬虫采用的是xpath或正则表达式这种技术来进行内容的提取，而缺乏对JavaScript语义上的理解，是无法模拟出触发JS的异步调用并解析返回的异步回调逻辑和内容。而且，在AJAX的应用中，JS会对DOM结构进行修改与变动，甚至于页面的所有内容都是通过JavaScript动态绘制的，这种动态页面生成技术与DOM结构相对不变的静态页面相差甚多。所以从此处可以看出，传统的爬虫程序是基于HTTP协议驱动的，用来爬取静态页面足够了，但是对于新型的AJAX这样的技术，爬虫引擎的核心必须是基于事件驱动的。

若是爬虫程序的可行性得到了解决，那么其面临的最大问题就是爬虫性能的提升和突破网站对爬虫的封锁。大部分情况下，采用多线程高并发、高频率进行数据抓取是可行的，但是这有一个前提，那就是网站没有采用任何反爬虫措施，例如验证码、防火墙、限制访问频率等等。但更多的时候，真正有价值的信息，一定是伴随着严格的反爬取策略的，一旦网站检测出当前请求者是爬虫程序而非真正的用户，那么最严格的情况下，有可能会直接将当前IP地址加入黑名单，一旦被加入黑名单，则意味着当前运行爬虫程序的服务器再也无法访问这个网站，更别提爬取数据了。所以，由此诞生出很多反封锁的办法，例如维护一个代理IP池来防止网站封IP，但是这也带来了新的问题，那就是IP池是否稳定，和切换IP的速度等等，这些都是开发出一款稳定的、功能完善的网络爬虫而无法避免、需要考虑的问题。

1.3 C/S架构下的软件开发

C/S是Client/Server（客户端/服务器模式）的简写。这里的服务器通常采用高性能的PC或工作站，并采用专业的数据库系统，如Oracle、Mysql、SQL Server。客户端需要安装专门开发的的客户端软件。

这种架构模式的优点是能充分发挥客户端机器的性能，大多数工作任务可以在客户端处理后再提交给服务器。因为部分任务放在了本机，所以对应的优点就是客户端响应速度比较快。但也有它不足的地方，例如客户端需要有专门的安装包，还需要根据平台特性与功能更新做出版本迭代，开发成本较大。

1.4 本文章节安排

本文章节具体安排如下：

1. ：引言部分。主要用于介绍背景知识，包括网络爬虫技术发展现状，以及客户端/服务器架构下的软件技术的发展现状。
2. ：网络爬虫技术。对网络爬虫技术进行系统的阐述，介绍网络爬虫的几种搜索策略，以及网络爬虫如何模拟用户行为，并对多个开源爬虫框架进行了对比测试。

第三章：基于客户端/服务器模式下的软件开发，阐述什么是C/S架构，解释其通信流程，与使用的底层网络协议，以及如何使用IO复用提升服务器性能。

第四章：基于网络爬虫的天气搜索工具的设计与实现。主要介绍天气搜索工具的设计步骤与实现细节，并对爬取结果进行处理与分析。

第二章□□网络爬虫技术*（居中、小三号、黑体）*

2.1网络爬虫的基本架构及工作流程*（四号、黑体、顶格）*

网络爬虫（Web crawler）是一种能够自动提取网页信息程序的统称。从功能上来讲，爬虫程序一般分为数据采集、数据处理、数据存储三个模块。网络爬虫的定义有狭义与广义之分，广义爬虫的定义为：所有能够利用http协议搜索web文档的软件都称之为网络爬虫。狭义爬虫的定义 为：利用标准的http 协议，根据超级链接和Web 文档检索的方法遍历万维网信息空间的软件程序。

网络爬虫的功能是根据已有URL去爬取对应的网站，并从得到的html文档中提取出设计者需要的信息和新的URL,然后就可以进行递归的爬取。所以说，网络爬虫的本质就是模拟用户使用浏览器发出http请求，虽然浏览器和网络爬虫是两种不同的客户端程序，但都用相同的方式去获取网页：客户端程序连接到DNS服务器上，DNS服务器将传给它的域名翻译成web服务器的IP地址，然后客户端通过IP地址和端口去向web服务器发起http请求，http协议是OSI七层模型中的应用层协议，它使用的底层协议分别是TCP（传输层协议）和IP（网络层协议）。连接建立之后，客户端的请求被web服务器收到，服务器会将相应的结果返回给客户端，并且一般来讲，HTTP协议默认的端口是80端口，是一个知名端口。

但是浏览器和爬虫程序还有有着本质的区别的，因为浏览器需要用户的点击来完成操作http请求，而爬虫程序是自动完成http请求的，所以爬虫程序需要有一整套完善的架构才能完成全部的工作任务。

在网络爬虫的系统框架中，主要过程结构由控制器，解析器，资源库三部分。控制器（也称调度器）的主要工作是负责给多线程中的各个爬虫线程分配工作任务。解析器的主要工作是根据url队列中的网页地址下载对应网页，并进行页面的分析处理，主要是将一些JS脚本标签、CSS代码内容、HTML标签等内容处理掉，爬虫的基本工作是由解析器完成的。资源库是用来存放下载到的网页资源的，将解析器解析出的数据保存起来，一般都采用大型的数据库存储，并对其建立索引。

下面是爬虫的工作流程图与步骤解析。



爬虫技术已经发展了十几年，整体框架上已经相对成熟，下面是通用的爬虫框架工作流程：

1）首先从所要爬取的网站中挑选一些合适的网页，将这些网页的url作为初始url（也成为种子url）

2）将初始url放到一个待抓取url的队列中，之后调度器将从这个队列中取出url进行网页下载。

3）爬虫程序从待抓取url队列中一次读取url，并将url通过DNS解析转换为web服务器IP地址

4）再由网页下载器根据IP地址和url中域名后的相对路径，发起http请求，下载目标网页

5）已经下载的网页，爬虫程序会将这个网页的url信息存入到一个已抓取url队列中，这个队列中记录着所有已经被下载器下载过的网页的url,通过这种机制，可以防止重复多次抓取同一个网页。

6）爬虫程序会对下载好的html文档进行分析，将其中的url链接提取出来，使用Bloom Fliter之类比较高效的算法，测试已爬取队列中是否存在这个url，如果不在，则说明这个一个未被爬取的url，那么就将它放入带爬取队列中。

7）调度器再次从带爬取队列中取出url进行爬取，如此形成一个完整的循环，直到待爬取队列为空结束。

2.1网络爬虫的搜索策略

网络爬虫是从一个或若干个初始网页URL开始，获取初始网页上的URL，在抓取网页的过程中，不断的从当前页面上抽取新的URL放入带爬取队列，重复这个过程直到爬取完全部页面，或满足系统设置的一定停止条件。因为这种数据采集过程像是一个爬虫或蜘蛛在网络上漫游，所以它才被称为网络爬虫系统或网络蜘蛛系统。然而由于网络资源太过于庞大复杂，无数网页资源之间的关系盘根错节，就算是再精明的爬虫、蜘蛛，可能也会在这网络森林中迷失方向，所以我们需要给爬虫从网页上爬取数据的过程设置一些搜索策略，以便更好的进行数据搜集工作。

网络爬虫通过请求网站站点上的HTML文档访问某一站点，遍历web空间，不断从一个站点爬到另一个站点，搜索与分析网站内包含的信息，对信息建立索引，再将索引关系加入到网页数据库中，并利用HTML语言的标记结构来搜索信息及获取指向其他网站的URL地址，通过这种方式，可以完全不依赖用户的干预实现在网络上的自动爬取。

2.1.1 深度优先搜索策略*（四号、黑体、顶格）*

深度优先搜索策略所遵循的原理就是尽可能的“往深处走”，从起点开始尽可能深的搜索图。在计算机科学的图论中，深度优先就是对于一个节点来说，如果这个节点还有以它为起点而且尚未被探索完的边，那么就继续从这个节点探索下去，直到这个节点的所有边都已经被探寻过，搜索将回溯到上层节点。这一过程一直持续到所有从源节点开始的可以到达的节点都被探索完毕为止。但是深度优先搜索策略可能会使爬虫陷入一个网站系统内部，不利于搜索比较靠近网站首页的网页信息，而且有的时候可能会导致爬虫陷入过深的递归搜索中导致爬虫出不来的情况。

2.1.2 广度优先搜索策略*（四号、黑体、顶格）*

广度优先搜索算法是最便捷的搜索算法之一，同时这种算法也是很多重要的图论算法的原型。例如Prime最小生成树算法与Dijkstra有向图的单源最短路径算法，都采用了类似广度优先搜索类似的思想。广度优先搜索算法的思想是沿着树的宽度遍历树的节点，遍历一个节点，并将其所有的子节点加入队列中，按“层”遍历。在爬虫程序的具体的实现中，就是将新下载的网页包含的URL链接直接追加到带爬取URL队列的末尾。

有实验表明，这种策略的实际效果很好，虽然看似死板与机械，但实际上的网页抓取顺序基本上是按照网页的重要性进行排序。之所以如此，有研究人员认为：如果某个网页入度很高（包含很多入链），那么更有可能被广度优先遍历策略早早的抓到，从某种意义上讲，入链个数也体现了一个网页的重要性，即实际上广度优先遍历策略也隐含了一些网页优先级假设。

2.1.3 聚焦搜索策略*（四号、黑体、顶格）*

聚焦搜索策略的出现，主要是因为随着互联网的不断发展与不断扩大，网页信息开始呈指级增长，那么传统的搜索策略在面对如此海量与庞大的网页信息会显得力不从心，因为其中掺杂了大量的与想要获取的信息关联性较低的或毫无关联内容，面对这种情况，定向抓取相关网页资源的聚焦爬虫便应运而生。聚焦爬虫的爬行策略只挑出所需要的特定主题的页面。聚焦爬虫会给它所下载下来的网页进行评价并排序，排序越靠前的，会越先被访问，这种策略会保证爬虫能优先跟踪那些最有可能包含目标页面链接的页面，决定这种搜索策略的关键是如何评价链接价值，或者成为链接价值的计算方法。

