链滴

Golang 爬虫快速入门 | 获取 B 站全站的视频数据

作者：zhshch

原文链接：https://ld246.com/article/1586935963072

来源网站：链滴

原文首发并持续更新于 https://imagician.net/archives/92/，欲了解更多信息可以前往我的博客https://imagician.net/

提到爬虫，总会联想到Python。似乎Python是爬虫的唯一选择。爬虫只是完成一个访问页面然后收数据的任务，用任何语言来写都能实现。相比较Python快速实现但是庞大的体型，Golang来写爬虫乎是更好的又一选择。

HTTP请求

Golang语言的HTTP请求库不需要使用第三方的库，标准库就内置了足够好的支持：

```
package main

import (
    "fmt"
    "net/http"
    "io/ioutil"
)

func fetch (url string) string {
    fmt.Println("Fetch Url", url)

    // 创建请求
    req, _ := http.NewRequest("GET", url, nil)
    // 创建HTTP客户端
    client := &http.Client{}
    // 发出请求
    resp, err := client.Do(req)
    if err != nil {
        fmt.Println("Http get err:", err)
        return ""
    }
    if resp.StatusCode != 200 {
        fmt.Println("Http status code:", resp.StatusCode)
        return ""
    }
    // 读取HTTP响应正文
    defer resp.Body.Close()
    body, err := ioutil.ReadAll(resp.Body)
    if err != nil {
        fmt.Println("Read error", err)
        return ""
    }
    return string(body)
}

func main(){
    fmt.Println(fetch("https://github.com"))
}
```

使用官方的HTTP包可以快速的请求页面并得到返回数据。

就像Python有Scrapy库，爬虫框架可以很大程度上简化HTTP请求、数据抽取、收集的流程，同时还

提供更多的工具来帮助我们实现更复杂的功能。

Golang爬虫框架——Goribot

https://github.com/zhshch2002/goribot是一个用Golang写成的爬虫轻量框架，有不错的扩展性分布式支持能力，文档在https://imagician.net/goribot/。

获取Goribot：

```
go get -u github.com/zhshch2002/goribot
```

使用Goribot实现上文的代码的功能要看起来简洁不少。

```
package main

import (
    "fmt"
    "github.com/zhshch2002/goribot"
)

func main() {
    s := goribot.NewSpider()
    s.AddTask(
        goribot.GetReq("https://github.com"),
        func(ctx *goribot.Context) {
            fmt.Println(ctx.Resp.Text)
        },
    )
    s.Run()
}
```

