Echo Cheat Sheet

Routing · Middleware · Binding · Groups

SETUP & INSTANCE

```
go get github.com/labstack/echo/v4
e := echo.New()
e.Use(middleware.Logger())
e.Use(middleware.Recover())
e.Start(":8080")
e.Logger.SetLevel(log.DEBUG)
```

ROUTING

```
e.GET("/users", listUsers)
e.POST("/users", createUser)
e.PUT("/users/:id", updateUser)
e.DELETE("/users/:id", deleteUser)
e.Static("/static", "public")
e.File("/favicon.ico",
"public/favicon.ico")
```

MIDDLEWARE

```
e.Use(middleware.Logger())
e.Use(middleware.Recover())
e.Use(middleware.CORS())
e.Use(middleware.RateLimiter(...))
// Custom:
e.Use(func(next echo.HandlerFunc)
echo.HandlerFunc {
return func(c echo.Context) error {
// before
return next(c)
}
})
```

REQUEST & RESPONSE

```
c.Param("id")           // route param
c.QueryParam("page")    // query string
c.FormValue("name")     // form field
return c.JSON(200, user)
return c.String(200, "Hello")
return c.XML(200, xmlData)
return c.File("index.html")
return c.Redirect(301, "/new")
```

BINDING & VALIDATION

```
type CreateUserReq struct {
Name  string `json:"name"
validate:"required"`
Email string `json:"email"
validate:"required,email"`
}
var req CreateUserReq
if err := c.Bind(&req); err != nil {
return err }
if err := c.Validate(&req); err != nil {
return err }
e.Validator = &CustomValidator{v:
validator.New()}
```

GROUPING

```
g := e.Group("/api/v1")
g.Use(JWTMiddleware())
g.GET("/users", listUsers)    //
/api/v1/users
admin := g.Group("/admin")
admin.Use(AdminOnly())
admin.GET("/stats", getStats) //
/api/v1/admin/stats
```

STATIC FILES

```
e.Static("/static", "assets")
e.StaticFS("/static",
echo.MustSubFS(os.DirFS("."), "assets"))
// Serve index.html for SPA:
e.GET("/*", func(c echo.Context) error {
return c.File("public/index.html")
})
```

ERROR HANDLING

```
e.HTTPErrorHandler = func(err error, c
echo.Context) {
code := http.StatusInternalServerError
if he, ok := err.(*echo.HTTPError); ok {
code = he.Code }
c.JSON(code, map[string]string{"error":
err.Error()})
}
return
echo.NewHTTPError(http.StatusBadRequest,
"invalid input")
```

COMMON CONTEXT METHODS

```
c.Set("user", user)   // store in
context
c.Get("user")        // retrieve value
c.Request()           // *http.Request
c.Response()         // *echo.Response
c.RealIP()           // client IP
respecting X-Real-IP
c.Logger()           // scoped logger
c.IsWebSocket()      // check WebSocket
upgrade
```

COMMON GOTCHAS

Return errors from handlers don't just log and continue
Validator must be set on echo instance before validation
echo.HTTPError message is returned to client don't expose internals
Middleware order matters Logger before Recover
Use c.Set/c.Get for request-scoped data, not global vars