Python爬虫利器四之PhantomJS的用法

Python 崔庆才 2年前 (2016-03-21) 85325浏览 29评论

前言

大家有没有发现之前我们写的爬虫都有一个共性，就是只能爬取单纯的html代码，如果页面是JS渲染的该怎么办呢？如果我们单纯去分析一个个后台的请求，手动去摸索JS渲染的到的一些结果，那简直没天理了。所以，我们需要有一些好用的工具来帮助我们像浏览器一样渲染JS处理的页面。

其中有一个比较常用的工具，那就是

PhantomJS

> Full web stack No browser required
>
> PhantomJS is a headless WebKit scriptable with a JavaScript API. It has **fast** and**native** support for various web standards: DOM handling, CSS selector, JSON, Canvas, and SVG.

PhantomJS是一个无界面的,可脚本编程的WebKit浏览器引擎。它原生支持多种web 标准：DOM 操作，CSS选择器，JSON，Canvas 以及SVG。

好，接下来我们就一起来了解一下这个神奇好用的库的用法吧。

安装

PhantomJS安装方法有两种，一种是下载源码之后自己来编译，另一种是直接下载编译好的二进制文件。然而自己编译需要的时间太长，而且需要挺多的磁盘空间。官方推荐直接下载二进制文件然后安装。

大家可以依照自己的开发平台选择不同的包进行下载

下载地址

当然如果你不嫌麻烦，可以选择

下载源码

然后自己编译。

目前（2016/3/21）最新发行版本是 v2.1，

安装完成之后命令行输入

```
phantomjs -v
```

如果正常显示版本号，那么证明安装成功了。如果提示错误，那么请重新安装。

本文介绍大部分内容来自于官方文档，博主对其进行了整理，学习更多请参考

官方文档

快速开始

第一个程序

第一个程序当然是Hello World，新建一个 js 文件。命名为 helloworld.js

```
console.log('Hello, world!');
phantom.exit();
```

命令行输入

```
phantomjs helloworld.js
```

程序输出了 Hello，world！程序第二句话终止了 phantom 的执行。

注意：`phantom.exit();`这句话非常重要，否则程序将永远不会终止。

页面加载

可以利用 phantom 来实现页面的加载，下面的例子实现了页面的加载并将页面保存为一张图片。

```
var page = require('webpage').create();
page.open('http://cuiqingcai.com', function (status) {
    console.log("Status: " + status);
    if (status === "success") {
        page.render('example.png');
    }
    phantom.exit();
});
```

首先创建了一个webpage对象，然后加载本站点主页，判断响应状态，如果成功，那么保存截图为 example.png

以上代码命名为 pageload.js，命令行

```
phantomjs pageload.js
```

发现执行成功，然后目录下多了一张图片，example.png

静觅

热门排行

1. 强势回归博客，顺便祝大家小年　评论 (9)　♡10喜欢
2. 弃用多说，改用畅言　评论 (9)　♡6喜欢
3. 干货！IT小伙伴们实用的网站及　评论 (7)　♡45喜欢

Python **Python爬虫利器一之Requests库的用法**

前言 之前我们用了 urllib 库，这个作为入门的工具还是不错的，对了解一些爬虫的基本理念，掌握爬虫爬取的流程有所帮助。入门之

JavaScript **jQuery易忽略的知识点总结**

前言 之前在用jQuery，不过有时候用着用着一些用法发现并没有用到过，比较陌生，现在重新梳理一下，把易忽略的知识点总结一下，长

Other **PhpStorm使用File Watchers自动编译less**

综述 PhpStorm可以使用File Watchers自动编译Less，有了这个IDE，妈妈再也不用担心我

因为这个 render 方法，phantom 经常会用到网页截图的功能。

测试页面加载速度

下面这个例子计算了一个页面的加载速度，同时还用到了命令行传参的特性。新建文件保存为 loadspeed.js

```
var page = require('webpage').create(),
  system = require('system'),
  t, address;

if (system.args.length === 1) {
  console.log('Usage: loadspeed.js <some URL>');
  phantom.exit();
}

t = Date.now();
address = system.args[1];
page.open(address, function(status) {
  if (status !== 'success') {
    console.log('FAIL to load the address');
  } else {
    t = Date.now() - t;
    console.log('Loading ' + system.args[1]);
    console.log('Loading time ' + t + ' msec');
  }
  phantom.exit();
});
```

