How to get JSON response from http.Get

Asked 7 years ago Active 5 months ago Viewed 180k times

133

58

I'm trying read JSON data from web, but that code returns empty result. I'm not sure what I'm doing wrong here.

```
package main

import "os"
import "fmt"
import "net/http"
import "io/ioutil"
import "encoding/json"

type Tracks struct {
    Toptracks []Toptracks_info
}

type Toptracks_info struct {
    Track []Track_info
    Attr  []Attr_info
}

type Track_info struct {
    Name       string
    Duration   string
    Listeners  string
    Mbid       string
    Url        string
    Streamable []Streamable_info
    Artist     []Artist_info
    Attr       []Track_attr_info
}

type Attr_info struct {
    Country    string
    Page       string
    PerPage    string
    TotalPages string
    Text      string
    Fulltrack string
}

type Artist_info struct {
    Name string
    Mbid string
    Url  string
}

type Track_attr_info struct {
    Rank string
}

func get_content() {
    // json data
    url := "http://ws.audioscrobbler.com/2.0/?
method=geo.gettoptracks&api_key=c1572082105bd40d247836b5c1819623&format=json&count


    res, err := http.Get(url)

    if err != nil {
        panic(err.Error())
    }

    body, err := ioutil.ReadAll(res.Body)

    if err != nil {
        panic(err.Error())
    }

    var data Tracks
    json.Unmarshal(body, &data)
    fmt.Printf("Results: %v\n", data)
    os.Exit(0)
}

func main() {
    get_content()
}
```

json go

edited Aug 11 '19 at 14:36
Dave C
6,274 4 35 54

asked Jun 17 '13 at 20:35
Akshaydeep Giri
2,221 5 22 32

4 Answers

Active Oldest Votes

263

The ideal way is *not* to use `ioutil.ReadAll`, but rather use a decoder on the reader directly. Here's a nice function that gets a url and decodes its response onto a `target` structure.

```
var myClient = &http.Client{Timeout: 10 * time.Second}

func getJson(url string, target interface{}) error {
    r, err := myClient.Get(url)
    if err != nil {
        return err
    }
    defer r.Body.Close()

    return json.NewDecoder(r.Body).Decode(target)
}
```

Example use:

```
type Foo struct {
    Bar string
}

func main() {
    foo1 := new(Foo) // or &Foo{}
    getJson("http://example.com", foo1)
    println(foo1.Bar)

    // alternately:

    foo2 := Foo{}
    getJson("http://example.com", &foo2)
    println(foo2.Bar)
}
```

You should not be using the default `*http.Client` structure in production as this answer originally demonstrated! (Which is what `http.Get` /etc call to). The reason is that the default client has no timeout set; if the remote server is unresponsive, you're going to have a bad day.

edited Jan 30 at 8:43
Barlas Apaydin
6,665 10 48 81

answered Jun 30 '15 at 5:06
Connor Peet
4,980 3 18 31

5 It seems like you need to use Uppercase for the names of the items in the struct e.g. `type WebKeys struct {        Keys []struct {            X5t string                X5c []string        } }` even when the actual params in the JSON you're parsing are in lower case. JSON example: `{        "keys": [{                "x5t": "foo",                "x5c": "baaaar"            }] }` – Wilson May 9 '16 at 3:26

1 @Roman, no. If an error is returned, the response value is nil. (An error means we couldn't read any valid HTTP response, there's no body to close!) You can test this by pointing .Get() at a non-existent URL. Tis method is demonstrated in the second code block in the net/http docs. – Connor Peet Aug 6 '17 at 15:29

1 @NamGVU saves a potential allocation and allows the use of http keep-alive to reuse connections. – Connor Peet Nov 27 '17 at 22:16

2 @ConnorPeet You made my day thanks ! I wonder what you meant with "You should not be using the default *http.Client structure in production". Did you mean that one should use `&http.Client{Timeout: 10 * time.Second}` or use a whole other library/strategy ? – Jona Rodrigues Jan 18 '18 at 20:05

6 Just a warning to others - `json.NewDecoder(r.Body).Decode(target)` will *not* return an error for certain types of malformed JSON! I just wasted a few hours trying to understand why I kept getting an empty response - turns out the source JSON had an extra comma where it shouldn't have been. I suggest you use `json.Unmarshal` instead. There's also a good writeup about other potential dangers of using `json.Decoder` here – adamc Oct 6 '18 at 10:33

25

Your Problem were the slice declarations in your data `structs` (except for `Track`, they shouldn't be slices...). This was compounded by some rather goofy fieldnames in the fetched json file, which can be fixed via structtags, see godoc.

The code below parsed the json successfully. If you've further questions, let me know.

```
package main

import "fmt"
import "net/http"
import "io/ioutil"
import "encoding/json"

type Tracks struct {
    Toptracks Toptracks_info
}
```

```
    Attr  Attr_info `json: "@attr"`
}

type Track_info struct {
    Name       string
    Duration   string
    Listeners  string
    Mbid       string
    Url        string
    Streamable Streamable_info
    Artist     Artist_info
    Attr       Track_attr_info `json: "@attr"`
}

type Attr_info struct {
    Country    string
    Page       string
    PerPage    string
    TotalPages string
    Total      string
}

type Streamable_info struct {
    Text      string `json: "#text"`
    Fulltrack string
}

type Artist_info struct {
    Name string
    Mbid string
    Url  string
}

type Track_attr_info struct {
    Rank string
}

func perror(err error) {
    if err != nil {
        panic(err)
    }
}

func get_content() {
    url := "http://ws.audioscrobbler.com/2.0/?
```

```
        res, err := http.Get(url)
        perror(err)
        defer res.Body.Close()

        decoder := json.NewDecoder(res.Body)
        var data Tracks
        err = decoder.Decode(&data)
        if err != nil {
            fmt.Printf("%T\n%s\n%#v\n",err, err, err)
            switch v := err.(type){
                case *json.SyntaxError:
                    fmt.Println(string(body[v.Offset-40:v.Offset]))
            }
        }
        for i, track := range data.Toptracks.Track{
            fmt.Printf("%d: %s %s\n", i, track.Artist.Name, track.Name)
        }
    }

    func main() {
        get_content()
    }
```

