# 1：简介

新浪博客（http://blog.sina.com.cn/s/articlelist\_1273160402\_0\_1.html）每篇文章的标题，发表时间、url、标签、详细内容

使用的工具

nodejs+mysql

# 2：获取所有文章的url

## 分析html页面

在全部博文的页面，分析页面的html

<http://blog.sina.com.cn/s/articlelist_1273160402_0_1.html>

发现每一个博文列表的html如下格式所示

<div class="articleCell SG\_j\_linedot1">

<p class="atc\_main SG\_dot">

<span class="atc\_ic\_f"></span>

<span class="atc\_title">

<a title="戴耘\_日内瓦的重逢与记忆" target="\_blank" href="http://blog.sina.com.cn/s/blog\_4be2e2d20102vtri.html">戴耘\_日内瓦的重逢与记忆</a>

</span>

<span class="atc\_ic\_b"></span>

</p>

<p class="atc\_info">

<span class="atc\_data" id="count\_4be2e2d20102vtri">

(<span title="评论数">0</span>/<span title="阅读数">24</span>)

</span>

<span class="atc\_tm SG\_txtc">2015-05-08 07:25</span>

<span class="atc\_set"></span>

</p>

</div>

a标签

* a标签中的title
* a标签中的href是每篇文章的url
* class 为”atc\_tm”的span标签中有我们需要的时间信息
* 上面黄色背景色的为id

获取当前页面中的url title time id

function readArticleList(url,callback) {

//读取分类页面

request(url, function (err, res) {

if (err) {

return console.error(err);

}

//根据网页内容创建dom操作对象

var $ = cheerio.load(res.body.toString());

//读取博文的列表

var articleList = [];

$('.articleList .articleCell').each(function () {

var $this = $(this);

var $title = $this.find('.atc\_title a');

var $time = $this.find('.atc\_tm');

var item = {

title: $title.text().trim(),

url: $title.attr('href'),

time: $time.text().trim()

};

//从url中取出文章的id

//http://blog.sina.com.cn/s/blog\_4be2e2d20102vtri.html

var s = item.url.match(/blog\_([a-zA-Z0-9]+)\.html/);

if (Array.isArray(s)) {

item.id = s[1];

articleList.push(item);

}

});

callback(null,articleList);

}

## 获取下一页中的文章列表

观察页面跳转部分的html

<div class="SG\_page">

<ul class="SG\_pages">

<li class="SG\_pgon" title="当前所在页">1</li>

<li><a href="http://blog.sina.com.cn/s/articlelist\_1273160402\_0\_2.html" title="跳转至第 2 页">2</a></li>

<li><a href="http://blog.sina.com.cn/s/articlelist\_1273160402\_0\_3.html" title="跳转至第 3 页">3</a></li>

。。。。。。。。。。。。。。此处省略重复

<li><a href="http://blog.sina.com.cn/s/articlelist\_1273160402\_0\_9.html" title="跳转至第 9 页">9</a></li>

<li><a href="http://blog.sina.com.cn/s/articlelist\_1273160402\_0\_10.html" title="跳转至第 10 页">10</a></li>

<li class="SG\_pgelip">...</li>

<li class="SG\_pgnext"><a href="http://blog.sina.com.cn/s/articlelist\_1273160402\_0\_2.html" title="跳转至第 2 页">下一页&nbsp;&gt;</a></li>

&nbsp;&nbsp;<span style="color:#888888;">共12页</span>

</ul>

</div>

如果我们想要跳转到第二页中，我们要做的工作是检查class 为“SG\_pgnext”的l，i标签的href值是否存在。

* 如果存在，则跳转到第二个页面中，然后从第二个页面开始获取。
* 如果不存在，则结束页面获取。

function readArticleList(url,callback) {

//读取分类页面

request(url, function (err, res) {

if (err) {

return console.error(err);

}

//根据网页内容创建dom操作对象

var $ = cheerio.load(res.body.toString());

//读取博文的列表

var articleList = [];

$('.articleList .articleCell').each(function () {

var $this = $(this);

var $title = $this.find('.atc\_title a');

var $time = $this.find('.atc\_tm');

var item = {

title: $title.text().trim(),

url: $title.attr('href'),

time: $time.text().trim()

};

//从url中取出文章的id

//http://blog.sina.com.cn/s/blog\_4be2e2d20102vtri.html

var s = item.url.match(/blog\_([a-zA-Z0-9]+)\.html/);

if (Array.isArray(s)) {

item.id = s[1];

articleList.push(item);

}

});

