Gin Cheat Sheet

Routing · Middleware · Binding · JSON

SETUP & ROUTER

```
go get github.com/gin-gonic/gin
r := gin.Default()  // includes
Logger+Recovery
r.Run(":8080")      // listen and serve
r := gin.New()      // bare router (no
middleware)
gin.SetMode(gin.ReleaseMode) //
production
```

ROUTES & HTTP METHODS

```
r.GET("/users", listUsers)
r.POST("/users", createUser)
r.PUT("/users/:id", updateUser)
r.DELETE("/users/:id", deleteUser)
r.PATCH  r.HEAD  r.OPTIONS  r.Any
r.NoRoute(func(c *gin.Context){
c.JSON(404, gin.H{"error":"not found"})
})
```

ROUTE PARAMS & QUERY

```
r.GET("/users/:id", func(c *gin.Context)
{
id := c.Param("id")
page := c.DefaultQuery("page", "1")
search := c.Query("q")
c.JSON(200, gin.H{"id": id, "page":
page})
})
c.QueryArray("ids")  // repeated
?ids=1&ids=2
```

MIDDLEWARE

```
// Global:
r.Use(gin.Logger(), gin.Recovery())
r.Use(CorsMiddleware())
// Group-level:
api := r.Group("/api", AuthMiddleware())
// Custom:
func AuthMiddleware() gin.HandlerFunc {
return func(c *gin.Context) {
if !isValid(c) { c.AbortWithStatus(401);
return }
c.Next()
}
}
```

REQUEST BINDING

```
// Bind JSON body:
var req CreateUserReq
if err := c.ShouldBindJSON(&req); err !=
nil {
c.JSON(400, gin.H{"error": err.Error()})
return
}
// Bind query params:
c.ShouldBindQuery(&params)
// Bind form:
c.ShouldBind(&form)
// With validation tags:
`binding:"required,email"`
```

RESPONSE RENDERING

```
c.JSON(200, gin.H{"users": users})
c.JSON(http.StatusCreated, newUser)
c.String(200, "Hello %s", name)
c.XML(200, xmlData)
c.File("./static/index.html")
c.Stream(200, func(w io.Writer) bool {
... })
c.Status(204)  // no body
```

GROUPING ROUTES

```
v1 := r.Group("/v1")
{
v1.GET("/users", ...)   // /v1/users
v1.POST("/users", ...)
admin := v1.Group("/admin", AdminOnly())
admin.GET("/stats", ...)  //
/v1/admin/stats
}
```

ERROR HANDLING

```
c.Error(err)  // attach error to context
c.AbortWithStatusJSON(401,
gin.H{"error": "unauthorized"})
c.AbortWithStatus(403)
// Global error handler:
r.Use(func(c *gin.Context) {
c.Next()
for _, e := range c.Errors {
log.Error(e) }
})
```

VALIDATION

```
type CreateUser struct {
Name  string `json:"name"
binding:"required,min=2,max=50"`
Email string `json:"email"
binding:"required,email"`
Age   int    `json:"age"
binding:"gte=0,lte=130"`
}
// Custom validator:
v, _ :=
binding.Validator.Engine().(*validator.Validate)
v.RegisterValidation("isbn",
validateISBN)
```

COMMON CONTEXT METHODS

```
c.Set("user", user)       // set value
in context
c.Get("user")             // retrieve
(returns any, bool)
c.MustGet("user")         // panics if
not found
c.Request                  //
*http.Request
c.Writer                   //
http.ResponseWriter
c.ClientIP()               // client IP
c.GetHeader("Authorization")
```

COMMON GOTCHAS

c.JSON() after c.Abort() still writes always return after Abort
Bind() calls c.Abort on failure; ShouldBind() doesn't
gin.H is just map[string]any
ReleaseMode silences debug logs enable in production
Goroutines in handlers: copy context values before starting