

使用C#也能网页抓取

修改于2021-09-13 17:56:15 阅读 579



原创

作者介绍



Oxylabs中文站

关注

专栏

文章	阅读量	获赞	作者排名
38	18.7K	114	4753

精选专题

腾讯云原生专题
云原生技术干货，业务实践地。

活动推荐

在编写网页抓取代码时，您要做出的**第一个决定是选择您的编程语言**。您可以使用多种语言进行编写，例如**Python、JavaScript、Java、Ruby或C#**。所有提到的语言都提供强大的网络抓取功能。

在本文中，我们将**探索C#并向您展示如何创建一个真实的C#公共网络爬虫**。请记住，即使我们使用C#，您也可以将此信息调整为.NET平台支持的所有语言，包括**VB.NET和F#**。

01.C#网页抓取工具

在编写任何代码之前，第一步是**选择合适的C#库或包**。这些C#库或包将具有下载HTML页面、解析它们以及从这些页面中提取所需数据的功能。一些最流行的C#包如下：

- ScrapySharp
- Puppeteer Sharp
- Html Agility Pack

Html Agility Pack是最受欢迎的C#包，仅Nuget就有近5,000万次下载。其流行有多种原因，其中最重要的原因是**该HTML解析器能够直接使用浏览器下载网页**。这个包可以容忍格式错误的HTML并支持XPath。此外，它甚至可以解析本地HTML文件；因此，我们将在本文中进一步使用这个包。

ScrapySharp为C#编程添加了更多功能。这个包支持CSS选择器并且可以模拟网络浏览器。虽然**ScrapySharp**被认为是一个强大的C#包，但程序员使用它进行维护的概率并不是很高。

Puppeteer Sharp是著名的Node.js Puppeteer项目的.NET端口。它使用相同的Chromium浏览器来加载页面。此外，这个包采用了async-await风格的代码，支持异步及预操作管理。如果您已经熟悉这个C#包并且需要一个浏览器来呈现页面，那么**Puppeteer Sharp**可能是一个不错的选择。

02.使用C#构建网络爬虫

如前所述，现在我们将演示**如何编写将使用Html Agility Pack的C#公共网络抓取代码**。我们将使用带有**Visual Studio Code**的.NET 5 SDK。此代码已在 .NET Core 3和.NET 5上测试过，它应该适用于其他版本的.NET。

云Webify部署项目

腾讯云自媒体分享计划

入驻云加社区，共享百万资源包。

立即入驻

运营活动



03.设置开发环境

对于C#开发环境，请安装**Visual Studio Code**。请注意，如果您使用**Visual Studio**和**Visual Studio Code**编写C#代码，则需要注意它们是两个完全不同的应用程序。

安装**Visual Studio Code**后，安装**.NET 5.0**或更高版本。您还可以使用**.NET Core 3.1**。安装完成后，打开终端并运行以下命令以验证**.NET CLI**或命令行界面是否正常工作：

```
1 | dotnet --version
```

该行命令会输出安装的.NET的版本号。

04.项目结构和依存关系

该代码将成为.NET项目的一部分。为简单起见，创建一个**控制台应用程序**。然后，创建一个文件夹，您将在其中编写C#代码。打开终端并导航到该文件夹。输入以下命令：

```
1 | dotnet new console
```

