
Table of Contents

概述	1.1
第一个demo	1.2
configs详解——之成员	1.3
configs详解——之field	1.4
爬虫进阶开发——之内置方法	1.5
爬虫进阶开发——之回调函数	1.6
爬虫进阶开发——之技巧篇	1.7
如何实现模拟登录？	1.7.1
file_get_contents 设置代理抓取页面	1.7.2
如果内容页有分页，该如何爬取到完整数据？	1.7.3
如何快速请求页面测试爬虫规则	1.7.4
多任务爬虫	1.7.5
Swoole下的多任务爬虫	1.7.6
Workerman下的多任务爬虫	1.7.7
如何实现爬虫集群负载？	1.7.8

PHP蜘蛛爬虫开发文档

《我用爬虫一天时间“偷了”知乎一百万用户，只为证明PHP是世界上最好的语言》所使用的程序框架

编写PHP网络爬虫，需要具备以下技能:

- 爬虫采用PHP编写
- 从网页中抽取数据需要用XPath(后面会开放支持CSS选择器)
- 很多情况下都会用到正则表达式
- Chrome的开发者工具是神器, 很多AJAX请求需要用它来分析

第一个demo

爬虫采用PHP编写,下面以糗事百科为例,来看一下我们的爬虫长什么样子:

```
$configs = array(
    'name' => '糗事百科',
    'domains' => array(
        'qiushibaike.com',
        'www.qiushibaike.com'
    ),
    'scan_urls' => array(
        'http://www.qiushibaike.com/'
    ),
    'content_url_regexes' => array(
        "http://www.qiushibaike.com/article/\d+"
    ),
    'list_url_regexes' => array(
        "http://www.qiushibaike.com/8hr/page/\d+\?s=\d+"
    ),
    'fields' => array(
        array(
            // 抽取内容页的文章内容
            'name' => "article_content",
            'selector' => "//*[ @id='single-next-link']",
            'required' => true
        ),
        array(
            // 抽取内容页的文章作者
            'name' => "article_author",
            'selector' => "//div[contains(@class,'author')]/h2"
        ),
        array(
            'required' => true
        )
    ),
);
$spider = new phpspider($configs);
$spider->start();
```

爬虫的整体框架就是这样, 首先定义了一个`$configs`数组, 里面设置了待爬网站的一些信息, 然后通过调用 `$spider = new phpspider($configs);` 和 `$spider->start();` 来配置并启动爬虫.

`$configs`对象如何定义, 后面会作详细介绍.^_^

configs详解——之成员

爬虫的整体框架是这样：首先定义了一个`$configs`数组，里面设置了待爬网站的一些信息，然后通过调用 `$spider = new phpspider($configs);` 和 `$spider->start();` 来配置并启动爬虫。

`$configs`数组中可以定义下面这些元素

name

定义当前爬虫名称

String类型 可选设置

举个例子:

```
'name' => '糗事百科'
```

save_running_state

保存爬虫运行状态

设置为`true`则程序会调用`redis`作为任务数据存储

需要配合`redis`来保存采集任务数据，供程序下次执行使用

布尔类型 可选设置

save_running_state默认值为`false`，即不保存爬虫运行状态

举个例子:

```
'save_running_state' => true
```

tasknum

同时工作的爬虫任务数

需要配合redis保存采集任务数据，供进程间共享使用

需要配合自带的worker类、Swoole或者Workerman的多进程使用

整型 可选设置

tasknum默认值为**1**，即单进程任务爬取

举个栗子:

开启5个进程爬取网页

```
'tasknum' => 5
```

多任务爬虫

Swoole下的多任务爬虫

Workerman下的多任务爬虫

注意: 保存爬虫运行状态 和 多任务 都需要 **redis** 来保存采集任务数据，修改

config/inc_config.php 文件中的 **redis**配置

```
$GLOBALS['config']['redis'] = array(
    'host'      => '127.0.0.1',
    'port'      => 6379,
    'pass'      => '',
    'prefix'    => 'phpspider',
    'timeout'   => 30,
);
```

domains

定义爬虫爬取哪些域名下的网页，非域名下的url会被忽略以提高爬取速度

数组类型 不能为空

举个栗子:

```
'domains' => array(  
    'qiushibaike.com',  
    'www.qiushibaike.com'  
)
```

scan_urls

定义爬虫的入口链接, 爬虫从这些链接开始爬取, 同时这些链接也是监控爬虫所要监控的链接

数组类型 不能为空

举个例子:

```
'scan_urls' => array(  
    'http://www.qiushibaike.com/'  
)
```

content_url_regexes

定义内容页url的规则

内容页是指包含要爬取内容的网页 比

如 `http://www.qiushibaike.com/article/115878724` 就是糗事百科的一个内容页

数组类型 正则表达式 最好填写以提高爬取效率

举个例子:

```
'content_url_regexes' => array(  
    "http://www.qiushibaike.com/article/\d+",  
)
```

list_url_regexes

定义列表页url的规则

对于有列表页的网站,使用此配置可以大幅提高爬虫的爬取速率

列表页是指包含内容页列表的网页 比

如 `http://www.qiushibaike.com/8hr/page/2/?s=4867046` 就是糗事百科的一个列表页

数组类型 正则表达式

举个栗子:

```
'list_url_regexes' => array(  
    "http://www.qiushibaike.com/8hr/page/\d+\?s=\d+"  
)
```

fields

定义内容页的抽取规则

规则由一个个 `field` 组成,一个 `field` 代表一个数据抽取项

数组类型 不能为空

举个栗子:

```
'fields' => array(  
    array(  
        'name' => "content",  
        'selector' => "//*[@id='single-next-link']",  
        'required' => true,  
    ),  
    array(  
        'name' => "author",  
        'selector' => "//div[contains(@class,'author')]/h2",  
    )  
)
```