不同的链接价值评价方法计算出的结果是不同的, 那么最终得出的链接的“重要程度”的排序也不同， 从而决定了不同的搜索策略。由于url链接包含于网页页面之中，而通常具有较高价值的网页页面包含的url链接也具有较高的价值，因而对链接价值的评价有时也转换为对页面价值的评价. 这种策略通常运用在专业搜索引擎中，因为这种搜索引擎只关心某一特定主题的页面。但有的时候，也会因此会漏掉一些评价不高，但是可能确实包含了有用信息的相关网页，所以不能盲目使用聚焦搜索策略，还应针对问题具体分析，才能得到最好的抓取效果。

2.3网络爬虫伪装用户

随着互联网的发展，网络爬虫已经成了很普及的网络技术，一个内容驱动型的网站，容易受到网络爬虫的青睐与光顾，这是不可避免的。一些智能的搜索引擎的爬虫的爬取频率是比较合理的，对网站资源的占用与消耗都较低，但是仍旧有很多比较糟糕的爬虫，肆无忌惮的并发几十上百请求来进行网页的循环重复抓取，这种网络爬虫对网站资源的消耗是十分巨大的，特别是对一些中小型网站来说，很容易就会造成网站访问压力增大，导致网站访问速度慢，甚至无法访问。

所以，一般来说，网站从三个方面进行反爬虫：用户请求的报文头部（Headers），用户行为，网站的数据的加载方式。从用户请求的Headers反爬虫是最常见的网站反爬虫策略，很多网站都会对报头Headers的User-Agent部分进行检测，还有一部分网站会对Referer部分进行检测（一些资源网站的防盗链就是通过检测Referer来进行的）。用户行为检测是有些网站是通过检测用户行为来判断是否是爬虫的，例如一个IP地址短时间内多次访问了同一页面，或者同一用户账户短时间内多次进行了相同的操作，那么很有可能会被认为是网络爬虫而被采取相应措施。有些网站的信息必须登录后才能获取到，例如新浪微博，不登录账号，只能看到大V的前十条微博，想要看到全部内容，网站要求用户必须处于登录态，也即需要Cookie验证，才能获取到相关数据。

那么，当我们需要爬取这个网站上的数据的时候，这些策略我们应该如何去应对呢？其实我们可以把爬虫程序完全伪装成为用户的正常浏览行为，从而躲避大部分的反爬虫策略。通过以下几种策略，可以将爬虫进行伪装，从而防止被网站的反爬虫机制发现。当然，访问频率，并发数目这些是无法逃避的，需要有足够的耐心和足够多的机器、有足够多的代理IP，那总是可以抓取到足够多的信息的。

2.3.1 User Agent

从用户请求的报头Headers进行分析，是最常见的反爬虫策略。很多网站都会对Headers的User-Agent部分进行检测，看是否存在User-Agent或者User-Agent是否合法，因为正常用户通过浏览器进行网页访问，一定是有User-Agent的。还有一部分网站会对Referer部分进行检测（一些资源网站的防盗链就是检测Referer）。如果遇到了这一类反爬虫机制，我们可以直接在爬虫中添加Headers，将浏览器的User-Agent部分复制到爬虫的报头Headers中；或者直接将Referer值修改为目标网站的域名。

2.3.2 IP代理

一般来说，网站会针对用户行为进行反爬虫。比如说同一IP地址，短时间内多次尝试访问同一类型的页面，或者同一用户账号短时间内多次进行了相同操作，若发生这些情况，那么很有可能会被认为是网络爬虫。大多数网站的反爬策略都是前一种，对于这种情况，使用代理IP就可以很好的解决问题，可以专门维护一个代理IP池，有了大量的代理ip之后，我们可以每对网站发几次请求 就更换一个ip地址，更换IP的方法，在requests或者urllib2中很容易实现，，这样可以很简单绕过第一种反爬虫机制。对于第二种限制同一账号相同操作次数过多的情况，可以在每次发出请求后，随机的间隔几秒，再进行下一次请求的发出。也可以发出几次请求之后，主动地进行退出登录，然后重新登陆，继续发出请求来绕过同一账户，短时间内进行相同请求的容易被反爬虫的限制，但单一账号如果操作次数过多，也很容易被发现，这种时候，最好可以维护一个账号池，将多个账号放入其中，可以达到更好的效果。

2.3.3 Cookie

有些网站需要用户登录后才能访问某个页面和获取定制化的信息，在登录之前，如果想抓取某个页面内容是不允许的，因为根本就获取不到，网页源码中就没有这部分内容。所以，我们可以先在浏览器中进行登录，然后利用urllib2库保存登录的Cookie，在爬虫发出的请求中，将Cookie带上，就可以模拟出用户登录了，从而获得登陆后才可以得到的信息。

2.3几种开源爬虫框架

分网络爬虫框架主要可以根据规模分为两种：分布式爬虫，和单机爬虫。

现在比较流行的分布式爬虫，是Apache公司的的Nutch。可是对于大多数的用户来说，Nutch却是这几类爬虫里，最不好的选择，因为Nutch主要是为搜索引擎而设计的爬虫，而大多数用户需要的只是一个能够做精准数据爬取的爬虫。在Nutch运行的工作流程里，有三分之二是为了设计一款搜索引擎而编写的。对精确地数据搜集没有太大的意义。也就是说，不推荐使用Nutch做数据抽取，这会浪费很多的时间在不必要的计算上。而且如果你想要通过对Nutch进行二次开发，来让它很好的适用于精确数据爬取的任务上，基本上就是要破坏Nutch的完整框架，这将会把Nutch改的面目全非，况且，如果有修改Nutch的能力，还不如自己重新设计一个分布式爬虫框架。而且，Nutch是依赖hadoop运行的，而hadoop本身会消耗很多的资源与时间。如果服务器数量较少，爬取速度反而不如单机爬虫快。所以，如果目的不是要设计一款搜索引擎，那就尽量不要选择Nutch作为爬虫。

单机爬虫，单机爬虫的开源框架有很多种，比较出名的有Larbin（C++），scrapy（python）。Larbin设计目的是能够跟踪所爬取的页面的url，并进行扩展的抓取。 Larbin只是一个单纯的爬虫，也就是说Larbin只有抓取网页的功能，解析数据的事情则由用户自己完成。另外，Larbin也不提供如何存储到数据库以及建立索引的解决方案，这些都交由用户去自行处理。Latbin的最初的设计原则是简单、高度可配置，所以基于Larbin的爬虫是非常高效的。再来说说Scrapy，Scrapy是一个为了爬取网页内包含的数据、提取结构性数据而编写的基于Python的爬虫应用框架。 它可以被应用在包括数据挖掘，信息处理或存储历史数据等一系列的程序中。 其最初开始的设计目的是页面抓取(或者说网络抓取)， 也可以应用在获取API所返回的数据或者编写通用的网络爬虫之中。下面的图表展现了Scrapy的架构，以及scrapy的各个组件在系统中发生的数据流向的概览。并对每个组件都做了一些简单的介绍。



Scrapy Engine引擎是用来控制整个Scrapy系统的数据处理流程的核心部件，也会进行事务处理的触发。Scheduler（调度器），调度程序从Scrapy引擎接受请求并将请求进行排序，再列入队列，并在Scrapy引擎发出请求后，将结果返还给他们。Downloader下载器的主要职责是根据url地址抓取网页，并将网页内容返还给蜘蛛( Spiders)。Spiders蜘蛛是由用户自己定义的，用来解析网页内容、抓取指定的URL链接并返回的类，每个Spiders都可以处理一个域名或一组域名。每个Spiders都可以用来定义特定网站的抓取和解析规则。

2.4网络爬虫的几种应用

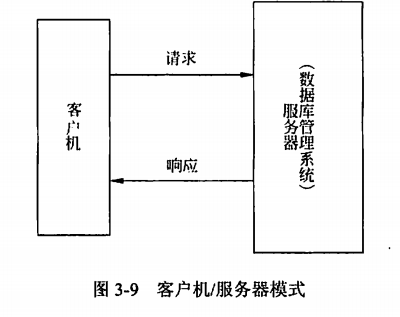
行随着互联网的蓬勃发展，每天都有海量的数据汇入其中，但是这些数据是分布在无数个网页上的，用户如果有需要查找的信息，那么就需要有一种能有效而快速的信息搜寻方式为其提供服务，有时候人们需要快速、便捷的获取和一些关键词相关的信息，这时候搜索引擎就能够为我们提供帮助，例如百度搜索、谷歌搜索等，这些搜索引擎也是通过无数个爬虫，将网络上林林总总的信息 全部进行了爬取，用户如果需要知道天气，那么百度一下天气关键词，就可以立刻得到各种天气网站的链接地址。除此之外，如果用户有一些个性化的数据采集需求，比如我想要知道知乎网站上，点赞数排名前十的问题与回答，或者我想要对新浪微博上一段时间之内的所有微博做一下数据分析，那么传统的使用关键词搜索的方式是无法满足这种需求的，这种情况下，就需要进行个性化订制的爬虫，才能够满足形形色色的需求，毕竟互联网包含了无数、海量的数据，通过网络爬虫进行合理的挖掘与分析，可以让其隐藏的价值得到更充分的体现。

