大话网络爬虫

小菜-邹嵩

2018年6月

网络爬虫概述

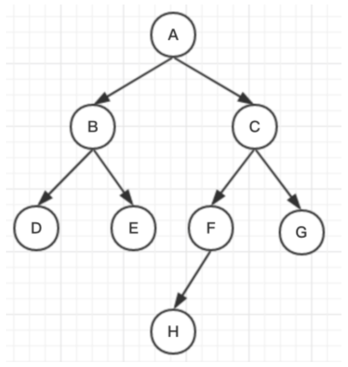
爬虫的分类：通用爬虫、定题爬虫

通用爬虫从一个或若干初始网页的链接开始，获得初始网页上的链接，在抓取网页的过程中不断从当前页面上抽取新的链接放入队列，直到满足系统的一定停止条件。

定题爬虫则需要根据一定的网页分析算法过滤与主题无关的链接保留有用的链接并将其放入队列。另外，所有被爬虫抓取的网页将会进行一定的分析、过滤，并存储起来。

采集的策略：IP地址采集策略、深度优先策略、广度优先策略

IP地址采集策略一般是通用爬虫会使用，按IP地址递增的方式采集。比如通过Ping命令可以知道百度的一个IP地址是115.239.210.27，则可以在浏览器中通过http://115.239.210.27来访问百度。



深度优先策略是先到达叶子节点，如左图则访问顺序是: A->B->D->E->C->F->H->G

广度优先策略则是先到达子节点，如左图则访问顺序是: A->B->C->D->E->F->G->H

网络爬虫的基本模块

爬虫的基本模块：采集调度、下载器、数据解析、数据存储

采集调度包含采集策略的实现，链接去重功能。调度器也可以分为单机调度、分布式调度，单机调度一般直接把链接数据存在内存中，分布式调度可以通过传统数据库、Redis或其它方式实现。

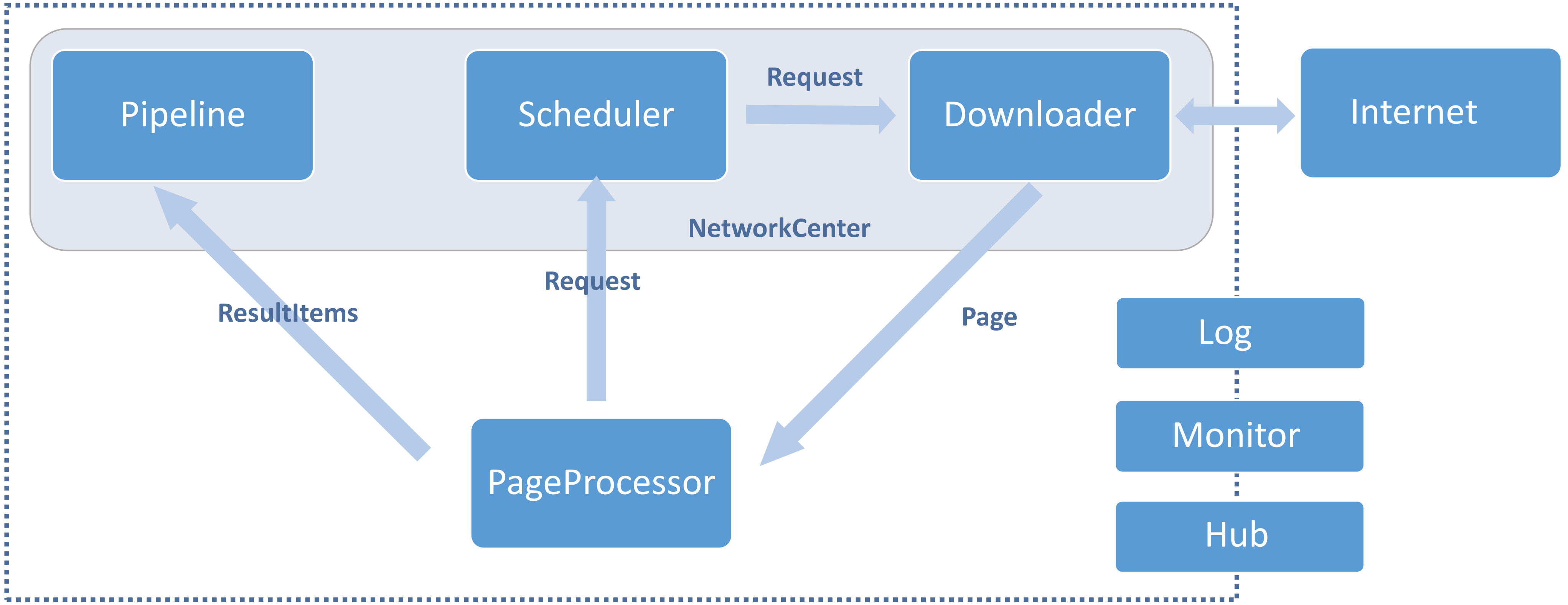
内容下载可以通过Http请求、无头浏览器、Selelinum操作真实浏览器来实现。

数据解析是指对下载好的内容做抽取，常用手段有Xpath、Css选择器、正则表达式。

数据存储的方式因人而宜，因场景而宜。可以存文件、数据库、ES。数据量小的常用的关系型数据库就能满足，在我的实践中mysql即便单表10E只要使用恰当，也还是能用来分析的(OLAP, OLTP的区别)，对我司来说，有些计算需要10几分钟到1小时并不是什么问题。