程序判断了参数的多少，如果参数不够，那么终止运行。然后记录了打开页面的时间，请求页面之后，再纪录当前时间，二者之差就是页面加载速度。

```
phantomjs loadspeed.js http://cuiqingcai.com
```

运行结果

```
Loading http://cuiqingcai.com
Loading time 11678 msec
```

这个时间包括JS渲染的时间，当然和网速也有关。

代码评估

> To evaluate JavaScript code in the context of the web page, use `evaluate()` function. The execution is "sandboxed", there is no way for the code to access any JavaScript objects and variables outside its own page context. An object can be returned from `evaluate()`, however it is limited to simple objects and can't contain functions or closures.

利用 evaluate 方法我们可以获取网页的源代码。这个执行是“沙盒式”的，它不会去执行网页外的 JavaScript 代码。evalute 方法可以返回一个对象，然而返回值仅限于对象，不能包含函数（或闭包）

```
var url = 'http://www.baidu.com';
var page = require('webpage').create();
page.open(url, function(status) {
  var title = page.evaluate(function() {
    return document.title;
  });
  console.log('Page title is ' + title);
  phantom.exit();
});
```

以上代码获取了百度的网站标题。

```
Page title is 百度一下，你就知道
```

任何来自于网页并且包括来自 evaluate() 内部代码的控制台信息，默认不会显示。

需要重写这个行为，使用 onConsoleMessage 回调函数，示例可以改写成

```
var url = 'http://www.baidu.com';
var page = require('webpage').create();
page.onConsoleMessage = function (msg) {
    console.log(msg);
};
page.open(url, function (status) {
    page.evaluate(function () {
        console.log(document.title);
    });
    phantom.exit();
});
```

这样的话，如果你用浏览器打开百度首页，打开调试工具的console，可以看到控制台输出信息。

重写了 onConsoleMessage 方法之后，可以发现控制台输出的结果和我们需要输出的标题都打印出来了。

```
一张网页，要经历怎样的过程，才能抵达用户面前？
一位新人，要经历怎样的成长，才能站在技术之巅？
探寻这里的秘密；
体验这里的挑战；
成为这里的主人；
加入百度，加入网页搜索，你，可以影响世界。

请将简历发送至 %c ps_recruiter@baidu.com（ 邮件标题请以“姓名-应聘XX职位-来自console”命名） color:red
职位介绍：http://dwz.cn/hr2013
百度一下，你就知道
```

啊，我没有在为百度打广告！

屏幕捕获

> Since PhantomJS is using WebKit, a real layout and rendering engine, it can capture a web page as a screenshot. Because PhantomJS can render anything on the web page, it can be used to convert contents not only in HTML and CSS, but also SVG and Canvas.

因为 PhantomJS 使用了 WebKit内核，是一个真正的布局和渲染引擎，它可以像屏幕截图一样捕获一个web界面。因为它可以渲染网页中的人和元素，所以它不仅用到HTML，CSS的内容转化，还用在SVG，Canvas。可见其功能是相当强大的。

下面的例子就捕获了github网页的截图。上文有类似内容，不再演示。

```
var page = require('webpage').create();
page.open('http://github.com/', function() {
  page.render('github.png');
  phantom.exit();
});
```