第三章□□基于客户端/服务器模式下的软件开发

3.1客户端/服务器模式介绍*（四号、黑体、顶格）*

客户端／服务器( Client/Server，也即c/s)模式，是基于网络资源的不对等，为了实现资源共享共享而提出的一种模式。C/S模式将应用一分为二，服务器（后台）负责数据的存储管理与处理客户端发送的请求，客户机（前台）完成与用户的交互任务。Client/Server模式具有十分强大的数据操作和事务处理的能力，而且模型的核心思想比较简单，易于让人们理解与接受。

下图是c/s模式的结构示意图，由两部分构成：前台是客户机，通常是用户的个人电脑端。后端是服务器，能够接受预处理客户端发送的请求。运行数据库管理系统，提供数据库的查询、管理功能，



3.1.1什么是C/S架构

客户端/服务器模式，是一种网络架构，简称C/S架构，这种模式把客户端（client）与服务器（Server）区分开来，每一个客户端都可以向一个服务器程序发出请求。C/S模式是一种概念上的叫法，它并不是一种是网络设备或计算机设备。在这种模式中，发送出请求的是客户，而进行响应的一方被称为服务器。有的时候，服务器可能不止一台，一台服务器无法完成某项工作，那它就会向其它服务器发送请求，这种情况下，发送请求的服务器，就相当于是另一台服务器的客户端。从一个连接建立的角度上看，主动启动通信的是客户端，被动的等待通信连接的是服务器。与C/S对应的模式，就是B/S（浏览器/服务器模式）。

3.1.2 C/S架构与B/S架构的区别

B/S结构（Browser/Server，浏览器/服务器模式），是web网站兴起之后诞生的一种网络结构模式，浏览器是客户端最主要的一款应用软件。这种模式下，相当于统一了客户端，将系统全部功能的核心部都分集中到服务器上，让服务器去处理、分析数据，极大的简化了客户端的开发任务。客户机只要安装一个浏览器，如Chrome或Microsoft Edge，服务器端安装数据库管理系统，例如SQL Server或MYSQL等。浏览器通过向服务器发起请求，服务器响应请求，并对后端的数据库进行操作，同时，B/S模式因为用户端是浏览器，那么就可以做到平台无关性。即使是不同的操作系统，用户都可以通过通用的浏览器进行访问。

同时，我们也需要关注到相比B/S来说C/S模式的优缺点。C/S模式的有很明显，客户端与服务器直接连接，没有中间环节，响应速度快，而且由于客户端软件是安装在pc的，那么这种模式可以充分发回PC的性能，很多数据可以在客户端处理后再提交给服务器，降低了服务器的工作负荷，可以提高服务器并行处理的请求数量。同时这种模式是将应用于服务分离，是系统具有稳定性与灵活性。但是缺点也非常巨大，那就是对客户端的操作系统会有限制，比如适用于window7的客户端安装包可能就不适用于windows8系统，更不用说Linux、Unix等与windows完全不同的操作系统，那一定是要针对每一种操作系统都开发一套客户端软件出来的同时由于产品发展较快，那就意味着版本迭代也很频发，代价较高。

3.2客户端与服务器的通信过程及原理*（四号、黑体、顶格）*

*NET*应用程序基本都是使用客户端服务器架构的。这里的客户端与服务器指的是两个进程或两个应用程序，通过网络在互通信和交换信息。两个过程中，一个作为客户端进程，另一个进程作为服务器进程。其中客户端是主动发起通信的一方，而服务器是被动的接收请求、并相应请求的一方。

要想了解学习客户端与服务器的通信方式，首先要了解一下Socket。Socket的英文含义是“插座”，而在网络编程中，它常被称为“套接字”，套接字被用来描述IP地址和端口，是一个通信链的“句柄”。Socket就像它的英文含义“插座”一样，以打电话为例，我们想要打电话，不但要知道对方的电话号码，而且还要买一部电话机。电话两端的通话双方相当于相互通信的两个程序，电话号码就相当于是对方的IP地址，而买一部电话机就相当于申请一个Socket套接字。然后向对方进行呼叫，相当于主动发起了连接请求，如果另一端此时处于空闲状态，那么当他拿起话筒，双方就就可以进行交流通话了，相当于连接已经建立成功。而双方通话的过程，是一方向电话机发出信息，另一方接收到消息的过程，相当于是向Socket写数据和从Socket读数据。

既然这个通信的流程是双方一起建立的，那么一定要有一套规范的建立起连接的过程。下面就是客户端与服务端通过Socket建立起TCP连接和关闭连接的过程。



服务器端首先调用socket()创建一个套接字，但此时套接字只是一个空的结构，我们需要让它绑定一个地址，地址包括IP地址与端口，所以再通过bind()函数进行端口绑定，之后服务器通过listen()函数监听某一个端口，因为服务器不知道谁会连接自己，也就是说服务器不知道下一个连接自己的客户端的信息，所以它只能事先和客户端约定达成一致性：客户端会将请求发送到服务器指定的某个端口，那么当请求到达后，服务器就可以通过listen()函数监听到这个请求了。完成之前的一些列初始化工作后，服务器会调用accept()函数，accept()函数是一个阻塞函数，它会一直阻塞直到有客户端发出的请求连接消息到达监听的端口。

客户端调用socket()创建Socket套接字进行初始化工作，再调用connect()发出SYN段并阻塞等待服务器应答，此时客户端处于SYN\_SENT状态，服务器接收到请求后，会应答一个SYN-ACK段，此时服务器处于SYN\_RCVD状态，当这条消息到达客户端口，客户端会处于ESTABLISHED状态，客户端确认这条消息之后，会向服务器发送一条ACK段消息，并从coinnect()返回，服务器收到这条ACK消息之后，会从从accept()返回，此时双方都处于ESTABLISHED状态。这样，一条连接就建立成功了。这就是俗称的TCP“三次握手”。



TCP协议提供全双工的通信服务，全双工意思就是双方同时可以收发消息，就像我们目前使用的手机，就是全双工的通信方式。一般来说，客户端/服务器程序的流程，是由客户端主动的发起连接请求，而服务器端则属于被动的接收与处理请求，就像是一问一答的方式。服务器从accept()函数返回后，会立刻调用read()函数，读取客户端通过TCP协议传输过来的内容，因为TCP协议是面向连接的可靠地协议，所以一般来说无需担心内容会出错。读socket套接字就像读管道一样，如果没有数据到达就会陷入阻塞等待的状态，

客户端调用write()函数发送请求给服务器，服务器收到后从read()函数的阻塞状态返回，并对客户端的法所过来的请求进行处理，在此期间客户端会调用read()函数并阻塞等待服务器的应答。服务器调用write()函数，将处理结果写到Socket套接字中，通过TCP协议发送给客户端，然后再次调用read()阻塞等待下一条请求，客户端收到服务器的应答后，后从阻塞状态的read()函数返回，并发出下一条请求，再如此循环下去。

如果客户端没有更多的请求需要发送了，那它就会调用close()函数关闭这条连接，此时会通过TCP协议发送一条FIN段，协议层若接收到这个字段，那么read()函数就会返回0，那么服务器就知道客户端关闭了连接，它也会调用close()函数关闭连接。注意，这个地方不建议调用shutdown()函数关闭连接，因为TCP协议是全双工的，也就是说，一端关闭了连接，只是意味着这一方不准备再向Socket中写入数据发送到另一端，但是仍旧可以接收到对方发送数据。也即如果一方调用shutdown()函数，则连接处于半关闭状态，仍可接收对方发来的数据。如果想要彻底关闭这条连接，那就应该调用close()函数而不是shutdown()函数，这样连接的两个传输方向都被关闭，就不能再发送数据了。



同时，需要注意，服务器端最少要有两个socket，其中一个是负责监听指定端口，监听是否有客户端发来的连接请求，但这个套接字不负责与请求的客户端进行通信，同时这个套接字是比较特殊的。另一个套接字是每当服务器端成功接收到客户端新的连接请求时，会在服务器创建一个用与发出请求的客户端进行通信的socket套接字，之后服务器都会使用这个套接字与客户端通信，而第一个套接字仍在监听是否有新的连接请求到达。

3.2.1 底层网络协议

网络编程中使用Socket进行多机之间的互联与通信，在创建Socket的时候，可以指定使用的底层网络协议。Socket可以支持不同的OSI第4层（传输层）协议，包括大名鼎鼎的TCP协议和UDP协议。简单来说，TCP协议是基于连接的协议，每条被发送的消息都需要对端进行确认，才能证明这条消息对方接收成功，所以可靠性较高。而UDP协议是面向无连接的协议，也就是说，它在发送消息之前，不需要建立连接，可以直接将消息发送到目标地址，因为对端不需要确认消息，因此UDP协议不是可靠的协议，相比来说，使用UDP协议来传输，效率要高一点。但是，出于对所传输的信息的可靠性的保证，我们选择了TCP协议来进行本课题的底层网络协议，下面对TCP协议进行简单的介绍。

使用TCP协议建立连接时，客户端和服务器总共需要发送3个包。俗称“三次握手”。在断开连接的时候，在客户端与服务器之间需要发送四个包，俗称“四次挥手”。下面是TCP报文格式的内部结构：



TCP报文中有几个字段需要重点的解释一下：

消息序列号：Seq，大小为32位，用来标识从tcp发送端向目的端发送的字节流的片段编号，发送方发送数据的时候，会对此字段进行标记。

确认序列号：Ack，大小为32位，当ACK标志位为1时，这条字段才有效，且Ack=Seq+1，有两个含义，不但意味着确认成功，而且也指出了吓一跳需要的消息的编号。

还包含6个标志位：即URG、ACK、PSH、RST、SYN、FIN等，具体含义分别是URG：紧急指针位；ACK：确认序号位；PSH：接收方应该尽快将这个报文交给应用层；RST：重置连接位；SYN：新连接标志位；FIN：结束连接标志位。

