Golang

Код: DEV-040

Длительность: 32 ч.

Описание:

Данный курс прежде всего ориентирован на разработчиков с опытом программирования на других языках, поэтому в процессе обучения особое внимание уделяется специфическим возможностям и отличиям Go. Детально и с применением практических примеров рассматривается внутренняя реализация встроенных контейнеров и особенности работы управляющих конструкций.

Язык программирования Go имеет собственную реализацию объектно-ориентированного подхода, поэтому в курсе уделяется внимание декомпозиции предметной области на основе интерфейсов. Кроме того, язык Go разрабатывался с учетом эффективного использования конкурентных и параллельных вычислений. В связи с этим курс включает целый ряд примеров использования конкурентных алгоритмов. Рассмотрение каждой новой темы сопровождается решением практических задач.

Отдельно изучаются вопросы использования возможностей встроенной библиотеки, а также установка и использование пакетов сторонних разработчиков. Язык Go имеет обширный набор инструментов для поддержки процесса разработки и тестирования, изучению которых посвящена отдельная часть курса.

В курсе также рассматривается ряд практических вопросов, в частности взаимодействие с реляционными базами данных, а в конце курса рассматривается пример реализации REST-интерфейса.

Цели:

- Научиться использовать типы данных и синтаксические конструкции языка Go.
- Понять объектно-ориентированную модель языка и научиться проектировать в соответствии с ней программные компоненты.
- Понять основные сложности разработки конкурентного программного кода и способы их устранения.
- Научиться использовать встроенные инструменты языка и разрабатывать код в соответствии с ожиданиями сообщества разработчиков.
- Научиться решать практические задачи на уровне, достаточном для промышленного применения.

Разбираемые темы:

1. Environment setup (теория – 1 ч., практика – 1 ч.)
   Как установить и настроить компилятор языка Go. Пример написания, компиляции и запуска простой программы на языке Go.
2. Go IDEs (теория – 1 ч., практика – 1 ч.)
   Какие интегрированные среды разработки можно использовать для написания кода на языке Go, как их установить и настроить.
3. Go tools (теория – 1 ч., практика – 1 ч.)
   Основной инструментарий поддержки процесса разработки. Как скомпилировать и запустить программу на языке Go, как получить доступ к документации, а также добавить документацию в собственный код.
4. Directory structure. Go modules. Packages (теория – 1 ч., практика – 1 ч.)
   Как организовать структуру директории проекта так, чтобы она соответствовала общепринятым правилам. Способы декомпозиции кода на отдельные составляющие для удобства последующей поддержки и повторного использования.
5. Installing dependencies (теория – 1 ч., практика – 1 ч.)
   Использование готовых решений вместо написания собственного кода. Управление зависимостями и распределенное хранение пакетов.
6. Data types and variables declaration. Zero values. Type conversions (теория – 1 ч., практика – 3 ч.)
   Представление примитивных значений в языке и выполнение базовых операций над ними.
7. Containers: arrays, slices, maps (теория – 2 ч., практика – 2 ч.)
   Как хранить последовательности значений: массивы, срезы и хеши, их внутреннее представление и примеры использования.
8. Branching and loops. break, continue, fallthrough, goto (теория – 1 ч., практика – 3 ч.)
   Описание алгоритмов с помощью циклов и ветвлений, отличие синтаксических конструкций в Go от других языков программирования.
9. Functions. Multiple return values. Variadic parameters. Function literals and closures (теория – 2 ч., практика – 2 ч.)
   Определение функций и передача параметров. Функции с произвольным количеством аргументов. Функциональные возможности языка, анонимные функции и замыкания.
10. Types declarations. Constants, Iota. Structures. Embedded structs. Anonymous structs. Nested structs (теория – 2 ч., практика – 2 ч.)
    Способы определения пользовательских типов и констант, описание последовательностей значений. Как описать сущности предметной области при помощи конструкций языка.
11. Pointers. Value and references types (теория – 2 ч., практика – 2 ч.)
    Управление памятью и указатели.
12. Defer statement (теория – 2 ч., практика – 2 ч.)
    Оператор отложенного вызова функций.
13. Error handling. Panic, recover (теория – 2 ч., практика – 2 ч.)
    Как обнаружить и обработать ошибки времени выполнения. Способы написания надежного и отказоустойчивого кода.

14. Methods and interfaces. Method expressions (теория – 2 ч., практика – 2 ч.)
    Понятие интерфейса и его реализации. Построение объектно-ориентированного кода на основе интерфейсов. Методы.
15. Concurrency vs parallelism. Goroutines. Memory model (теория – 2 ч., практика – 2 ч.)
    Как делать несколько вещей одновременно. Организация конкурентного выполнения на основе горутин. Модель памяти в Go.
16. Channels. Blocking and unblocking channels. select statement (теория – 2 ч., практика – 2 ч.)
    Способ написания надежного конкурентного кода: обмен данными между горутинами на основе каналов.
17. Waitgroups, mutexes, atomics (теория – 2 ч., практика – 2 ч.)
    Синхронизация отдельных частей программы.
18. Reading and writing files (теория – 1 ч., практика – 3 ч.)
    Как сохранять данные между отдельными запусками программы.
19. Standard library: fmt, log, strings, time, sort, http, math, regexp (теория – 1 ч., практика – 3 ч.)
    Встроенные механизмы языка для решения повседневных задач.
20. Testing and benchmarking (теория – 3 ч., практика – 3 ч.)
    Как гарантировать стабильность программного кода с помощью модульных тестов, а также найти наиболее эффективное решение задачи.
21. JSON encoding and decoding (теория – 3 ч., практика – 3 ч.)
    Обмен данными с программами, написанными на других языках.
22. Databases. PostgreSQL. MongoDB (теория – 3 ч., практика – 3 ч.)
    Хранение данных в реляционных базах.
23. REST application example (теория – 3 ч., практика – 3 ч.)
    Пример полноценного бек-енд приложения.
24. Unsafe package (теория – 3 ч., практика – 3 ч.)
    Расширения языка для выполнения небезопасных функций.

Целевая аудитория:

- Основная:
  Разработчики с опытом программирования на других языках
- Дополнительная:
  Архитекторы, системные проектировщики, тестировщики

Предварительная подготовка – общее:

Умение писать код на каком-либо языке программирования.