如此之实现了一个单一的功能，即访问“https://github.com”并打印出结果。如此的应用还不足以用框架。那我们来入手一个更复杂点的爬虫应用。

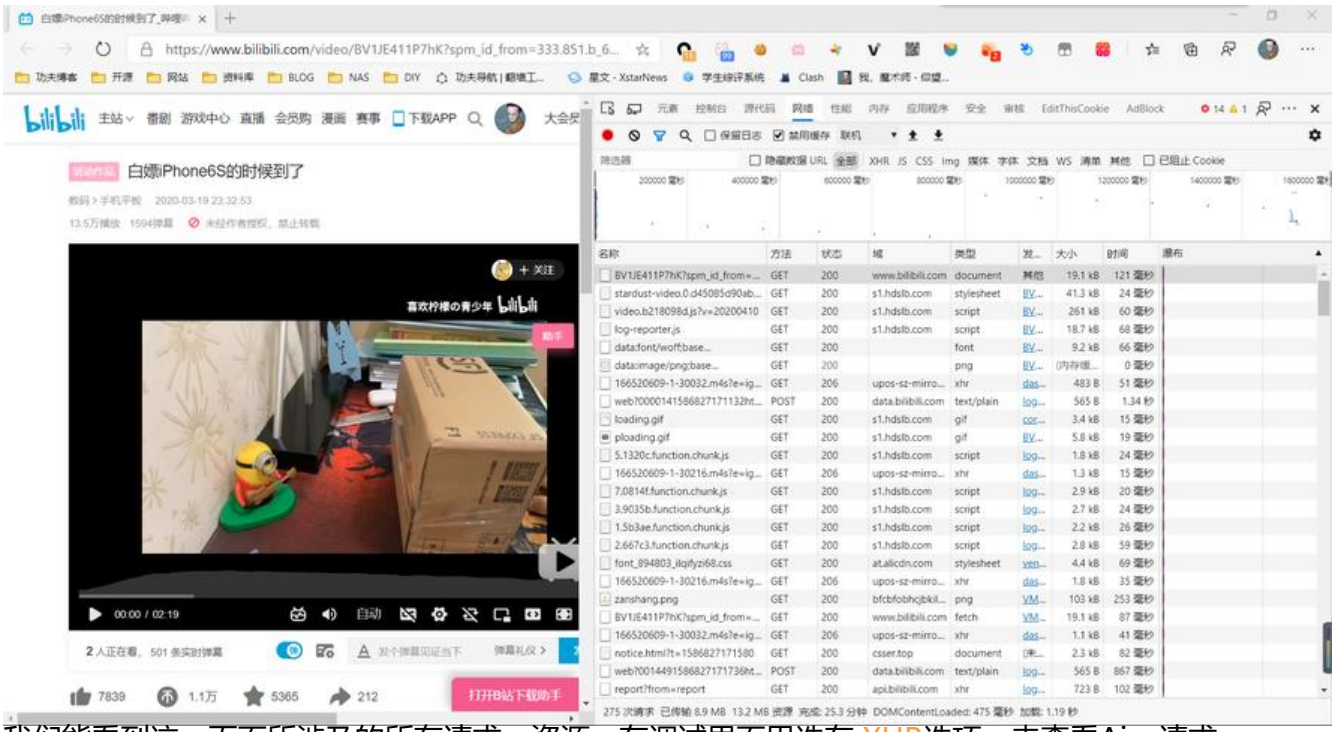
用Goribot爬取B站信息

我们来建立一个复杂点的爬虫应用，预期实现两个功能：

1. 沿着链接自动发现新的视频链接
2. 提取标题、封面图、作者和视频数据（播放量、投币、收藏等）

研究B站页面

首先我们来研究一下B站的视频页面，以https://www.bilibili.com/video/BV1JE411P7hK?spm_id_from=333.851.b_62696c695f7265706f72745f6469676974616c.21为例，按F12打开调试界面，切换 Network（网络）选项卡。