三次握手的目的是根据IP地址连接服务器的指定端口，和在客户端和服务器之间建立起一条TCP连接,并同步连接双方的序列号和确认号，同时交换TCP窗口大小，因为TCP有打开拥塞控制的机制，就是通过滑动窗口来进行流量限制的，所以滑动窗口的大小，对通信双方来说，都是一个一直在动态调整的数值，需要及时告诉对端。在Socket编程中，客户端主动执行connect()的时候，将会触发TCP的三次握手。

3.2.2 I/O多路复用

在linux下编写服务器程序的时候，我们不得不考虑的一个问题就是，服务器程序能否同时处理多个客户端的发起的连接请求。

由于进程的执行过程是线性的(即顺序执行)，所以我们调用系统I/O(read()，write()，accept()等等函数)，进程可能会被阻塞，此时进程就阻塞在这个I/O调用上，无法执行其他操作。这种情况下，若是一个服务器进程和一个客户端进程通信，服务器端调用read(sockfd1，bud，bufsize)，此时客户端进程没有发送数据，那么read()函数将会阻塞住，直到客户端调用write(sockfd，but，size)发送来了数据。在只有一个客户端和服务器通信时，这没什么问题，但是当有多个客户端与服务器进行通信时，若服务器阻塞于其中一个客户的套接字sockfd1，那么当另一个客户的数据到达套接字sockfd2时，服务器是无法处理的，因为它仍旧阻塞在read(sockfd1，bud，bufsize)上；此时问题就出现了，服务器不能及时处理另一个客户端的服务请求。如果是同步阻塞模型下使用多线程的方式的处理，那就是在主线程中创建一个循环，每当有新的客户端的请求到达的时候，主线程监听到这个请求，会创建一个新的线程，并将这个与这个会话相关的套接字作为参数传给新的线程，从而让新的线程全权负责处理这个客户端的请求，新的IO请求会阻塞在新的线程中，但这种同步则色模型下的多线程的方法最致命的一个缺点是，每个进程虽然在理论上拥有全部内存大小的虚拟内存空间，但是实际上操作系统也是要占用一部分的，而且每一个线程都拥有自己的线程栈，这个栈的大小一般来说是固定的，这就意味着，一个进程能开辟的线程数量是有限制的，一般来说，4G大小的linuxx系统的计算机，能最多开辟出不到两千个线程，这就意味着，如果采用这种方法，服务器最多只能为不到两千个客户端服务，是不具有上万的并发的能力的。面对两种情况，就必须使用I/O多路复用来解决。

I/O多路复用是建立在内核提供的多路分离函数select（）和poll（）的基础之上的，使用select（）可以避免同步-非阻塞IO模型中的循环等待的问题。事实上，select（）和poll（）这两个函数也会使进程阻塞。



Select（）被调用后会先阻塞，当有活动套接字才返回到用户线程，但是和阻塞I/O不同的是，select（）和poll（）不是阻塞在用户线程，所以这两个函数可以同时阻塞多个I/O操作，并且可以同时对多个读操作和多个写操作的I/O函数进行状态检测，直到被监测到有数据可读或可写（实际上就是监听多个socket）。Select（）函数被调用后，进程会被阻塞，内核负责监视所有select负责的socket套接字，当有任何一个socket的数据已经准备好，select就会返回套接字可读的消息，用户就可以调用recvfrom（）处理数据。正因为阻塞IO函数只能阻塞一个I/O操作，而I/O多路复用模型缺能够阻塞多个I/O操作，可以在单线程的情况下同时键测多个描述符的状态，而这是同步阻塞IO模型中通过使用多线程的方式才能够达到的，所以才把它叫做多路复用。

第四章□□基于网络爬虫的天气搜索工具的设计与实现

4.1总体框架介绍*（四号、黑体、顶格）*

况本课题的软件架构设计，主要包括以下几部分。

服务器：包含一个用来保存天气信息的Mysql数据库，和一个长期运行的用来相应客户端请求的Server程序，以及使用Scrapy框架编写的网络爬虫程序。

客户端：使用Qt编写的客户端，用户登录与查询天气都在这个客户端上操作。

下面是天气搜索工具软件的各个功能模块：



4.2 数据爬取

数据爬取部分，是由Python编写的的网络爬虫程序实现的。网络爬虫程序在编写好后，可以自动的、不受用户干扰的运行，并从目标站点搜索与下载内容。

本课题使用的框架是Python的开源爬虫框架Scrapy，它被用于抓取Web节点并且能通过Xpath技术从html文档中提取结构化的数据与信息。它最大的好处是提供了很多类型的爬虫基类，我们可以根据个人的需求进行选择和修改。

爬虫代码的文件结构组成如下图：



对爬虫项目的主要文件的说明如下：

scrapy.cfg：当前项目的配置文件信息，主要是为Scrapy命令行工具提供了一些基础的配置信息。

\_\_init\_\_.py:一个python包是一个带有特殊文件 \_\_init\_\_.py 的目录。\_\_init\_\_.py文件定义了包的属性与方法。事实上它也可以什么都不定义，可以只是一个空的文件，但是这个文件必须存在。因为如果 \_\_init\_\_.py 不存在，这个目录就仅仅只是一个目录，而不会被认为是一个包，那么它就不能被python程序导入，或者在它内部包含其它的模块和嵌套其他的包。

items.py：用于数据存储模板的设置，让数据结构化，如天气、风速、日期等数据。

pipelines.py：用于处理从爬虫Spider返回的数据。

settings.py：整个爬虫项目的配置文件，包含了很多配置内容，比如延迟下载的时间，是否遵循网站的robots.txt的口令约束，以及并发数量等。

spiders：爬虫程序所在的目录，里面有爬虫的主体文件。

middlewares.py:下载中间件是处于引擎和下载器之间的一层组件，可以有多个下载中间件同时被加载运行。

cron.sh:自己编写的定时脚本任务，用来定期进行数据库内容的保存，以及爬虫运行日志的保存。

weather.log:保存爬虫程序的运行日志，以便出错误时方便定位错误。

爬虫项目的总体框架基本就是这样，接下来将介绍各个模块的功能是如何用Python语言来实现的。

4.2.1 网站结构的分析

本课题中，爬虫所爬取的网站是中国天气网（http://www.weather.com.cn）

爬虫程序从中国天气网中爬取相关的城市的天气数据，而在设计爬虫程序之前，我们需要先对网站的网页结构进行分析，了解各个部分的布局，以及关键的天气信息的位置在哪里

事实上，Python安装的时候就自带有html/xml的解析器，这个解析器可以很好地处理不规范的标记并生成相关的剖析树。Lxml库支持Xpath的规范，XPath 是一门在 XML 文档中查找目标信息的语言。XPath 可用来在 XML 文档中对目标元素和相关属性进行遍历查找与定位。同时，在Python中也可以不采用xpath而采用正则表达式来匹配自己所要的信息。正则表达式是一种用来根据一定的规则来匹配字符串的非常强发个武器、工具。它的设计思想是通过使用一种描述性的语言来给需要查找的字符串定义一个规则，凡是符合这个规则的字符串就是匹配了，否则，就认为该字符串是不合法的。

分析网页还需要知道HTML的具体结构，HTML的全称是“超文本标记语言”，所谓的标记就是通过很多个标签来描述一个网页的具体部分。以<>开始以</>结束的标志都是标签。这些标签通常是成对的出现，标签的内部也可以嵌套子标签，也即类似于先辈-子孙的关系。位于最外面的html标签是所有标签中级别最高的，它没有父节点。Head节点、body节点次之，层层嵌套，最后才是包含了具体内容的文本、链接等信息。一些常用的标签如下：

<head>：这个节点包含了很多与网页相关的重要信息。例如，网页的标题在<head>下的<title>中定义，我们也可以在<head>下的<style>节点中定义对网页内容使用的CSS样式。

<body>：这个节点包含了更多的子节点，基本上我们都可以看到的网页内容（文本、链接、图片等）都在<body>节点中。例如：

<P>标签，表示一个段落；

<H1>标签，它的内容是文本部分的标题；

<a>标签，内部包含了url链接；

<Img>标签，表示包含有图片链接；

<Form>标签，表示一个表单；

<div>标签，表示一个区块。

知道网页结构与标签的含义，我们已经可以对中国天气网的网页源码进行具体分析了，从而写出爬虫去抓取这些内容。

下面是中国天气网的部分网页源码：



从中可以看出，西安近七天的天气信息，是保存在<div id="7d" class="c7d">标签下的<ul class="t clearfix">标签的第一至第七个<li class="\*">子节点中的，这里用一个具体的例子来进行分析某一天的天气数据：



从图中可以分析出，<li>节点潜逃了一些子节点，这些子节点分别包含了24日这一天的全部天气数据，包括：

<h1>节点，日期

<p class=”wea”>节点，天气气象

<p class=”tem”>下的<span>节点：最高温度；<i>节点：最低温度

<p class=”win”>节点，<em>节点：风向；<i>节点：风速

这是对网页上“七天”按钮所链接的网页内容的分析，通过分析，现在我们已经知道<div id="7d" class="c7d">标签下的节点都包含得的信息以及信息位于什么位置。

要获取这些数据，我们可以通过xpath技术进行定位：

sel = response.xpath('//\*[@id="7d"]/ul/li')

item['weatherWea']=sel.xpath('p[1]/text()').extract()

item['weatherTem1']=sel.xpath('p[@class="tem"]/span/text()').extract()

item['weatherTem2']=sel.xpath('p[@class="tem"]/i/text()').extract()

item['weatherWin']=sel.xpath('p[@class="win"]/i/text()').extract()

通过这种办法，成功的提取出了未来七天的天气气象、风向、风速等信息，并将其根据items.py中定义的字典结构，添加到item内，传给pipeline进行解析与存储。