edited Jul 7 '16 at 16:04
Myles McDonnell
10.9k 12 52 89

answered Jun 17 '13 at 22:10
tike
1,984 12 18

1 There is something in the response body. – peterSO Jun 17 '13 at 23:28

6 In my case, I was missing UPPER-CASE first character in the "struct" fields. – abourget Jun 8 '14 at 19:30

Answer below is right, using a Decoder directly on the response.Body avoids unnecessary allocations and is generally more ideomatic. Corrected my answer, thanks for pointing it out. – tike Dec 12 '15 at 18:30

@abourget omg thank you for this comment. Just spend 1 hours looking for problems in parser, confirming with wireshark that response is correct... thanks – agilob Mar 26 '17 at 20:15

You need upper case property names in your structs in order to be used by the json packages.

You also need to pass the your data object by reference (`&data`).

```
package main

import "os"
import "fmt"
import "net/http"
import "io/ioutil"
import "encoding/json"

type tracks struct {
    Toptracks []toptracks_info
}

type toptracks_info struct {
    Track []track_info
    Attr  []attr_info
}

type track_info struct {
    Name       string
    Duration   string
    Listeners  string
    Mbid       string
    Url        string
    Streamable []streamable_info
    Artist     []artist_info
    Attr       []track_attr_info
}

type attr_info struct {
    Country    string
    Page       string
    PerPage    string
    TotalPages string
    Total      string
}

type streamable_info struct {
    Text      string
    Fulltrack string
}
```

```
    Url  string
}

type track_attr_info struct {
    Rank string
}

func get_content() {
    // json data
    url := "http://ws.audioscrobbler.com/2.0/?
method=geo.gettoptracks&api_key=c1572082105bd40d247836b5c1819623&format=json&count


    res, err := http.Get(url)

    if err != nil {
        panic(err.Error())
    }

    body, err := ioutil.ReadAll(res.Body)

    if err != nil {
        panic(err.Error())
    }

    var data tracks
    json.Unmarshal(body, &data)
    fmt.Printf("Results: %v\n", data)
    os.Exit(0)
}

func main() {
    get_content()
}
```

edited Jun 17 '13 at 20:47

answered Jun 17 '13 at 20:42
Daniel
33.1k 8 78 70

still not work, is this working for you ? same empty response – Akshaydeep Giri Jun 17 '13 at 20:47

8

The results from `json.Unmarshal` (into `var data interface{}`) do not directly match your Go type and variable declarations. For example,

```
package main

import (
    "encoding/json"
    "fmt"
    "io/ioutil"
    "net/http"
    "os"
)

type Tracks struct {
    Toptracks []Toptracks_info
}

type Toptracks_info struct {
    Track []Track_info
    Attr  []Attr_info
}

type Track_info struct {
    Name       string
    Duration   string
    Listeners  string
    Mbid       string
    Url        string
    Streamable []Streamable_info
    Artist     []Artist_info
    Attr       []Track_attr_info
}

type Attr_info struct {
    Country    string
    Page       string
    PerPage    string
    TotalPages string
    Total      string
}

type Streamable_info struct {
    Text      string
    Fulltrack string
    Name string
    Mbid string
    Url  string
}

type Track_attr_info struct {
    Rank string
}

func get_content() {
    // json data
    url := "http://ws.audioscrobbler.com/2.0/?
method=geo.gettoptracks&api_key=c1572082105bd40d247836b5c1819623&format=json&count

    url += "&limit=1" // limit data for testing
    res, err := http.Get(url)
    if err != nil {
        panic(err.Error())
    }
    body, err := ioutil.ReadAll(res.Body)
    if err != nil {
        panic(err.Error())
    }
    var data interface{} // TopTracks
    err = json.Unmarshal(body, &data)
    if err != nil {
        panic(err.Error())
    }
    fmt.Printf("Results: %v\n", data)
    os.Exit(0)
}

func main() {
    get_content()
}
```

Output:

```
Results: map[toptracks:map[track:map[name:Get Lucky (feat. Pharrell Williams)
listeners:1863 url:http://www.last.fm/music/Daft+Punk/_/Get+Lucky+
(feat.+Pharrell+Williams) artist:map[name:Daft Punk mbid:056e4f3e-d505-4dad-
8ec1-d04f521cbb56 url:http://www.last.fm/music/Daft+Punk] image:
```

```
map[#text:http://userserve-ak.last.fm/serve/300x300/88137413.png
size:extralarge]] @attr:map[rank:1] duration:369 mbid: streamable:map[#text:1
fulltrack:0]] @attr:map[country:Netherlands page:1 perPage:1 totalPages:500
total:500]]]
```

answered Jun 17 '13 at 23:41

peterSO

121k 22 208 209