protechtraining.com

"Charting the Course ...

... to Your Success!"

Concurrent Programming in Go

Course Summary

Description

Concurrency in one of the core features of the Go Language; however those who have only programmed in languages where concurrency is managed through thread manipulation or other techniques are generally used to thinking in terms of developing an application and then adding in concurrency constructs. It takes a shift in thinking to design and write applications that are concurrent from the ground up. Concurrent applications in Go tend to be more efficient, simpler and scalable as solutions for classes of problems in today's highly concurrent world.

This course starts by exploring the Go concurrency model in depth with detailed, example-driven analyses of goroutines, channels, the select statement and other features available in the Go sync and related libraries. More importantly, the correct use and best practices for using these constructs is covered in depth, including synchronization of goroutines with channels, recovering from error conditions and working with chains of goroutine dependencies and more.

The course continues with an exploration of Go concurrency patterns like multiplex/de-multiplex, fan-in/fan-out, pipe-lining, channel generators, daisy chains, subscriber/publisher and others. These concurrency patterns and concepts are then used to show students how to design a concurrent application from the ground up beginning with a concurrent design model, then implementing the model in Go and testing the final result.

The class is designed to be about 50% hands on labs and exercises, about 25% theory and 25% instructor led hands on learning where students code along with the instructor.

Topics

- The Go concurrency CPS model.
- The goroutine in detail, usage, costs and benefits – how it is implemented.
- Channels – properties, types, buffering, how they work under the hood.
- Channels as first class objects.
- Advanced, effective and efficient select statement usage.
- Quit channels, blocking, nil channels, timeouts.
- Avoiding race problem and other classic concurrency issues by using channels.
- Goroutine synchronization with channels – blocking and listening channels.
- Cleaning up concurrent applications, handling abnormal conditions, error channels.
- Modeling a concurrent application, visual task, data and work product flows.
- Goroutine channel generators.
- Delegation and integration of work flow – fan-in/fan-out and multiplexing/de-multiplexing.
- Pipelining, filtering, daisy chaining and other concurrent patterns.
- Managing multiple channels using maps, structs and interfaces.
- Best practices for concurrent application and goroutine design.
- Dealing with legacy shared resources (eg. memory management and CGO)
- Best practices for handing deadlocks and errors.
- Working with the sync library – WaitGroups, mutexes, etc.
- Testing and debugging concurrent applications.
- Best practices and common pitfalls to avoid

Concurrent Programming in Go

Course Summary (cont'd)

Audience

This course is intended for intermediate level Go programmers who can comfortably write code using at a level consistent with completing ProTech's "Introduction to Go Programming for Developers" (PT20182) course.

Prerequisites

This course is intended for intermediate level Go programmers who can comfortably write code using at a level consistent with completing ProTech's "Introduction to Go Programming for Developers" (PT20182) course. Students who do not have this level of programming competence will not be able to follow the material. Because of time limitations, there will be no opportunity for remedial instruction in Go during the class. This is an essential prerequisite.

Duration

Two days