
Geziyor

Geziyor is a blazing fast web crawling and web scraping framework. It can be used to crawl websites and extract structured data from them. Geziyor is useful for a wide range of purposes such as data mining, monitoring and automated testing.



Features

- **JS Rendering**
- 5.000+ Requests/Sec
- Caching (Memory/Disk/LevelDB)
- Automatic Data Exporting (JSON, CSV, or custom)
- Metrics (Prometheus, Expvar, or custom)
- Limit Concurrency (Global/Per Domain)
- Request Delays (Constant/Randomized)
- Cookies, Middlewares, robots.txt
- Automatic response decoding to UTF-8
- Proxy management (Single, Round-Robin, Custom)

See scraper Options for all custom settings.

Status

We highly recommend you to use Geziyor with go modules.

Usage

This example extracts all quotes from *quotes.toscrape.com* and exports to JSON file.

```
1 func main() {
2     geziyor.NewGeziyor(&geziyor.Options{
3         StartURLs: []string{"http://quotes.toscrape.com/"},
4         ParseFunc: quotesParse,
5         Exporters: []export.Exporter{&export.JSON{}}},
6     ).Start()
7 }
8
9 func quotesParse(g *geziyor.Geziyor, r *client.Response) {
```

```

10     r.HTMLDoc.Find("div.quote").Each(func(i int, s *goquery.Selection)
11     {
12         g.Exports <- map[string]interface{}{
13             "text": s.Find("span.text").Text(),
14             "author": s.Find("small.author").Text(),
15         }
16     })
17     if href, ok := r.HTMLDoc.Find("li.next > a").Attr("href"); ok {
18         g.Get(r.JoinURL(href), quotesParse)
19     }

```

See tests for more usage examples.

Documentation

Installation

```
1 go get -u github.com/geziyor/geziyor
```

If you want to make JS rendered requests, make sure you have Chrome installed.

NOTE: macOS limits the maximum number of open file descriptors. If you want to make concurrent requests over 256, you need to increase limits. Read this for more.

Making Normal Requests

Initial requests start with `StartURLs []string` field in `Options`. Geziyor makes concurrent requests to those URLs. After reading response, `ParseFunc func(g *Geziyor, r *Response)` called.

```

1 geziyor.NewGeziyor(&geziyor.Options{
2     StartURLs: []string{"http://api.ipify.org"},
3     ParseFunc: func(g *geziyor.Geziyor, r *client.Response) {
4         fmt.Println(string(r.Body))
5     },
6 }).Start()

```

If you want to manually create first requests, set `StartRequestsFunc`. `StartURLs` won't be used if you create requests manually.

You can make requests using `Geziyor` methods:

```

1 geziyor.NewGeziyor(&geziyor.Options{
2     StartRequestsFunc: func(g *geziyor.Geziyor) {
3         g.Get("https://httpbin.org/anything", g.Opt.ParseFunc)

```

```
4     g.Head("https://httpbin.org/anything", g.Opt.ParseFunc)
5 },
6 ParseFunc: func(g *geziyor.Geziyor, r *client.Response) {
7     fmt.Println(string(r.Body))
8 },
9 }).Start()
```

Making JS Rendered Requests

JS Rendered requests can be made using `GetRendered` method. By default, geziyor uses local Chrome application CLI to start Chrome browser. Set `BrowserEndpoint` option to use different chrome instance. Such as, "ws://localhost:3000"

```
1 geziyor.NewGeziyor(&geziyor.Options{
2     StartRequestsFunc: func(g *geziyor.Geziyor) {
3         g.GetRendered("https://httpbin.org/anything", g.Opt.ParseFunc)
4     },
5     ParseFunc: func(g *geziyor.Geziyor, r *client.Response) {
6         fmt.Println(string(r.Body))
7     },
8     //BrowserEndpoint: "ws://localhost:3000",
9 }).Start()
```

Extracting Data

We can extract HTML elements using `response.HTMLDoc`. HTMLDoc is Goquery's Document.

HTMLDoc can be accessible on Response if response is HTML and can be parsed using Go's built-in HTML parser If response isn't HTML, `response.HTMLDoc` would be `nil`.

```
1 geziyor.NewGeziyor(&geziyor.Options{
2     StartURLs: []string{"http://quotes.toscrape.com/"},
3     ParseFunc: func(g *geziyor.Geziyor, r *client.Response) {
4         r.HTMLDoc.Find("div.quote").Each(func(_ int, s *goquery.
5             Selection) {
6             log.Println(s.Find("span.text").Text(), s.Find("small.
7                 author").Text())
8         })
9     }).Start()
```

Exporting Data

You can export data automatically using exporters. Just send data to `Geziyor.Exports` chan. Available exporters

```
1 geziyor.NewGeziyor(&geziyor.Options{
2     StartURLs: []string{"http://quotes.toscrape.com/"},
3     ParseFunc: func(g *geziyor.Geziyor, r *client.Response) {
4         r.HTMLDoc.Find("div.quote").Each(func(_ int, s *goquery.
5             Selection) {
6             g.Exports <- map[string]interface{}{
7                 "text": s.Find("span.text").Text(),
8                 "author": s.Find("small.author").Text(),
9             }
10        })
11    },
12    Exporters: []export.Exporter{&export.JSON{}}),
13    Start()
```

Custom Requests - Passing Metadata To Callbacks

You can create custom requests with `client.NewRequest`

Use that request on `geziyor.Do(request, callback)`

```
1 geziyor.NewGeziyor(&geziyor.Options{
2     StartRequestsFunc: func(g *geziyor.Geziyor) {
3         req, _ := client.NewRequest("GET", "https://httpbin.org/
4             anything", nil)
5         req.Meta["key"] = "value"
6         g.Do(req, g.Opt.ParseFunc)
7     },
8     ParseFunc: func(g *geziyor.Geziyor, r *client.Response) {
9         fmt.Println("This is our data from request: ", r.Request.Meta["
10             key"])
11     },
12    Start()
```

Proxy - Use proxy per request

If you want to use proxy for your requests, and you have 1 proxy, you can just set these env values: `HTTP_PROXY` `HTTPS_PROXY` And `geziyor` will use those proxies.

Also, you can use in-order proxy per request by setting `ProxyFunc` option to `client.RoundRobinProxy` Or any custom proxy selection function that you want. See `client/proxy.go` on how to implement that kind of custom proxy selection function.

Proxies can be HTTP, HTTPS and SOCKS5.

Note: If you use [http](#) scheme for proxy, It'll be used for http requests and not for https requests.

```
1 geziyor.NewGeziyor(&geziyor.Options{
2     StartURLs:      []string{"http://httpbin.org/anything"},
3     ParseFunc:      parseFunc,
4     ProxyFunc:       client.RoundRobinProxy("http://some-http-proxy.
                        com", "https://some-https-proxy.com", "socks5://some-socks5-
                        proxy.com"),
5 }).Start()
```

Benchmark

8748 request per seconds on *Macbook Pro 15" 2016*

See tests for this benchmark function:

```
1 >> go test -run none -bench Requests -benchtime 10s
2 goos: darwin
3 goarch: amd64
4 pkg: github.com/geziyor/geziyor
5 BenchmarkRequests-8      200000      108710 ns/op
6 PASS
7 ok      github.com/geziyor/geziyor  22.861s
```