SinaWeiBoCrawler项目说明

SinaWeiBoCrawler项目旨在从新浪微博上爬取指定用户的微博，以便备份和进行进一步分析。

初期，爬取新浪微博主要分三个步骤：一是登录新浪微博、二是获取新浪微博的页面、三是从页面中抽取出微博的发送者、转发者、内容、时间、地点、发送方式等信息。

登录新浪微博通过使用浏览器上登录微博的cookie实现。初期，使用了CookieContainer类实现，把cookie键值对填入后赋给httpWebRequest.CookieContainer。

获取新浪微博的页面使用HttpWebRequest类实现，通过httpWebRequest.GetResponse().GetResponseStream()方法得到响应请求的数据流，从中得到HTML文本。这一步骤和下一步骤，即从页面中抽取微博信息，在之前的爬取亚马逊商品数据时都实现过，因此感觉难度不大。

开始具体实现后，首先遇到的问题是发现微博页面一次只载入15条微博，后面还有两段15条左右的微博只有在页面拉到底后才会加载显示。在杨寒星的帮助下通过Chrome的网络传输捕获工具找到了这两段微博对应的json数组文件，以及传送这两段json数组文件的地址和地址组成规律（其中最重要的max\_id、end\_id和pagebar；max\_id是指当前页面所显示的微博中最下面一条微博的编号（在HTML文本中以“mid”的属性体现），当页面刚刚显示时，这个值时第15条微博的编号，加载一段微博后，即变为刚加加的那段微博的编号；end\_id是指当前页面头一条微博的编号，这个值在整个页面中范围内是不变的；pagebar是指加载微博段数。请求加载第一段微博，pagebar=0、加载第二段微博，pagebar=1）。解决方法是得到页面HTML文本后分析得到end\_id和max\_id，再构造两段微博的URL，取得json文件，从中提取两段微博的HTML文本。

遇到的第二个问题是发现使用CookieContainer构造的cookie只能用于获取页面，但无法获取两段json文件。解决方法是自行构造cookie字符串赋给httpWebRequest.Headers["Cookie"]，构造的cookie字符串格式为“键=值;……”。在解决这个问题时，由于尚不知是CookieContainer无效的问题，以为是HTTP头构建不正确，所以完全模仿Chrome中捕获的HTTP头设置，设httpWebRequest.Headers[“Accept-Encoding”]=” gzip,deflate,sdch”结果导致获得的是经过gzip压缩过的数据流，发现后还用了网上找到的解压gzip代码获得原始数据，绕了一个弯路。

遇到的第三个问题是发现获取页面后，其中HTML文本只会显示一个微博页面的框架，其中的微博内容等都是在客户端执行js函数后再渲染出来的。解决方法办法是抛弃httpWebRequest，使用WebBrowser控件，在窗体中获得页面，因为WebBrowser控件使用IE浏览器的内核，完全执行浏览器操作，能够自动执行js函数，能够获得js改变过的DOM节点。这样做的效率较低，因为要等浏览器把页面渲染完全，但在当时也没有更好的办法。使用一个页面加载完全的事件来触发后面的获得HTML文本的操作，但是发现在渲染页面的过程中，该事件会被多次触发（经过分析每次触发该事件的URL，发现只要js请求一个链接即会触发该事件），又无法事先确定会触发几次，因此只能使后面的获得HTML文本的操作等待足够时间（设为10秒），保证页面渲染完成。

后来在网上看到把WebBrowser控件写入控制台程序的方法，再次修改这部分代码加入控制台程序。由于这类控件需要运行在一个单独的线程单元中（STA Thread），所以开设了一个单独的线程，并设为STA线程供其运行。由此该程序变为一个多线程程序。

遇到的第四个问题是上述解决HTML文本需要通过js函数生成的办法在爬取其他用户个人主页时不管用。其原因是和爬取自己首页不同，爬取他人个人主页时页面加载完全的事件只触发一次，而触发时事实上js函数还未执行完成（由此可以看出该事件当页面加载完成时即触发，和js完全无关。之前没有问题只是因为js请求了其他链接，在WebBrowser获取其他资源的等待时间中把js函数的执行时间给掩盖了），因此根本无法得到什么时候页面HTML文本生成完成。当WebBrowser开始接受网页后，包含WebBrowser控件的线程又无法再继续执行其他代码或使用其他线程（包括Timer）暂停，因为执行任何轮询/阻塞代码都会使WebBrowser暂停运行，其中的js函数也暂停运行；也没用方法从主线程中获得WebBrowser中HTML文本（基于WebBrowser运行于单线程单元的原因），这使得我无法从WebBrowser中获得HTML文本，只能放弃WebBrowser。

通过杨寒星的帮助，发现页面的HTML文本处于同一页面的js代码段的一个json数组中，因此还是通过从页面的数据流中把该json数组以字符串形式提取出来，再反序列化为json数组，再获取其中的HTML文本。这个过程在GetHtmlFromHttpResponse函数中完成。这样该程序又变为一个单线程程序，所有网络通信部分使用HttpWebRequest和HttpWebResponse完成。