上面的例子从网页中抽取内容和作者,抽取规则是针对糗事百科的内容页写的

proxy

代理服务器

如果爬取的网站根据IP做了反爬虫,可以设置此项

String类型 可选设置

举个例子:

```
'proxy' => 'http://proxy.abuyun.com:9010'
```

proxy_auth

代理服务器验证

如果爬取的网站根据IP做了反爬虫,可以设置此项

String类型 可选设置

举个例子:

```
'proxy_auth' => 'name:pass'
```

注意:如果对方根据IP做了反爬虫技术,你可能需要到 [阿布云代理](#) 申请代理通道或者 第三方免费代理IP,然后在这里填写代理信息

interval

爬虫爬取每个网页的时间间隔

单位:秒

整型 可选设置

举个例子:

```
'interval' => 1
```

timeout

爬虫爬取每个网页的超时时间

单位:秒

整型 可选设置

timeout默认值为**5**秒

举个例子:

```
'timeout' => 5
```

collect_fails

爬虫爬取每个网页允许的失败次数

网络不好可能导致爬虫在超时时间内抓取失败, 可以设置此项允许爬虫重复爬取

整型 可选设置

collect_fails默认值为**0**, 即不重复爬取

举个例子:

```
'collect_fails' => 5 // 重复爬取5次
```

user_agent

爬虫爬取网页所使用的浏览器类型

```
phpspider::AGENT_ANDROID
```

```
phpspider::AGENT_IOS
```

```
phpspider::AGENT_PC
```

```
phpspider::AGENT_MOBILE
```

枚举类型 可选设置

栗子1:

使用内置的枚举类型

phpspider::AGENT_ANDROID, 表示爬虫爬取网页时, 使用安卓手机浏览器

phpspider::AGENT_IOS, 表示爬虫爬取网页时, 使用苹果手机浏览器

phpspider::AGENT_PC, 表示爬虫爬取网页时, 使用**PC**浏览器

phpspider::AGENT_MOBILE, 表示爬虫爬取网页时, 使用移动设备浏览器

```
'user_agent' => phpspider::AGENT_ANDROID
```

栗子2:

使用自定义类型

```
'user_agent' => "Mozilla/5.0"
```

export

爬虫爬取数据导出

export_type : 导出类型 csv、sql、db

export_file : 导出 csv、sql 文件地址

export_table : 导出db、sql数据表

注意导出到数据库的表和字段要和上面的**fields**对应

数组类型 可选设置

栗子1:

导出CSV结构数据到文件

```
'export' => array(  
    'type' => 'csv',  
    'file' => PATH_DATA.'/qiushibaike.csv', // data目录下  
)
```

栗子2:

导出SQL语句到文件

```
'export' => array(  
    'type' => 'sql',  
    'file' => PATH_DATA.'/qiushibaike.sql',  
    'table' => '数据表',  
)
```

栗子3:

导出数据到Mysql

```
'export' => array(
    'type' => 'db',
    'table' => '数据表', // 如果数据表没有数据新增请检查表结构和字段名
                        是否匹配
)
```

注意: 导出到 **Mysql** 需要修改 **config/inc_config.php** 文件中的 **mysql**配置

```
$GLOBALS['config']['db'] = array(
    'host'   => '127.0.0.1',
    'port'   => 3306,
    'user'   => 'root',
    'pass'   => 'root',
    'name'   => 'demo',
);
```

configs详解——之field

`field` 定义一个抽取项, 一个 `field` 可以定义下面这些东西

name

给此项数据起个变量名

变量名中不能包含.

如果抓取到的数据想要以文章或者问答的形式发布到网站(WeCenter, WordPress, Discuz!等), `field`的命名请参考两个完整demo中的命名, 否则无法发布成功

String类型 不能为空

举个例子:

给 `field` 起了个名字叫 `content`

```
array(  
    'name' => "content",  
    'selector' => "//*[@id='single-next-link']"  
)
```

selector

定义抽取规则, 默认使用xpath

如果使用其他类型的, 需要指定`selector_type`

String类型 不能为空

举个例子:

使用xpath来抽取糗事百科的笑话内容, `selector`的值就是内容的xpath

```
array(  
  'name' => "content",  
  'selector' => "//*[@id='single-next-link']"  
)
```

selector_type

抽取规则的类型

目前可用 `xpath`, `regex`

默认 `xpath`

枚举类型

栗子1: `selector`默认使用`xpath`

```
array(  
  'name' => "content",  
  'selector' => "//*[@id='single-next-link']" // xpath抽取规则  
)
```

栗子2: 使用正则表达式来抽取数据

```
array(  
  'name' => "content",  
  'selector' => '#<div\sclass="content">([^\s/]+)</div>#i' // regex抽取规则  
)
```

required

定义该 `field` 的值是否必须, 默认`false`

赋值为`true`的话, 如果该 `field` 没有抽取到内容, 该`field`对应的整条数据都将被丢弃

布尔类型

举个栗子:

```
array(  
  'name' => "content",  
  'selector' => "//*[@id='single-next-link']",  
  'required' => true  
)
```

repeated

定义该 `field` 抽取到的内容是否是有多项, 默认 `false`
赋值为`true`的话, 无论该 `field` 是否真的是有多项, 抽取到的结果都是数组结构

布尔类型

举个例子:

爬取的网页中包含多条评论, 所以抽取评论的时候要将`repeated`赋值为`true`

```
array(  
  'name' => "comments",  
  'selector' => "//*[@id='zh-single-question-page']//a[contains(@class, 'zm-item-tag')]",  
  'repeated' => true  
)
```

children

为此 `field` 定义子项
子项的定义仍然是一个 `fields` 数组
没错, 这是一个树形结构

数组类型

举个例子:

抓取糗事百科的评论, 每个评论爬取了内容, 点赞数

```
array(  
  'name' => "article_comments",  
  'selector' => "//div[contains(@class, 'comments-wrap')]",  
  'children' => array(  
    array(  
      'name' => "replay",  
      'selector' => "//div[contains(@class, 'replay')]",  
      'repeated' => true,  
    ),  
    array(  
      'name' => "report",  
      'selector' => "//div[contains(@class, 'report')]",  
      'repeated' => true,  
    )  
  )  
)
```


