
sessions



Gin middleware for session management with multi-backend support:

- cookie-based
- Redis
- memcached
- MongoDB
- GORM
- memstore
- PostgreSQL

Usage

Start using it

Download and install it:

```
1 go get github.com/gin-contrib/sessions
```

Import it in your code:

```
1 import "github.com/gin-contrib/sessions"
```

Basic Examples

single session

```
1 package main
2
3 import (
4     "github.com/gin-contrib/sessions"
5     "github.com/gin-contrib/sessions/cookie"
6     "github.com/gin-gonic/gin"
7 )
8
9 func main() {
10     r := gin.Default()
11     store := cookie.NewStore([]byte("secret"))
12     r.Use(sessions.Sessions("mysession", store))
13 }
```

```
14  r.GET("/hello", func(c *gin.Context) {
15      session := sessions.Default(c)
16
17      if session.Get("hello") != "world" {
18          session.Set("hello", "world")
19          session.Save()
20      }
21
22      c.JSON(200, gin.H{"hello": session.Get("hello")})
23  })
24  r.Run(":8000")
25 }
```

multiple sessions

```
1  package main
2
3  import (
4      "github.com/gin-contrib/sessions"
5      "github.com/gin-contrib/sessions/cookie"
6      "github.com/gin-gonic/gin"
7  )
8
9  func main() {
10     r := gin.Default()
11     store := cookie.NewStore([]byte("secret"))
12     sessionNames := []string{"a", "b"}
13     r.Use(sessions.SessionsMany(sessionNames, store))
14
15     r.GET("/hello", func(c *gin.Context) {
16         sessionA := sessions.DefaultMany(c, "a")
17         sessionB := sessions.DefaultMany(c, "b")
18
19         if sessionA.Get("hello") != "world!" {
20             sessionA.Set("hello", "world!")
21             sessionA.Save()
22         }
23
24         if sessionB.Get("hello") != "world?" {
25             sessionB.Set("hello", "world?")
26             sessionB.Save()
27         }
28
29         c.JSON(200, gin.H{
30             "a": sessionA.Get("hello"),
31             "b": sessionB.Get("hello"),
32         })
33     })
34     r.Run(":8000")
}
```

```
35 }
```

Backend Examples

cookie-based

```
1 package main
2
3 import (
4     "github.com/gin-contrib/sessions"
5     "github.com/gin-contrib/sessions/cookie"
6     "github.com/gin-gonic/gin"
7 )
8
9 func main() {
10     r := gin.Default()
11     store := cookie.NewStore([]byte("secret"))
12     r.Use(sessions.Sessions("mysession", store))
13
14     r.GET("/incr", func(c *gin.Context) {
15         session := sessions.Default(c)
16         var count int
17         v := session.Get("count")
18         if v == nil {
19             count = 0
20         } else {
21             count = v.(int)
22             count++
23         }
24         session.Set("count", count)
25         session.Save()
26         c.JSON(200, gin.H{"count": count})
27     })
28     r.Run(":8000")
29 }
```

Redis

```
1 package main
2
3 import (
4     "github.com/gin-contrib/sessions"
5     "github.com/gin-contrib/sessions/redis"
6     "github.com/gin-gonic/gin"
7 )
8
```

```

 9 func main() {
10     r := gin.Default()
11     store, _ := redis.NewStore(10, "tcp", "localhost:6379", "", []byte("
    secret"))
12     r.Use(sessions.Sessions("mysession", store))
13
14     r.GET("/incr", func(c *gin.Context) {
15         session := sessions.Default(c)
16         var count int
17         v := session.Get("count")
18         if v == nil {
19             count = 0
20         } else {
21             count = v.(int)
22             count++
23         }
24         session.Set("count", count)
25         session.Save()
26         c.JSON(200, gin.H{"count": count})
27     })
28     r.Run(":8000")
29 }
```

Memcached

ASCII Protocol

```

1 package main
2
3 import (
4     "github.com/bradfitz/gomemcache/memcache"
5     "github.com/gin-contrib/sessions"
6     "github.com/gin-contrib/sessions/memcached"
7     "github.com/gin-gonic/gin"
8 )
9
10 func main() {
11     r := gin.Default()
12     store := memcached.NewStore(memcache.New("localhost:11211"), "", []
        byte("secret"))
13     r.Use(sessions.Sessions("mysession", store))
14
15     r.GET("/incr", func(c *gin.Context) {
16         session := sessions.Default(c)
17         var count int
18         v := session.Get("count")
19         if v == nil {
20             count = 0
21         } else {
22             count = v.(int)
23             count++

```

```

24     }
25     session.Set("count", count)
26     session.Save()
27     c.JSON(200, gin.H{"count": count})
28 })
29 r.Run(":8000")
30 }

```

Binary protocol (with optional SASL authentication)

```

1 package main
2
3 import (
4     "github.com/gin-contrib/sessions"
5     "github.com/gin-contrib/sessions/memcached"
6     "github.com/gin-gonic/gin"
7     "github.com/memcachier/mc"
8 )
9
10 func main() {
11     r := gin.Default()
12     client := mc.NewMC("localhost:11211", "username", "password")
13     store := memcached.NewMemcacheStore(client, "", []byte("secret"))
14     r.Use(sessions.Sessions("mysession", store))
15
16     r.GET("/incr", func(c *gin.Context) {
17         session := sessions.Default(c)
18         var count int
19         v := session.Get("count")
20         if v == nil {
21             count = 0
22         } else {
23             count = v.(int)
24             count++
25         }
26         session.Set("count", count)
27         session.Save()
28         c.JSON(200, gin.H{"count": count})
29     })
30     r.Run(":8000")
31 }