此命令的输出应该是已成功创建控制台应用程序的信息。

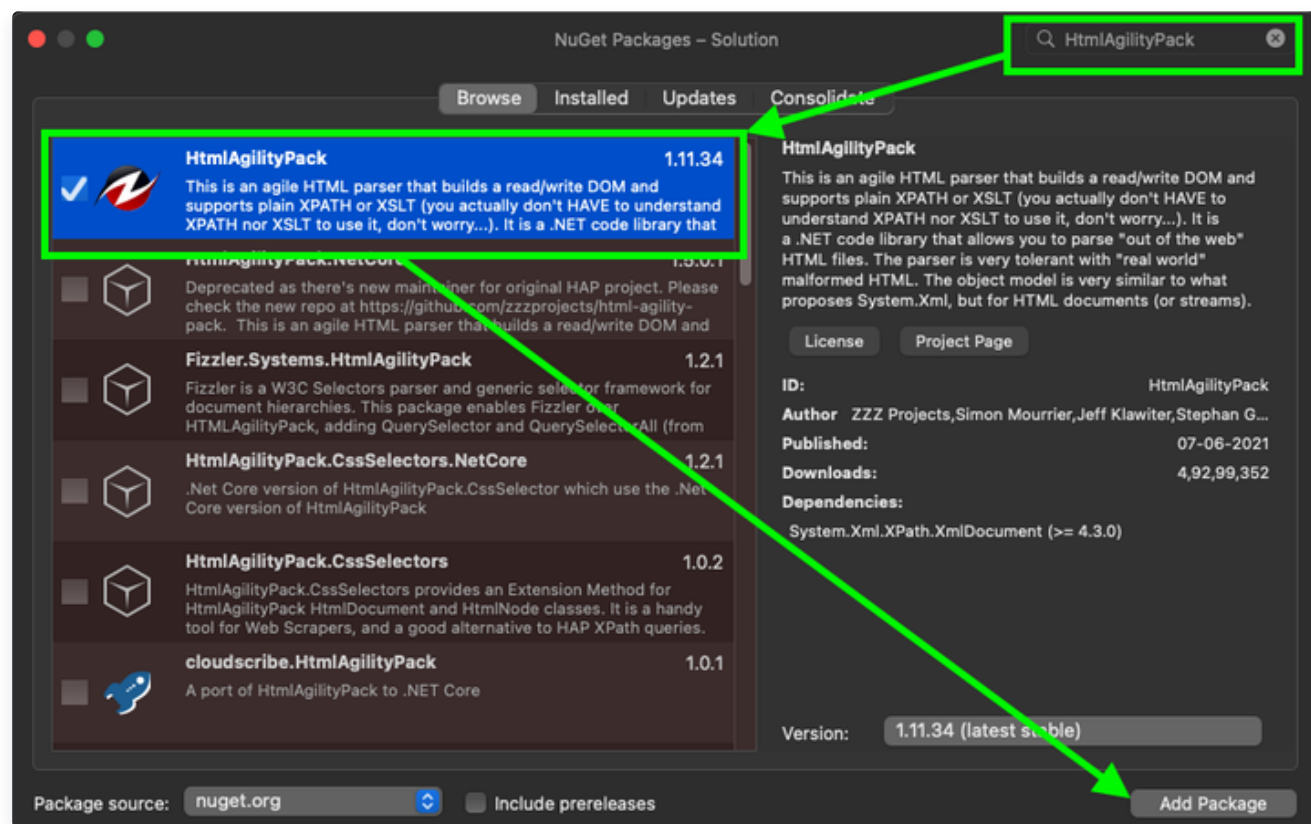
到时间安装所需的软件包了。使用C#抓取公共网页，**Html Agility Pack**将是一个不错的选择。您可以使用以下命令为该项目安装它：

```
1 | dotnet add package HtmlAgilityPack
```

再安装一个包，以便我们可以轻松地将抓取的数据导出到CSV文件：

```
1 | dotnet add package CsvHelper
```

- 选择项目;
- 单击管理项目依赖项。这将打开NuGet包窗口;
- 搜索HtmlAgilityPack并选择它;
- 最后, 搜索CsvHelper, 选择它, 然后单击添加包。



Visual Studio中的Nuget包管理器

安装了这些包后, 我们可以继续编写用于抓取线上书店的代码。

05. 下载和解析网页数据

何URL和浏览器读取和解析文件。

在我们的例子中，我们需要做的就是从URL获取HTML。**Html Agility Pack**没有使用.NET本机函数，而是提供了一个方便的类—**HtmlWeb**。这个类提供了一个**Load**函数，它可以接受一个URL并返回一个**HtmlDocument**类的实例，它也是我们使用的包的一部分。有了这些信息，我们可以编写一个函数，接受一个URL并返回**HtmlDocument**这个实例。