数据解析模块不一定必要的，比如通用爬虫的应用之一搜索引擎则一般不做特定解析，直接入到全文检索服务中。

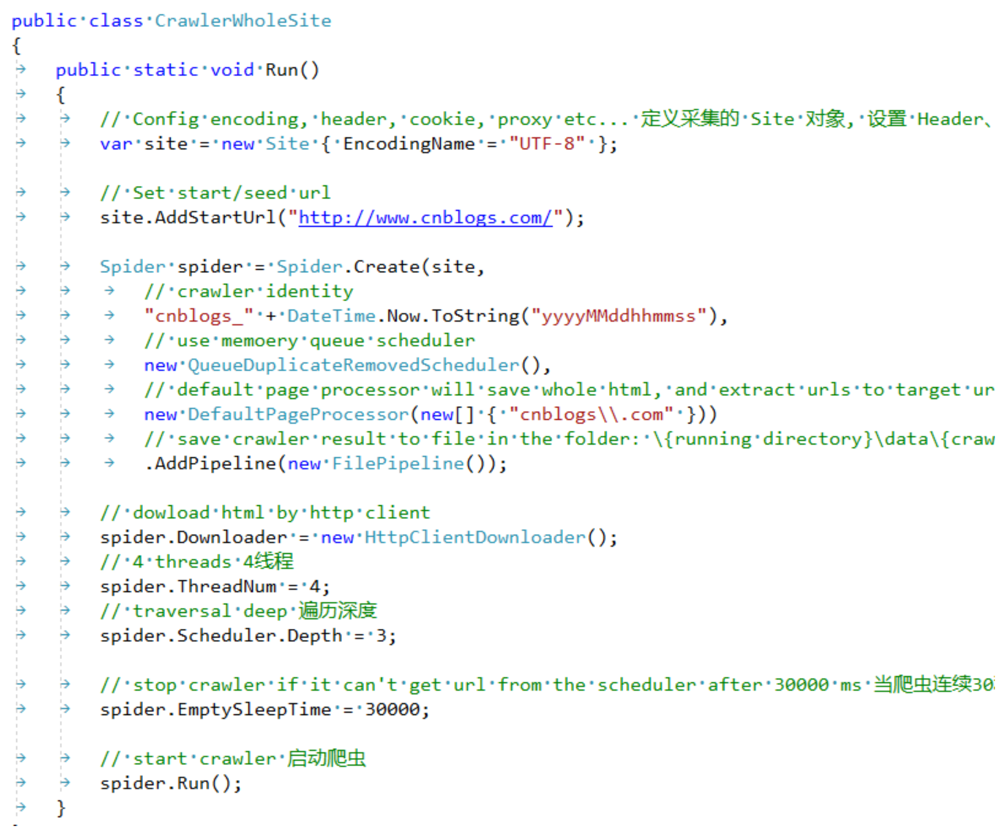
DotnetSpider的设计



- 网络中心: 全局通信、拨号
- Hub: 任务调度，管理

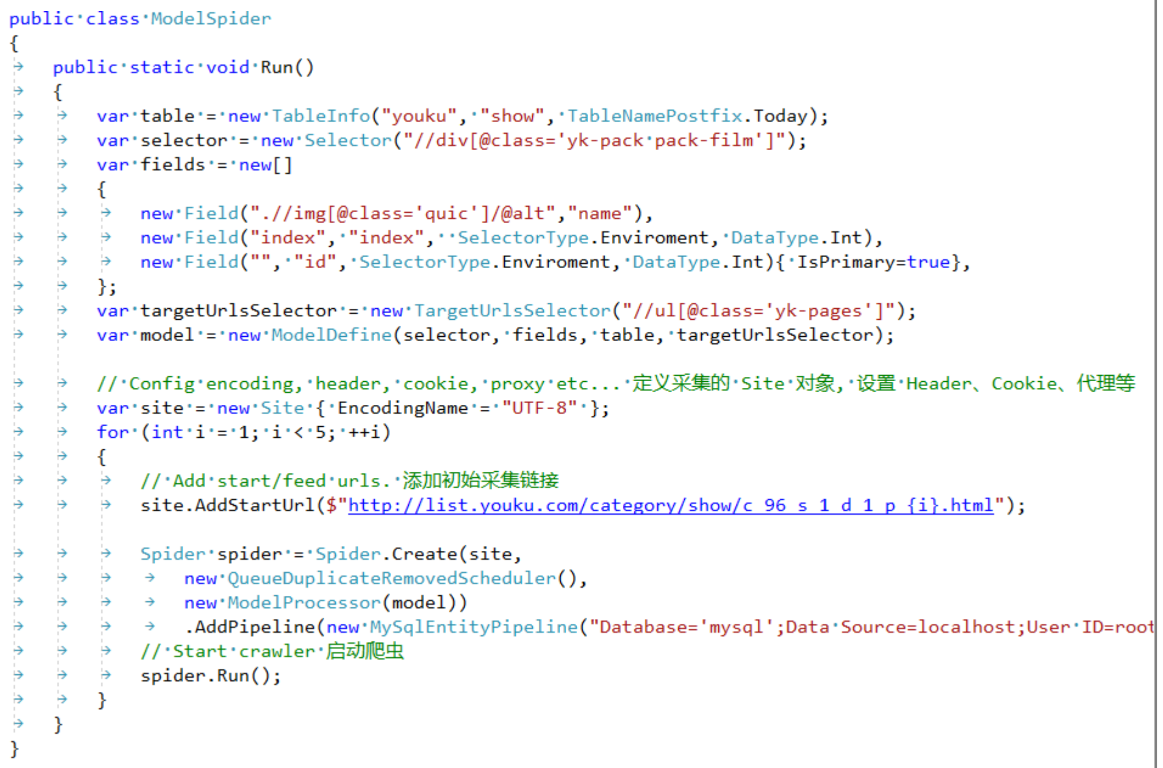
DotnetSpider的使用

通用爬虫模式



```
public class CrawlerWholeSite
{
    public static void Run()
    {
        // Config encoding, header, cookie, proxy etc... 定义采集的 Site 对象, 设置 Header、
        var site = new Site { EncodingName = "UTF-8" };

        // Set start/seed url
        site.AddStartUrl("http://www.cnblogs.com/");

        Spider spider = Spider.Create(site,
            // crawler identity
            "cnblogs_" + DateTime.Now.ToString("yyyyMMddhhmmss"),
            // use memoery queue scheduler
            new QueueDuplicateRemovedScheduler(),
            // default page processor will save whole html, and extract urls to target ur
            new DefaultPageProcessor(new[] { "cnblogs\\.com" }))
            // save crawler result to file in the folder: \{running directory}\data\{craw
            .AddPipeline(new FilePipeline());

        // dowload html by http client
        spider.Downloader = new HttpClientDownloader();
        // 4 threads 4线程
        spider.ThreadNum = 4;
        // traversal deep 遍历深度
        spider.Scheduler.Depth = 3;

        // stop crawler if it can't get url from the scheduler after 30000 ms 当爬虫连续30
        spider.EmptySleepTime = 30000;

        // start crawler 启动爬虫
        spider.Run();
    }
```

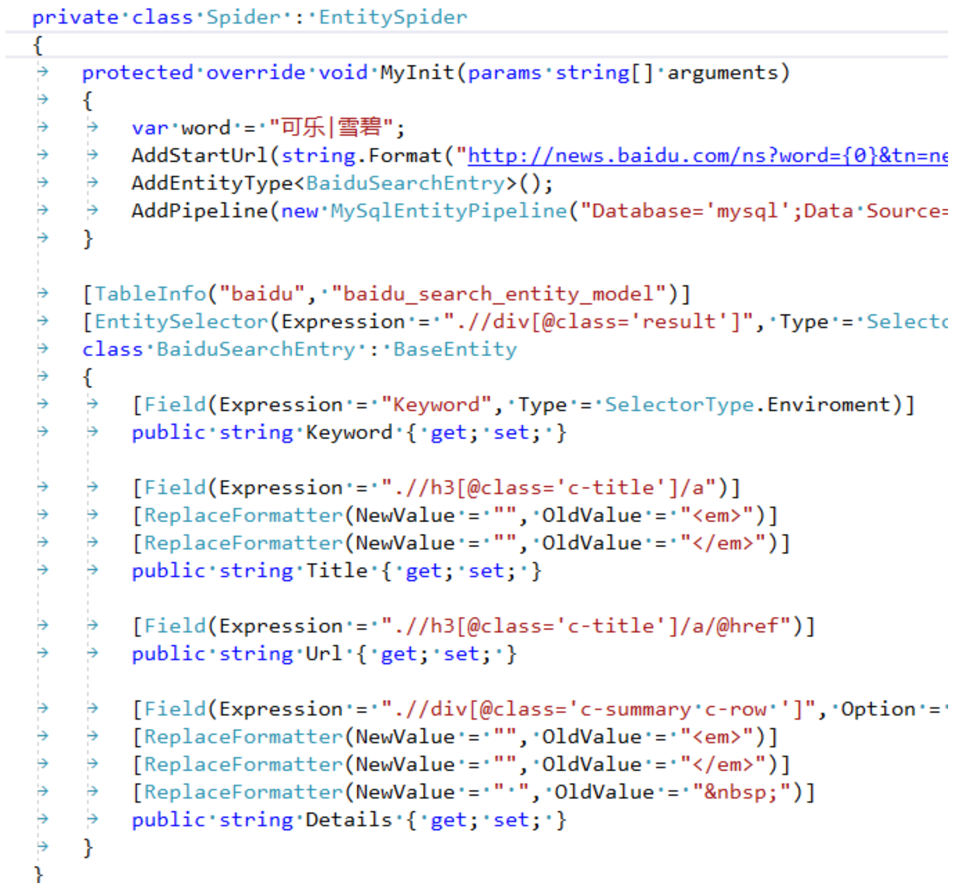
Model爬虫模式



```
public class ModelSpider
{
    public static void Run()
    {
        var table = new TableInfo("youku", "show", TableNamePostfix.Today);
        var selector = new Selector("//div[@class='yk-pack pack-film']");
        var fields = new[]
        {
            new Field(".//img[@class='quic']/@alt","name"),
            new Field("index", "index",  SelectorType.Enviroment, DataType.Int),
            new Field("", "id", SelectorType.Enviroment, DataType.Int){ IsPrimary=true},
        };
        var targetUrlsSelector = new TargetUrlsSelector("//ul[@class='yk-pages']");
        var model = new ModelDefine(selector, fields, table, targetUrlsSelector);

        // Config encoding, header, cookie, proxy etc... 定义采集的 Site 对象, 设置 Header、Cookie、代理等
        var site = new Site { EncodingName = "UTF-8" };
        for (int i = 1; i < 5; ++i)
        {
            // Add start/feed urls. 添加初始采集链接
            site.AddStartUrl($"http://list.youku.com/category/show/c_96_s_1_d_1_p_{i}.html");

            Spider spider = Spider.Create(site,
                new QueueDuplicateRemovedScheduler(),
                new ModelProcessor(model))
                .AddPipeline(new MySqlEntityPipeline("Database='mysql';Data Source=localhost;User ID=root
            // Start crawler 启动爬虫
            spider.Run();
        }
    }
}
```

