



HTTP Basic Auth ⁺

HTTP Basic Auth- Sesi 9

Http Basic Auth

HTTP Basic Auth adalah salah satu teknik otentikasi http request. Metode ini membutuhkan informasi username dan password untuk disisipkan dalam header request (dengan format tertentu), jadi cukup sederhana, tidak memerlukan cookies maupun session. Lebih jelasnya silakan baca [RFC-7617](#).

Informasi username dan password tidak serta merta disisipkan dalam header, informasi tersebut harus di-encode terlebih dahulu ke dalam format yg sudah ditentukan sesuai spesifikasi, sebelum dimasukan ke header.

Berikut adalah contoh penulisan basic auth.

```
// Request Header
```

```
Authorization: Basic c29AGHTYIUEuW14Klp5R32jd==
```

HTTP Basic Auth- Sesi 9

Http Basic Auth

Informasi disisipkan dalam request header dengan key Authorization, dan value adalah Basic spasi hasil enkripsi dari data username dan password.

Data username dan password digabung dengan separator tanda titik dua (:), lalu di-encode dalam format encoding Base 64.

```
// Username password encryption  
  
base64encode("username:password")  
  
// Hasilnya adalah dXNlcm5hbWU6cGFzc3dvcmQ=
```

Go menyediakan fungsi untuk meng-handle request basic auth dengan cukup mudah, jadi tidak perlu untuk memarsing header request terlebih dahulu untuk mendapatkan informasi username dan password.

HTTP Basic Auth- Sesi 9

Struktur Folder Project & Endpoint

Kita akan membuat sebuah web service sederhana, isinya satu buah endpoint. Endpoint ini kita manfaatkan sebagai dua endpoint, dengan pembeda adalah informasi pada query string-nya.

- Endpoint /student, menampilkan semua data siswa.
- Endpoint /student?id=s001, menampilkan data siswa sesuai dengan id yang di minta.

Data siswa sendiri merupakan slice object yang disimpan di variabel global.

OK, langsung saja kita praktekan. Siapkan 3 buah file berikut, tempatkan dalam satu folder proyek.

- main.go
- middleware.go
- student.go

HTTP Basic Auth- Sesi 9

Routing

Buka file main.go, isi dengan kode berikut.

```
package main

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

func main() {
    http.HandleFunc("/student", ActionStudent)

    server := new(http.Server)
    server.Addr = ":9000"

    fmt.Println("server started at localhost:9000")
    server.ListenAndServe()
}
```



HTTP Basic Auth- Sesi 9

Routing

Siapkan handler untuk rute /student.

```
func ActionStudent(w http.ResponseWriter, r *http.Request) {  
    if !Auth(w, r)        { return }  
    if !AllowOnlyGET(w, r) { return }  
  
    if id := r.URL.Query().Get("id"); id != "" {  
        OutputJSON(w, SelectStudent(id))  
        return  
    }  
  
    OutputJSON(w, GetStudents())  
}
```



HTTP Basic Auth- Sesi 9

Routing

Di dalam rute `/student` terdapat beberapa validasi.

- Validasi `!Auth(w, r)`; Nantinya akan kita buat fungsi `Auth()` untuk mengecek apakah request merupakan valid basic auth request atau tidak.
- Validasi `!AllowOnlyGET(w, r)`; Nantinya juga akan kita siapkan fungsi `AllowOnlyGET()`, gunanya untuk memastikan hanya request dengan method `GET` yang diperbolehkan masuk.

Setelah request lolos dari 2 validasi di atas, kita cek lagi apakah request ini memiliki parameter student id.

- Ketika tidak ada parameter student id, maka endpoint ini mengembalikan semua data user yang ada, lewat pemanggilan fungsi `GetStudents()`.
- Sedangkan jika ada parameter student id, maka hanya user dengan id yg diinginkan yg dijadikan nilai balik, lewat fungsi `SelectStudent(id)`.



