



EXPERT INSIGHT

Golang для профи

Создаем профессиональные утилиты,
параллельные серверы и сервисы

Третье издание



Михалис Цукалос

Packt

Mastering Go

Third Edition

Harness the power of Go to build professional utilities
and concurrent servers and services

Mihalis Tsoukalos

Packt

BIRMINGHAM - MUMBAI

Golang для профи

Третье издание

Создаем профессиональные утилиты,
параллельные серверы и сервисы

Михалис Цукалос



Санкт-Петербург · Москва · Минск

2024

ББК 32.988.02-018.1
УДК 004.43
Ц85

Цукалос Михалис

Ц85 Golang для профи: Создаем профессиональные утилиты, параллельные серверы и сервисы. 3-е изд. — СПб.: Питер, 2024. — 624 с.: ил. — (Серия «Для профессионалов»).

ISBN 978-5-4461-1999-8

Язык Go — это простой и понятный язык для создания высокопроизводительных систем будущего. Используйте Go в реальных производственных системах. В новое издание включены такие темы, как создание серверов и клиентов RESTful, знакомство с дженериками Go и разработка серверов и клиентов gRPC.

Третье издание «Golang для профи» исследует практические возможности Go и описывает такие продвинутые темы, как параллелизм и работа сборщика мусора Go, использование Go с Docker, разработка мощных утилит командной строки, обработка данных в формате JSON (JavaScript Object Notation) и взаимодействие с базами данных. Кроме того, книга дает дополнительные сведения о работе внутренних механизмов Go, знание которых позволит оптимизировать код на Go и использовать типы и структуры данных новыми и необычными способами.

Также охватываются некоторые нюансы и идиомы языка Go, предлагаются упражнения и приводятся ссылки на ресурсы для закрепления полученных знаний.

Станьте опытным программистом на Go, создавая системы и внедряя передовые методы программирования на Go в свои проекты!

16+ (В соответствии с Федеральным законом от 29 декабря 2010 г. № 436-ФЗ.)

ББК 32.988.02-018.1
УДК 004.43

Права на издание получены по соглашению с Packt Publishing. Все права защищены. Никакая часть данной книги не может быть воспроизведена в какой бы то ни было форме без письменного разрешения владельцев авторских прав.

Информация, содержащаяся в данной книге, получена из источников, рассматриваемых издательством как надежные. Тем не менее, имея в виду возможные человеческие или технические ошибки, издательство не может гарантировать абсолютную точность и полноту приводимых сведений и не несет ответственности за возможные ошибки, связанные с использованием книги. В книге возможны упоминания организаций, деятельность которых запрещена на территории Российской Федерации, таких как Meta Platforms Inc., Facebook, Instagram и др. Издательство не несет ответственности за доступность материалов, ссылки на которые вы можете найти в этой книге. На момент подготовки книги к изданию все ссылки на интернет-ресурсы были действующими.

ISBN 978-1801079310 англ.

© Packt Publishing 2021.
First published in the English language under the title 'Mastering Go — Third Edition — (9781801079310)'

ISBN 978-5-4461-1999-8

© Перевод на русский язык ООО «Прогресс книга», 2023
© Издание на русском языке, оформление ООО «Прогресс книга», 2023
© Серия «Для профессионалов», 2023

Краткое содержание

Об авторе	17
О научном редакторе	18
Предисловие	19
Глава 1. Краткое введение в Go.....	24
Глава 2. Основные типы данных Go	68
Глава 3. Составные типы данных	118
Глава 4. Рефлексия и интерфейсы	148
Глава 5. Пакеты и функции Go.....	194
Глава 6. Даём указания системе UNIX.....	254
Глава 7. Параллельное выполнение в Go	313
Глава 8. Создание веб-сервисов	370
Глава 9. Работа с TCP/IP и WebSocket.....	423
Глава 10. Работа с REST API.....	461
Глава 11. Тестирование и профилирование кода.....	523
Глава 12. Работа с gRPC	575
Глава 13. Дженерики Go	590
Приложение. Сборщик мусора Go	607