除了 png 格式的转换，PhantomJS还支持 jpg，gif，pdf等格式。

测试样例

其中最重要的方法便是 viewportSize 和 clipRect 属性。

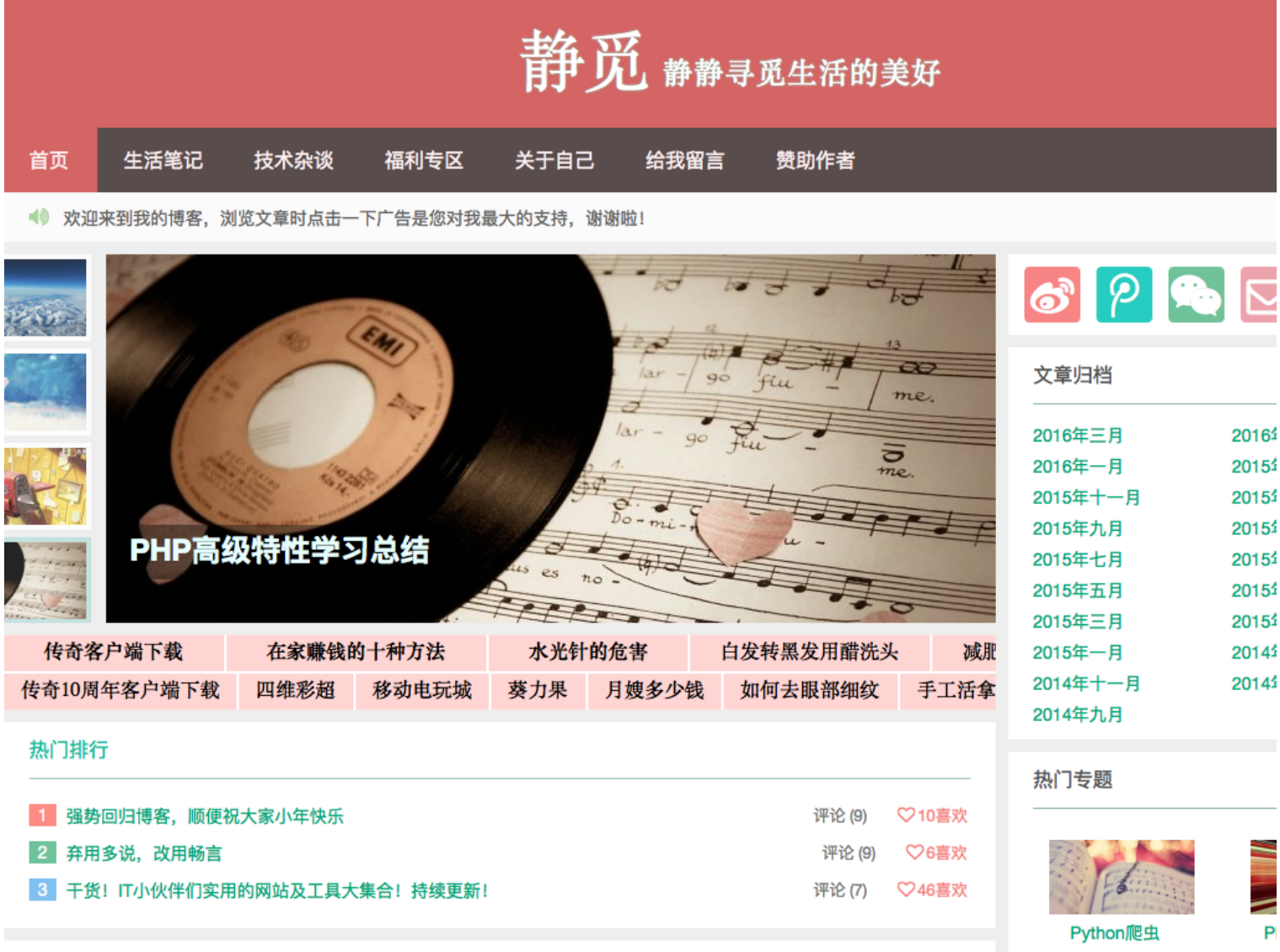
viewportSize 是视区的大小，你可以理解为你打开了一个浏览器，然后把浏览器窗口拖到了多大。

clipRect 是裁切矩形的大小，需要四个参数，前两个是基准点，后两个参数是宽高。

通过下面的小例子感受一下。

```
var page = require('webpage').create();
//viewportSize being the actual size of the headless browser
page.viewportSize = { width: 1024, height: 768 };
//the clipRect is the portion of the page you are taking a screenshot of
page.clipRect = { top: 0, left: 0, width: 1024, height: 768 };
//the rest of the code is the same as the previous example
page.open('http://cuiqingcai.com/', function() {
  page.render('germy.png');
  phantom.exit();
});
```

运行结果



就相当于把浏览器窗口拖到了 1024×768 大小，然后从左上角裁切出了 1024×768 的页面。

网络监听

> Because PhantomJS permits the inspection of network traffic, it is suitable to build various analysis on the network behavior and performance.

