

This page links to resources for learning about server programming in Go - both web services and mobile backends. The items are organized into sections by topic.

## Getting Started

- Read [Writing Web Applications with the Go standard library](#).
- Read [Build a Web Application With Go](#) from the author of the [BeeGo web framework](#)
- Read [Webapps in Go the anti textbook](#)
- Read [Building Web Applications in Go](#) from the author of the [Negroni](#) and [Martini](#) webserver toolkits. First learn the absolute basics before going to this book.
- Read [Building Your Own Web Framework in Go](#) a 5-part series.
- Watch [Go: code that grows with grace](#)
- Download a [full working 3-tier application example](#) from the Sourcegraph Team.

## Middleware

A topic you will see discussed frequently is "middleware". If you're not familiar with that term, we suggest you start out by reading a few of these articles:

- [Middleware in Go: Best practices and examples](#) 2014-11-13
- Custom Handlers [Part 1 - Avoiding Globals](#), [Part 2 - Error Handling](#) 2014-07-16
- [Making and Using HTTP Middleware](#) 2014-10-21
- [Writing HTTP Middleware in Go](#) 2013-10-09

## Toolkits and Frameworks

Before you decide to adopt a third party web framework or toolkit, keep in mind that the Go standard library provides all of the tools you need to build a sophisticated, modern web application. Keeping with Go's preference for simplicity and composability over complexity and magic, we suggest you [see how far the standard library can take you](#).

If you decide you need a bit more infrastructure, start by looking at some of the toolkits and libraries available.

### Toolkits & Libraries & Microframeworks

- [Gorilla Toolkit](#)
- [Negroni Toolkit - Idiomatic HTTP Middleware for Go](#)
- [Echo Framework - Fast and Unfancy](#)
- [Goji Web Microframework](#)
- [Go Craft Middleware](#)
- [Go RESTful](#) - Toolkit for RESTful service APIs
- [limiter](#) - Simple rate-limiting middleware for Go
- [Kite Micro-service framework](#)
- [Alice - Painless middleware chaining for Go](#)
- [YAM - Yet Another Mux](#)
- [Bone - Fast HTTP Router](#)

### Frameworks

- [BeeGo Framework](#)
- [Frodo](#) - Go mini web framework inspired by Laravel(PHP), Slim(PHP) and ExpressJS(Node.js)
- [GinGonic](#)
- [Macaron](#) - Productive, modular web framework in Go.
- [Revel Web Framework](#)

- [Ringo](#) - Lightweight MVC web framework inspired by Rails, Gin.
- [Utron](#) - Lightweight MVC framework for web applications.
- [Iris](#) - Fast MVC framework for web applications.

## Communication

- [Package net/http](#) provides HTTP client and server implementations.
- [Package encoding/json](#) implements encoding and decoding of JSON objects as defined in RFC 4627.
- [Package net/rpc](#) provides access to the exported methods of an object across a network or other I/O connection.
- [Package os/exec](#) runs external commands.

## Presentation

- [Package text/template](#) implements data-driven templates for generating textual output.
- [Package html/template](#) implements data-driven templates for generating HTML output safe against code injection.

## Profiling and Performance

- Read [Profiling Go Programs](#)
- Read [Arrays, slices \(and strings\): The mechanics of 'append'](#)
- Read the [Frequently Asked Questions \(FAQ\)](#), especially
  - [Why does Go perform badly on benchmark X?](#)
  - [Why do garbage collection? Won't it be too expensive?](#)
- [Package bufio](#) implements buffered I/O.
- [Package runtime/pprof](#) writes runtime profiling data in the format expected by the pprof visualization tool.
- [Package net/http/pprof](#) serves via its HTTP server runtime profiling data in the format expected by the pprof visualization tool.

## Tracing, Monitoring, Logging, and Configuration

- [Package expvar](#) provides a standardized interface to public variables, such as operation counters in servers.
- [Package flag](#) implements command-line flag parsing.
- [Package log](#) implements a simple logging package.
- [Package glog](#) implements logging analogous to the Google-internal C++ INFO/ERROR/V setup.

## Storage

- [Package os](#) provides a platform-independent interface to operating system functionality.
- [Package path/filepath](#) implements utility routines for manipulating filename paths in a way compatible with the target operating system-defined file paths.
- [Package database/sql](#) provides a generic interface around SQL (or SQL-like) databases.

## Platforms

### Google Cloud Platform

- Read [Go, Cloud Endpoints and App Engine, Part 1, Part 2](#)
- Read [Google Cloud Platform: Go Runtime Environment](#)
- Watch [Go and the Google Cloud Platform](#)
- Read [Go on App Engine: tools, tests, and concurrency](#)
- Get [Google Cloud Platform Go Libraries](#)

- Read [Deploying Go servers with Docker](#)
- Search packages for [Google Cloud](#) or [gcloud](#)
- Search packages for [App Engine](#) or [GAE](#)

### **Amazon Web Services**

- The [aws-sdk-go](#) repository provides automatically generated AWS clients in Go. It has [official support](#) from Amazon.
- [Package goamz](#) enables Go programs to interact with the Amazon Web Services.
- Search packages for [AWS](#) or [Amazon services](#)

### **Microsoft Azure**

- Microsoft OpenTech's [azure-sdk-for-go](#) provides a Golang package that makes it easy to consume and manage Microsoft Azure Services.
- Search packages for [Azure](#)

### **Openstack / Rackspace**

- [Gophercloud](#) is a Golang SDK for working with OpenStack clouds.
- Search packages for [Openstack](#) or [Rackspace](#)

### **IBM BlueMix**

- [Write your first Golang app on BlueMix](#)