[手把手教你做爬虫---基于NodeJs](http://blog.csdn.net/yezhenxu1992/article/details/50820629)

**原文地址：http://blog.csdn.net/yezhenxu1992/article/details/50820629**

**我写的：**

**就是利用cheerio和request两个模块来做爬虫，cheerio用于解析html，解析方法跟jquery很像**

**目标：爬取北京大学软件与微电子学院的所有新闻，并将内容及图片存储到本地**。

**设计思路**：经过对北京大学软件与微电子学院的新闻网址<http://www.ss.pku.edu.cn/index.php/newscenter/news/>内容及网页格式的分析，我发现了这样一个规律：在每篇文章中，都会有下一篇文章url的链接。所以，我的做法是：给定一个初始（最新的）网页的url，如[http://www.ss.pku.edu.cn/index.php/newscenter/news/2391](http://www.ss.pku.edu.cn/index.php/newscenter/news/2391" \t "_blank)，然后进行一次请求，爬取到下一篇文章的url，用新的url再继续请求，递归调用，直到遍历完所有的新闻网页。值得一提的是，我的亮点之一，是可以通过控制一个变量i，来控制爬取文章的数量。

神奇的旅程即将开启 ….

**步骤1：正所谓“工欲善其事，必先利其器”**

安装包我就无私地贡献出来：“http://pan.baidu.com/s/1i4uQcLZ

1）下载nodejs   
2）下载javaScript编辑器webStorm

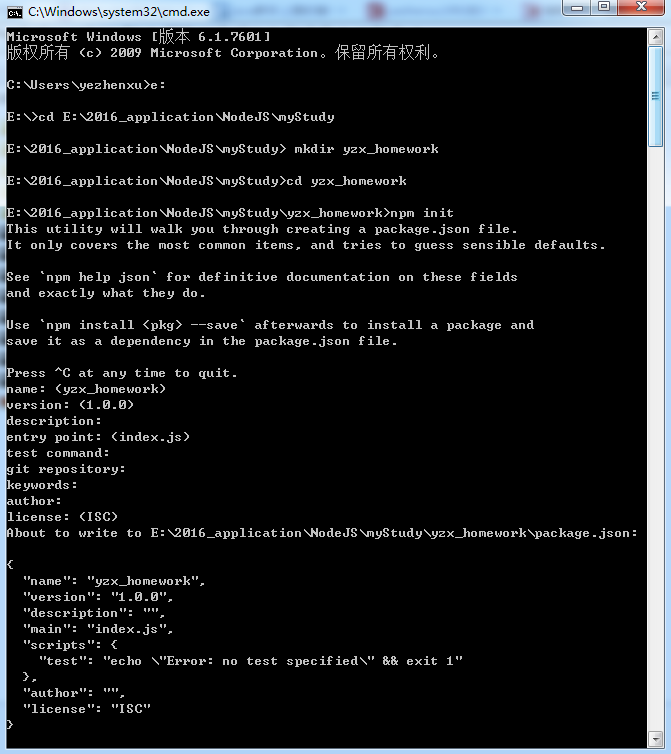
如果你是个聪明的家伙，你一定能完成这两个安装。如果安装过程中，不幸遇到各种bug，那么请你自行问度娘，安装好了再继续往下看。

**步骤2：建立工程**

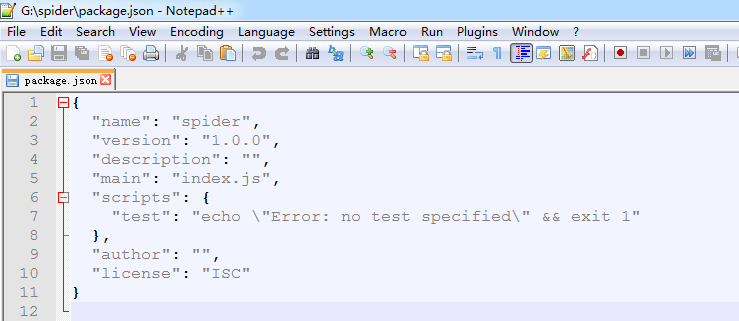
友情提醒：请原谅我对webStorm还不太熟悉，我自己也才刚安装一天，还处于学习阶段，所以在DOS操作吧！cmd进入DOS，感觉还是挺酷的！

1）在DOS下cd 进入到你想要创建项目的路径。   
2）mkdir yzx\_homework （创建一个yzx\_homework文件夹）   
3）cd yzx\_homework   
4) npm init （初始化工程）