因为 PhantomJS 有网络通信的检查功能，它也很适合用来做网络行为的分析。

> When a page requests a resource from a remote server, both the request and the response can be tracked via onResourceRequested and onResourceReceived callback.

当接受到请求时，可以通过改写onResourceRequested和onResourceReceived回调函数来实现接收到资源请求和资源接受完毕的监听。例如

```
var url = 'http://www.cuiqingcai.com';
var page = require('webpage').create();
page.onResourceRequested = function(request) {
  console.log('Request ' + JSON.stringify(request, undefined, 4));
};
page.onResourceReceived = function(response) {
  console.log('Receive ' + JSON.stringify(response, undefined, 4));
};
page.open(url);
```

运行结果会打印出所有资源的请求和接收状态，以JSON格式输出。

页面自动化处理

> Because PhantomJS can load and manipulate a web page, it is perfect to carry out various page automations.

因为 PhantomJS 可以加载和操作一个web页面，所以用来自动化处理也是非常适合的。

DOM操作

> Since the script is executed as if it is running on a web browser, standard DOM scripting and CSS selectors work just fine.

脚本都是像在浏览器中运行的，所以标准的 JavaScript 的 DOM 操作和 CSS 选择器也是生效的。

例如下面的例子就修改了 User-Agent，然后还返回了页面中某元素的内容。

```
var page = require('webpage').create();
console.log('The default user agent is ' + page.settings.userAgent);
page.settings.userAgent = 'SpecialAgent';
page.open('http://www.httpuseragent.org', function(status) {
  if (status !== 'success') {
    console.log('Unable to access network');
  } else {
    var ua = page.evaluate(function() {
      return document.getElementById('myagent').textContent;
    });
    console.log(ua);
  }
  phantom.exit();
});
```

运行结果

```
The default user agent is Mozilla/5.0 (Macintosh; Intel Mac OS X) AppleWebKit/538.1 (KHTML, like Gec
ko) PhantomJS/2.1.0 Safari/538.1
Your Http User Agent string is: SpecialAgent
```

首先打印出了默认的 User-Agent，然后通过修改它，请求验证 User-Agent 的一个站点，通过选择器得到了修改后的 User-Agent。

使用附加库

在1.6版本之后允许添加外部的JS库，比如下面的例子添加了jQuery，然后执行了jQuery代码。

```
var page = require('webpage').create();
page.open('http://www.sample.com', function() {
  page.includeJs("http://ajax.googleapis.com/ajax/libs/jquery/1.6.1/jquery.min.js", function() {
    page.evaluate(function() {
      $("button").click();
    });
    phantom.exit()
  });
});
```

引用了 jQuery 之后，我们便可以在下面写一些 jQuery 代码了。

Webpage对象

在前面我们介绍了 webpage 对象的几个方法和属性，其实它本身还有其它很多的属性。具体的内容可以参考

Webpage　　Webpage用例

里面介绍了 webpage的所有属性，方法，回调。

命令行

Command-line Options

PhantomJS提供的命令行选项有：

> –help or -h lists all possible command-line options. Halts immediately, will not run a script passed as argument. [帮助列表]

–version or -v prints out the version of PhantomJS. Halts immediately, will not run a script passed as argument. [查看版本]

–cookies-file=/path/to/cookies.txt specifies the file name to store the persistent Cookies. [指定存放cookies的路径]

–disk-cache=[true|false] enables disk cache (at desktop services cache storage location, default is false). Also accepted: [yes|no]. [硬盘缓存开关，默认为关]

–ignore-ssl-errors=[true|false] ignores SSL errors, such as expired or self-signed certificate errors (default is false). Also accepted: [yes|no]. [忽略ssl错误，默认不忽略]

–load-images=[true|false] load all inlined images (default is true). Also accepted: [yes|no]. [加载图片，默认为加载]

–local-storage-path=/some/path path to save LocalStorage content and WebSQL content. [本地存储路径，如本地文件和SQL文件等]

–local-storage-quota=number maximum size to allow for data. [本地文件最大大小]

–local-to-remote-url-access=[true|false] allows local content to access remote URL (default is false). Also accepted: [yes|no]. [是否允许远程加载文件，默认不允许]

–max-disk-cache-size=size limits the size of disk cache (in KB). [最大缓存空间]