在获取微博页面信息的步骤中，难度也比预想的要大一些。主要原因是微博内容结构复杂。主要问题如下：一个是转发微博：转发微博相当于一个子微博，很多DOM结构和本条微博高度重复，设置过滤器时需要小心分辨；二是“还有X条对原微博的转发”的情况。这是新浪对于多个用户转发同一条微博的情况做出的简化处理，对于这样一个框，也需要额外处理（其也相当于一个子微博，但细节上有所不同，需要不同的爬取逻辑）；第三个是微博内容的多样性：其中有文字、@其他用户、表情图片、#话题#等等，这些需要对微博内容这个部分循环处理，对于不同的元素提取不同的内容；另外使用的HTML解析器对<em>标记很不友好，无法识别，导致额外的编码负担。第四是在获取的文本内容中，含有HTML编码的字符和一些回车、换行符，对于保存内容产生很大干扰。

在编写爬取网页版新浪微博工具的过程中，还完成了爬取移动版新浪微博的工作。相比网页版，移动版的HTML结构简单很多，比较容易。

在编写爬取微博页面代码的过程中，新浪微博的个人主页的URL和页面HTML文本结构也在发生变化。因此通过观察分析其改变，需要适时地修改已有代码。所以才出现了现在在代码中的查询URL。

在完成最基本的爬取微博发送者和微博内容、发送时间、发送方式等之后，还编写了爬取个人主页资料、爬取赞数、评论数、转发数等信息的代码；另外还统一了爬取网页版和移动版微博两者调用的接口，实现了简洁的调试方式。在爬取信息丰富化之后，把存储信息的数据结构从List<Feed>数组变为了User实例，其中能保存更多的用户个人信息。

爬取得到的User实例的保存方式是存为XML文件，既便于序列化，也便于分析时读取文件，进行反序列化直接生成对应的User实例。

完成爬取网页版新浪微博和移动版新浪微博后，有一个问题就是如何方便地调用两个程序，即在每次运行前，需要设定的参数、变量和修改的代码最少。我的解决办法是使MobileCrawler类和WebCrawler类共同实现一个ICrawler接口。这个接口定义一些在Main函数中需要调用的函数，这样在Main函数中只存在一个实现ICrawler接口的实例，调用的是一些ICrawler接口中的方法，至于这个实例是MobileCrawler类还是WebCrawler类，只需修改Main函数中一句给ICrawler赋值的语句即可。

至于cookie的问题，我把cookie保存在一个txt文件中。cookieStr作为一个静态成员，当程序构造一个MobileCrawler类或WebCrawler类的实例时，静态构造函数会调用ReadInCookieFromFile静态函数，用来读取相应的文件，填入cookieStr中。

至此，在爬取一个新的用户时，需要修改的只有两处：一是Main函数中的taskCount值，该值为要爬取的网页数目；二是WebCrawler类或MobileCrawler类中的userID，该静态成员用于保存要爬取用户的ID。为了进一步简化操作，该值应该可以抽象为ICrawler中的一个字段，使得同样可以在Main函数中对ICrawler的实例进行修改，这可以在之后对程序的改进中实现。

除需要访问的URL的基本部分外，整个项目中还有数个硬编码在代码中的数值，主要都是为了使爬虫模拟真人用户的行为，以防被服务器屏蔽。一：如果爬取页面超过25个，则以25为周期，重新生成一个爬虫；二：每次访问页面后暂停7秒，使用Thread.Sleep方法实现。

进一步的，我增加了一个名为FansAndFollowCrawler的类，用于爬取一个用户的关注/粉丝列表。由于某些函数需要的形参不同，因此该类没有实现ICrawler接口。有了爬取网页版微博的坚实基础，爬取用户的关注/粉丝列表的总体技术难度不大。

在爬取用户的关注/粉丝列表时，我增加了可以下载关注者/粉丝的头像的功能。该功能由WebClient类实现，因为下载头像文件无需登录用户，而且该类的DownloadAsync方法可以较方便地下载图片、声音等文件（该方法有相对应的同步版本），因为是异步方法，也不需要阻塞之后的程序，只需当下载完成后触发一个事件，释放相应的WebClient实例即可。

最后，简述一下该项目中各个程序代码文件的主要作用：Fan.cs：定义一个粉丝/关注者及其包含的信息；FansAndFollowCrawler.cs：爬取用户关注/粉丝列表；Feed：定义一条微博及其包含的信息；HttpHeader.cs：定义一个HTTP头；ICrawler.cs：统一的爬虫接口，供WebCrawler类和MobileCrawler类实现；JsonJsPart.cs：定义一个和特定json数组对应的类，供对原始HTML内容中一个特定的json数组进行逆序列化时使用，目的是从该json数组中提取出包含微博内容的HTML文本；JsonWeiboPart.cs：定义一个和特定json数组对应的类，供对包含两段异步加载的微博的json数组进行逆序列化时使用，目的是从该json数组中提取出包含微博内容的HTML文本；MobileCrawler.cs：爬取移动版微博；Program.cs：程序入口和把爬取结果保存为XML文件；User.cs：定义一个微博用户及其包含的信息和微博；WebCrawler.cs：爬取网页版微博。