此时需要填写一些项目信息，你可以根据情况填写，当然也可以一路回车。



创建完项目后，会生成一个package.json的文件。该文件包含了项目的基本信息。



5）安装第三方包（后面程序会直接调用包的模块）

说明：由于http模块、fs模块都是内置的包，因此不需要额外添加。

这里安装cheerio包，和request包。

在dos中，cd进入spider文件夹，然后   
npm install cheerio –save

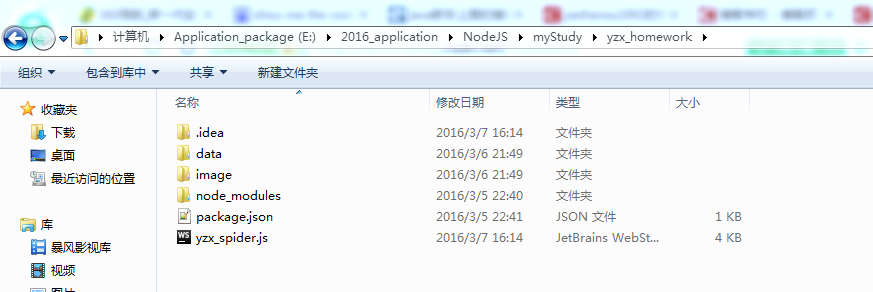
安装完cheerio包后，继续安装request包， npm install request –save

说明：npm（nodejs package manager），nodejs包管理器；   
–save的目的是将项目对该包的依赖写入到package.json文件中。

6）在spider文件夹下

1. 创建子文件夹data（用于存放所抓取的新闻文本内容）
2. 创建子文件夹image（用于存放所抓取的图片资源）
3. 创建一个yzx\_spider.js文件

整个项目的目录结构如下图所示：



**步骤三：”talk is cheep，show me the code .”**

打开yzx\_spider.js，并一行一行的敲代码。

你要是个经常“ctrl+c” + “ctrl+v”的家伙，那我就呵呵了！

var http = require('http');

var fs = require('fs');

var cheerio = require('cheerio');

var request = require('request');

var i = 0;

var url = "http://www.ss.pku.edu.cn/index.php/newscenter/news/2391";

//初始url

function fetchPage(x) { //封装了一层函数

startRequest(x);

}

function startRequest(x) {

//采用http模块向服务器发起一次get请求

http.get(x, function (res) {

var html = ''; //用来存储请求网页的整个html内容

var titles = [];

res.setEncoding('utf-8'); //防止中文乱码

//监听data事件，每次取一块数据

res.on('data', function (chunk) {

html += chunk;

});

//监听end事件，如果整个网页内容的html都获取完毕，就执行回调函数

res.on('end', function () {

var $ = cheerio.load(html); //采用cheerio模块解析html

var time = $('.article-info a:first-child').next().text().trim();

var news\_item = {

//获取文章的标题

title: $('div.article-title a').text().trim(),

//获取文章发布的时间

Time: time,

//获取当前文章的url

link: "http://www.ss.pku.edu.cn" + $("div.article-title a").attr('href'),

//获取供稿单位

author: $('[title=供稿]').text().trim(),

//i是用来判断获取了多少篇文章

i: i = i + 1,

};

console.log(news\_item); //打印新闻信息

var news\_title = $('div.article-title a').text().trim();

savedContent($,news\_title); //存储每篇文章的内容及文章标题

savedImg($,news\_title); //存储每篇文章的图片及图片标题

//下一篇文章的url

var nextLink="http://www.ss.pku.edu.cn" + $("li.next a").attr('href');

str1 = nextLink.split('-'); //去除掉url后面的中文

str = encodeURI(str1[0]);

//这是亮点之一，通过控制I,可以控制爬取多少篇文章.

if (i <= 500) {

fetchPage(str);

}

});

}).on('error', function (err) {

console.log(err);

});

}

//该函数的作用：在本地存储所爬取的新闻内容资源

function savedContent($, news\_title) {

$('.article-content p').each(function (index, item) {

var x = $(this).text();

var y = x.substring(0, 2).trim();

if (y == '') {

x = x + '\n';

//将新闻文本内容一段一段添加到/data文件夹下，并用新闻的标题来命名文件

fs.appendFile('./data/' + news\_title + '.txt', x, 'utf-8', function (err) {

if (err) {

console.log(err);

}

});

}

})

}

//该函数的作用：在本地存储所爬取到的图片资源

function savedImg($,news\_title) {

$('.article-content img').each(function (index, item) {

var img\_title = $(this).parent().next().text().trim(); //获取图片的标题

if(img\_title.length>35||img\_title==""){

img\_title="Null";}

var img\_filename = img\_title + '.jpg';

var img\_src = 'http://www.ss.pku.edu.cn' + $(this).attr('src'); //获取图片的url

//采用request模块，向服务器发起一次请求，获取图片资源

request.head(img\_src,function(err,res,body){

if(err){

console.log(err);