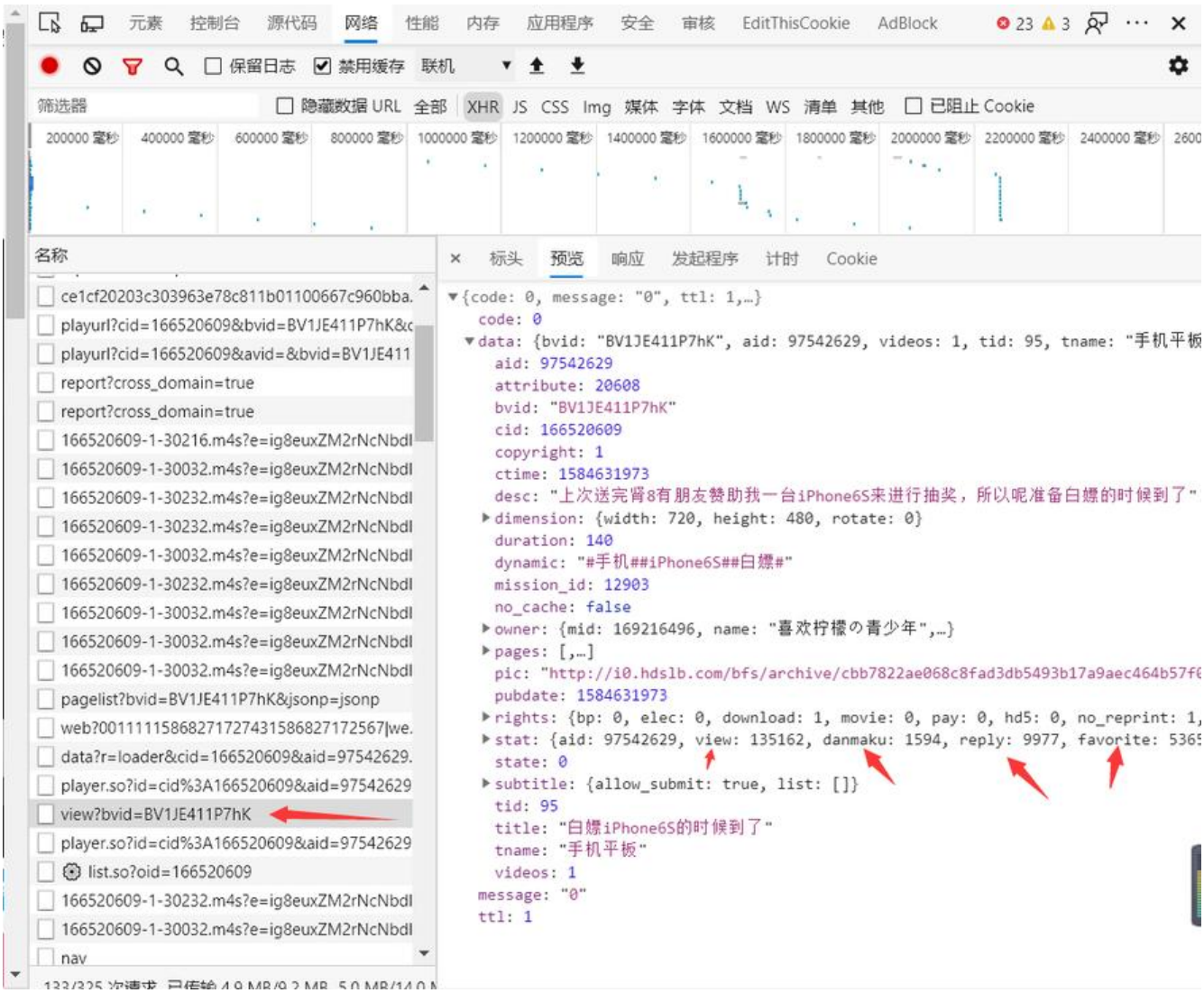


我们能看到这一页面所涉及的所有请求、资源。在调试界面里选在 XHR选项，来查看Ajax请求。

你可以通过点选不同的请求，在右侧弹出的面板里查看具体内容。在新面板里点击 Preview（预览）以查看服务器响应的内容。

那么，交给你一个任务，依次查看XHR下的所有请求，找到最像是服务器返回的点赞、收藏、播放量据的哪一个。

很好，那来看看你找到是这个吗?



你已经成功达成了一个爬虫工程师的成就——从Ajax请求里寻找目标数据。

那我们切换到 Header（标头）选项，来看看这个请求对应的参数，最好能找到这个响应和视频Id的系。

发现了视频Id——BV号。

我们以及解决了核心问题，获取B站的视频数据，对于自动搜寻视频，我们可以设定一个起始链接，后搜寻 <a>标签来延伸爬取。

搭建爬虫

完整代码在后文。

创建爬虫

```
s := goribot.NewSpider( // 创建一个爬虫并注册扩展
    goribot.Limiter(true, &goribot.LimitRule{ // 添加一个限制器，限制白名单域名和请求速录限制
        Glob: "*.bilibili.com",              // 以防对服务器造成过大压力以及被B站服务器封禁
        Rate: 2,
    }),
    goribot.RefererFiller(), // 自动填写Referer，参见Goribot(https://imagician.net/goribot/)关于
展的部分
    goribot.RandomUserAgent(), // 随机UA
    goribot.SetDepthFirst(true), // 使用深度优先策略，就是沿着一个页面，然后去子页面而非同级
面
)
```

