Echo - Practice Sheet

10 Q&A on routing, middleware, binding, JWT auth, and testing

Q1. What is Echo and how does it compare to Gin?

A: Both use radix-tree routing. Echo provides built-in middleware (Logger, Recover, CORS, JWT), automatic TLS via Let's Encrypt, and HTTP/2 support out of the box. Its echo.Context API is arguably more ergonomic with typed header/cookie helpers.

Q2. How do you apply middleware globally vs to a route group in Echo?

A: e.Use(middleware.Logger()) applies globally to all routes. g := e.Group('/admin', AdminMiddleware) applies to the group only. Individual routes: e.GET('/profile', handler, AuthMiddleware). Middleware functions have signature func(next echo.HandlerFunc) echo.HandlerFunc.

Q3. How does request binding and validation work in Echo?

A: c.Bind(&req) reads JSON body, form values, or query params into a struct based on tags. Echo does not validate by default. Register a validator (e.Validator = &MyValidator{}) that implements Validate(i interface{}) error and call c.Validate(&req) explicitly.

Q4. How do Echo Groups help organize routes?

A: g := e.Group('/api/v1') prefixes all routes in the group. Apply shared middleware once: g.Use(JWTMiddleware). Create sub-groups: admin := g.Group('/admin', AdminOnly). This mirrors Express.Router and NestJS Module organization patterns.

Q5. How does Echo's custom error handler work?

A: Set e.HTTPErrorHandler = func(err error, c echo.Context). Check if err is *echo.HTTPError for status/message; otherwise default to 500. This centralizes error formatting, allowing you to return consistent JSON error shapes across the entire application.

Q6. What does echo.Context provide for reading request data?

A: c.Param('id') reads path params, c.QueryParam('q') reads query strings, c.FormValue('field') reads form fields, c.Request().Header.Get('X-Api-Key') reads headers, and c.Get/Set passes values through middleware to handlers via the request context.

Q7. How do you implement JWT authentication in Echo?

A: Register echojwt.WithConfig(echojwt.Config{SigningKey: []byte(secret)}) as middleware on a Group. It reads the Bearer token, validates the signature, and populates the jwt.Token in the context. Retrieve claims with c.Get('user').(*jwt.Token).Claims.

Q8. What are Echo's response helpers and what do they return?

A: c.JSON(200, obj) marshals obj and sets Content-Type application/json. c.XML, c.String, c.HTML, c.File, c.Stream cover other content types. c.NoContent(204) sends an empty response. c.Redirect(301, url) sets Location header and status.

Q9. How do you handle WebSocket connections in Echo?

A: Use github.com/gorilla/websocket inside an Echo handler. Register a standard echo.HandlerFunc, upgrade the connection with upgrader.Upgrade(c.Response(), c.Request(), nil), then loop on conn.ReadMessage() and conn.WriteMessage() for full-duplex

Q10. How do you test Echo handlers without starting an HTTP server?

A: req := httptest.NewRequest(http.MethodGet, '/users/1', nil); rec := httptest.NewRecorder(); c := e.NewContext(req, rec); handler(c). Access rec.Code and rec.Body.String() for assertions. No listening port is required.