}

});

request(img\_src).pipe(fs.createWriteStream('./image/'+news\_title + '---' + img\_filename)); //通过流的方式，把图片写到本地/image目录下，并用新闻的标题和图片的标题作为图片的名称。

})

}

fetchPage(url); //主程序开始运行

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25
* 26
* 27
* 28
* 29
* 30
* 31
* 32
* 33
* 34
* 35
* 36
* 37
* 38
* 39
* 40
* 41
* 42
* 43
* 44
* 45
* 46
* 47
* 48
* 49
* 50
* 51
* 52
* 53
* 54
* 55
* 56
* 57
* 58
* 59
* 60
* 61
* 62
* 63
* 64
* 65
* 66
* 67
* 68
* 69
* 70
* 71
* 72
* 73
* 74
* 75
* 76
* 77
* 78
* 79
* 80
* 81
* 82
* 83
* 84
* 85
* 86
* 87
* 88
* 89
* 90
* 91
* 92
* 93
* 94
* 95
* 96
* 97
* 98
* 99
* 100
* 101
* 102
* 103
* 104
* 105
* 106
* 107

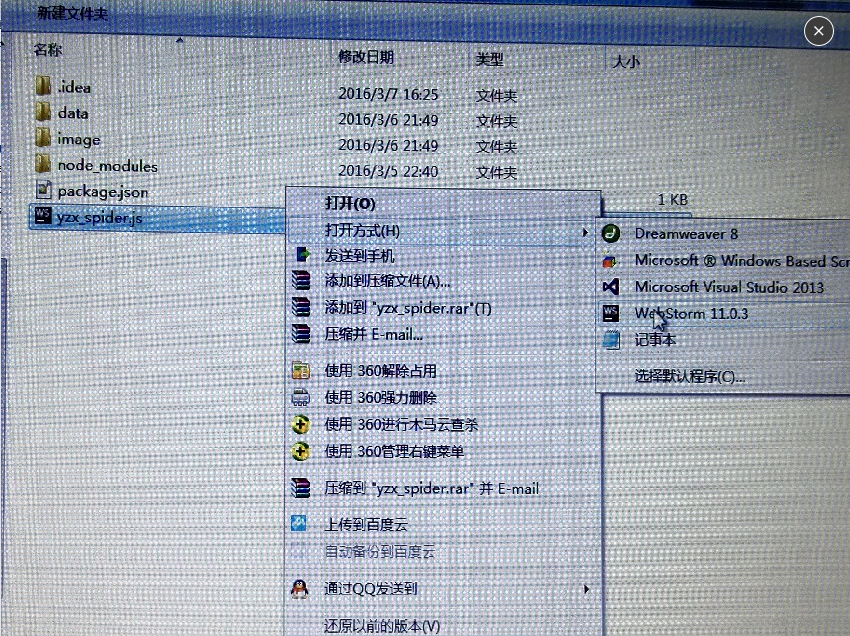
恭喜你，你总算熬出头了！

我依然相信，你是一行一行敲代码的家伙！

运行程序： 见证奇迹的时刻到了！

运行程序：两种方法：

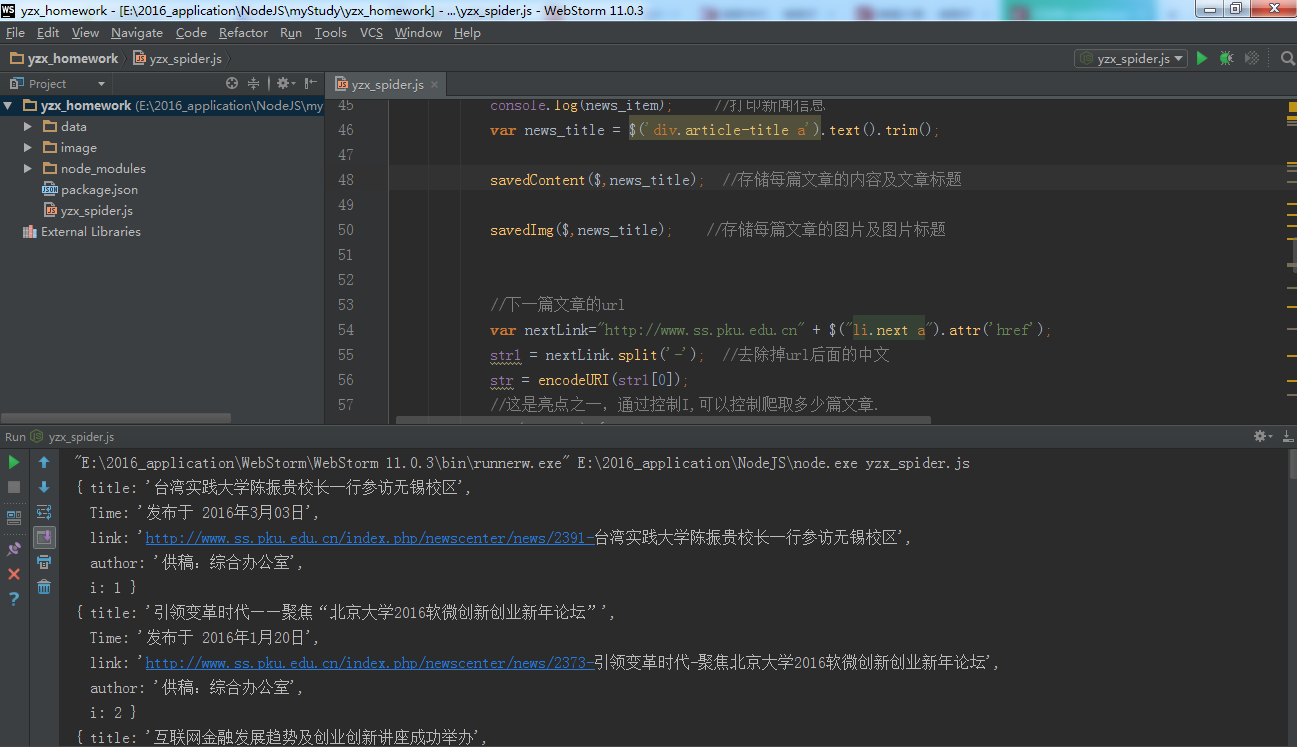
1） 右键yzx\_spider.js，打开方式选择webStorm



然后进入webStorm，点击右上角绿色的三角标志，程序便开始运行。

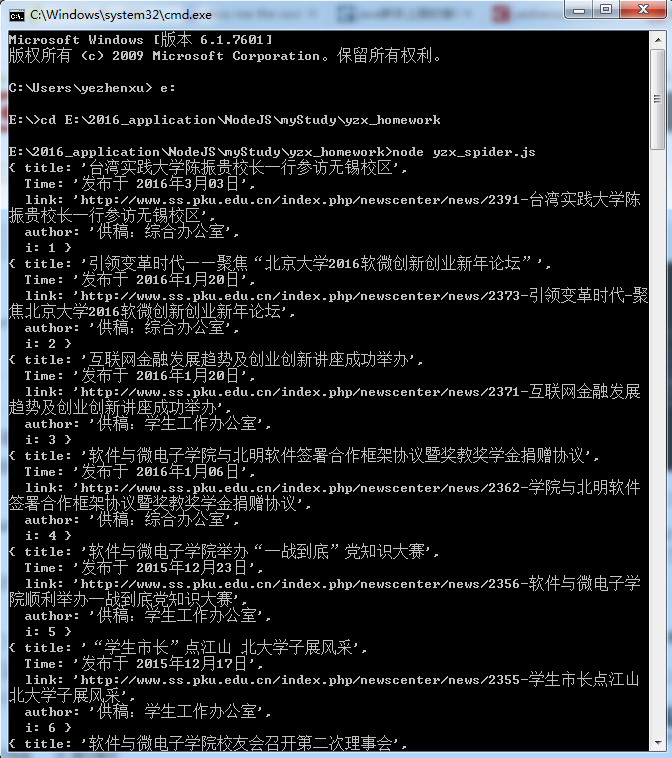
控制台有内容输出，证明你成功了！！！

控制台输出的内容，对应的代码是console.log(news\_item)，输出了新闻的信息。



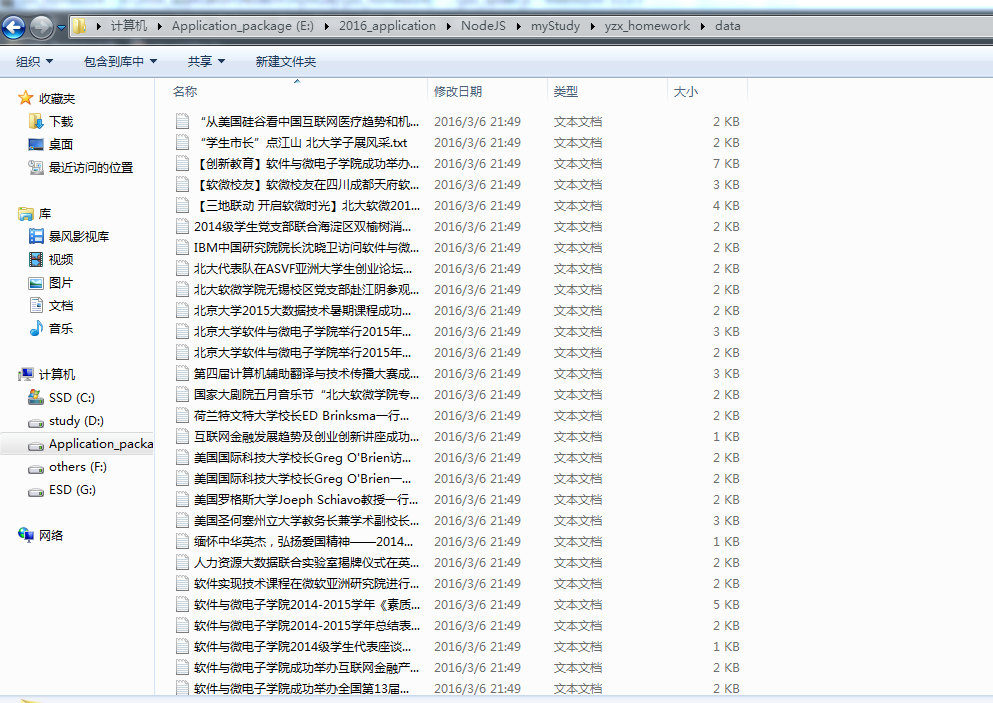