获取视频数据

```
var getVideoInfo = func(ctx *goribot.Context) {
    res := map[string]interface{}{
        "bvid":  ctx.Resp.Json("data.bvid").String(),
        "title": ctx.Resp.Json("data.title").String(),
        "des":   ctx.Resp.Json("data.des").String(),
        "pic":   ctx.Resp.Json("data.pic").String(),  // 封面图
        "tname": ctx.Resp.Json("data.tname").String(), // 分类名
        "owner": map[string]interface{}{ //视频作者
            "name": ctx.Resp.Json("data.owner.name").String(),
            "mid":  ctx.Resp.Json("data.owner.mid").String(),
            "face": ctx.Resp.Json("data.owner.face").String(), // 头像
        },
        "ctime":   ctx.Resp.Json("data.ctime").String(), // 创建时间
        "pubdate": ctx.Resp.Json("data.pubdate").String(), // 发布时间
        "stat": map[string]interface{}{ // 视频数据
            "view":     ctx.Resp.Json("data.stat.view").Int(),
            "danmaku":  ctx.Resp.Json("data.stat.danmaku").Int(),
            "reply":    ctx.Resp.Json("data.stat.reply").Int(),
            "favorite": ctx.Resp.Json("data.stat.favorite").Int(),
            "coin":     ctx.Resp.Json("data.stat.coin").Int(),
            "share":    ctx.Resp.Json("data.stat.share").Int(),
            "like":     ctx.Resp.Json("data.stat.like").Int(),
            "dislike":  ctx.Resp.Json("data.stat.dislike").Int(),
        },
    }
    ctx.AddItem(res) // 保存到蜘蛛的Item处理队列
}
```

这是一个函数，自动解析响应里的Json数据，也就是刚才看的Ajax结果。解析完数据后保存到蜘蛛的Item处理队列。

发现新视频

```
var findVideo goribot.CtxHandlerFun
findVideo = func(ctx *goribot.Context) {
    u := ctx.Req.URL.String()
    fmt.Println(u)
    if strings.HasPrefix(u, "https://www.bilibili.com/video/") { // 判断是否为视频页面
        if strings.Contains(u, "?") {
            u = u[:strings.Index(u, "?")]
        }
        u = u[31:] // 截取视频中的BV号
        fmt.Println(u)

        // 创建一个从BV号获取具体数据的任务，使用上一个策略
        ctx.AddTask(goribot.GetReq("https://api.bilibili.com/x/web-interface/view?bvid="+u), ge
VideoInfo)
    }
    ctx.Resp.Dom.Find("a[href]").Each(func(i int, sel *goquery.Selection) {
        if h, ok := sel.Attr("href"); ok {
            ctx.AddTask(goribot.GetReq(h), findVideo) // 用同样的策略处理子页面
```

```
        }
    })
}
```

收集Item

我们在**获取视频数据**里获取了Ajax数据，并保存到Item队列。我们在这里处理这些Item以避免读写件和数据库对爬取主线程的阻塞。

```
s.OnItem(func(i interface{}) interface{} {
    fmt.Println(i) // 我们暂时不做处理，就先打印出来
    return i
})
```

OnItem的具体使用要参考Goribot文档的相关内容。

最后 Run 吧

```
// 种子任务
s.AddTask(goribot.GetReq("https://www.bilibili.com/video/BV1at411a7RS"), findVideo)
s.Run()
```