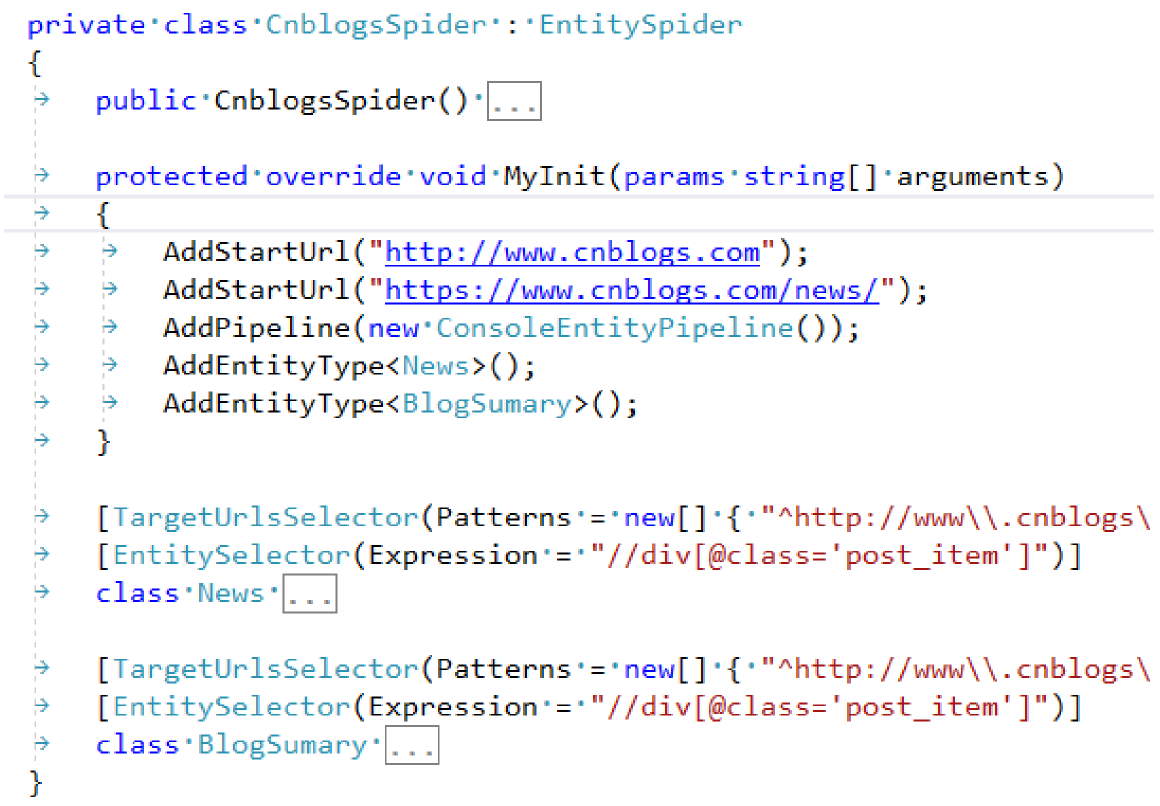
DotnetSpider的使用

Entity爬虫模式



```
private class Spider : EntitySpider
{
    protected override void MyInit(params string[] arguments)
    {
        var word = "可乐|雪碧";
        AddStartUrl(string.Format("http://news.baidu.com/ns?word={0}&tn=ne
        AddEntityType<BaiduSearchEntry>();
        AddPipeline(new MySqlEntityPipeline("Database='mysql';Data Source=
    }

    [TableInfo("baidu", "baidu_search_entity_model")]
    [EntitySelector(Expression = ".//div[@class='result']", Type = Select
    class BaiduSearchEntry : BaseEntity
    {
        [Field(Expression = "Keyword", Type = SelectorType.Enviroment)]
        public string Keyword { get; set; }

        [Field(Expression = ".//h3[@class='c-title']/a")]
        [ReplaceFormatter(NewValue = "", OldValue = "<em>")]
        [ReplaceFormatter(NewValue = "", OldValue = "</em>")]
        public string Title { get; set; }

        [Field(Expression = ".//h3[@class='c-title']/a/@href")]
        public string Url { get; set; }

        [Field(Expression = ".//div[@class='c-summary c-row ']", Option =
        [ReplaceFormatter(NewValue = "", OldValue = "<em>")]
        [ReplaceFormatter(NewValue = "", OldValue = "</em>")]
        [ReplaceFormatter(NewValue = " ", OldValue = "&nbsp;")]
        public string Details { get; set; }
    }
}
```

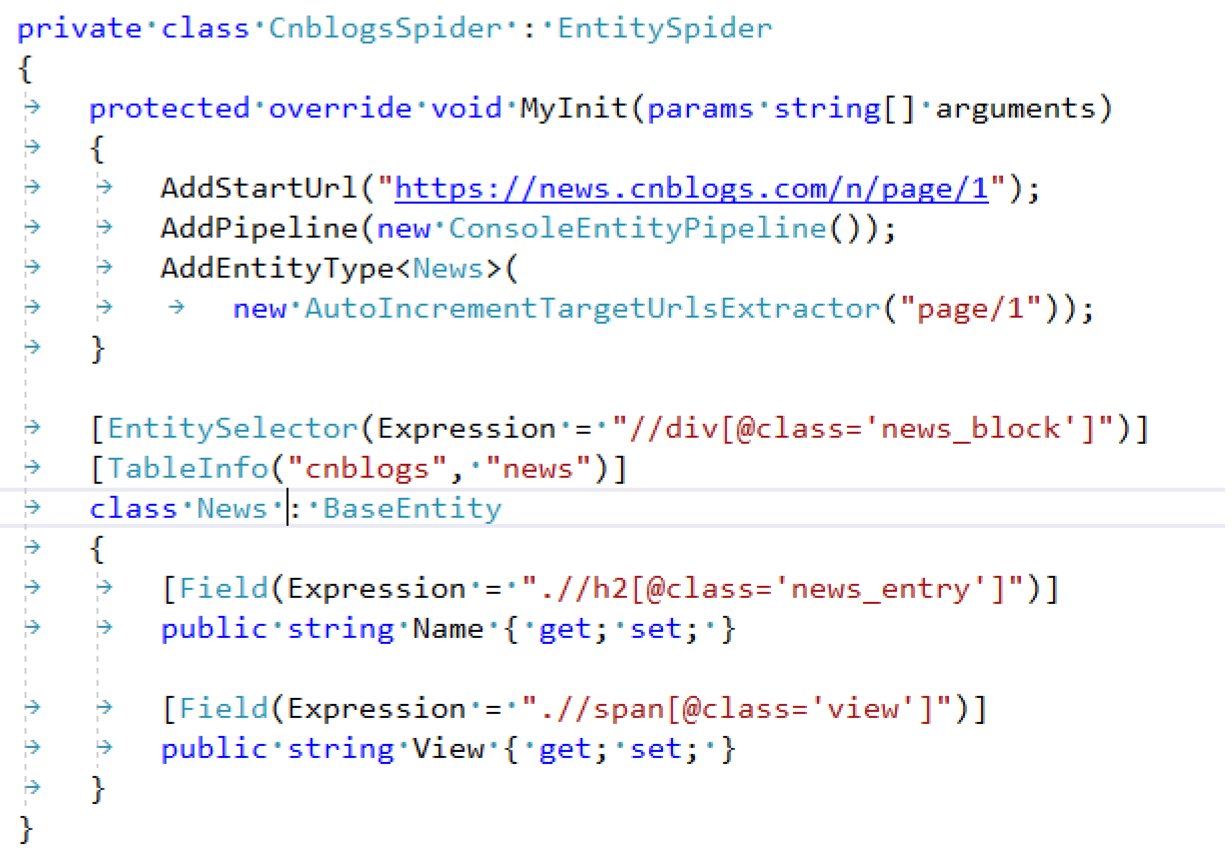
Multiy Entity



```
private class CnblogsSpider : EntitySpider
{
    public CnblogsSpider() ...

    protected override void MyInit(params string[] arguments)
    {
        AddStartUrl("http://www.cnblogs.com");
        AddStartUrl("https://www.cnblogs.com/news/");
        AddPipeline(new ConsoleEntityPipeline());
        AddEntityType<News>();
        AddEntityType<BlogSumary>();
    }

    [TargetUrlsSelector(Patterns = new[] { "^http://www\\.cnblogs\
    [EntitySelector(Expression = "//div[@class='post_item']")]
    class News ...

    [TargetUrlsSelector(Patterns = new[] { "^http://www\\.cnblogs\
    [EntitySelector(Expression = "//div[@class='post_item']")]
    class BlogSumary ...
}
```