打开**Program.cs**文件并在类中输入此函数**Program**：

```
1 // Parses the URL and returns HtmlDocument object
2 static HtmlDocument GetDocument (string url)
3 {
4     HtmlWeb web = new HtmlWeb();
5     HtmlDocument doc = web.Load(url);
6     return doc;
7 }
```

这样，代码的第一步就完成了。下一步是解析文档。

06.解析HTML：获取书籍链接

在这部分代码中，我们将从网页中提取所需的信息。在这个阶段，文档现在是一个类型的对象**HtmlDocument**。这个类公开了两个函数来选择元素。这两个函数都接受**XPath**输入并返回**HtmlNode** or **HtmlNodeCollection**。

下面是这两个函数的签名：

```
1 public HtmlNodeCollection SelectNodes(string xpath);
2 public HtmlNode SelectSingleNode(string xpath);
```

我们就**SelectNodes**先讨论一下。

对于这个例子——C#网络爬虫——我们将从[这个页面](#)中抓取所有书籍的详细信息。

在了解标记后，您要选择的XPath应该是这样的：

```
1 | //h3/a
```

现在可以将此XPath传递给**SelectNodes**函数。

```
1 | XmlDocument doc = GetDocument(url);
2 | XmlNodeCollection linkNodes = doc.DocumentNode.SelectNodes("//h3/a");
```

请注意，该**SelectNodes**函数是由

XmlDocument的**DocumentNode**属性调用的。

变量**linkNodes**是一个集合。我们可以写一个**foreach**循环，并从每个链接一个一个地获取**href**值。我们只需要解决一个小问题——那就是页面上的链接是相对链接。因此，在我们抓取这些提取的链接之前，需要将它们转换为绝对URL。

为了转换相对链接，我们可以使用**Uri**该类。我们使用此构造函数来获取**Uri**具有绝对URL的对象。

```
1 | dotnet --version
```

一旦我们有了Uri对象，我们就可以简单地检查该**AbsoluteUri**属性以获取完整的URL。

我们将所有这些写在一个函数中，以保持代码的组织性。

```
1 | static List<string> GetBookLinks(string url)
2 | {
3 |     var bookLinks = new List<string>();
4 |     XmlDocument doc = GetDocument(url);
5 |     XmlNodeCollection linkNodes = doc.DocumentNode.SelectNodes("//h3/a");
6 |     var baseUri = new Uri(url);
7 |     foreach (var link in linkNodes)
8 |     {
9 |         string href = link.Attributes["href"].Value;
10 |         bookLinks.Add(new Uri(baseUri, href).AbsoluteUri);
    }
```

在这个函数中，我们从一个空**List<string>对象**开始。在**foreach**循环中，我们将所有链接添加到此对象并返回它。

现在，就可以修改**Main()**函数了，以便我们可以测试到目前为止编写的C#代码。修改函数如下：

```
1 static void Main(string[] args)
2
3 {
4     var bookLinks = GetBookLinks("http://books.toscrape.com/catalogue/category/bc
5     Console.WriteLine("Found {0} links", bookLinks.Count);
6 }
```

