Chi - Practice Sheet

10 Q&A on lightweight routing, middleware, context, and net/http

Q1. What is Chi and why is it described as idiomatic Go?

A: Chi is a lightweight router that implements http.Handler and uses only standard net/http no custom context types. Middleware is standard func(http.Handler) http.Handler. Any existing net/http handler works out of the box without adaptation.

Q2. How does Chi middleware chaining work?

A: Chi uses alice-style functional composition. r.Use(middleware.Logger) applies globally. r.Group(func(r chi.Router) { r.Use(AuthMiddleware); r.Get('/admin', handler) }) scopes middleware to the group. Each middleware wraps the next http.Handler.

Q3. How do you read URL parameters in Chi?

A: chi.URLParam(r, 'id') reads the named parameter from the URL. In a handler: id := chi.URLParam(r, 'id'). Chi stores params in the request context under its own key, making them accessible without a custom Context type via the standard library.

Q4. What is chi.Mount() and how do you use it?

A: r.Mount('/api', apiRouter) attaches a sub-router at a prefix. apiRouter is itself a *chi.Mux. Mount strips the prefix before passing the request to the sub-router. This enables fully independent route trees for modular codebases.

Q5. How do you pass values between middleware and handlers in Chi?

A: Use standard context: ctx := context.WithValue(r.Context(), userKey{}, user); r = r.WithContext(ctx). Retrieve in the handler: user := r.Context().Value(userKey{}).(User). Chi recommends unexported struct keys to avoid collisions with other packages.

Q6. How do you implement JWT authentication middleware in Chi?

A: Write a func(next http.Handler) http.Handler that reads the Authorization header, validates the JWT, and on success calls next.ServeHTTP(w, r.WithContext(ctx)) with the user in the context. On failure, write 401 and return without calling next.

Q7. What built-in middleware does Chi provide?

A: middleware.Logger (structured request logging), middleware.Recoverer (panic recovery), middleware.Throttle(N) (concurrent request limiter), middleware.Compress (gzip), middleware.StripSlashes, middleware.Heartbeat('/health'), and middleware.RealIP.

Q8. How do you define RESTful resource routes in Chi?

A: r.Get('/users', listUsers); r.Post('/users', createUser); r.Route('/users/{id}', func(r chi.Router) { r.Get('/', getUser); r.Put('/', updateUser); r.Delete('/', deleteUser) }). Chi does not have a resources() shortcut explicit routes are idiomatic.

Q9. How does Chi handle 404 and 405 responses?

A: r.NotFound(handler) sets a custom 404 handler for unmatched routes. r.MethodNotAllowed(handler) sets a 405 handler when a route exists but not for the used HTTP method. Without custom handlers, Chi returns plain text 404/405 by default.

Q10. How do you write tests for Chi handlers?

A: Chi handlers are plain http.HandlerFunc, so use httptest.NewRequest and httptest.NewRecorder directly. Set chi URL params with chitest: rctx := chi.NewRouteContext(); rctx.URLParams.Add('id', '42'); r = r.WithContext(context.WithValue(r.Context(), chi.RouteCtxKey, rctx)).