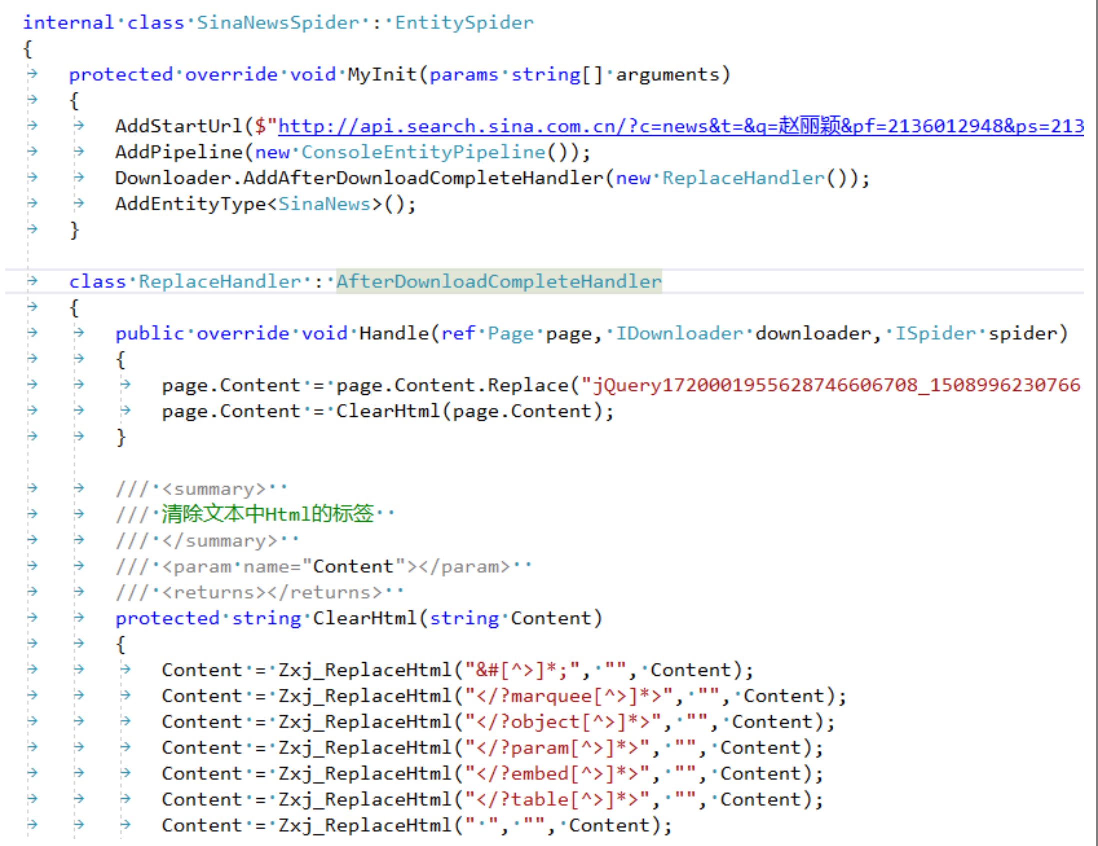
DotnetSpider的使用

TargetUrlsExtractor



```csharp
private class CnblogsSpider : EntitySpider
{
    protected override void MyInit(params string[] arguments)
    {
        AddStartUrl("https://news.cnblogs.com/n/page/1");
        AddPipeline(new ConsoleEntityPipeline());
        AddEntityType<News>(
            new AutoIncrementTargetUrlsExtractor("page/1"));
    }

    [EntitySelector(Expression = "//div[@class='news_block']")]
    [TableInfo("cnblogs", "news")]
    class News : BaseEntity
    {
        [Field(Expression = ".//h2[@class='news_entry']")]
        public string Name { get; set; }

        [Field(Expression = ".//span[@class='view']")]
        public string View { get; set; }
    }
}
```

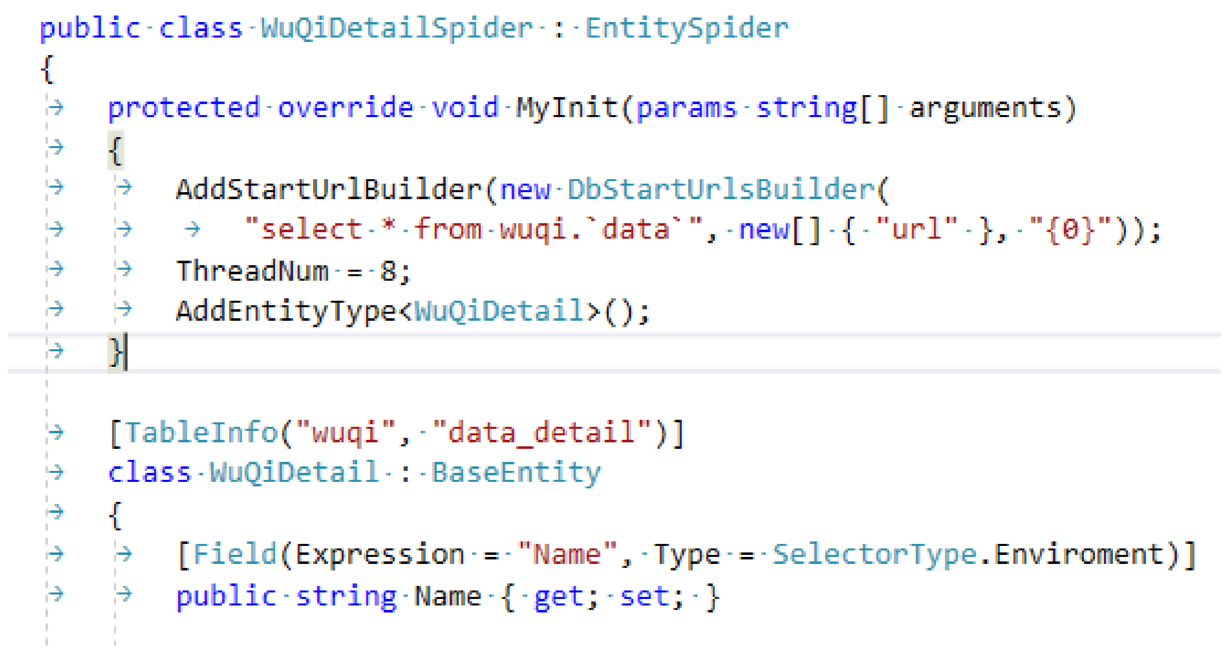
DownloadHandler



```csharp
internal class SinaNewsSpider : EntitySpider
{
    protected override void MyInit(params string[] arguments)
    {
        AddStartUrl($"http://api.search.sina.com.cn/?c=news&t=&q=赵丽颖&pf=2136012948&ps=213
        AddPipeline(new ConsoleEntityPipeline());
        Downloader.AddAfterDownloadCompleteHandler(new ReplaceHandler());
        AddEntityType<SinaNews>();
    }

    class ReplaceHandler : AfterDownloadCompleteHandler
    {
        public override void Handle(ref Page page, IDownloader downloader, ISpider spider)
        {
            page.Content = page.Content.Replace("jQuery17200019556287466067O8_1508996230766
            page.Content = ClearHtml(page.Content);
        }

        /// <summary>
        /// 清除文本中Html的标签
        /// </summary>
        /// <param name="Content"></param>
        /// <returns></returns>
        protected string ClearHtml(string Content)
        {
            Content = Zxj_ReplaceHtml("&#[^>]*;", "", Content);
            Content = Zxj_ReplaceHtml("</?marquee[^>]*>", "", Content);
            Content = Zxj_ReplaceHtml("</?object[^>]*>", "", Content);
            Content = Zxj_ReplaceHtml("</?param[^>]*>", "", Content);
            Content = Zxj_ReplaceHtml("</?embed[^>]*>", "", Content);
            Content = Zxj_ReplaceHtml("</?table[^>]*>", "", Content);
            Content = Zxj_ReplaceHtml(" ", "", Content);
```