```

MongoDB

```

1 package main
2
3 import (
4     "github.com/gin-contrib/sessions"
5     "github.com/gin-contrib/sessions/mongo/mongomgo"

```

```

6  "github.com/gin-gonic/gin"
7  "github.com/globalsign/mgo"
8  )
9
10 func main() {
11     r := gin.Default()
12     session, err := mgo.Dial("localhost:27017/test")
13     if err != nil {
14         // handle err
15     }
16
17     c := session.DB("").C("sessions")
18     store := mongomgo.NewStore(c, 3600, true, []byte("secret"))
19     r.Use(sessions.Sessions("mysession", store))
20
21     r.GET("/incr", func(c *gin.Context) {
22         session := sessions.Default(c)
23         var count int
24         v := session.Get("count")
25         if v == nil {
26             count = 0
27         } else {
28             count = v.(int)
29             count++
30         }
31         session.Set("count", count)
32         session.Save()
33         c.JSON(200, gin.H{"count": count})
34     })
35     r.Run(":8000")
36 }

```

mongo-driver

```

1  package main
2
3  import (
4      "context"
5      "github.com/gin-contrib/sessions"
6      "github.com/gin-contrib/sessions/mongo/mongodriver"
7      "github.com/gin-gonic/gin"
8      "go.mongodb.org/mongo-driver/mongo"
9      "go.mongodb.org/mongo-driver/mongo/options"
10 )
11
12 func main() {
13     r := gin.Default()
14     mongoOptions := options.Client().ApplyURI("mongodb://localhost:27017")
15     client, err := mongo.NewClient(mongoOptions)
16     if err != nil {

```

```

17     // handle err
18 }
19
20 if err := client.Connect(context.Background()); err != nil {
21     // handle err
22 }
23
24 c := client.Database("test").Collection("sessions")
25 store := mongodriver.NewStore(c, 3600, true, []byte("secret"))
26 r.Use(sessions.Sessions("mysession", store))
27
28 r.GET("/incr", func(c *gin.Context) {
29     session := sessions.Default(c)
30     var count int
31     v := session.Get("count")
32     if v == nil {
33         count = 0
34     } else {
35         count = v.(int)
36         count++
37     }
38     session.Set("count", count)
39     session.Save()
40     c.JSON(200, gin.H{"count": count})
41 })
42 r.Run(":8000")
43 }
```

memstore

```

1 package main
2
3 import (
4     "github.com/gin-contrib/sessions"
5     "github.com/gin-contrib/sessions/memstore"
6     "github.com/gin-gonic/gin"
7 )
8
9 func main() {
10     r := gin.Default()
11     store := memstore.NewStore([]byte("secret"))
12     r.Use(sessions.Sessions("mysession", store))
13
14     r.GET("/incr", func(c *gin.Context) {
15         session := sessions.Default(c)
16         var count int
17         v := session.Get("count")
18         if v == nil {
19             count = 0

```

```
20     } else {
21         count = v.(int)
22         count++
23     }
24     session.Set("count", count)
25     session.Save()
26     c.JSON(200, gin.H{"count": count})
27 })
28 r.Run(":8000")
29 }
```

GORM

```
1 package main
2
3 import (
4     "github.com/gin-contrib/sessions"
5     gormsessions "github.com/gin-contrib/sessions/gorm"
6     "github.com/gin-gonic/gin"
7     "gorm.io/driver/sqlite"
8     "gorm.io/gorm"
9 )
10
11 func main() {
12     db, err := gorm.Open(sqlite.Open("test.db"), &gorm.Config{})
13     if err != nil {
14         panic(err)
15     }
16     store := gormsessions.NewStore(db, true, []byte("secret"))
17
18     r := gin.Default()
19     r.Use(sessions.Sessions("mysession", store))
20
21     r.GET("/incr", func(c *gin.Context) {
22         session := sessions.Default(c)
23         var count int
24         v := session.Get("count")
25         if v == nil {
26             count = 0
27         } else {
28             count = v.(int)
29             count++
30         }
31         session.Set("count", count)
32         session.Save()
33         c.JSON(200, gin.H{"count": count})
34     })
35     r.Run(":8000")
36 }
```

PostgreSQL

```
1 package main
2
3 import (
4     "database/sql"
5     "github.com/gin-contrib/sessions"
6     "github.com/gin-contrib/sessions/postgres"
7     "github.com/gin-gonic/gin"
8 )
9
10 func main() {
11     r := gin.Default()
12     db, err := sql.Open("postgres", "postgresql://username:
        password@localhost:5432/database")
13     if err != nil {
14         // handle err
15     }
16
17     store, err := postgres.NewStore(db, []byte("secret"))
18     if err != nil {
19         // handle err
20     }
21
22     r.Use(sessions.Sessions("mysession", store))
23
24     r.GET("/incr", func(c *gin.Context) {
25         session := sessions.Default(c)
26         var count int
27         v := session.Get("count")
28         if v == nil {
29             count = 0
30         } else {
31             count = v.(int)
32             count++
33         }
34         session.Set("count", count)
35         session.Save()
36         c.JSON(200, gin.H{"count": count})
37     })
38     r.Run(":8000")
39 }
```