爬虫进阶开发——之内置方法

本节介绍爬虫的内置方法

set_useragent(\$useragent)

一般在 `on_start` 回调函数（在[爬虫进阶开发——之回调函数](#)中会详细描述）中调用, 设置浏览器useragent

`@param $useragent` 需添加的useragent

默认使用 **useragent : Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/47.0.2526.106 Safari/537.36**

[点击查看“常见浏览器useragent大全”](#)

举个栗子:

```
$spider->on_start = function($phpspider)
{
    $phpspider->set_useragent("Mozilla/4.0 (compatible; MSIE 6.0
; ) Opera/UCWEB7.0.2.37/28/");
};
```

add_header(\$key, \$value)

一般在 `on_start` 回调函数（在[爬虫进阶开发——之回调函数](#)中会详细描述）中调用, 用来添加一些HTTP请求的Header

`@param $key` Header的key, 如User-Agent,Referer等

`@param $value` Header的值

举个栗子:

Referer是HTTP请求Header的一个属性, `http://www.9game.cn/kc/` 是Referer的值

```
$spider->on_start = function($phpspider)
{
    $phpspider->add_header("Referer", "http://www.9game.cn/kc/")
;
};
```

add_cookie(\$key, \$value, \$domain='')

一般在 `on_start` 回调函数（在[爬虫进阶开发——之回调函数](#)中会详细描述）中调用，用来添加一些HTTP请求的Cookie

@param \$key Cookie的key

@param \$value Cookie的值 @param \$domain 默认放到全局Cookie，设置域名后则放到相应域名下

举个栗子：

cookie是由键-值对组成的，BAIDUID是cookie的key，

FEE96299191CB0F11954F3A0060FB470:FG=1则是cookie的值

```
$spider->on_start = function($phpspider)
{
    $phpspider->add_cookie("BAIDUID", "FEE96299191CB0F11954F3A00
60FB470:FG=1");
    // 把Cookie设置到 www.phpspider.org 域名下
    $phpspider->add_cookie("NAME", "phpspider", "www.phpspider.o
rg");
};
```

get_cookie(\$name, \$domain = '')

一般在 `on_start` 回调函数（在[爬虫进阶开发——之回调函数](#)中会详细描述）中调用，用来得到某个域名所附带的某个Cookie

@param \$name Cookie的名称

@param \$domain configs的domains成员中的元素

举个栗子：得到 `s.weibo.com` 域名所附带的 Cookie，并将 Cookie 添加到 `weibo.com` 的域名中

```

$configs = array(
    'domains' => array(
        's.weibo.com',
        'weibo.com'
    )
    // configs的其他成员
    ...
);

$spider->on_start = function($phpspider)
{
    $cookie = $phpspider->get_cookie("SUB", "s.weibo.com");
    // 把Cookie设置到 weibo.com 域名下
    $phpspider->add_cookie("SUB", $cookie, "weibo.com");
};

```

add_cookies(\$cookies, \$domain='')

一般在 `on_start` 回调函数（在[爬虫进阶开发——之回调函数](#)中会详细描述）中调用, 用来添加一些HTTP请求的Cookie

@param \$cookies 多个Cookie组成的字符串

@param \$domain 默认放到全局Cookie，设置域名后则放到相应域名下

举个例子:

cookies是多个cookie的键-值对组成的字符串，用;分隔。BAIDUID和BIDUPSID是cookie的key，FEE96299191CB0F11954F3A0060FB470:FG=1和FEE96299191CB0F11954F3A0060FB470是cookie的值，键-值对用=相连

```

$spider->on_start = function($phpspider)
{
    $phpspider->add_cookies("BAIDUID=FEE96299191CB0F11954F3A0060FB470:FG=1; BIDUPSID=FEE96299191CB0F11954F3A0060FB470;");
    // 把Cookie设置到 www.phpspider.org 域名下
    $phpspider->set_cookie("NAME", "www.phpspider.org");
};

```

get_cookies(\$domain = '')

一般在 `on_start` 回调函数（在[爬虫进阶开发——之回调函数](#)中会详细描述）中调用，用来得到某个域名所附带的所有Cookie

@param \$domain configs的domains成员中的元素

@return array 返回的是所有Cookie的数组

举个例子：得到 `s.weibo.com` 域名所附带的 Cookie，并将 Cookie 添加到 `weibo.com` 的域名中

```
$configs = array(
    'domains' => array(
        's.weibo.com',
        'weibo.com'
    )
    // configs的其他成员
    ...
);

$spider->on_start = function($phpspider)
{
    $cookies = $phpspider->get_cookies("s.weibo.com");
    // 返回的是数组，可以输出看看所有的Cookie内容
    print_r($cookies);
    // 数组转化成String
    $cookies = implode(";", $cookies);
    // 把Cookie设置到 weibo.com 域名下
    $phpspider->add_cookies($cookies, "weibo.com");
};
```

add_url(\$url, \$options = array())

一般在 `on_scan_page` 和 `on_list_page` 回调函数（在[爬虫进阶开发——之回调函数](#)中会详细描述）中调用，用来往待爬队列中添加url

@param \$url 待添加的url

@param \$options 成员包括method, headers, fields, context_data, reserve和,proxy和proxy_auth, 如下所示:

@param \$options['method'] 默认为"get"请求, 也支持"post"请求

@param \$options['headers'] 此url的Headers, 可以为空

@param \$options['fields'] 发送请求时需添加的参数, 可以为空

@param \$options['context_data'] 此url附加的数据, 可以为空

@param \$options['proxy'] 访问此url时使用的代理服务器, 不使用请留空

@param \$options['proxy_auth'] 代理服务器验证

栗子1:

```
$spider->on_scan_page = function($page, $content, $phpspider)
{
    $regex = "#http://pic.qiushibaike.com/system/pictures/\d+#";
    $urls = array();
    preg_match_all($regex, $content, $out);
    $urls = empty($out[0]) ? array() : $out[0];
    if (!empty($urls)) {
        foreach ($urls as $url)
        {
            $phpspider->add_url($url);
        }
    }
    ...
    return false;
};
```