2）如果你对webStorm还一头雾水，亦或安装失败，你可以选择在dos下运行。

cd 到你创建工程文件夹yzx\_homework下，然后 node yzx\_spider.js 程序就跑起来了。

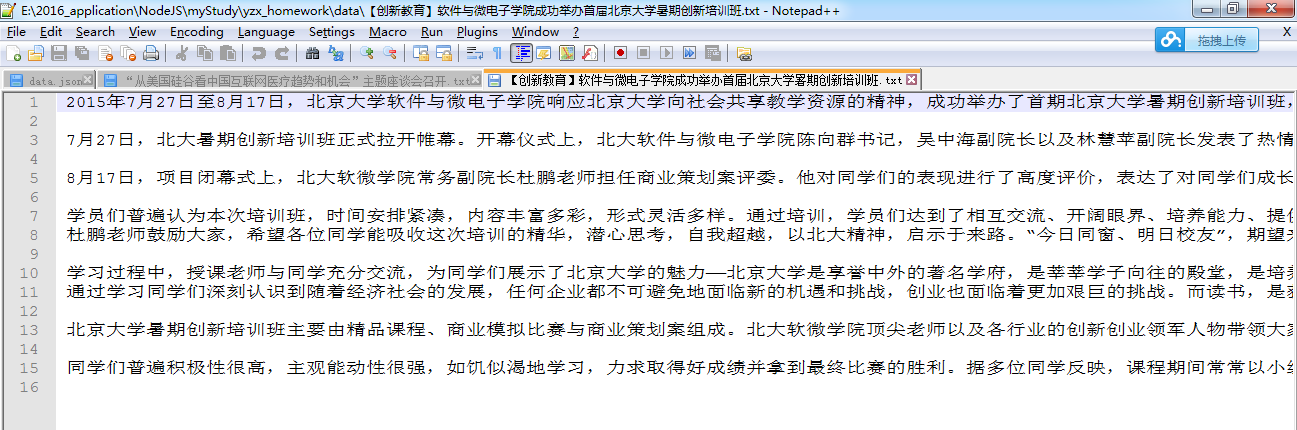


**本地存储的资源**

1）**文本资源**：就没刻意保存所有文章的标题了，最后以文章的标题命名.txt文件。



2）**文章的具体内容**：文章内容按段落处理，且段与段之间空一行。



3）**图片资源**：



4）**图片命名说明**：就没刻意保存文章图片的标题了。图片的命名格式为：文章标题+ “—”+图片标题。

如下图所示：图片的标题为“座谈会现场”和“参数队员合影”。



**感悟与体会：**经过这次NodeJs爬虫项目的学习，我熟悉了整个网页Html元素的布局方式及结构，能够很从容地针对所要爬取的内容，设计相对应的选择器。同时，我也体会到了NodeJs模块化的便捷及魅力。

**后记**：爬虫项目，其中关键就在于选择器的设计。cheerio包的选择器$，和jQuery选择器规则几乎是一样的。歇一会，如果还有精力，我就总结一下我这两天学习选择器的体会和小小经验。