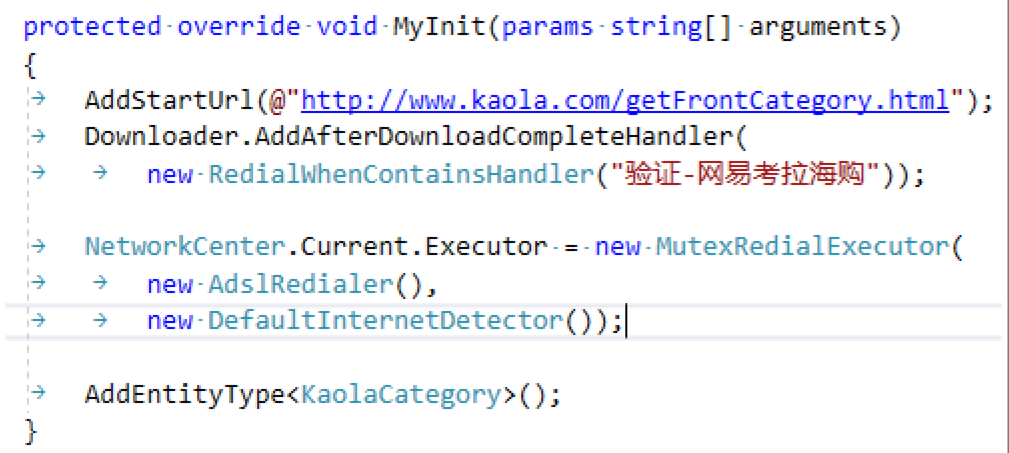
DotnetSpider的使用

StartUrlBuilder



```
public class WuQiDetailSpider : EntitySpider
{
	protected override void MyInit(params string[] arguments)
	{
		AddStartUrlBuilder(new DbStartUrlsBuilder(
			"select * from wuqi.`data`", new[] { "url" }, "{0}"));
		ThreadNum = 8;
		AddEntityType<WuQiDetail>();
	}

	[TableInfo("wuqi", "data_detail")]
	class WuQiDetail : BaseEntity
	{
		[Field(Expression = "Name", Type = SelectorType.Enviroment)]
		public string Name { get; set; }
```

Redial



```
protected override void MyInit(params string[] arguments)
{
	AddStartUrl(@"http://www.kaola.com/getFrontCategory.html");
	Downloader.AddAfterDownloadCompleteHandler(
		new RedialWhenContainsHandler("验证-网易考拉海购"));

	NetworkCenter.Current.Executor = new MutexRedialExecutor(
		new AdslRedialer(),
		new DefaultInternetDetector());

	AddEntityType<KaolaCategory>();
}
```

DotnetSpider的使用

- **WebDriverDownloader**

  通过WebDriver可以操作Phantomjs, Chrome, Firefox, IE

- **CookieInjector**

  在每一个Downloader执行第一次下载前, 注入Cookie到HttpClient的CookieContainer中

- **HttpClient Group**

  某些网站的请求访问是链式的, 每个token之间的计算是相关的，需要保证使用相同的HttpClient访问

- **EntityPipeline**

  支持MySql, MySqlFile, SqlServer, MongoDb, Cassandra…

反爬技术介绍

IP 限制->帐号限制

代理、ADSL拨号、注册大量帐号

Html->Ajax->Sign

纯Html输出时, 采集没有任何难度。当使用ajax异步加载数据时, 纯Http无法得到最终Html,
可以直接使用Fiddler之类抓包工具得到Http请求，使用代码直接构造正确的请求取得数据。
如果请求加sign了, 有能力的逆向js取得加密算法，没能力的直接使用WebDriver。