栗子2:

```

$spider->on_list_page = function($page, $content, $phpspider)
{
    ...
    $next_url = str_replace($page['url'], "page=".$current_page_
num, "page=".$page_num);
    $options = array(
        'method' => 'post',
        'fields' => array(
            'page' => $page_num,
            'size' => 18
        )
    );
    $phpspider->add_url($next_url, $options);
    return false;
};

```

request_url(\$url, \$options = array())

一般在 `on_start` , `on_download_page` , `on_scan_page` 和 `on_list_page` 回调函数（在[爬虫进阶开发——之回调函数](#)中会详细描述）中调用, 下载网页, 得到网页内容

@param \$url 待添加的url

@param \$options 成员包括method, headers, fields, context_data, reserve, proxy和proxy_auth, 如下所示:

@param \$options['method'] 默认为"get"请求, 也支持"post"请求

@param \$options['headers'] 此url的Headers, 可以为空

@param \$options['fields'] 发送请求时需添加的参数, 可以为空

@param \$options['context_data'] 此url附加的数据, 可以为空

@param \$options['proxy'] 访问此url时使用的代理服务器, 不使用请留空

@param \$options['proxy_auth'] 代理服务器验证

举个栗子:

```

$spider->on_download_page = function($page, $phpspider)
{
    $url = "https://checkcoverage.apple.com/cn/zh/?sn=FK1QPNCEGR
YD";
    $options = array(
        'method' => 'post',
        'fields' => array(
            'sno' => 'FK1QPNCEGRYD',
            'CSRFToken' => $result[1]
        )
    );
    // 通过发送带参数的post请求，下载网页，并将网页内容赋值给result
    $request = $phpspider->request_url($url, $options);
    ...
    return $page;
};

```

log(\$msg, \$type = 'info')

框架提供不同级别的日志打印

@param \$msg 日志内容

@param \$type 日志类型包括 `info` 、 `debug` 、 `warn` 、 `error` , 如下所示:

info: 打印普通日志, 调试时使用

举个例子:

```

$phpspider->log("测试log日志");
$phpspider->log("测试log日志","info");

```

debug: 打印出错级别日志, 调试时使用

举个例子:

```

$phpspider->log("正在提取文章标题数据", "debug");

```

warn: 打印警告情况的日志

举个例子:

```
$phpspider->log("XX文件解析失败","warn");
```

error: 打印错误情况的日志

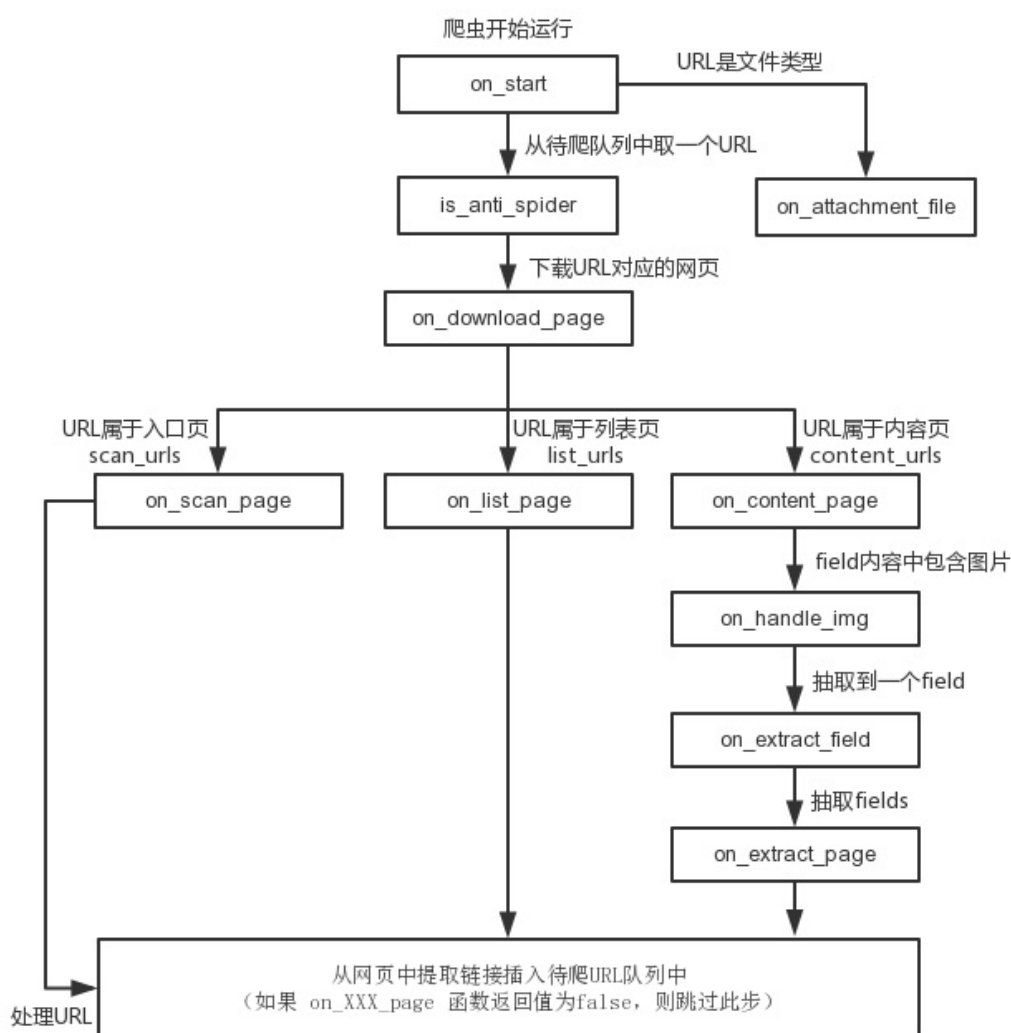
举个例子:

```
$phpspider->log("XPath错误","error");
```


爬虫进阶开发——之回调函数

回调函数是在爬虫爬取并处理网页的过程中设置的一些系统钩子,通过这些钩子可以完成一些特殊的处理逻辑。

下图是PHP蜘蛛爬虫爬取并处理网页的流程图,矩形方框中标识了爬虫运行过程中所使用的重要回调函数:



on_start(\$phpspider)

