Colly: обзор фреймворка для веб-скрейпинга на Go

Joanna Shevchuk

О себе

- Go + Python + Linux;
- Менторю Django Girls и курс PyLadies

Веб-скрейпинг

извлечение данных из веб-страниц для последующей структуризации.

API

Программный интерфейс приложения

описание способов, помогающих одной программе взаимодействовать с другой.

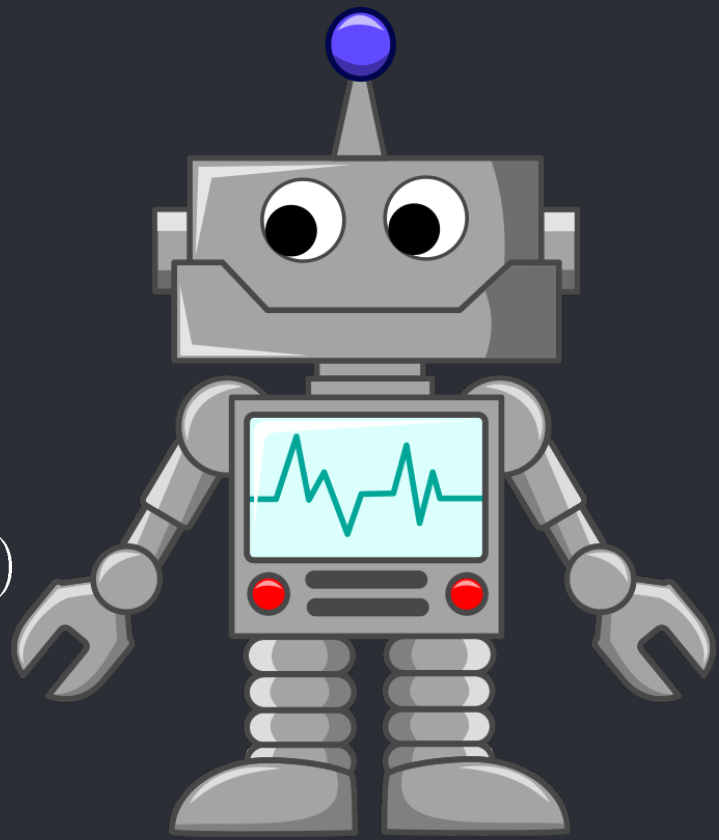
- ✔ Есть API - используем API.
- ☹ Нет API - используем скрейпер.

robots.txt

наш_сайт/robots.txt

User-agent: *
Disallow: /

(скрейпить нельзя помиловать **)



Colly: Fast and Elegant Scraping Framework for Gophers

Преимущества

- Толковый API и подробная документация;
- Много плюшек;
- На Go = привычнее;
- Конкурентность.

О плюшках:

- Скрейпит синхронно/асинхронно/параллельно;
- Автоматически кодирует не-Юникодные символы;
- Сам вычищает cookies;
- Обрабатывает robots.txt;
- Можно прикрутить БД: SQLite или MongoDB;

Недостатки

- Нет встроенного headless-браузера;

Headless browser

браузер без графического интерфейса (работает через командную строку).

```go
func main() {
    c := colly.NewCollector()

    c.OnHTML("a[href]", func(e *colly.HTMLElement) {
        e.Request.Visit(e.Attr("href"))
    })

    c.OnRequest(func(r *colly.Request) {
        fmt.Println("Visiting", r.URL)
    })

    c.Visit("http://go-colly.org/")
}
```

```go
type Collector struct {
    UserAgent string
    MaxDepth int
    AllowedDomains []string
    DisallowedDomains []string
    DisallowedURLFilters []*regexp.Regexp
    URLFilters []*regexp.Regexp
    AllowURLRevisit bool
    MaxBodySize int
    CacheDir string
    IgnoreRobotsTxt bool
    Async bool
    ParseHTTPErrorResponse bool
    ID  uint32
    DetectCharset bool
    RedirectHandler func(req *http.Request, via []*http.Request)
        error
    CheckHead bool
}
```

User Agent

как концепция

приложение, через определенный сетевой протокол обеспечивающее доступ к веб-контенту (например, браузер или скрейпер).

как элемент

строка, содержащая сведения о браузере или скрейпере: название, версия, платформа (ОС), движок.

Скрейпим статический сайт

```
package main

import (
    "encoding/csv"
    "log"
    "os"

    "github.com/gocolly/colly"
)

func main() {
    fName := "xkcd_store_items.csv"
    file, err := os.Create(fName)
    if err != nil {
        log.Fatalf("Cannot create file %q: %s\n", fName, err)
        return
    }
```

```
...
defer file.Close()
writer := csv.NewWriter(file)
defer writer.Flush()

writer.Write([]string{"Name", "Price", "URL", "Image URL"})

c := colly.NewCollector(

    colly.AllowedDomains("store.xkcd.com"),
)

...
```

```
    ...
    c.OnHTML(‘.next a[href]‘, func(e *colly.HTMLElement) {
        e.Request.Visit(e.Attr("href"))
    })

    c.Visit("https://store.xkcd.com/collections/everything")

    log.Printf("Scraping finished, check file %q for results\n",
        fName)


    log.Println(c)
}
```

Скрейпим динамический сайт (не через родной API)

Instagram

Под капотом есть goquery.

- Исходный код страницы
- Ищем переменную window._sharedData

```
c := colly.NewCollector()

c.OnHTML("body > script:first-of-type", func(e *colly.HTMLElement
    ) {

    jsonData := e.Text[strings.Index(e.Text, "{") : len(e.Text)
        -1]
```

```
data := struct {
    EntryData struct {
        ProfilePage []struct {
            User struct {
                Id    string `json:"id"`
                Media struct {
                    Nodes []struct {
                        ImageURL     string `json:"display_src
                            "`
                        ThumbnailURL string `json:"
                            thumbnail_src"`
                        IsVideo      bool   `json:"is_video"`
                        Date         int    `json:"date"`
                        Dimensions   struct {
                            Width  int `json:"width"`
                            Height int `json:"height"`
                        }
                    }
    ...
```

```
...
                    PageInfo pageInfo `json:"page_info"`
                } `json:"media"`
            } `json:"user"`
        } `json:"ProfilePage"`
    } `json:"entry_data"`
}{}

err := json.Unmarshal([]byte(jsonData), &data)
if err != nil {
    log.Fatal(err)
}
```

```
    page := data.EntryData.ProfilePage[0]
    actualUserId = page.User.Id
    for _, obj := range page.User.Media.Nodes {

        if obj.IsVideo {
            continue
        }
        c.Visit(obj.ImageURL)
    }

}
```

```
const nextPageURLTemplate string = ‘https://www.instagram.com/
    graphql/query/?query_id=17888483320059182&variables={"id":"%s
    ","first":12,"after":"%s"}‘

//...
c.OnResponse(func(r *colly.Response) {
    if strings.Index(r.Headers.Get("Content-Type"), "image") > -1
         {
        r.Save(outputDir + r.FileName())
        return
    }
}
```

http://go-colly.org/

https://github.com/gocolly/colly

https://godoc.org/github.com/gocolly/colly

https://godoc.org/github.com/gocolly/colly/extensions

Мои контакты

djeanne
joannashevchuk
djeanne

djeanne.github.io

joanne.shevchuk@gmail.com
djeanne@pm.me