要运行此代码，请打开终端并导航到包含此文件的目录，然后键入以下内容：

```
1 dotnet run
```

输出应如下所示：

```
1 Found 20 links
```

然后我们转到下一部分，我们将处理所有链接以获取图书数据。

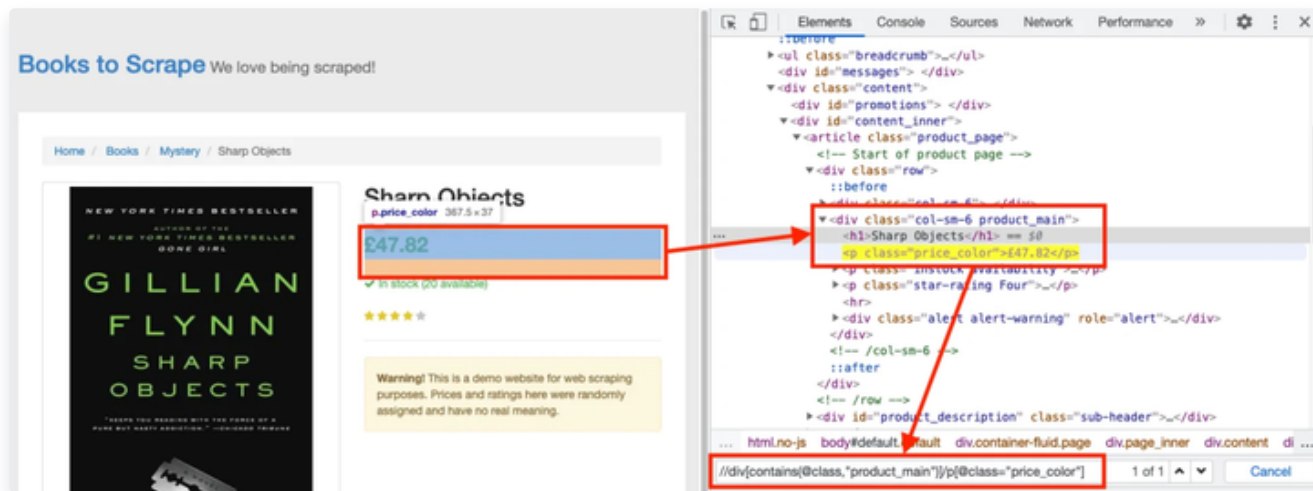
07.解析HTML：获取书籍详细信息

此时，我们有一个包含书籍URL的字符串列表。我们可以简单地编写一个循环，首先使用我们已经编写的函数**GetDocument**来获取文档。之后，我们将使用该**SelectSingleNode函数**来提取书名和价格。

为了让数据清晰有条理，我们从一个类开始。这个类将代表一本书，有两个属性-**Title**和**Price**。示例如下：

```
1 public class Book
2 {
```


然后，为**Title – //h1**在浏览器中打开一个书页。为价格创建 XPath 有点棘手，因为底部的附加书籍应用了相同的类。



价格的XPath

价格的XPath将是这样的：

```
1 | //div[contains(@class,"product_main")]/p[@class="price_color"]
```

请注意，XPath**包含双引号**。我们将不得不通过在它们前面加上反斜杠来转义这些字符。

现在我们可以使用**SelectSingleNode**函数来获取节点，然后使用**InnerText**属性获取元素中包含的文本。

我们可以将所有内容放在一个函数中，如下所示：

```
1 | static List<Book> GetBookDetails(List<string> urls)
2 | {
3 |     var books = new List<Book>();
4 |     foreach (var url in urls)
5 |     {
6 |         HtmlDocument document = GetDocument(url);
```



```
10     book.Title = document.DocumentNode.SelectSingleNode (priceXPath).InnerText;
11     book.Price = document.DocumentNode.SelectSingleNode(priceXPath).InnerText;
12     books.Add(book);
13 }
14 return books;
15 }
```

此函数将返回一个**Book对象**列表。是时候更新**Main()**函数了：

```
1 static void Main(string[] args)
2 {
3     var bookLinks = GetBookLinks("http://books.toscrape.com/catalogue/category/bc
4     Console.WriteLine("Found {0} links", bookLinks.Count);
5     var books = GetBookDetails(bookLinks);
6 }
```