爬虫初始化时调用,用来指定一些爬取前的操作

@param \$phpspider 爬虫对象

在函数中可以调用 `$phpspider->add_header($key, $value)` , `$phpspider->add_cookie($key, $value)` , `$phpspider->add_cookies($cookies)` 等

举个栗子:

在爬虫开始爬取之前给所有待爬网页添加一个Header

```
$spider->on_start = function($phpspider)
{
    $phpspider->add_header("Referer", "http://buluo.qq.com/p/index.html");
};
```

on_attachment_file(\$url, \$fileinfo)

发现附件文件触发

@param \$url 附件URL地址

@param \$fileinfo 附件文件信息

@param \$fileinfo['basename'] 文件名+后缀名

@param \$fileinfo['filename'] 文件名

@param \$fileinfo['fileext'] 后缀名

@param \$fileinfo['filesize'] 文件大小

@param \$fileinfo['atime'] 文件最后访问时间 @param \$fileinfo['mtime'] 文件最后修改时间

```
$configs = array(
    'domains' => array(
        'ocnt0imhl.bkt.clouddn.com',
    ),
    // 直接把要采集的图片放到入口数组，请求的时候就会触发回调函数
    'scan_urls' => array(
        "http://ocnt0imhl.bkt.clouddn.com/imgs/1637/2015-07/k306n1wzvqk669nm.jpg",
    ),
    // 按规则匹配到的jpg也会触发回调函数
    'list_url_regexes' => array(
        "http://ocnt0imhl.bkt.clouddn.com/imgs/\d+/\d+-\d+\d+.jp
```

```
g",
    ),
);

$spider->on_start = function($phpspider)
{
    // 也可以在这里直接请求，也会触发回调函数
    $url = "http://ocnt0imhl.bkt.clouddn.com/imgs/1637/2015-07/k
306n1wzvq669nm.jpg";
    $options = array(
        'method' => 'post',
        'data' => array(
            'export' => 'jpg',
        )
    );
    $phpspider->request_url($url, $options);
};

$spider->on_attachment_file = function($attachment_url, $mime_type)
{
    // 输出文件URL地址和文件类型
    var_dump($attachment_url, $mime_type);

    if ($mime_type == 'jpg')
    {
        // 以纳秒为单位生成随机数
        $filename = uniqid();
        // 在data目录下生成图片
        $filepath = PATH_DATA."/{ $filename }.jpg";
        // 用系统自带的下载器wget下载
        exec("wget { $attachment_url } -O { $filepath }");

        // 用PHP函数下载，容易耗尽内存，慎用
        // $data = file_get_contents($attachment_url);
        // file_put_contents($filepath, $attachment_url);
    }
};
```

is_anti_spider(\$url, \$content)

判断当前网页是否被反爬虫了, 需要开发者实现

@param \$url 当前网页的url

@param \$content 当前网页内容

@return 如果被反爬虫了, 返回true, 否则返回false

举个栗子:

```
$spider->is_anti_spider = function($url, $content)
{
    // $content中包含"404页面不存在"字符串
    if (strpos($content, "404页面不存在") !== false)
    {
        return true; // 返回当前网页被反爬虫了
    }
    return false;
};
```

on_download_page(\$page, \$phpspider)

在一个网页下载完成之后调用. 主要用来对下载的网页进行处理.

@param \$page 当前下载的网页页面的对象

@param \$phpspider 爬虫对象

@return 返回处理后的网页

@param \$page['url'] 当前网页的URL

@param \$page['raw'] 当前网页的内容

@param \$page['request'] 当前网页的请求对象

举个栗子:

比如下载了某个网页, 希望向网页的body中添加html标签, 处理过程如下:

```
$spider->on_download_page = function($page, $phpspider)
{
    $page_html = "<div id=\"comment-pages\"><span>5</span></div>";
    $index = strpos($page['row'], "</body>");
    $page['raw'] = substr($page['raw'], 0, $index) . $page_html . substr($page['raw'], $index);
    return $page;
};
```

on_scan_page(\$page, \$content, \$phpspider)

在爬取到入口url的内容之后, 添加新的url到待爬队列之前调用. 主要用来发现新的待爬url, 并且能给新发现的url附加数据 (点此查看“url附加数据”实例解析)。

@param \$page 当前正在爬取的网页页面的对象

@param \$content 当前网页内容

@param \$phpspider 当前爬虫对象

@return 返回false表示不需要再从此网页中发现待爬url

@param \$page['url'] 当前网页的URL

@param \$page['raw'] 当前网页的内容

@param \$page['request'] 当前网页的请求对象

此函数中通过调用\$phpspider->add_url(\$url, \$options)函数来添加新的url到待爬队列。

栗子1:

实现这个回调函数并返回false, 表示爬虫在处理这个scan_url的时候, 不会从中提取待爬url

```
$spider->on_scan_page = function($page, $content, $phpspider)
{
    return false;
};
```

栗子2:

生成一个新的url添加到待爬队列中, 并通知爬虫不再从当前网页中发现待爬url

```

$spider->on_scan_page = function($page, $content, $phpspider)
{
    $array = json_decode($page['raw'], true);
    foreach ($array as $v)
    {
        $lastid = $v['id'];
        // 生成一个新的url
        $url = $page['url'] . $lastid;
        // 将新的url插入待爬的队列中
        $phpspider->add_url($url);
    }
    // 通知爬虫不再从当前网页中发现待爬url
    return false;
};

```

on_list_page(\$page, \$content, \$phpspider)

在爬取到入口url的内容之后, 添加新的url到待爬队列之前调用. 主要用来发现新的待爬url, 并且能给新发现的url附加数据 (点此查看“url附加数据”实例解析)。

