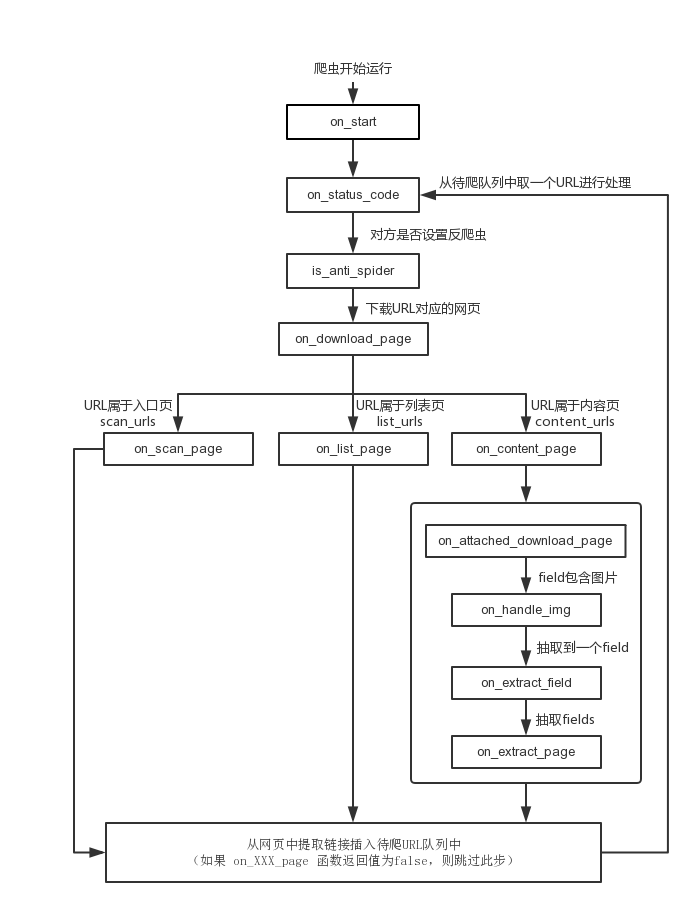
**爬虫进阶开发——之回调函数**

**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requests::set\_header($key,$value), requests::set\_cookie($key,$value),requests::set\_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)

{

requests::set\_header("Referer", "http://buluo.qq.com/p/index.html");

};

**on\_status\_code($status\_code, $url, $content, $phpspider)**

**判断当前网页是否被反爬虫了, 需要开发者实现**

@param $status\_code 当前网页的请求返回的HTTP状态码  
@param $url 当前网页URL  
@param $content 当前网页内容  
@param $phpspider 爬虫对象  
@return $content 返回处理后的网页内容，不处理当前页面请返回false

举个栗子:

$spider->on\_status\_code = function($status\_code, $url, $content, $phpspider)

{

// 如果状态码为429，说明对方网站设置了不让同一个客户端同时请求太多次

if ($status\_code == '429')

{

// 将url插入待爬的队列中,等待再次爬取

$phpspider->add\_url($url);

// 当前页先不处理了

return false;

}

// 不拦截的状态码这里记得要返回，否则后面内容就都空了

return $content;

};

**is\_anti\_spider($url, $content, $phpspider)**

**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  
@param $phpspider 爬虫对象  
@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, $phpspider)

{

// $content中包含"404页面不存在"字符串

if (strpos($content, "404页面不存在") !== false)

{

// 如果使用了代理IP，IP切换需要时间，这里可以添加到队列等下次换了IP再抓取

// $phpspider->add\_url($url);

return true; // 告诉框架网页被反爬虫了，不要继续处理它

}

// 当前页面没有被反爬虫，可以继续处理

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 返回处理后的网页内容

@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\_download\_attached\_page($content, $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 $content 当前下载的网页内容  
@param $phpspider 爬虫对象  
@return 返回处理后的网页内容

举个栗子:  
比如下载的网页需去掉前后两边的中括号，并将处理后的数据返回，处理过程如下：

$spider->on\_download\_attached\_page = function($content, $phpspider)

{

$content = trim($content);

$content = ltrim($content, "[");

$content = rtrim($content, "]");

$content = json\_decode($content, true);

return $content;

};

**on\_fetch\_url($url, $phpspider)**

**在一个网页获取到URL之后调用. 主要用来对获取到的URL进行处理.**

@param $url 当前获取到的URL  
@param $phpspider 爬虫对象  
@return 返回处理后的URL，为false则此URL不入采集队列

栗子1:  
如果获取到URL包含#filter，不入URL采集队列

$spider->on\_fetch\_url = function($url, $phpspider)

{

if (strpos($url, "#filter") !== false)

{

return false;

}

return $url;

};

栗子2:  
把获取到URL的&替换成&，并将处理后的数据返回

$spider->on\_fetch\_url = function($url, $phpspider)

{

$url = str\_replace("&amp;", "&", $url);

return $url;

};

**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 当前下载的网页页面的对象  
@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 当前下载的网页页面的对象  
@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 当前下载的网页页面的对象  
@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**

举个栗子:  
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 $page 当前下载的网页页面的对象  
@return 返回处理后的数据, 注意数据类型需要跟传进来的$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:

<span class="item gender" ><i class="icon icon-profile-male"></i></span>  
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 当前下载的网页页面的对象  
@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;

// 返回false不处理，当前页面的字段不入数据库直接过滤

// 比如采集电影网站，标题匹配到“预告片”这三个字就过滤

//if (strpos($data['title'], "预告片") !== false)

//{

// return false;

//}

};