//检查是否有下一页

var nextUrl = $('.SG\_pgnext a').attr('href');

if (nextUrl) {

//读取下一页

readArticleList(nextUrl,function(err,articleList2){

if(err) return callback(err);

//合并结果

callback(null,articleList.concat(articleList2));

});

}

else{ //返回结果集

callback(null,articleList);

}

});

}

这个过程是利用了递归。

# 3：将文章类表存放在数据库中

步骤：

*\* 查询文章列表是否存在，  
\* 如果已经存在了，做update操作  
\* 如果不存在，则做insert操作*

function articleListSave(articleList,callback){

pool.getConnection(function(err,connection){

var queryArticleList\_sql="select count(\*) cont from article\_list where id=?";

connection.query(queryArticleList\_sql,[articleList.id],function(err,result){

if (err) {

console.log("queryArticleList\_sql Error: " + err.message);

return;

}

if(result[0].cont!=0){

//已经存放在则做更新操作

var updateArticleList\_sql="update article\_list set title=? , url=? ,time=? where id=?";

connection.query(updateArticleList\_sql,

[articleList.title,articleList.url,articleList.time,articleList.id],

function(err,result){

if (err) {

console.log("updateArticleList\_sql Error: " + err.message);

return;

}

callback(err,result);

connection.release();

});

}

else{

//不存在的话，则做插入操作

var insertArticleList="insert into article\_list(id,title,time,url) values(?,?,?,?) ";

connection.query(insertArticleList,

[articleList.id,articleList.title,articleList.time,articleList.url],

function(err,result){

if (err) {

console.log("insertArticleList Error: " + err.message);

return;

}

callback(err,result);

connection.release();

});

}

});

})};

注意：

在数据库中该数据存在与不存在的判断依据。

将以上的内容合并

read.readArticleList(url,function(err,articleList){

if(err) console.error(err.stack);

async.eachSeries(articleList,function(article,next){

save.articleListSave(article,function(err,result){

if(err){

console.log('readArticleList into mysql Error'+err.message);

}

next();

});

},function(err){

//当遍历完了所有的回调函数之后，执行该命令

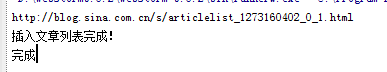
if(err) return console.error(err.stack);

console.log('完成');

});

console.log('插入文章列表完成！');

});



注意：

在这里不能使用 articleList.forEach(),,

articleList.forEach(function(value,index){

save.articleListSave(value,function(err,result){

if(err){

console.log('readArticleList into mysql Error'+err.message);

}

});

});

会发现在数据库中根本就存储不了数据，原因在于：

【异步编程】

4：获取文章的详细信息

步骤：

1：获取html页面的dom操作对象。

2：根据dom对象，获取文章的tags 内容 等信息

function readArticleDetail(url,callback){

//读取博文页面

console.log('url:'+url);

request(url,function(err,res){

if(err) console.error(err);

//根据网页内容创建DOM操作对象

var $= cheerio.load(res.body.toString());

console.log($);

//获取文章

var tags=[];

$('.blog\_tag h3 a').each(function(){

var tag=$(this).text().trim();

if(tag){

tags.push(tag);

}

});

//获取文章内容

var content =$('.articalContent').text();

//从url中获取id

//http://blog.sina.com.cn/s/blog\_4be2e2d20102vtri.html

var s = url.match(/blog\_([a-zA-Z0-9]+)\.html/);

if (Array.isArray(s)) {

var id = s[1];

}

callback(null,{tags:tags,content:content,id:id,url:url});

});

}

在数据库中的操作和操作 article\_list表的操作类似，这里就不重复叙述。

# 4：数据库的表结构

CREATE TABLE `article\_detail` (

`id` varchar(20) DEFAULT NULL,

`url` varchar(255) DEFAULT NULL,

`tags` varchar(255) DEFAULT NULL,

`content` longtext,

`pk\_id` int(11) NOT NULL AUTO\_INCREMENT,

PRIMARY KEY (`pk\_id`)

) ENGINE=InnoDB AUTO\_INCREMENT=1109 DEFAULT CHARSET=utf8;

CREATE TABLE `article\_detail` (

`id` varchar(20) DEFAULT NULL,

`url` varchar(255) DEFAULT NULL,

`tags` varchar(255) DEFAULT NULL,

`content` longtext,

`pk\_id` int(11) NOT NULL AUTO\_INCREMENT,

PRIMARY KEY (`pk\_id`)

) ENGINE=InnoDB AUTO\_INCREMENT=1109 DEFAULT CHARSET=utf8;

这个是最后的结果，表明爬虫工具的第一步完成了。