这个网络抓取项目的最后一部分是**将数据导出为CSV**。

08.导出数据

如果您尚未安装**CsvHelper**，则可以通过

```
1 dotnet add package CsvHelper
```

在终端内运行命令来完成此操作。

导出功能非常简单。首先，我们需要创建一个**StreamWriter**并发送CSV文件名作为参数。接下来，我们将使用此对象创建一个**CsvWriter**。最后，我们可以使用该**WriteRecords**函数在一行代码中编写所有书籍。

为了确保所有资源都正确关闭，我们可以使用**using**块。我们还可以将所有内容包装在一个函数中，如下所示：

```
3 using (var writer = new StreamWriter("./books.csv"))
4 using (var csv = new CsvWriter(writer, CultureInfo.InvariantCulture))
5 {
6     csv.WriteRecords(books);
7 }
8 }
```

最后，我们可以从**Main()函数**中调用这个函数：

```
1 static void Main(string[] args)
2 {
3     var bookLinks = GetBookLinks("http://books.toscrape.com/catalogue/category/bc
4     var books = GetBookDetails(bookLinks);
5     exportToCSV(books);
6 }
```

要运行此代码，请打开终端并运行以下命令：

```
1 dotnet run
```

在几秒钟内，您将创建一个**books.csv**文件。

09.结论

如果您想用C#编写一个网络爬虫，您可以使用多个包。在本文中，我们展示了如何使用Html Agility Pack，这是一个功能强大且易于使用的包。也是一个可以进一步增强的简单示例；例如，您可以尝试将上述逻辑添加到此代码中以处理多个页面。

如果您想了解更多有关使用其他编程语言进行网络抓取的工作原理，可以查看使用Python进行网络抓取的指南。我们还有一个[关于如何使用JavaScript编写网络爬虫的分步教程](#)

常见问题

要。不过您将能够在Python和C#中找到示例的网页抓取工具。

Q：网络抓取合法吗？

A：如果在不违反任何法律的情况下使用代理，则它们可能是合法的。然而，在与代理进行任何活动之前，您应该就您的特定案件获得专业的法律建议。可以参见我们的文章[“网络抓取合法吗？”](#)

原创声明，本文系作者授权腾讯云开发者社区发表，未经许可，不得转载。

如有侵权，请联系 cloudcommunity@tencent.com 删除。

[Python](#)[C 语言](#)[C++](#)[C#](#)[举报](#)

点赞 3

分享

[登录](#) 后参与评论

0 条评论


c#使用WebClient登录网站抓取登录后的网页

C#登录网站实际上就是模拟浏览器提交表单，然后记录浏览器响应返回的会话Cookie值，再次发送请求时带着这个会话cookie值去请求就可以实现模拟登录的效果了。

 全栈程序员站长

使用Pyppeteer抓取渲染网页

GitHub地址是：<https://miyakogi.github.io/pyppeteer>

 SeanCheney

使用Python轻松抓取网页

抓取网页入门其实挺简单的。在之前的文章中我们介绍了怎么用C#和JAVA两种方法来抓取网页，这一期给大家介绍一种更容易，也是使用最广泛的一种抓取...

 Oxylabs中文站

使用Java进行网页抓取

用于网页抓取的流行语言有Python、JavaScript和Node.js、PHP、Java、C#等。因为有很多选择，想要确定哪种语言最合适并不容易。每种语言都...

 Oxylabs中文站

小白也能轻松为网页加各种部件

小轻相信，许多人对网页是很感兴趣并且是很想学习的。当初创立小轻网及小轻论坛网页就是为了帮助大家学习一些技术经验，同时也学会如何去找资源。今...


小白也能轻松为网页加各种部件

小轻相信，许多人对网页是很感兴趣并且是很想学习的。当初创立小轻网及小轻论坛网页就是为了帮助大家学习一些技术经验，同时也学会如何去找资源。今天，我们给原有网页加一...

 半夜喝可乐

Python使用Tor作为代理进行网页抓取

在网络抓取的过程中，我们经常会遇见很多网站采取了防爬取技术，或者说因为自己采集网站信息的强度和采集速度太大，给对方服务器带去了太多的压力，所以你一直用同一个代理...

