Go doing a GET request and building the Querystring

Asked 8 years ago Modified 2 years, 10 months ago Viewed 237k times



147

I am pretty new to Go and don't quite understand everything as yet. In many of the modern languages Node.js, Angular, jQuery, PHP you can do a GET request with additional query string parameters.



Doing this in Go isn't quite a simple as it seems, and I can't really figure it out as yet. I really don't want to have to concatenate a string for each of the requests I want to do.



Here is the sample script:

```
package main
import (
   "fmt"
    "io/ioutil"
    "net/http"
)
func main() {
   client := &http.Client{}
   req, _ := http.NewRequest("GET", "http://api.themoviedb.org/3/tv/popular", nil)
   req.Header.Add("Accept", "application/json")
   resp, err := client.Do(req)
    if err != nil {
       fmt.Println("Errored when sending request to the server")
       return
    }
   defer resp.Body.Close()
   resp_body, _ := ioutil.ReadAll(resp.Body)
   fmt.Println(resp.Status)
   fmt.Println(string(resp_body))
}
```

In this example you can see there is a URL, which requires a GET variable of api_key with your api key as the value. The problem being that this becomes hard coded in the form of:

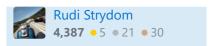
```
req, _ := http.NewRequest("GET", "http://api.themoviedb.org/3/tv/popular?
api_key=mySuperAwesomeApiKey", nil)
```

Is there a way to build this query string dynamically?? At the moment I will need to assemble the URL prior to this step in order to get a valid response.

http

go





- 1 So what is wrong with concatenating a string? Salvador Dali Jun 4, 2015 at 19:34
- 10 I suppose nothing, but it's not really a elegant sollution, just thought there is a better way of doing things in Go. You see the action changes, the method and then you have to string everything together.
 - Rudi Strydom Jun 4, 2015 at 19:40 /
- You can use <u>url.Values</u> 's <u>Encode</u> method. You could also use <u>URL.String</u> to build up the whole URL. Dave C Jun 4, 2015 at 20:46

3 Answers

Highest score (default)

Sorted by:





As a commenter mentioned you can get Values from net/url which has an Encode method. You could do something like this (req.URL.Query() returns the existing url.Values)

305







```
package main
import (
    "fmt"
    "log"
    "net/http"
    "os"
)
func main() {
    req, err := http.NewRequest("GET", "http://api.themoviedb.org/3/tv/popular",
nil)
    if err != nil {
        log.Print(err)
        os.Exit(1)
    }
    q := req.URL.Query()
    q.Add("api_key", "key_from_environment_or_flag")
    q.Add("another_thing", "foo & bar")
    req.URL.RawQuery = q.Encode()
    fmt.Println(req.URL.String())
    // Output:
    // http://api.themoviedb.org/3/tv/popular?
another_thing=foo+%26+bar&api_key=key_from_environment_or_flag
}
```

http://play.golang.org/p/L5XCrw9VIG

Share Improve this answer

edited Dec 5, 2016 at 20:25

answered Jun 5, 2015 at 2:41

Follow



³ Awesome thank you man! That's exactly what I was looking for! - Rudi Strydom Jun 5, 2015 at 7:47

- @artificerpi if it was critical for some reason you could reimplement what they do inside the Encode method golang.org/src/net/url/url.go?s=24222:24253#L845 But I would wonder why it mattered. jcbwlkr Jan 15, 2018 at 18:58
- 4 You don't need to use NewRequest if you're not doing anything with it. You can just use url.Parse("https://something.com/") instead or even create an URL object directly. Richard Mar 25, 2019 at 5:43
- 1 @JakeBoomgaarden the http.NewRequest function does not actually call the remote server. It just constructs a *http.Request variable for you to use. You can tweak the request to your liking then send it off by passing it to the *http.Client.Do method. jcbwlkr Sep 8, 2020 at 19:52



Use r.URL.Query() when you appending to existing query, if you are building new set of params use the url.Values struct like so

57







```
package main
import (
    "fmt"
    "log"
    "net/http"
    "net/url"
    "os"
)
func main() {
   req, err := http.NewRequest("GET","http://api.themoviedb.org/3/tv/popular", nil)
    if err != nil {
       log.Print(err)
        os.Exit(1)
    }
    // if you appending to existing query this works fine
    q := req.URL.Query()
   q.Add("api_key", "key_from_environment_or_flag")
    q.Add("another_thing", "foo & bar")
    // or you can create new url. Values struct and encode that like so
   q := url.Values{}
    q.Add("api_key", "key_from_environment_or_flag")
    q.Add("another_thing", "foo & bar")
   req.URL.RawQuery = q.Encode()
   fmt.Println(req.URL.String())
    // Output:
http://api.themoviedb.org/3/tv/popularanother_thing=foo+%26+bar&api_key=key_from_environum.
```

Share Improve this answer Follow

TDD allyraza allyraza 1,346 ● 11 ● 7



Using NewRequest just to create an URL is an overkill. Use the net/url package:

53







```
package main
import (
    "fmt"
    "net/url"
)
func main() {
    base, err := url.Parse("http://www.example.com")
    if err != nil {
       return
    }
    // Path params
    base.Path += "this will get automatically encoded"
    // Query params
    params := url.Values{}
    params.Add("q", "this will get encoded as well")
    base.RawQuery = params.Encode()
   fmt.Printf("Encoded URL is %q\n", base.String())
}
```

Playground: https://play.golang.org/p/YCTvdluws-r

Share Improve this answer Follow

answered Jul 11, 2019 at 9:29

Janek Olszak
4,017 • 1 • 28 • 22

- 1 Agreed, great answer! pow-il Nov 18, 2020 at 17:02
- 2 This should really be the accepted answer. Paul Nov 28, 2020 at 12:45

Totally agree, creating a request is overkill for something small like this. – kurczynski Mar 15, 2022 at 15:17