@param \$page 当前正在爬取的网页页面的对象 @param \$content 当前网页内容

@param \$phpspider 当前爬虫对象

@return 返回false表示不需要再从此网页中发现待爬url

@param \$page['url'] 当前网页的URL

@param \$page['raw'] 当前网页的内容

@param \$page['request'] 当前网页的请求对象

此函数中通过调用\$phpspider->add_url(\$url, \$options)函数来添加新的url到待爬队列。

栗子1:

实现这个回调函数并返回false, 表示爬虫在处理这个scan_url的时候, 不会从中提取待爬url

```
$spider->on_list_page = function($page, $content, $phpspider)
{
    return false;
};
```

栗子2:

生成一个新的url添加到待爬队列中，并通知爬虫不再从当前网页中发现待爬url

```
$spider->on_list_page = function($page, $content, $phpspider)
{
    $array = json_decode($page['raw'], true);
    foreach ($array as $v)
    {
        $lastid = $v['id'];
        // 生成一个新的url
        $url = $page['url'] . $lastid;
        // 将新的url插入待爬的队列中
        $phpspider->add_url($url);
    }
    // 通知爬虫不再从当前网页中发现待爬url
    return false;
};
```

on_content_page(\$page, \$content, \$phpspider)

在爬取到入口url的内容之后，添加新的url到待爬队列之前调用。主要用来发现新的待爬url，并且能给新发现的url附加数据（点此查看“url附加数据”实例解析）。

@param \$page 当前正在爬取的网页页面的对象

@param \$content 当前网页内容

@param \$phpspider 当前爬虫对象

@return 返回false表示不需要再从此网页中发现待爬url

@param \$page['url'] 当前网页的URL

@param \$page['raw'] 当前网页的内容

@param \$page['request'] 当前网页的请求对象

此函数中通过调用`$phpspider->add_url($url, $options)`函数来添加新的url到待爬队列。

栗子1:

实现这个回调函数并返回`false`，表示爬虫在处理这个`scan_url`的时候，不会从中提取待爬url

```
$spider->on_content_page = function($page, $content, $phpspider)
{
    return false;
};
```

栗子2:

生成一个新的url添加到待爬队列中，并通知爬虫不再从当前网页中发现待爬url

```
$spider->on_content_page = function($page, $content, $phpspider)
{
    $array = json_decode($page['raw'], true);
    foreach ($array as $v)
    {
        $lastid = $v['id'];
        // 生成一个新的url
        $url = $page['url'] . $lastid;
        // 将新的url插入待爬的队列中
        $phpspider->add_url($url);
    }
    // 通知爬虫不再从当前网页中发现待爬url
    return false;
};
```

on_handle_img(\$fieldname, \$img)

在抽取到**field**内容之后调用，对其中包含的**img**标签进行回调处理

@param \$fieldname 当前field的name. 注意: 子field的name会带着父field的name, 通过.连接.

@param \$img 整个img标签的内容

@return 返回处理后的img标签的内容

很多网站对图片作了延迟加载, 这时候就需要在这个函数里面来处理

举个例子:

汽车之家论坛帖子的图片大部分是延迟加载的, 默认会使

用 `http://x.autoimg.cn/club/lazyload.png` 图片url, 我们需要找到真实的图片url并替换, 具体实现如下:

```
$spider->on_handle_img = function($fieldname, $img)
{
    $regex = '/src="(https?:\\/.*)"/i';
    preg_match($regex, $img, $rs);
    if (!$rs)
    {
        return $img;
    }
    $url = $rs[1];
    if ($url == "http://x.autoimg.cn/club/lazyload.png")
    {
        $regex2 = '/src9="(https?:\\/.*)"/i';
        preg_match($regex, $img, $rs);
        // 替换成真是图片url
        if (!$rs)
        {
            $new_url = $rs[1];
            $img = str_replace($url, $new_url);
        }
    }
    return $img;
};
```

on_extract_field(\$fieldname, \$data, \$page)

当一个field的内容被抽取到后进行的回调, 在此回调中可以对网页中抽取的内容作进一步处理

@param \$fieldname 当前field的name. 注意: 子field的name会带着父field的name, 通过.连接.

@param \$data 当前field抽取到的数据. 如果该field是repeated, data为数组类型, 否则是String

@param \$page 当前正在爬取的网页页面的对象

@return 返回处理后的数据, 注意数据类型需要跟传进来的data类型匹配

@param \$page['url'] 当前网页的URL

@param \$page['raw'] 当前网页的内容

@param \$page['request'] 当前网页的请求对象

举个栗子:

比如爬取知乎用户的性别信息, 相关网页源码如下:

那么可以这样写:

```
$configs = array(
    // configs的其他成员
    'fields' => array(
        array(
            'name' => "gender",
            'selector' => "//span[contains(@class, 'gender')]/i/
@class",
        ),
        ...
    )
);

$spider->on_extract_field = function($fieldname, $data, $page)
{
    if ($fieldname == 'gender')
    {
        // data中包含"icon-profile-male"，说明当前知乎用户是男性
        if (strpos($data, "icon-profile-male") !== false)
        {
            return "男";
        }
        // data中包含"icon-profile-female"，说明当前知乎用户是女性
        elseif (strpos($data, "icon-profile-female") !== false)
        {
            return "女";
        }
        else
        {
            return "未知";
        }
    }
    return $data;
};
```

on_extract_page(\$page, \$data)

在一个网页的所有field抽取完成之后，可能需要对field进一步处理，以发布到自己的网站

@param \$page 当前正在爬取的网页页面的对象

@param \$data 当前网页抽取出来的所有field的数据

@return 返回处理后的数据, 注意数据类型不能变

@param \$page['url'] 当前网页的URL

@param \$page['raw'] 当前网页的内容

@param \$page['request'] 当前网页的请求对象

举个栗子:

比如从某网页中得到time和title两个field抽取项, 现在希望把time的值添加中括号后拼凑到title中, 处理过程如下:

```
$spider->on_extract_page = function($page, $data)
{
    $title = "[{$data['time']}]" . $data['title'];
    $data['title'] = $title;
    return $data;
};
```