但是只知道近七天的大致天气走向是不够的，我们还需要获得每一天的各个时段的天气走向，需要获得实时的天气数据，所以接下来还需要对“今天”按钮所链接的网页的源码进行分析。

通过Chrome的元素审查，找到了一天内各个时段的天气走向的标签的位置：



可以看到，所有内容位于div<class=“curve\_livezs”>节点下，其中温度位于div<class = tem>的子节点，时间位于div<class=”time”>下的子节点。如果我们能够得到这两部分内容，就可以得到某一天之内各个时段的详细的温度走势了。

所以，我再次根据通过这种分析方法，得到了信息的xpath：

item['weatherTime']=response.xpath('//\*[@id="curve"]/div[1]/em[%d]/text()' % i).extract()

item['weatherTem']=response.xpath('//\*[@id="curve"]/div[4]/em[%d]/text()' % i).extract()

item['weatherWea']=response.xpath('//\*[@id="curve"]/div[6]/em[%d]/text()' % i).extract()

但是后来运行爬虫程序的时候，发现无法通过这些节点的xpath获取到任何数据，所以进行了深入的探究，我发现网页的源码中并没有这些节点，但是在Chrome的元素审查中却有这些节点，并且可以通过开发者工具提取出这些节点的xpath。于是我意识到，这部分的节点实际上是由浏览器解析网页源码并动态加载出来的，不是实际存在于网页源码中的内容，而爬虫程序只会下载网页源码并对网页源码进行解析，没有做浏览器动态加载JS生成新的内容那一部分工作，所以爬虫程序是无法获取到这部分信息的。

但是，这部分信息虽然是浏览器动态解析加载出来的，可原始的数据一定是存在于源码中的，所以我再次对网页源码进行分析，发现全部的天气数据都在一个标签内，这个标签的位置如下图：



从而可以提取出它的xpath：

sel.xpath('//\*[@id="today"]/script/text()').extract()

所以，未来七天的天气走向，和每日的各时段的天气信息都可以通过xpath分析得到其内容，下一步就可以进行爬虫程序的设计与实现了。

4.2.2 Spider的设计实现

网络爬虫的各个模块的功能在4.2节已经做了简单的介绍，下面详细描述其实现细节。

Items.py主要包含了一个类TodayWeatherItem，其功能是配置需采集页面的字段实例。

class TodayWeatherItem(scrapy.Item):

content = scrapy.Field()

city\_code = scrapy.Field()

因为中国天气网上一个页面只有一个城市的详细天气信息，而且上一小节提到了，网页上“今天”按钮所链接的网站的一些标签都是浏览器动态加载生成的，而这些标签的内容全部都保存在一个<script>标签中，所以爬虫只需要每次从网页中获两个标签内容就可以，一个是<script>内的json串content，另一个是从当前网页源码中得到这些天气数据所属的城市，也就是获得城市天气代码city\_code。

settting.py主要包含的是一些配置项，以及一些数据库的连接参数。

BOT\_NAME = 'today\_weather'

SPIDER\_MODULES = ['today\_weather.spiders']

NEWSPIDER\_MODULE = 'today\_weather.spiders'

DEBUG = True

if DEBUG:

dbuser = 'root'

dbpass = '123456'

dbname = 'bs\_db'

dbhost = '127.0.0.1'

dbport = '3306'

else:

dbuser = 'root'

dbpass = '\*\*\*\*\*'

dbname = 'bs\_db'

dbhost = '\*\*\*\*\*\*\*\*'

dbport = '3306'

#遵循robots.txt中的口令约束

ROBOTSTXT\_OBEY = True

ITEM\_PIPELINES = {

#'today\_weather.pipelines.TodayWeatherPipeline': 300,

'today\_weather.pipelines.MySQLStorePipeline': 400,

}

weatherSpider.py,是爬虫程序的主体，用来爬取网页内容，并存入item中。

在设计上，爬虫程序需要实现这样的功能：从数据库读取用户注册的城市信息，再根据城市天气代码生成url，将url传给爬虫进行网页下载，再对下载在的内容使用xpath技术提取出我们想要的内容。所以，这个爬虫程序是分为好几各模块的，摸一个模块都有各自的功能，我会分开介绍。

get\_urls()函数，用来从数据库中读取用户注册的城市列表，从而根据城市天气代码生成目标url，再根据url进行网页爬取。下面是get\_urls()函数的实现。

import MySQLdb

from ..settings import dbuser,dbpass,dbname,dbhost,dbport

#从settings中，扩展出几个数据库连接参数，供爬虫程序使用。

def get\_urls():

try:

conn=MySQLdb.connect(user=dbuser, passwd=dbpass, db=dbname, host=dbhost, charset="utf8", use\_unicode=True)

cur=conn.cursor()#填写参数并连接数据

cur.execute('select \* from register\_city;')#执行查询

s=[]

results = cur.fetchall()#获取全部查询结果

result=list(results)

for r in result:

#将数据库中的城市代码列表，构建成url列表的形式，并返回url列表

s.append(("http://www.weather.com.cn/weather1d/%s.shtml" % r))#生成url列表

conn.close()

return s

except MySQLdb.Error,e:

print "Mysql Error %d: %s" % (e.args[0], e.args[1])

爬虫主体类是CatchWeatherSpider类，继承自scrapy.Spider类，功能是下载网页并进行内容解析。

class CatchWeatherSpider(scrapy.Spider):

name = 'weather'#爬虫程序的名字

allowed\_domains = ['weather.com.cn']

start\_urls = get\_urls() #获取初始url列表，将爬取这个列表。

#下载后的网页内容，交由parse()函数处理

def parse(self, response):

sel=Selector(response)

#调试

sites=sel.xpath('//\*[@id="today"]/script/text()').extract()

print('test--1:%s' % sites)

#从网页源码中获取城市代码，用来分辨天气记录是属于哪一个城市的

city\_code\_path = sel.xpath('//\*[@id="someDayNav"]/li[1]/a/@href').extract()

city\_code\_path = city\_code\_path[0].encode('utf-8')

#city\_code\_path 的形式是："/weather1d/101110102.shtml"，需要用正则表达式提取city\_code

city\_code = re.findall(r'/weather1d/(.+?).shtml',city\_code\_path)[0]

#获取网页script节点中存储的全部内容

contents = sel.xpath('//\*[@id="today"]/script/text()').extract()

contents = contents[0].encode('utf-8')

content\_list = re.findall(r'"(.\*?)"', contents)

i = content\_list.index("7d")

for x in content\_list[i+1:]:

item = TodayWeatherItem()

item["content"] = x

item["city\_code"] = city\_code

#这条可以在运行结果上看到每一条记录的实际内容(而不是utf-8码)

print("%s:%s" % (city\_code,x))

yield item

Pipelines.py主要包含了一个 MySQLStorePipeline类，用来处理CatchWeatherSpider类的parse（）函数传来的item对象，并对其中包含的数据进行处理与入库。

class MySQLStorePipeline(object):

def \_\_init\_\_(self):#初始化函数，连接数据库。

self.conn = MySQLdb.connect(user=dbuser, passwd=dbpass, db=dbname, host=dbhost, charset="utf8", use\_unicode=True)

self.cursor = self.conn.cursor()

#查询之前，清空旧数据：

self.cursor.execute("truncate table weather7day\_full;")

self.conn.commit()

def process\_item(self, item, spider):

curTime = datetime.datetime.now()

try:

self.cursor.execute("""INSERT INTO weather7day\_full (city\_code,weatherDate, weatherWea, weatherTem, weatherWinf,weatherWinl, updateTime)

VALUES (%s, %s, %s, %s, %s, %s, %s)""",

(

item['city\_code'],

item['content'].split(',')[0],

item['content'].split(',')[2],

item['content'].split(',')[3],

item['content'].split(',')[4],

item['content'].split(',')[5],

curTime,

)

)

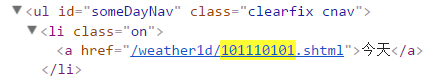
self.conn.commit() #提交修改

except MySQLdb.Error, e:#异常处理

print "Error %d: %s" % (e.args[0], e.args[1])

return item

上面是获得每一天的详细数据的爬虫的实现，还需要对未来七天天气走向的数据进行爬取。所以还需要从网页源码中提取出当前城市的城市天气代码。



通过分析网页源码，得到了城市代码的位置，其xpath是：

sel.xpath('//\*[@id="someDayNav"]/li[1]/a/@href').extract()

再通过正则表达式技术，将连接中的城市代码（高亮部分）取出来：

re.findall(r'/weather1d/(.+?).shtml',city\_code\_path)[0]

下面就是爬取未来七天天气走向的网络爬虫的源码：

class CatchWeatherSpider(scrapy.Spider):

name = 'weather'

allowed\_domains = ['weather.com.cn']

start\_urls = get\_urls()

def parse(self, response):

sel=Selector(response)

#从网页源码中获取城市代码，用来分辨天气记录是属于哪一个城市的

city\_code\_path = sel.xpath('//\*[@id="someDayNav"]/li[1]/a/@href').extract()

city\_code\_path = city\_code\_path[0].encode('utf-8')

#city\_code\_path 的形式是："/weather1d/101110102.shtml"，需要用正则表达式提取city\_code

city\_code = re.findall(r'/weather1d/(.+?).shtml',city\_code\_path)[0]

for sel in response.xpath('//\*[@id="7d"]/ul/li'):

item = WeatherItem()

item["city\_code"] = city\_code

item['weatherDate'] = sel.xpath('h1/text()').extract()

#item['weatherWea'] = sel.xpath('p[@class="wea"]/text()').extract()

#print(response.xpath('//\*[@id="around"]/div[1]/ul/li[1]/a/span/text()').extract())

