

Go Language Foundations

Core programming and a small web-service project

FA-0622 · 3 days

Software Development

Learn Go fundamentals and extend a small HTTP service with tests, data access and introductory operational checks.

Introduction

This three-day course introduces Go to participants with prior programming experience. It connects syntax, types, composition, error handling and modules with concurrency and standard-library web development.

Participants build a small service using a prepared project and a limited data example. Persistence, observability and packaging are introductory; the workshop is not a guarantee of scalable or production-ready service design.

Learning outcomes

The course teaches participants to:

- Write small Go programs using core syntax, structs, methods and interfaces.
- Manage a module and use the build, test and formatting toolchain.
- Handle errors and test selected functions.
- Use basic goroutine, channel and cancellation patterns and identify shared-state risks.
- Implement simple HTTP handlers, JSON validation and error responses.
- Integrate a small persistence example and manage configuration appropriately.
- Package the service and describe further security, monitoring and deployment work.

Prerequisites

- Prior programming experience (in any language): variables, loops, functions/methods.
- Familiarity with using an IDE or text editor and basic command line (terminal).
- Understanding of HTTP / web concepts (requests/responses) is helpful but not strictly required.

A computer with a currently supported Go toolchain and compatible editor or IDE, prepared before the course. Use the official release policy and selected module requirements rather than treating Go 1.24 as the latest release.

- Ready to code hands-on and engage in real-world style examples.

Training outline

1. Day 1 – Language, Types and Tooling

Introduction & Setup

- Course overview, objectives, and expectations
- Install the selected supported toolchain and use a module-based project; explain GOPATH only as relevant background rather than the required project layout.
- Go toolchain: go build, go run, go test, go mod
- Project structure and version control basics for Go

Go Language Fundamentals

- Basic syntax: packages, imports, functions, variables, constants
- Data types, zero values, composite types (arrays, slices, maps)
- Control flow: if, for, switch, goto (rarely), and defer
- Functions: multiple return values, named returns, variadic parameters

Go Types, Structures & Composition

- Structs, methods, interfaces, embedding (composition)
- No traditional inheritance; using interface-based design and composition
- Pointers, copying values and how slices/maps share underlying data; Go function arguments are passed by value.
- Go's approach to OOP-like patterns and idiomatic Go style

Error Handling, Testing and Modules

- Error handling in Go: error as value, idiomatic patterns
- Using defer, panic, recover carefully
- Writing tests: testing package, benchmarks and examples
- Go modules: go.mod, semantic versioning, module-aware builds

2. Day 2 — Concurrency and HTTP Services

Standard Library Highlights & Concurrency

- Key packages: fmt, io, os, net/http, encoding/json — and how to use them effectively.
- Concurrency with goroutines and channels: creating goroutines, channel patterns, select statement.
- Synchronisation primitives: sync, atomic, context cancellation.
- Measure performance on a small example rather than assuming faster compilation or runtime behaviour guarantees throughput.

Modern Go Features & Best Practices

- Review selected release-appropriate features, including generics and type-alias examples where supported; do not describe old releases as the latest.
- Code structure: clean package design, separation of concerns, dependency injection-like patterns in Go.
- Tooling: go vet, go fmt/gofmt, tests, introductory profiling/benchmarks and dependency maintenance. The deprecated golint is not part of the recommended setup.
- Logging, configuration, environment variables, CI/CD basics in Go services.

Building Web Services in Go

- HTTP handlers and routing with net/http; compare an optional maintained router only where needed.
- Middleware patterns: logging, authentication, request metrics
- JSON payloads: encoding/decoding, validation, error responses
- RESTful service design: endpoints, status codes, versioning
- Configuration from environment or files; distinguish configuration from request-scoped context values and keep secrets out of source.

3. Day 3 — Persistence, Packaging and Operational

Awareness

Data Persistence & Integration

- Connecting to databases: basic use of database/sql, or an ORM/mapper package
- Struct tags for JSON/DB mapping, entity design
- Introductory transactions and connection-pool behaviour in the small training example.
- External integrations: calling external APIs, background tasks/goroutines

Packaging and Deployment Considerations

- Build for a selected OS/architecture and discuss cross-compilation limitations, including native/cgo dependencies.
- Review logging, metrics and tracing concepts using selected examples.
- A prepared containerisation example and microservice considerations, not a full production deployment.
- Error handling, graceful shutdowns, signal handling
- Input validation, output handling, TLS and secret-management awareness; no claim that a short checklist proves service security.