爬虫进阶开发——之技巧篇

本节是开发爬虫模板时需要了解的技巧。包括，在爬取网站过程中经常遇到的问题，回调函数和内置函数的使用技巧等。

如何实现模拟登录？

通过模拟登录，可以解决登录后才能爬取某些网站数据的问题。接下来给你介绍模拟登录的实现。

通过发送HTTP请求来实现模拟登录。

举个例子：

在 `on_start` 回调函数（在[爬虫进阶开发——之回调函数](#)中会详细描述）中，调用 `$phpspider->request_url()` 函数发送带参数的POST请求，就可以实现模拟登录：

```
$spider->on_start = function($phpspider)
{
    // 登录请求url
    $login_url = "http://www.waduanzi.com/login?url=http%3A%2F%2Fwww.waduanzi.com%2F";
    // 提交的参数
    $options = array(
        'method' => 'post',
        'fields' => array(
            "LoginForm[returnUrl]" => "http%3A%2F%2Fwww.waduanzi.com%2F",
            "LoginForm[username]" => "用户名",
            "LoginForm[password]" => "密码",
            "yt0" => "登录",
        )
    );
    // 发送登录请求
    $phpspider->request_url($login_url, $options);
    // 登录成功后本框架会把Cookie保存到www.waduanzi.com域名下，我们可以看看是否是已经收集到Cookie了
    $cookies = $phpspider->get_cookies("www.waduanzi.com");
    print_r($cookies); // 可以看到已经输出Cookie数组结构

    // 框架自动收集Cookie，访问这个域名下的URL会自动带上
    // 接下来我们来访问一个需要登录后才能看到的页面
    $url = "http://www.waduanzi.com/member";
    $html = $phpspider->request_url($url);
    echo $html; // 可以看到登录后的页面，非常棒
};
```

如何获得提交参数？

登录需要登录验证信息，下面我们来看看如何获得一个网站所需要的登录信息还是以挖段子(www.waduanzi.com)为例，看看如何获得下面的信息

1、打开挖段子网站点击登录按钮进入登陆页：

```
http://www.waduanzi.com/login?url=http%3A%2F%2Fwww.waduanzi.com%2F
```

2、鼠标点击右键 -> 检查 从而打开**Chrome**浏览器的开发者工具



选择**Network**选项卡，勾选**Preserve log**选项

欢迎加入挖段子网

分享

邮箱/手机

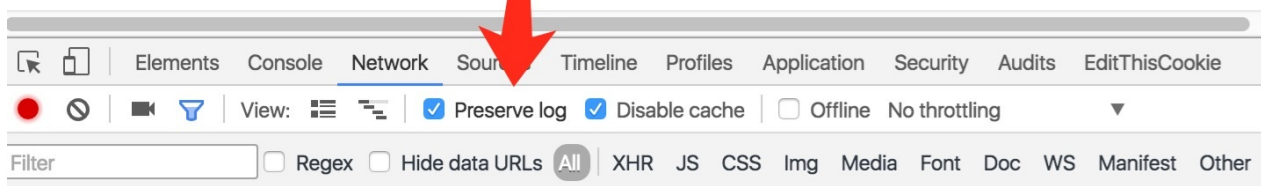
密码

3、填写登陆信息

> 还

4、点击登录

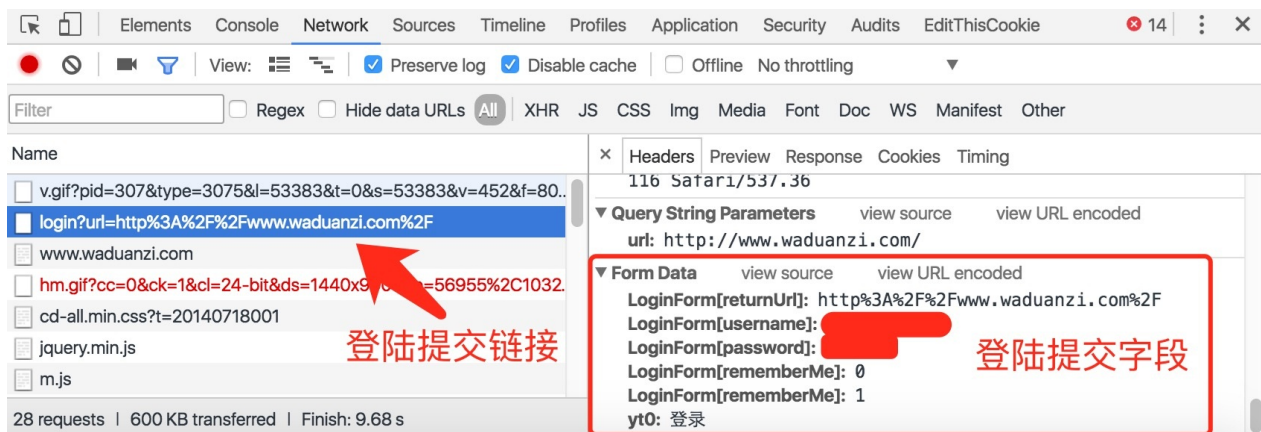
1、点击Network选项
2、勾选保留历史日志



Recording network activity...

Perform a request or hit **R** to record the reload.

3、填写登陆信息点击登录按钮，得到登录验证URL



4、上面的登录提交字段填入框架代码

```
'fields' => array(
    "LoginForm[returnUrl]" => "http%3A%2F%2Fwww.waduanzi.com%2F"
    ,
    "LoginForm[username]" => "用户名",
    "LoginForm[password]" => "密码",
    "yt0" => "登录",
)
```

大功告成^_^

file_get_contents 设置代理抓取页面

普通页面获取

```
$url = "http://www.epool1.com/archives/806/";
$content = file_get_contents($url);
preg_match_all("/<h1>(.*?)</h1>/is", $content, $matches);
print_r($matches[0]);
```

设置代理**IP**去采集数据

```
$context = array(
    'http' => array(
        'proxy' => 'tcp://192.168.0.2:3128', //这里设置你要使用的
        代理ip及端口号
        'request_fulluri' => true,
    ),
);
$content = stream_context_create($context);
$html = file_get_contents("http://www.epool1.com/archives/806/",
    false, $context);
echo $html;
```