#print(response.xpath('//\*[@id="curve"]/div[1]/em[1]/text()').extract())

item['weatherWea'] = sel.xpath('p[1]/text()').extract()

item['weatherTem1'] = sel.xpath('p[@class="tem"]/span/text()').extract()

item['weatherTem2'] = sel.xpath('p[@class="tem"]/i/text()').extract()

item['weatherWin'] = sel.xpath('p[@class="win"]/i/text()').extract()

yield item

4.2.3 Spider的伪装与使用代理IP

通过使用User-Agent的方法，将爬虫程序伪装成普通用户使用浏览器访问，可以一定程度上避免网站屏蔽，在爬虫代码中添加如下部分即可：

head = {}

#写入User Agent信息

head['User-Agent'] = 'Mozilla/5.0 (Windows NT 11.0; WOW64; Trident/7.1; .NET4.0; .NET4.0E; .NET CLR 2.0.41647; .NET CLR 3.1.45934; .NET CLR 3.5.45934; InfoPath.3; rv:12.0) like Gecko'

getProxies\_List()是设置代理IP的方法，下面是具体实现：

import requests

def getListProxies():

sessions = requests.session()

pages = session.get("http://www.xicidaili.com/nn", headers=headers)

soups = BeautifulSoup(pages.text, 'lxml')

proxyLists = []

taglists = soup.find\_all('tr', attrs={'class': re.compile("(odd)|()")})

for trtag in taglists:

tdlists = trtag.find\_all('td')

proxys = {'http': tdlists[1].string + ':' + tdlists[2].string,

'https': tdlists[1].string + ':' + tdlists[2].string}

urls = "http://ip.chinaz.com/getip.aspx";

try:

response = session.get(urls, proxies=proxys, timeout=5)

proxyLists.append(proxys)

if(len(proxyLists) == 10):

break;

except Exception, e:

continue

return proxyLists

4.2.3 使用多线程提升爬虫性能

Scrapy框架提供了很多API函数与参数设置的选项，scrapy自带twisted线程池，默认是10个线程。在配置中可以设置线程数：

CONCURRENT\_ITEMS = 100

#在item Pipeline中并行处理的最大并发项数。

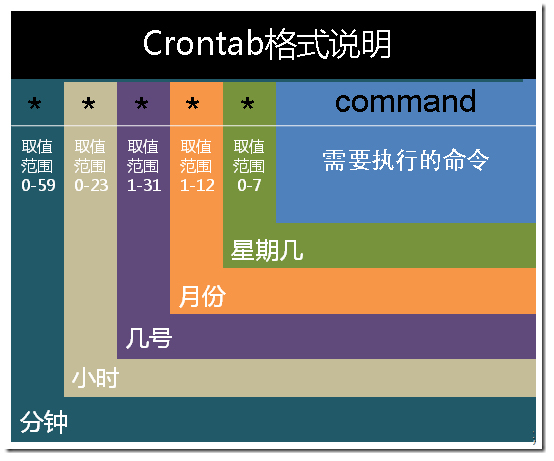
CONCURRENT\_REQUESTS = 32

#Scrapy下载器最大的并发下载量

4.2.3 Crontab实现定时数据爬取

载服务器程序是运行在Linux操作系统下的，爬虫程序需要定期的运行和更新最新的天气信息，但是由于scrapy不自带定时运行的功能，所以需要借助linux的crontab定时任务来实现这个功能。

Crontab的简介：



下面是根据crontab语法编写出的crontab定时任务：

# m h dom mon dow command

#两分钟爬取一次每日各时段的详细天气信息

2 \* \* \* \* sudo sh /home/by/python\_test/today\_weather/cron.sh

#每隔一小时爬去一次未来七天的天气走向。

0 \* \* \* \* sudo sh /home/by/python\_test/weather/cron.sh

4.3 数据存储

数据库存储着整个系统的全部天气数据、用户登录信息、用户详细信息和一份全国城市天气代码对照表。

表4.1 数据库结构表

|  |  |
| --- | --- |
| 表名 | 功能 |
| city\_code | 全部的城市代码：省、市、地区、城市代码 |
| register\_city | 所有已经被注册的城市的城市代码集合 |
| user\_info | 每一个用户所注册的城市代码集合 |
| user\_login | 用户的用户名与密码 |
| weather7day | 近七天的大致天气(仅有每天的最高温度与最低温度 |
| weather7day\_full | 近七天的详细天气（按小时统计），包括今天的详细天气 |

表4.1为数据库中的全部表格及其功能含义。

下面详细介绍每一个表的设计：

City\_code表：

| city\_code | CREATE TABLE `city\_code` (

`province` varchar(20) DEFAULT NULL,

`city` varchar(20) DEFAULT NULL,

`area` varchar(20) DEFAULT NULL,

`city\_code` varchar(20) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COMMENT='全部的城市代码：省、市、地区、城市代码'

Register\_city表:

| register\_city | CREATE TABLE `register\_city` (

`city\_code` varchar(20) NOT NULL,

PRIMARY KEY (`city\_code`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='所有已经被注册的城市的城市代码集合'

User\_info表：

| user\_info | CREATE TABLE `user\_info` (

`username` varchar(20) NOT NULL,

`city` varchar(20) NOT NULL,

PRIMARY KEY (`username`,`city`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='每一个用户所注册的城市'

User\_login表：

| user\_login | CREATE TABLE `user\_login` (

`username` varchar(20) NOT NULL,

`password` varchar(20) NOT NULL,

PRIMARY KEY (`username`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1 COMMENT='用户的用户名与密码'

Weather7day表:

| weather7day | CREATE TABLE `weather7day` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`city\_code` varchar(10) NOT NULL,

`weatherDate1` varchar(10) DEFAULT NULL,

`weatherDate2` varchar(10) DEFAULT NULL,

`weatherWea` varchar(10) NOT NULL,

`weatherTem1` varchar(10) NOT NULL,

`weatherTem2` varchar(10) NOT NULL,

`weatherWin` varchar(10) NOT NULL,

`updateTime` datetime NOT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=29 DEFAULT CHARSET=utf8 COMMENT='近七天的大致天气(仅有每天的最高温度与最低温度)'

Weather7day\_full表：

| weather7day\_full | CREATE TABLE `weather7day\_full` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`city\_code` varchar(10) NOT NULL,

`weatherDate` varchar(10) DEFAULT NULL,

`weatherWea` varchar(10) NOT NULL,

`weatherTem` varchar(10) NOT NULL,

`weatherWinf` varchar(10) NOT NULL,

`weatherWinl` varchar(10) NOT NULL,

`updateTime` datetime NOT NULL,

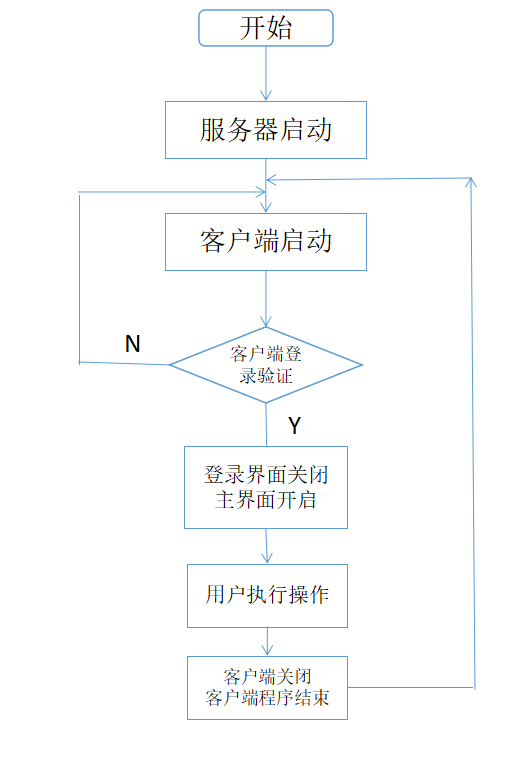
PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=161 DEFAULT CHARSET=utf8 COMMENT='近七天的详细天气（按小时统计），包括今天的详细天气'

4.4 服务器程序的设计实现

服务器程序使用python语言编写，是一个可以长期运行的服务器程序，当有客户端连接请求被发送过来的时候，服务器接手请求并对请求进行处理，根据请求的不同，服务器可以提供的服务分别有：用户登录验证，今日天气信息查询，未来七天天气走向查询，注册城市等功能。

下图是服务器与客户端的启动流程，服务器是长期运行的，客户端用完就可以结束，并在再次需要的时候启动。



4.4.1 使用网络套接字与Client通信

服务器使用socket与客户端通信，下面是服务器创建套接字，并监听指定端口是否有客户端连接，以及数据的发送与接收部分。

import socket

# Address

HOST = ""

PORT = 8955

# Configure socket

skt = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

#AF\_INET的含义是让套接字使用IPV4协议作为网络层协议

#SOCK\_STREAM的含义让套接字使用TCP协议作为传输层协议

skt.setsockopt(socket.SOL\_SOCKET,socket.SO\_REUSEADDR,1)

#SO\_REUSEADDR是让处于TIME\_WAIT状态的端口再次被使用

#套接字绑定端口

skt.bind((HOST, PORT))

#最大连接数为1024

Skt.listen(1024)

#服务器开始监听指定端口

conn, addr = s.accept()

#若连接已经建立，通过套接字接收客户端发送过来的消息

request = conn.recv(1024)

#通过套接字发送回复到客户端

conn.sendall(reply)

#关闭套接字

conn.close()