图片化数据

- 纯图片化：OCR
- 图片当backgroud偏移: 图片比对

CSS Before

- 正则匹配

测试环境检测

检测是否在使用Selinium访问, 直接禁止。使用自定义Chrominum或者其它浏览器。

反爬技术介绍

验证码，手机验证码

OCR，打码平台，短信接手平台

喂毒

多次采集，数据验证，社工学找到反爬机制

链式请求

单线程保持链式请求

检测代理

- 使用无线网卡共享热点给手机, 通过Wireshark抓包
- 在手机上使用Packet Capture抓包

反爬技术介绍

- **App访问Api加Sign**

  逆向smali代码，找到计算Sign的算法。

- **通过so文件计算Sign**

  逆象so，找app漏洞，系统层级分析

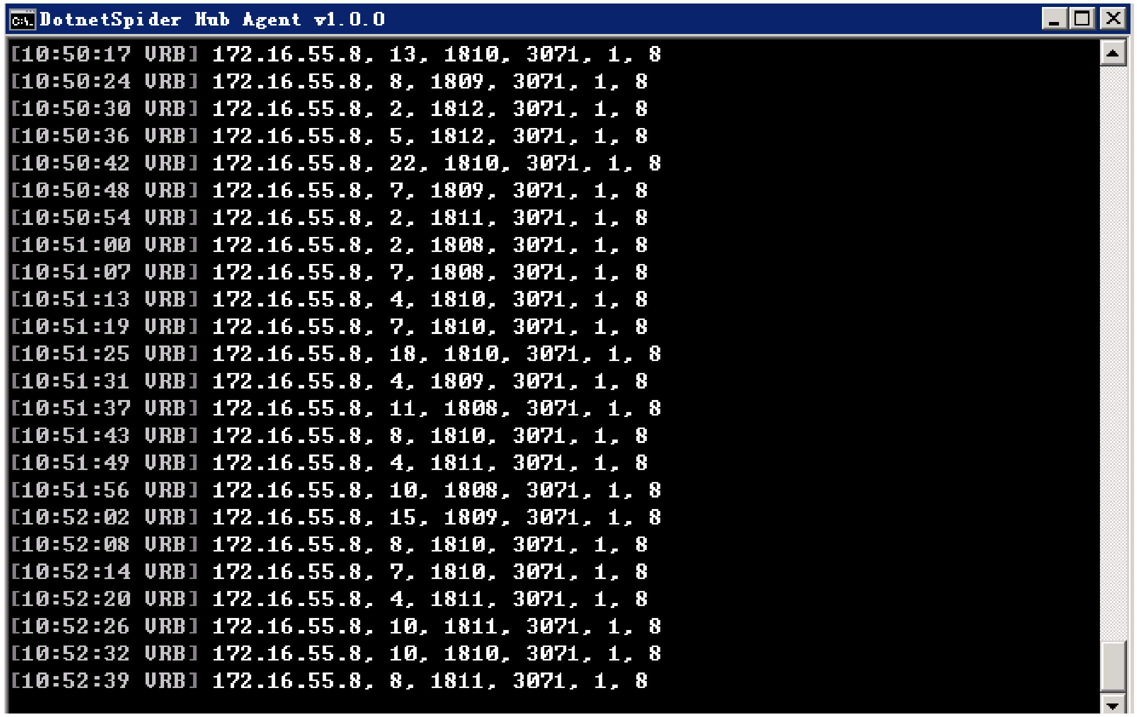
- **加壳**

  脱壳

- **其它很多很多**

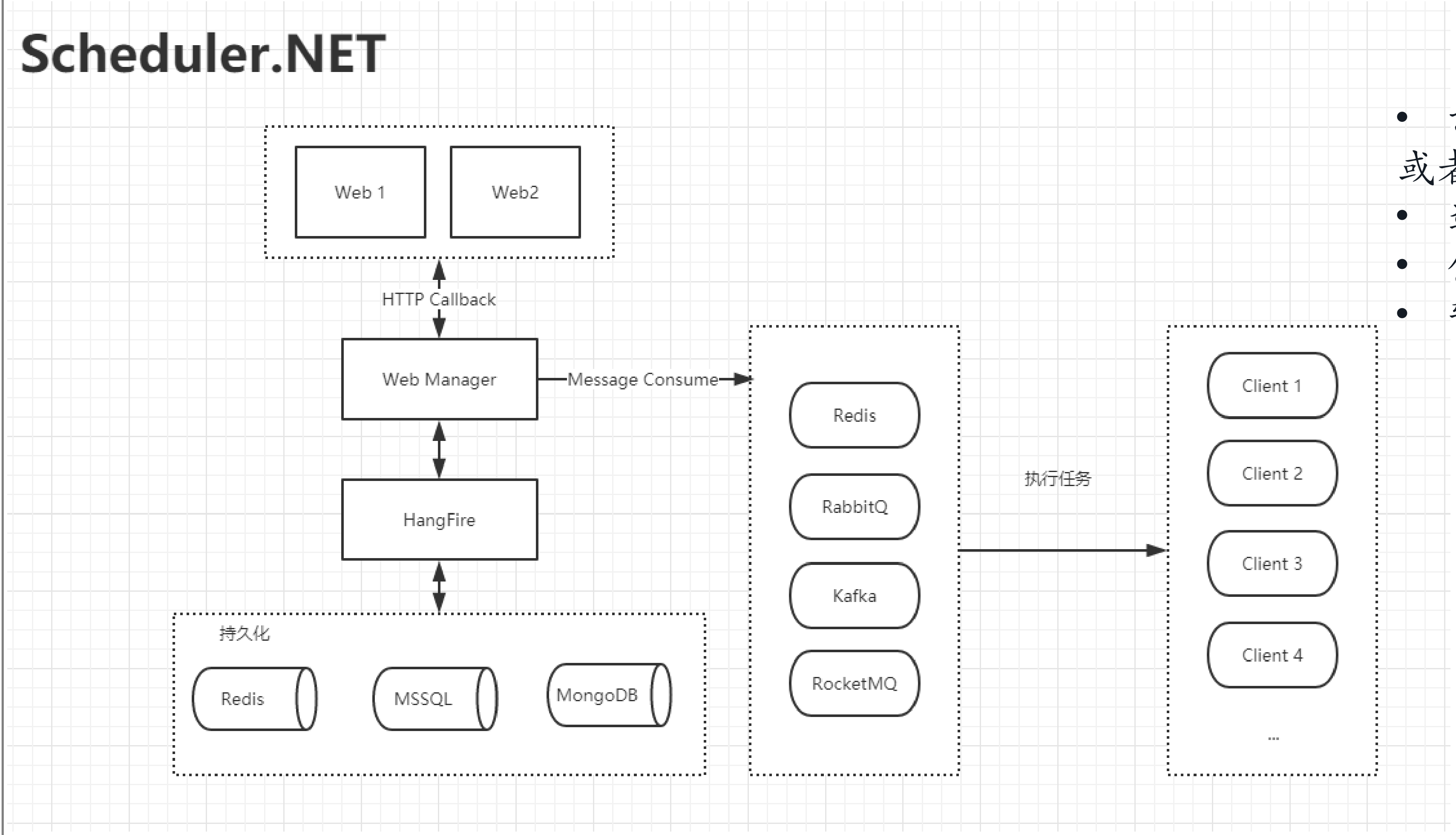
Agent

- 心跳上报采集节点状态: CPU, 内存, 任务数等
- 通过Http轮循获取控制消息：启动任务、退出任务、节点



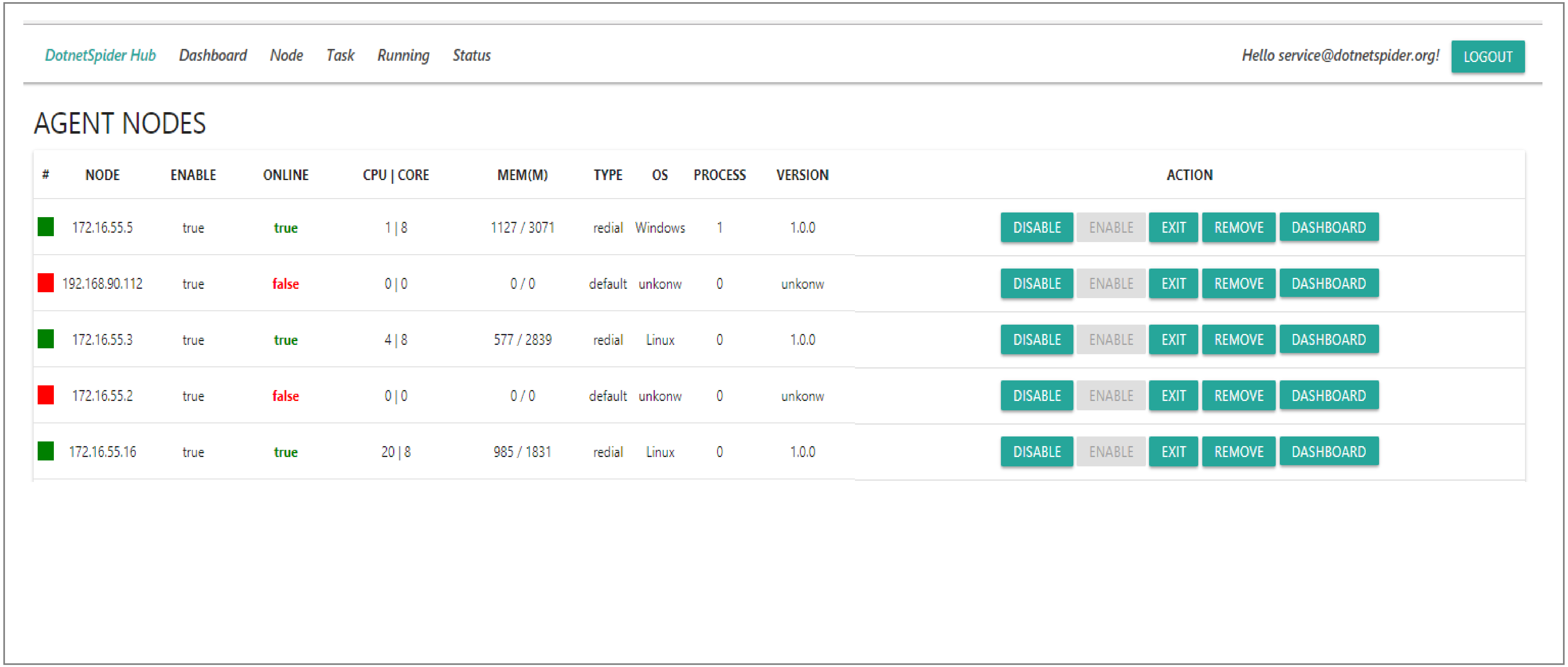
```
[10:50:17 VRB] 172.16.55.8, 13, 1810, 3071, 1, 8
[10:50:24 VRB] 172.16.55.8, 8, 1809, 3071, 1, 8
[10:50:30 VRB] 172.16.55.8, 2, 1812, 3071, 1, 8
[10:50:36 VRB] 172.16.55.8, 5, 1812, 3071, 1, 8
[10:50:42 VRB] 172.16.55.8, 22, 1810, 3071, 1, 8
[10:50:48 VRB] 172.16.55.8, 7, 1809, 3071, 1, 8
[10:50:54 VRB] 172.16.55.8, 2, 1811, 3071, 1, 8
[10:51:00 VRB] 172.16.55.8, 2, 1808, 3071, 1, 8
[10:51:07 VRB] 172.16.55.8, 7, 1808, 3071, 1, 8
[10:51:13 VRB] 172.16.55.8, 4, 1810, 3071, 1, 8
[10:51:19 VRB] 172.16.55.8, 7, 1810, 3071, 1, 8
[10:51:25 VRB] 172.16.55.8, 18, 1810, 3071, 1, 8
[10:51:31 VRB] 172.16.55.8, 4, 1809, 3071, 1, 8
[10:51:37 VRB] 172.16.55.8, 11, 1808, 3071, 1, 8
[10:51:43 VRB] 172.16.55.8, 8, 1810, 3071, 1, 8
[10:51:49 VRB] 172.16.55.8, 4, 1811, 3071, 1, 8
[10:51:56 VRB] 172.16.55.8, 10, 1808, 3071, 1, 8
[10:52:02 VRB] 172.16.55.8, 15, 1809, 3071, 1, 8
[10:52:08 VRB] 172.16.55.8, 8, 1810, 3071, 1, 8
[10:52:14 VRB] 172.16.55.8, 7, 1810, 3071, 1, 8
[10:52:20 VRB] 172.16.55.8, 4, 1811, 3071, 1, 8
[10:52:26 VRB] 172.16.55.8, 10, 1811, 3071, 1, 8
[10:52:32 VRB] 172.16.55.8, 10, 1810, 3071, 1, 8
[10:52:39 VRB] 172.16.55.8, 8, 1811, 3071, 1, 8
```