设置需要验证的代理**IP**去采集数据

```
$auth = base64_encode('USER:PASS');    //LOGIN:PASSWORD 这里是代理
服务器的账户名及密码
$context = array(
    'http' => array(
        'proxy' => 'tcp://192.168.0.2:3128',    //这里设置你要使用的
代理ip及端口号
        'request_fulluri' => true,
        'header' => "Proxy-Authorization: Basic $auth",
    ),
);
$context = stream_context_create($context);
$html = file_get_contents("http://www.epool1.com/archives/806/",
    false, $context);
echo $html;
```

如果内容页有分页，该如何爬取到完整数据？

如果要爬取的某个内容页中有多个分页，该如何爬取这个内容页的完整数据呢？这里就无法使用 `on_list_page` 回调函数了，而需要使用 `field` 中的 `attached_url` 来请求其他分页的数据。

举个栗子：

爬取某网站文章时，发现有些文章有多个内容页面，处理过程如下：

```
$configs = array(
    // configs 的其他成员
    ...
    'fields' => array(
        array(
            'name' => "contents",
            'selector' => "//div[@id='pages']/a//@href",
            'repeated' => true,
            'children' => array(
                array(
                    // 抽取出其他分页的url待用
                    'name' => 'content_page_url',
                    'selector' => "//text()"
                ),
                array(
                    // 抽取其他分页的内容
                    'name' => 'page_content',
                    // 发送 attached_url 请求获取其他的分页数据
                    // attached_url 使用了上面抓取的 content_page_u
                    'source_type' => 'attached_url',
                    'attached_url' => 'content_page_url',
                    'selector' => "/*[@id='big-pic']"
                ),
            ),
        ),
    ),
);
```

在爬取到所有的分页数据之后，可以在 `on_extract_page` 回调函数中将这些数据组合成完整的数据

```
$spider->on_extract_field = function($fieldname, $data, $page)
{
    if ($fieldname == 'contents')
    {
        if (!empty($data))
        {
            $contents = $data;
            $data = "";
            foreach ($contents as $content)
            {
                $data .= $content['page_content'];
            }
        }
    }
    return $data;
};
```

如何快速请求页面测试爬虫

在运行爬虫框架前，我们可能需要做很多准备工作，比如：登录验证测试、内容提取规则测试，这个时候我们就可以把phpspider框架当做类库来使用，即不启用 `start()` 方法。

内容提取测试

比如我们来提取epooll这个站点的文章标题

```
$spider = new phpspider();
$url = "http://www.epool1.com/archives/806/";
$html = $spider->request_url($url);

$fieldname = "标题";
$selector = "//div[contains(@class, 'page-header')]/h1/a";

$result = $spider->get_fields_xpath($html, $selector, $fieldname);
print_r($result);
```

多任务爬虫

天下爬虫，唯快不破，配合多进程使用，phpspider可以快到你怕，下面我们来看看如何实现一个多任务爬虫。

```
$configs = array(
    'name' => '糗事百科',
    'tasknum' => 4,                // 爬虫任务数
    'save_running_state' => true, // 保存运行状态
    ...
);
$spider = new phpspider($configs);
// 单进程(单任务)模式
//$spider->start();

// 多进程(多任务)模式
$w = new worker();
// 直接使用上面配置的任务数作为worker进程数
$w->count = $configs['tasknum'];
$w->on_worker_start = function($worker) use ($spider) {

    $taskmaster = false;
    // 把第一个worker进程当做主任务
    if ($worker->worker_id == 1)
    {
        $taskmaster = true;
    }
    $spider->start($taskmaster, $worker->worker_id);

};
// 开始运行多进程
$w->run();
```


Swoole下的多任务爬虫

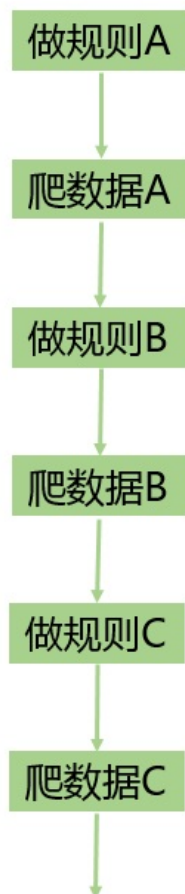
敬请期待 ^_^

Workerman下的多任务爬虫

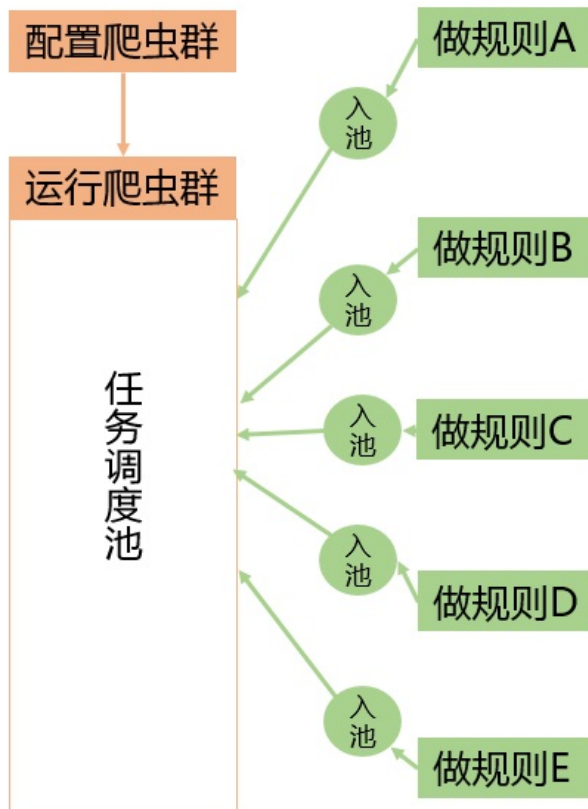
敬请期待 ^_^

如何实现爬虫集群负载

低效率的普通模式



高效率的爬虫群模式



思路如上，功能敬请期待^_^