4.4.2 Server验证登录用户合法性

服务器对客户端传递的参数进行判别，并掉调用不同函数进行处理后，将结果返回给客户端，下面是服务器的判别参数的模块：

# 接收和确认连接

while 1:

conn, addr = s.accept()

request = conn.recv(1024)

print 'Connected by', addr

if request == 'user\_info':

reply = select\_user\_info(request)

elif request == 'user\_login':

reply = select\_user\_login(request)

elif request == 'weather7day':

reply = select\_weather7day(request)

elif request == 'weather7day\_full':

reply = select\_weather7day\_full(request)

#reply = '----ok,here is all info you need----\n'

conn.sendall(reply)

conn.close()

用户登录验证模块，根据用户端传递的参数，验证用户名与密码是否匹配，若用户名存在且密码匹配，则返回”yes”，否则返回”no”。

def select\_user\_login(request):

connect = None

#从参数中取出用户名密码

username = list(request.split(‘,’))[0]

password = list(request.split(‘,’))[1]

try:

connect=MySQLdb.connect(user=dbuser, passwd=dbpass, db=dbname, host=dbhost, charset="utf8", use\_unicode=True)

cur=connect.cursor()

cur.execute(“select password from user\_login where username = ‘%s’;”,% username)

reply = 'no '

result = cur.fetchone()

if not result:

if password == result[0]:

reply=’yes’

if connect:

cur.close()

connect.close()

return reply

except MySQLdb.Error, e:

print "Error %d: %s" % (e.args[0], e.args[1])

sys.exit()

4.4.3 后端数据分析与查询

Scrapy爬虫程序会定时查询天气数据，并更新到mysql数据库中，总是将最新的天气数据保存到文件中。每当用户端发来天气查询的请求，服务器都会将文件内容作为结果返回给客户端。

下面是server程序中处理天气查询的部分：

def select\_weather7day\_full():

f = None

try:

f = open('/var/lib/mysql-files/weather7day\_full.txt')

print('weather7day\_full.txt is opened')

reply = f.read()

print("file content:\n%s"%reply)

return reply

except Exception as err:

print("File operation failed:" + str(err))

return ' '

finally:

if f:

f.close()

print('weather7day\_full.txt is closed')

pass

以及通过定时脚本任务来更新最新的数据到文件中，下面是脚本cron.sh的内容：

#! /bin/sh export PATH=$PATH:/usr/local/bin

#旧的数据文件若存在，则删除这个文件

file="/var/lib/mysql-files/weather7day\_full.txt"

if [ -f "$file" ];then

rm -f "$file"

fi

#启动爬虫程序，将标准输出和标准错误信息都导入到weather.log中

cd /home/by/python\_test/today\_weather/

nohup scrapy crawl weather >> weather.log 2>&1

#执行mysql命令，将最新的数据导出到文件中。

sql\_txt="select \* from weather7day\_full into outfile '/var/lib/mysql-files/weather7day\_full.txt' lines terminated by '\n';"

mysql -uroot -p123456 bs\_db -e "${sql\_txt}"

4.4.4 使用IO多路复用提升Server性能

服务器程序通过使用I/O多路复用函数，可以提升处理性能。

r\_list = [s,]

num = 0

while True:

rl,wl,error = select.select(r\_list, [], [])

num += 1

print('--------counts is %s'%num)

print("--------rl's length is %s"%len(rl))

for fd in rl:

if fd == s:

conn, addr = fd.accept()

r\_list.append(conn)

request = conn.recv(1024)

print 'Connected by', addr

print request

reply = "connect to server succeed!"

conn.sendall(reply)

else:

try:

request = conn.recv(1024)

print 'requests: ',request

if request == 'user\_info':

reply = select\_user\_info()

elif request == 'user\_login':

reply = select\_user\_login()

elif request == 'weather7day':

reply = select\_weather7day()

elif request == 'weather7day\_full':

reply = select\_weather7day\_full()

conn.sendall(reply)

#fd.sendall('second ...,,,,;;;')

except ConnectionAbortedError:

r\_list.remove(fd)

#sys.exit()

conn.close()

4.5 Client的设计与实现

客户端使用Qt进行开发，Qt是基于C++的一套GUI开发库，同时拥有非常丰富的接口，包括网络、线程、绘图库等等。本课题使用Qt开发客户端，用户从客户端登录，进入主界面，并可以从主界面进行城市注册、天气查询等操作。

4.5.1 使用Qt进行GUI开发

使用Qt进行客户端的开发，客户端界面有两部分，包括用户登录界面和登陆成功之后的主界面，QT提供了Qt Designer工具，能很方便的通过鼠标拖拽的方式去绘制界面。绘制完后自动生成一个图形界面的.h文件（如ui\_mainwindow.h），其中含有一个自动生成的Ui\_MainWindow类，这个类中核心的函数是setupUi。登录界面的类是公有继承自QDidalog的login类，界面的切换是在main函数中进行的：

int main(int argc, char \*argv[])

{

QApplication a(argc, argv);

login log; //创建登录界面对象

if(log.*exec*()==QDialog::Accepted){

log.close();

MainWindow w;

w.show();

return a.exec();

}

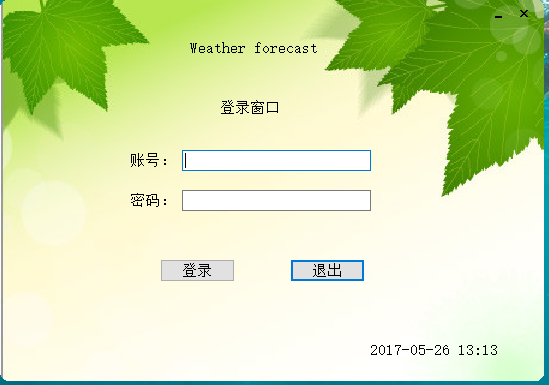
else

return 0;

}

启动log并等待用户输入，当用户点击登录按钮，而且用户名密码匹配成功的时候，login.cpp会调用accept函数，返回一个值Accepted，main函数接收到这个值，会将log关闭，创建一个MainWindow对象并将其显示，然后创建一个事件循环。

4.5.2 用户登录模块设计与实现



用户登录模块主要是通过前台输入用户名密码，然后建立socket连接服务器，将用户名密码通过字符串的形式发送给服务器，服务器在后端进行用户名密码的校验，若校验成功，则返回yes，客户端接收到yes，会通过验证；若收到no，则弹窗“用户名或密码错误”。

void login::on\_pushButton\_clicked()

{

QString user=ui->lineEdit->text();

QString passwd=ui->lineEdit\_2->text();

if( !user\_exist\_true(user,passwd) ){

QMessageBox::warning(this,tr("oh"),tr("User name or Password Wrong"),QMessageBox::Yes);

ui->lineEdit\_2->clear();

}

else{

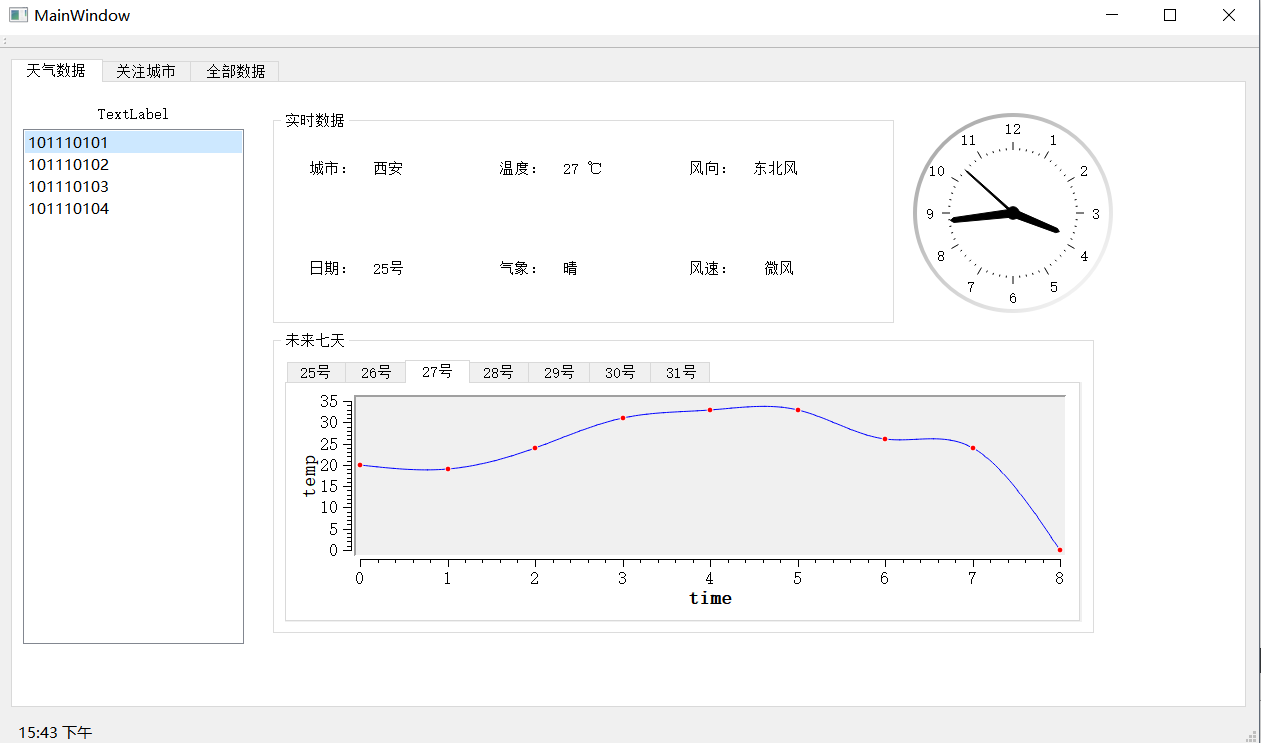
qDebug()<<"OK"<<endl;

accept();