完整代码如下

```
package main

import (
    "fmt"
    "github.com/PuerkitoBio/goquery"
    "github.com/zhshch2002/goribot"
    "strings"
)

func main() {
    s := goribot.NewSpider(
        goribot.Limiter(true, &goribot.LimitRule{
            Glob: "*.bilibili.com",
            Rate: 2,
        }),
        goribot.RefererFiller(),
        goribot.RandomUserAgent(),
        goribot.SetDepthFirst(true),
    )
    var getVideoInfo = func(ctx *goribot.Context) {
        res := map[string]interface{}{
            "bvid":  ctx.Resp.Json("data.bvid").String(),
            "title": ctx.Resp.Json("data.title").String(),
            "des":   ctx.Resp.Json("data.des").String(),
            "pic":   ctx.Resp.Json("data.pic").String(),   // 封面图
            "tname": ctx.Resp.Json("data.tname").String(), // 分类名
            "owner": map[string]interface{}{ //视频作者
                "name": ctx.Resp.Json("data.owner.name").String(),
                "mid":  ctx.Resp.Json("data.owner.mid").String(),
```

```
                "face": ctx.Resp.Json("data.owner.face").String(), // 头像
            },
            "ctime":   ctx.Resp.Json("data.ctime").String(), // 创建时间
            "pubdate": ctx.Resp.Json("data.pubdate").String(), // 发布时间
            "stat": map[string]interface{}{ // 视频数据
                "view":     ctx.Resp.Json("data.stat.view").Int(),
                "danmaku":  ctx.Resp.Json("data.stat.danmaku").Int(),
                "reply":    ctx.Resp.Json("data.stat.reply").Int(),
                "favorite": ctx.Resp.Json("data.stat.favorite").Int(),
                "coin":     ctx.Resp.Json("data.stat.coin").Int(),
                "share":    ctx.Resp.Json("data.stat.share").Int(),
                "like":     ctx.Resp.Json("data.stat.like").Int(),
                "dislike":  ctx.Resp.Json("data.stat.dislike").Int(),
            },
        }
        ctx.AddItem(res)
    }
    var findVideo goribot.CtxHandlerFun
    findVideo = func(ctx *goribot.Context) {
        u := ctx.Req.URL.String()
        fmt.Println(u)
        if strings.HasPrefix(u, "https://www.bilibili.com/video/") {
            if strings.Contains(u, "?") {
                u = u[:strings.Index(u, "?")]
            }
            u = u[31:]
            fmt.Println(u)
            ctx.AddTask(goribot.GetReq("https://api.bilibili.com/x/web-interface/view?bvid="+u),
etVideoInfo)
        }
        ctx.Resp.Dom.Find("a[href]").Each(func(i int, sel *goquery.Selection) {
            if h, ok := sel.Attr("href"); ok {
                ctx.AddTask(goribot.GetReq(h), findVideo)
            }
        })
    }
    s.OnItem(func(i interface{}) interface{} {
        fmt.Println(i)
        return i
    })
    s.AddTask(goribot.GetReq("https://www.bilibili.com/video/BV1at411a7RS").SetHeader("coo
ie", "_uuid=1B9F036F-8652-DCDD-D67E-54603D58A9B904750infoc; buvid3=5D62519D-8AB5
449B-A4CF-72D17C3DFB87155806infoc; sid=9h5nzg2a; LIVE_BUVID=AUTO78158115742055
5; CURRENT_FNVAL=16; im_notify_type_403928979=0; rpdid=|(k|~uu|lu||0J'ul)ukk)~kY; _ga=
A1.2.533428114.1584175871; PVID=1; DedeUserID=403928979; DedeUserID__ckMd5=08363
45687b3545; SESSDATA=b4f022fe%2C1601298276%2C1cf0c*41; bili_jct=2f00b7d205a97aa2e
1475f93bfcb1a3; bp_t_offset_403928979=375484225910036050"), findVideo)
    s.Run()
}
```

最后

爬虫框架只是工具，重要的是人怎么使用它。了解工具可以看项目_examples和文档。