–output-encoding=encoding sets the encoding used for terminal output (default is utf8). [默认输出编码，默认utf8]

–remote-debugger-port starts the script in a debug harness and listens on the specified port [远程调试端口]

–remote-debugger-autorun runs the script in the debugger immediately: ‘yes’ or ‘no’ (default) [在调试环境下是否立即执行脚本，默认否]

–proxy=address:port specifies the proxy server to use (e.g. –proxy=192.168.1.42:8080). [代理]

–proxy-type=[http|socks5|none] specifies the type of the proxy server (default is http). [代理类型，默认http]

–proxy-auth specifies the authentication information for the proxy, e.g. –proxy-auth=username:password). [代理认证]

–script-encoding=encoding sets the encoding used for the starting script (default is utf8). [脚本编码，默认utf8]

–ssl-protocol=[sslv3|sslv2|tlsv1|any’] sets the SSL protocol for secure connections (default is SSLv3). [SSL协议，默认SSLv3]

–ssl-certificates-path=<val> Sets the location for custom CA certificates (if none set, uses system default). [SSL证书路径，默认系统默认路径]

–web-security=[true|false] enables web security and forbids cross-domain XHR (default is true). Also accepted: [yes|no]. [是否开启安全保护和禁止异站Ajax，默认开启保护]

–webdriver starts in ‘Remote WebDriver mode’ (embedded GhostDriver): ‘[[:]]’ (default ‘127.0.0.1:8910’) [以远程WebDriver模式启动]

–webdriver-selenium-grid-hub URL to the Selenium Grid HUB: ‘URLTOHUB’ (default ‘none’) (NOTE: works only together with ‘–webdriver’) [Selenium接口]

–config=/path/to/config.json can utilize a JavaScript Object Notation (JSON) configuration file instead of passing in multiple command-line optionss [所有的命令行配置从config.json中读取]

注：JSON文件配置格式

```
{
  /* Same as: --ignore-ssl-errors=true */
  "ignoreSslErrors": true,

  /* Same as: --max-disk-cache-size=1000 */
  "maxDiskCacheSize": 1000,

  /* Same as: --output-encoding=utf8 */
  "outputEncoding": "utf8"

  /* etc. */
}

There are some keys that do not translate directly:

 * --disk-cache => diskCacheEnabled
 * --load-images => autoLoadImages
 * --local-storage-path => offlineStoragePath
 * --local-storage-quota => offlineStorageDefaultQuota
 * --local-to-remote-url-access => localToRemoteUrlAccessEnabled
 * --web-security => webSecurityEnabled
```

以上是命令行的基本配置

实例

在此提供官方文档实例，多对照实例练习，使用起来会更得心应手。

官方实例

结语

以上是博主对 PhantomJS 官方文档的基本总结和翻译，如有差错，希望大家可以指正。另外可能有的小伙伴觉得这个工具和 Python 有什么关系？不要急，后面会有 Python 和 PhantomJS 的综合使用的。

喜欢 (160) or 分享 (0)

我的个人微信公众号，联系我请直接在公众号留言即可~

扫码或搜索：进击的Coder

进击的Coder

微信公众号 扫一扫关注

想结交更多的朋友吗?

来进击的Coder瞧瞧吧

进击的Coder

QQ群号 99350970 立即加入

进击的Coder灌水太多?

这里是纯粹的技术领地

激进的Coder

QQ群号 627725766 立即加入

您的支持是博主写作最大的动力，如果您喜欢我的文章，感觉我的文章对您有帮助，请狠狠点击下面的

我要小额赞助

JS Python 爬虫

« Python爬虫利器一之Requests库的用法

Python爬虫利器五之Selenium的用法 »

TensorFlow layers模块用法

TensorFlow验证码识别

利用python库twilio来免费发送短信

[Python3网络爬虫开发实战] 后续章节

- TensorFlow layers模块用法
- 利用python库twilio来免费发送短信
- 爬虫代理哪家强？十大付费代理详细对比评测出炉！
- [Python3网络爬虫开发实战] 7.3-Splash负载均衡配
- TensorFlow验证码识别
- [Python3网络爬虫开发实战] 后续章节
- [Python3网络爬虫开发实战] 7.4-使用Selenium爬取淘
- [Python3网络爬虫开发实战] 7.2-Splash的使用