}

}

4.5.3 数据分析与曲线绘制



QVector<double> xs;

QVector<double> ys;

save\_value(xs,ys);

QwtPlotCurve \*curve=new QwtPlotCurve("Temp");

curve->setSamples(xs,ys);

curve->setPen(QPen(Qt::blue));//线条颜色设置为蓝色

curve->setRenderHint(QwtPlotItem::RenderAntialiased);//启用抗锯齿

curve->setStyle(QwtPlotCurve::Lines);//直线形式

curve->setCurveAttribute(QwtPlotCurve::Fitted, true);//是曲线更光滑

curve->setPen(QPen(Qt::blue));//设置画笔

//设置样本点的颜色、大小,Ellipse椭圆或圆，brush填充内部颜色，QPen节点的轮廓颜色

QwtSymbol \*symbol = new QwtSymbol( QwtSymbol::Ellipse,

QBrush(Qt::red), QPen(Qt::white,1), QSize(6,6));

//添加样本点形状

curve->setSymbol(symbol);

//ui->qwtPlot\_5->show();

curve->attach(ui->qwtPlot\_5);

ui->qwtPlot->*replot*();

结束语

*（三号、黑体、居中、与正文空一行）*

经过了近两个月的学习和工作，我终于完成了《基于python网络爬虫的天气信息搜索与预报系统》的毕业设计和论文。从二月多开始接触到论文题目到程序的设计与实现，再到论文的完成，对我来说每一步都是新的尝试和挑战，这个项目课题也算是我在大学生涯中独立完成的最大的一个了。在这几个月里，我学到了很多新的知识，也有很多以前不曾有过的感触。一开始对CSDN、博客园、Github等知名技术分享网站基本是一无所知，对网络爬虫，mysql数据库等相关技术也不甚了解，于是我开始了独立的学习和探索实践，通过查看相关的资料和文献，逐渐了解这些技术的应用背景与发展历程，并让自己的头脑的一些模糊的概念逐渐地清晰起来，使自己功能简单的作品一步步完善起来，每一次代码的修改都给我带来对课题更深刻的理解。我也充分认识到了技术博客、Github这些开源代码网站给我们学习提供的便利，因此我也一直在通过git将自己的代码上传到Github上和他人分享，也感受到网络爬虫技术在我们日常生活中的应用其实十分广泛。

本课题的论文凭借Python丰富的标准库和快速开发的特点，设计了一种基于中国天气网的网络爬虫程序，能为用户带带来快速获取指定城市天气的使用体验。并通过C/S模式，将程序架构分为两部分：客户端与服务器，服务器上运行爬虫程序、server程序、数据库管理工具等，客户端运行一个由Qt开发带界面的client程序。这些模块属于耦合度很低，但通过Socket网络通信，又有机的结合起来，成功实现了预期的功能。

虽然我的论文作品不是非常成熟，仍旧有许多不恰当的地方，比如对python网络编程的理解不是十分深刻，只实现了Select而没有成功实现epoll模式的Server程序；又比如对Mysql数据库的键的关系不太了解，数据库表之间的字段没有能够用主键/外键的方式建立起关系与关联，但每一段代码都有我的劳动，这让我很欣慰。

这次毕业设计的经历也定会让我终身受益，我感受到做毕设是需要自己认真地去对待，是自己真正独立学习和研究的过程，没有学习就没有研究的能力，也不能完成这次毕设，这也为我以后的工作与生活带来宝贵的经验。

致谢

*（三号、黑体、居中、致谢两字空两格、与正文空一行）*

首先感谢导师石薇老师，在我论文开题和写作过程中给予了悉心的指导，在我写作最为困难的时候对我提供了思路和方向。在导师严谨、负责的治学态度影响下，我认真的完成了本篇论文的每一部分，老师不厌其烦的指导和解惑给予我进一步学习和提升的机会。所以，在此对辛苦付出的吴老师道一声真挚的感谢！

同时也要感谢我的各位同学，在学习中互相分享和帮助，是我完成毕设论文的动力。

完成论文的过程中，本人虽参考了大量的文献，但水平有限或许存在一些错误。恳请各位专家老师指正，不尽感谢！

参考文献（三号、黑体、顶格）

[1] [凤祥云](http://xueshu.baidu.com/s?wd=author%3A%28%E5%87%A4%E7%A5%A5%E4%BA%91%29%20&tn=SE_baiduxueshu_c1gjeupa&ie=utf-8&sc_f_para=sc_hilight%3Dperson" \t "_blank)，[孙海艳](http://xueshu.baidu.com/s?wd=author%3A%28%E5%AD%99%E6%B5%B7%E8%89%B3%29%20&tn=SE_baiduxueshu_c1gjeupa&ie=utf-8&sc_f_para=sc_hilight%3Dperson" \t "_blank)，[张万臣](http://xueshu.baidu.com/s?wd=author%3A%28%E5%BC%A0%E4%B8%87%E8%87%A3%29%20&tn=SE_baiduxueshu_c1gjeupa&ie=utf-8&sc_f_para=sc_hilight%3Dperson" \t "_blank).[基于光纤通信技术的物联网传感器系统](http://xueshu.baidu.com/s?wd=paperuri%3A%28c77a9549c24b8365681bc6195aed1b3c%29&filter=sc_long_sign&tn=SE_xueshusource_2kduw22v&sc_vurl=http%3A%2F%2Fwww.cqvip.com%2FQK%2F91041X%2F201607%2F669514878.html&ie=utf-8&sc_us=7665783302043157044" \t "_blank)[J].激光杂志, 2016(7):131-134．

[2] [迪内希·钱德拉·维玛](http://xueshu.baidu.com/s?wd=author%3A%28%E8%BF%AA%E5%86%85%E5%B8%8C%C2%B7%E9%92%B1%E5%BE%B7%E6%8B%89%C2%B7%E7%BB%B4%E7%8E%9B%29%20&tn=SE_baiduxueshu_c1gjeupa&ie=utf-8&sc_f_para=sc_hilight%3Dperson" \t "_blank)，[帕利德·维玛](http://xueshu.baidu.com/s?wd=author%3A%28%E5%B8%95%E5%88%A9%E5%BE%B7%C2%B7%E7%BB%B4%E7%8E%9B%29%20&tn=SE_baiduxueshu_c1gjeupa&ie=utf-8&sc_f_para=sc_hilight%3Dperson" \t "_blank).大数据爆炸时代的移动通信技术与应用[M]. 郎为民,译．北京：机械工业出版社, 2016：20-30．

[3] [刘俊文](http://xueshu.baidu.com/s?wd=author%3A%28%E5%88%98%E4%BF%8A%E6%96%87%29%20&tn=SE_baiduxueshu_c1gjeupa&ie=utf-8&sc_f_para=sc_hilight%3Dperson" \t "_blank)，[赵子岩](http://xueshu.baidu.com/s?wd=author%3A%28%E8%B5%B5%E5%AD%90%E5%B2%A9%29%20&tn=SE_baiduxueshu_c1gjeupa&ie=utf-8&sc_f_para=sc_hilight%3Dperson" \t "_blank)，[徐慧明](http://xueshu.baidu.com/s?wd=author%3A%28%E5%BE%90%E6%85%A7%E6%98%8E%29%20&tn=SE_baiduxueshu_c1gjeupa&ie=utf-8&sc_f_para=sc_hilight%3Dperson" \t "_blank)，[张素香](http://xueshu.baidu.com/s?wd=author%3A%28%E5%BC%A0%E7%B4%A0%E9%A6%99%29%20&tn=SE_baiduxueshu_c1gjeupa&ie=utf-8&sc_f_para=sc_hilight%3Dperson" \t "_blank)．[量子通信技术在电力信息系统保密传输中的应用](http://xueshu.baidu.com/s?wd=paperuri%3A%2807b58d40f1334061b898700237944390%29&filter=sc_long_sign&tn=SE_xueshusource_2kduw22v&sc_vurl=http%3A%2F%2Fcpfd.cnki.com.cn%2FArticle%2FCPFDTOTAL-YDDX201609001069.htm&ie=utf-8&sc_us=2010724533587303457" \t "_blank)[C] [电力行业信息化年会](http://xueshu.baidu.com/usercenter/data/journal?cmd=jump&wd=confuri%3A%285b41c4bb46127d22%29%20%E7%94%B5%E5%8A%9B%E8%A1%8C%E4%B8%9A%E4%BF%A1%E6%81%AF%E5%8C%96%E5%B9%B4%E4%BC%9A&tn=SE_baiduxueshu_c1gjeupa&ie=utf-8&sc_f_para=sc_hilight%3Dpublish&sort=sc_cited" \t "_blank" \o "电力行业信息化年会), 2016

[4] DL/T5344-2006,电力光纤通信工程验收规范[S].2006.

[5] 姜锡洲.一种温热外敷药制备方案：中国,881056073[P].1989-07-26.

[6] 王明亮.关于中国学术期刊标准化数据库系统工程的进展[EB/OL]. (1998-08-16)[1998-10-04].http：//[www.cajcd](http://www.cajcd).edu.cn/pub/wml.tex/980810-2.html.

[7] 丁文祥.数字革命与竞争国际化[N].中国青年报,2000-11-20（15）.

[8] 张志祥.间断动力系统的随机扰动及其在守恒律方程中的应用[D].北京：北京大学数理学院,1998.

[9] World Health Organization.Factors regulating the immune response:report of WHO Scientific Group[R].Geneva:WHO,1970.

*（以上，如果需要两行的，第二行文字要位于序号的后边，与第一行文字对齐。中文的用五号宋体，外文的用五号Times New Roman字体。）*

附录X

*(采用三号字、黑体、顶格，与内容空一行，X表示A，B，C等)*

（小四号宋体，行距1.25倍）×××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××