 终身幼稚园

vConsole 让你在手机上也能轻松调试网页

vConsole相信各位并不陌生，它是一个轻量、可拓展、针对手机网页的前端开发者调试面板。有时候为了想在手机上对网页进行 Debug，可手机上没有F12，...

 qiangzai

卧槽，R 语言也能爬取网页的数据！

爬虫技术是一种从网页中获取数据的方式，是按照一定规则，自动地抓取网页数据的程序或者脚本。除了Python可以写爬虫程序外，R语言一样可以实现爬虫...

 Python研究者

Python爬虫进阶（一）使用Selenium进行网页抓取

萌新要学习Selenium了，安装是个坑。还要下载相关配件，可以参考python 安

教你如何使用微信网页版“抓取”微信撤回消息

有个高中微信搞笑群，常发一些搞笑的图片，但是发后就撤回了，一不小心就看不到了，所以就想着怎么查看撤回的图片或者文字。思路是这样的，当微信收...

 FB客服

java使用正则表达式抓取网页内容存为txt

前几天女友在网上看了一本电子书，想要下载下来，不过那个网站只能支持在线阅读，不提供下载，还好可以复制粘贴。

 the5fire


Puppeteer Sharp: 使用C#和Headless Chrome爬网页

Puppeteer 是谷歌构建的流行的Headless Chrome NodeJS API爬虫库。Puppeteer Sharp是用C#写的，由达里奥·孔德拉蒂...


 皇上得了花柳病

3D 穿梭效果？使用 UWP 也能搞定

这个效果太神奇了，他还问我能不能用 WPF 搞出来，因为我完全没用过 WPF 的 3D，我第一反应是“这太难为我了”。


 dino.c

小白也能懂！教你快速入门使用Burpsuite抓包

 网络安全自修室

【玩转Lighthouse】小白也能FRP内网穿透配置和使用

有时候在想互联网互联网，为什么在异地没有办法通过网络连接家里的NAS，远程控制家里的电脑呢？网上一顿恶补学习，原来是没有分配到 基于 IPV4 的公网 IP...

 用户6795856


『爬虫四步走』手把手教你使用Python抓取并存储网页数据！

爬虫是Python的一个重要的应用，使用Python爬虫我们可以轻松的从互联网中抓取我们想要的的数据，本文将基于爬取B站视频热搜榜单数据并存储为例，详细...

 刘早起

C#开发BIMFACE系列50 Web网页中使用jQuery加载模型与图纸

在前一篇博客《C#开发BIMFACE系列49 Web网页集成BIMFACE应用的技术方案》中介绍了目前市场主流的Web开发技术与应用框架，其中前端脚本的应用在国...

 张传宁IT讲堂

如何让你的网站也能跟我的一样使用HTTPS访问？

使用HTTPS访问我们的网站，不仅可以增加我们网站的安全性，更重要的是还能提升我们网站的逼格！我在为网站搭建SSL服务和CDN上有一些经验，在这里...

 冯文议



社区

专栏文章

阅读清单

互动问答

技术沙龙

技术视频

团队主页

腾讯云TI平台

活动

自媒体分享计划

邀请作者入驻

自荐上首页

资源

技术周刊

社区标签

开发者手册

开发者实验室

关于

视频介绍

社区规范

免责声明

联系我们

友情链接

腾讯云开发者



扫码关注腾讯云开发者
领取腾讯云代金券

热门产品

域名注册

云服务器

区块链服务

消息队列

网络加速

云数据库

域名解析

云存储

视频直播

热门推荐

人脸识别

腾讯会议

企业云

CDN 加速

视频通话

图像分析

MySQL 数据库

SSL 证书

语音识别

更多推荐

数据安全

负载均衡

短信

文字识别

云点播

商标注册

小程序开发

网站监控

数据迁移