Scheduler.NET



- 调度器接口化，可自由使用QuartZ或者Hangfire
- 通过WebApi添加、删除任务
- 任务类型支持Http回调、消息队列回调
- 暂未提供UI

节点监控

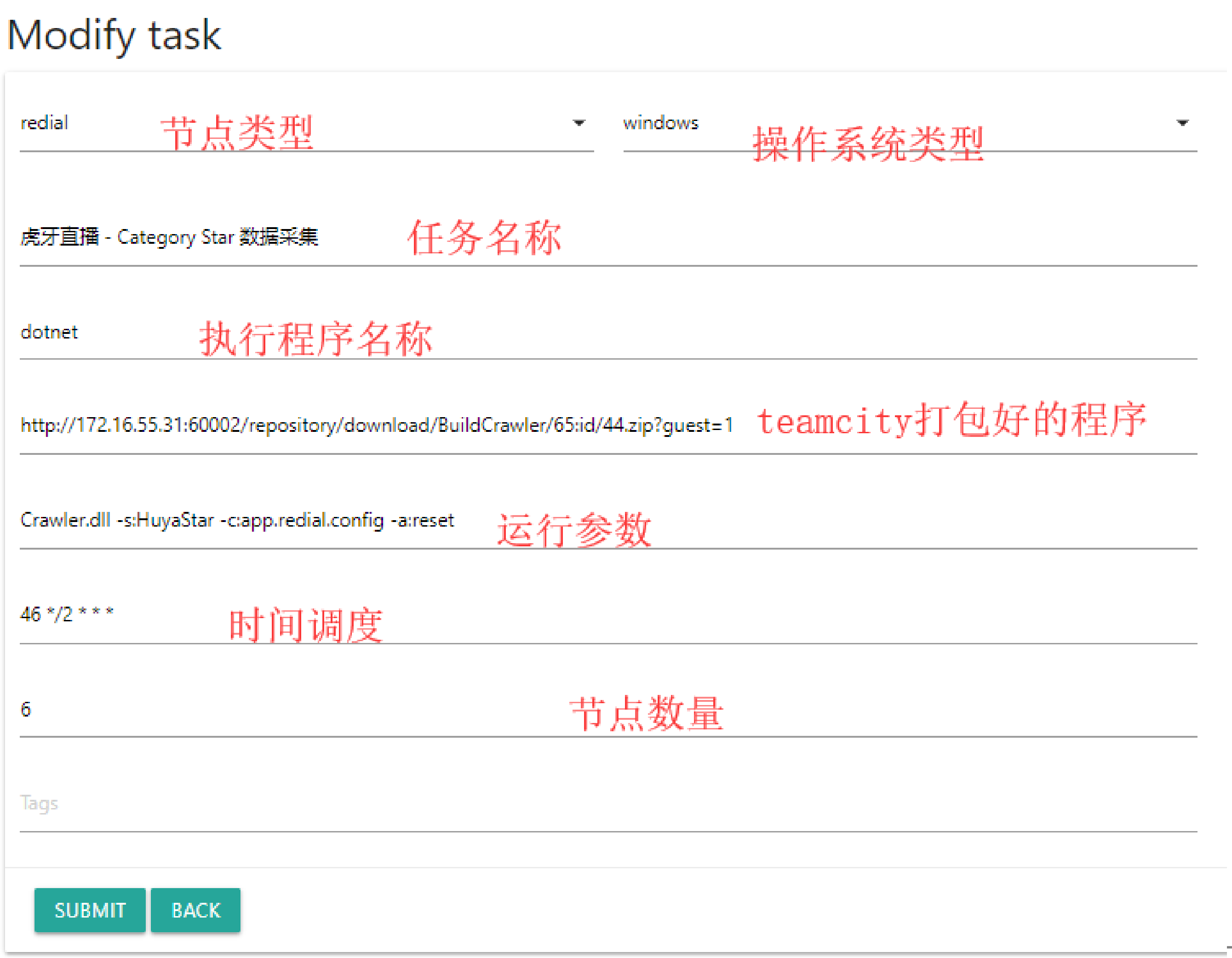


DotnetSpider Hub Dashboard Node Task Running Status Hello service@dotnetspider.org! LOGOUT

AGENT NODES

#	NODE	ENABLE	ONLINE	CPU | CORE	MEM(M)	TYPE	OS	PROCESS	VERSION	ACTION
🟩	172.16.55.5	true	true	1 | 8	1127 / 3071	redial	Windows	1	1.0.0	DISABLE ENABLE EXIT REMOVE DASHBOARD
🟥	192.168.90.112	true	false	0 | 0	0 / 0	default	unkonw	0	unkonw	DISABLE ENABLE EXIT REMOVE DASHBOARD
🟩	172.16.55.3	true	true	4 | 8	577 / 2839	redial	Linux	0	1.0.0	DISABLE ENABLE EXIT REMOVE DASHBOARD
🟥	172.16.55.2	true	false	0 | 0	0 / 0	default	unkonw	0	unkonw	DISABLE ENABLE EXIT REMOVE DASHBOARD
🟩	172.16.55.16	true	true	20 | 8	985 / 1831	redial	Linux	0	1.0.0	DISABLE ENABLE EXIT REMOVE DASHBOARD