HTTP Basic Auth- Sesi 9

Routing

Selanjutnya tambahkan satu fungsi lagi di main, `OutputJSON()`. Fungsi ini digunakan untuk mengkonversi data menjadi JSON string.

```
func OutputJSON(w http.ResponseWriter, o interface{}) {
    res, err := json.Marshal(o)
    if err != nil {
        w.Write([]byte(err.Error()))
        return
    }

    w.Header().Set("Content-Type", "application/json")
    w.Write(res)
}
```



HTTP Basic Auth- Sesi 9

Student Data

Buka file `student.go`, siapkan struct `Student` dan variabel untuk menampung data yang bertipe `[]Student`. Data inilah yang dijadikan nilai balik di endpoint yang sudah dibuat.

```
package main

var students = []*Student{}

type Student struct {
    Id    string
    Name  string
    Grade int32
}
```

HTTP Basic Auth- Sesi 9

Student Data

Buat function *GetStudents*, function ini mengembalikan semua data student. Dan buat juga function *SelectStudent(id)*, function ini mengembalikan data student sesuai dengan id terpilih.

```
func GetStudents() []*Student {  
    return students  
}  
  
func SelectStudent(id string) *Student {  
    for _, each := range students {  
        if each.Id == id {  
            return each  
        }  
    }  
  
    return nil  
}
```



HTTP Basic Auth- Sesi 9

Student Data

Kemudian implementasikan function *init*, buat beberapa dummy data untuk ditampung pada variabel *students*.

```
func init() {  
    students = append(students, &Student{Id: "s001", Name: "bourne", Grade: 2})  
    students = append(students, &Student{Id: "s002", Name: "ethan", Grade: 2})  
    students = append(students, &Student{Id: "s003", Name: "wick", Grade: 3})  
}
```

HTTP Basic Auth- Sesi 9

Function Auth & AllowOnlyGet

Function Auth

Buka file middleware.go, siapkan function *Auth*.

```
package main

import "net/http"

const USERNAME = "batman"
const PASSWORD = "secret"

func Auth(w http.ResponseWriter, r *http.Request) bool {
    username, password, ok := r.BasicAuth()
    if !ok {
        w.Write([]byte(`something went wrong`))
        return false
    }

    isValid := (username == USERNAME) && (password == PASSWORD)
    if !isValid {
        w.Write([]byte(`wrong username/password`))
        return false
    }

    return true
}
```



HTTP Basic Auth- Sesi 9

Function Auth & AllowOnlyGet

Tugas function *Auth* adalah memvalidasi apakah request merupakan valid basic auth request, dan juga apakah credentials yang dikirim cocok dengan data pada aplikasi kita. Informasi acuan credentials sendiri di hardcode pada konstanta `USERNAME` dan `PASSWORD`.

Function *Auth* mengembalikan 3 informasi:

1. Username
2. Password
3. Nilai balik ke-3 ini adalah representasi valid tidak nya basic auth request yang sedang berlangsung

Jika basic auth request tidak valid, maka tampilkan pesan error sebagai nilai balik. Sedangkan jika basic auth adalah valid, maka dilanjutkan ke proses otentikasi, mengecek apakah username dan password yang dikirim cocok dengan username dan password yang ada di aplikasi kita.

HTTP Basic Auth- Sesi 9

Function Auth & AllowOnlyGet

Function AllowOnlyGet

Function ini bertugas untuk memastikan bahwa request yang diperbolehkan hanya yang ber-method GET. Selainnya, maka akan dianggap invalid request.

```
func AllowOnlyGET(w http.ResponseWriter, r *http.Request) bool {  
    if r.Method != "GET" {  
        w.Write([]byte("Only GET is allowed"))  
        return false  
    }  
  
    return true  
}
```



HTTP Basic Auth- Sesi 9

Testing

Semuanya sudah siap, jalankan aplikasi.




Jangan menggunakan perintah `go run main.go`, dikarenakan dalam package main terdapat beberapa file lain yang harus di-ikut-sertakan pada saat runtime.

HTTP Basic Auth- Sesi 9

Testing

Test aplikasi kita kali ini menggunakan command curl.



```
$ curl -X GET --user batman:secret http://localhost:9000/student
$ curl -X GET --user batman:secret http://localhost:9000/student?id=s001
```