任务创建和编辑



Modify task

redial 节点类型

windows 操作系统类型

虎牙直播 - Category Star 数据采集 任务名称

dotnet 执行程序名称

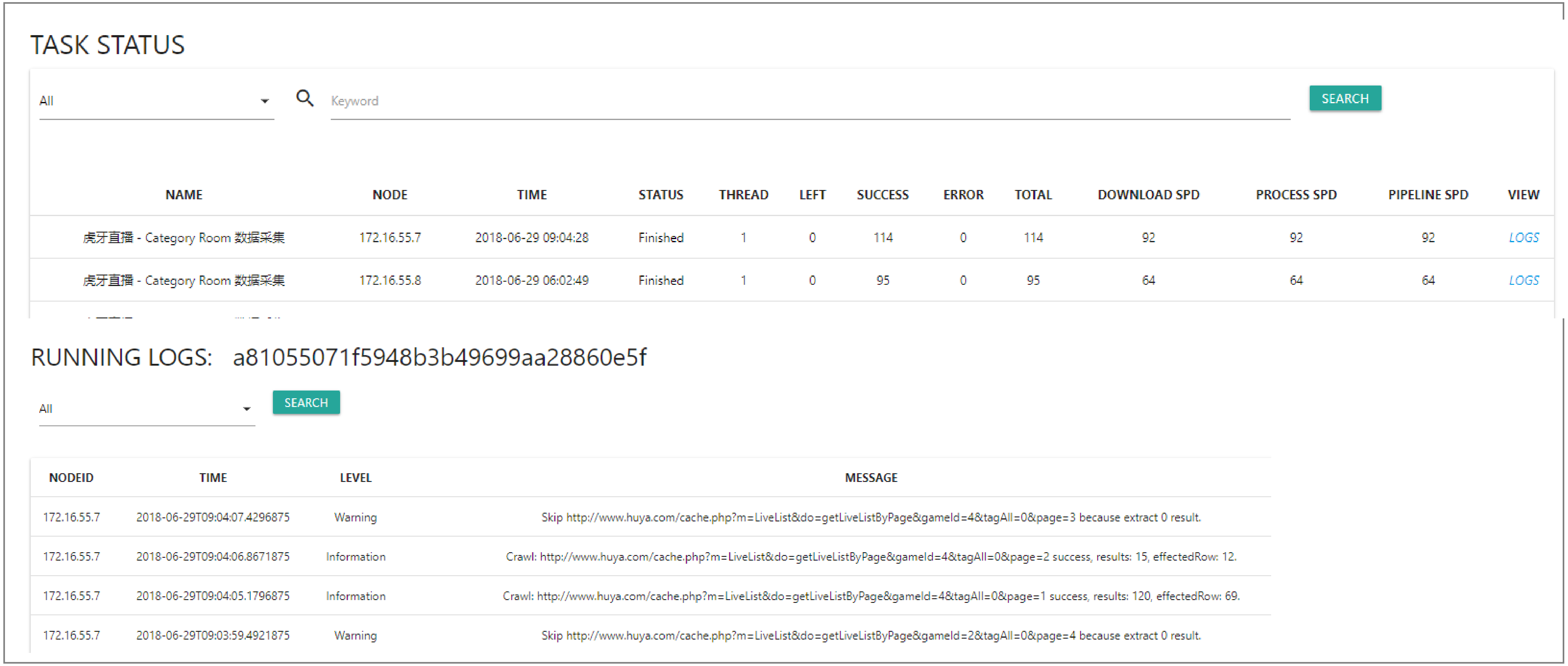
http://172.16.55.31:60002/repository/download/BuildCrawler/65:id/44.zip?guest=1 teamcity打包好的程序

Crawler.dll -s:HuyaStar -c:app.redial.config -a:reset 运行参数

46 */2 * * * 时间调度

6 节点数量

任务状态和日志



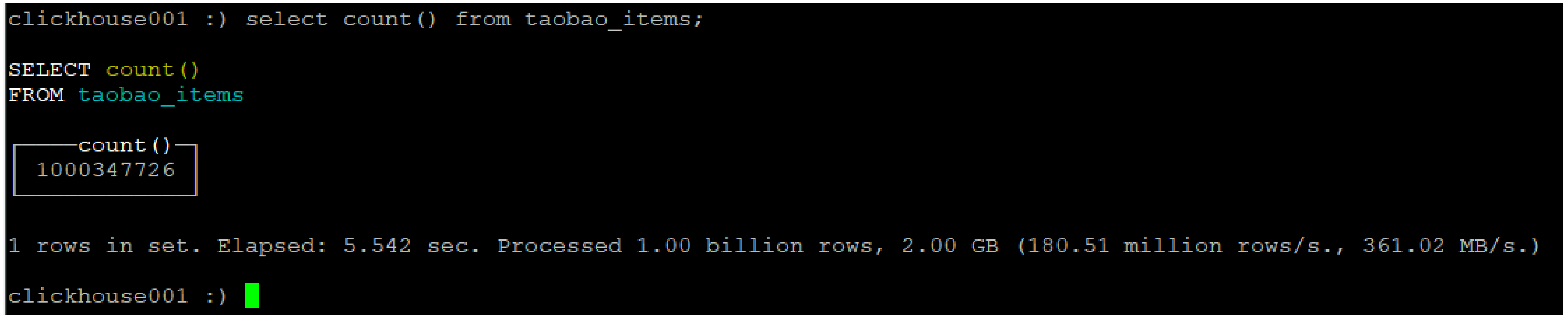
NAME	NODE	TIME	STATUS	THREAD	LEFT	SUCCESS	ERROR	TOTAL	DOWNLOAD SPD	PROCESS SPD	PIPELINE SPD	VIEW
虎牙直播 - Category Room 数据采集	172.16.55.7	2018-06-29 09:04:28	Finished	1	0	114	0	114	92	92	92	LOGS
虎牙直播 - Category Room 数据采集	172.16.55.8	2018-06-29 06:02:49	Finished	1	0	95	0	95	64	64	64	LOGS

RUNNING LOGS: a81055071f5948b3b49699aa28860e5f

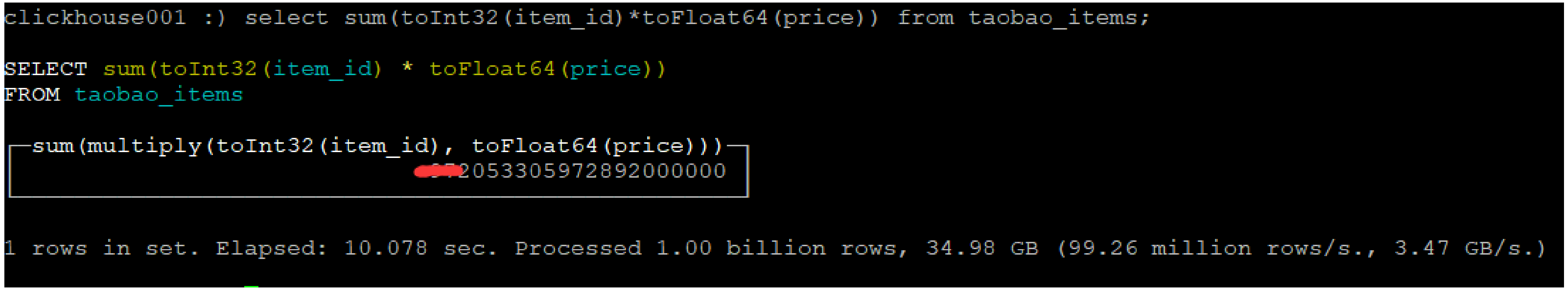
NODEID	TIME	LEVEL	MESSAGE
172.16.55.7	2018-06-29T09:04:07.4296875	Warning	Skip http://www.huya.com/cache.php?m=LiveList&do=getLiveListByPage&gameId=4&tagAll=0&page=3 because extract 0 result.
172.16.55.7	2018-06-29T09:04:06.8671875	Information	Crawl: http://www.huya.com/cache.php?m=LiveList&do=getLiveListByPage&gameId=4&tagAll=0&page=2 success, results: 15, effectedRow: 12.
172.16.55.7	2018-06-29T09:04:05.1796875	Information	Crawl: http://www.huya.com/cache.php?m=LiveList&do=getLiveListByPage&gameId=4&tagAll=0&page=1 success, results: 120, effectedRow: 69.
172.16.55.7	2018-06-29T09:03:59.4921875	Warning	Skip http://www.huya.com/cache.php?m=LiveList&do=getLiveListByPage&gameId=2&tagAll=0&page=4 because extract 0 result.

OLTP数据库选择: Clickhouse

HDD 4T*12 Raid 10, 32Core, 128G



```
clickhouse001 :) select count() from taobao_items;

SELECT count()
FROM taobao_items

┌──count()─┐
│ 1000347726 │
└──────────┘

1 rows in set. Elapsed: 5.542 sec. Processed 1.00 billion rows, 2.00 GB (180.51 million rows/s., 361.02 MB/s.)

clickhouse001 :)
```



```
clickhouse001 :) select sum(toInt32(item_id)*toFloat64(price)) from taobao_items;

SELECT sum(toInt32(item_id) * toFloat64(price))
FROM taobao_items

┌─sum(multiply(toInt32(item_id), toFloat64(price)))─┐
│                        [redacted]2053305972892000000 │
└───────────────────────────────────────────────────┘

1 rows in set. Elapsed: 10.078 sec. Processed 1.00 billion rows, 34.98 GB (99.26 million rows/s., 3.47 GB/s.)
```