Оглавление

Об авторе	17
О научном редакторе	18
Предисловие	19
Для кого эта книга	19
Структура издания	19
Как сделать книгу максимально полезной	21
Файлы с примерами кода	22
Цветные иллюстрации	22
Условные обозначения	22
От издательства	23
Глава 1. Краткое введение в Go	24
Введение в Go	25
История Go	26
Почему UNIX, а не Windows	27
Преимущества Go	28
Утилиты go doc и godoc	30
Hello World!	31
Введение в функции	32
Введение в пакеты	32
Запуск Go-кода	33
Компиляция Go-кода	33
Использование Go в качестве языка скриптов	33
Важные правила форматирования и кодирования	34
Важные особенности Go	35
Определение и использование переменных	36
Управление ходом выполнения программы	39
Итерации с помощью циклов for и range	41

Получение пользовательского ввода	43
Использование переменных ошибок для различения типов входных данных	48
Знакомство с моделью параллелизма в Go	49
Разработка утилиты <code>which(1)</code> на Go	51
Вывод информации в лог	54
Функции <code>log.Fatal()</code> и <code>log.Panic()</code>	56
Запись в пользовательский файл журнала	58
Вывод номеров строк в записях журнала	59
Обзор Go-дженериков	61
Разработка базового приложения телефонной книги	62
Упражнения	66
Резюме	66
Дополнительные ресурсы	67
Глава 2. Основные типы данных Go	68
Тип данных <code>error</code>	69
Числовые типы данных	72
Нечисловые типы данных	75
Строки, символы и руны	75
Время и даты	82
Go-константы	87
Генератор констант <code>iota</code>	88
Группировка схожих данных	90
Массивы	90
Срезы	91
Указатели	108
Генерация случайных чисел	112
Генерация случайных строк	113
Генерация безопасных случайных чисел	114
Обновление приложения телефонной книги	115
Упражнения	117
Резюме	117
Дополнительные ресурсы	117

Глава 3. Составные типы данных	118
Карты	119
Сохранение в карту <code>nil</code>	120
Перебор карт	122
Структуры	123
Определение новых структур	123
Использование ключевого слова <code>new</code>	124
Срезы структур	126
Регулярные выражения и сопоставление с образцом	128
О регулярных выражениях <code>Go</code>	129
Сопоставление имен и фамилий	130
Сопоставление целых чисел	131
Сопоставление полей записи	132
Улучшение приложения телефонной книги	133
Работа с CSV-файлами	133
Добавление индекса	137
Улучшенная версия приложения для телефонной книги	138
Упражнения	146
Резюме	147
Дополнительные ресурсы	147
Глава 4. Рефлексия и интерфейсы	148
Рефлексия	149
Изучение внутренней структуры <code>Go</code> -структуры	151
Изменение значений структуры с использованием рефлексии	153
Три недостатка рефлексии	154
Методы типа	155
Создание методов типа	155
Использование методов типа	156
Интерфейсы	159
Интерфейс <code>sort.Interface</code>	162
Пустой интерфейс	164
Утверждения типа и переключатели типов	166
Карта <code>map[string]interface{}</code>	170

Тип данных <code>error</code>	173
Написание собственных интерфейсов	176
Работа с двумя различными форматами файлов CSV	182
Объектно-ориентированное программирование в Go	185
Обновление приложения телефонной книги	189
Настройка значения CSV-файла	189
Использование пакета <code>sort</code>	190
Упражнения	192
Резюме	192
Дополнительные ресурсы	193
Глава 5. Пакеты и функции Go	194
Go-пакеты	195
Скачивание Go-пакетов	195
Функции	198
Анонимные функции	198
Функции, возвращающие несколько значений	199
Возвращаемые значения функции могут иметь имя	200
Функции, которые принимают другие функции в качестве параметров	201
Функции могут возвращать другие функции	202
Функции с переменным количеством параметров	204
Ключевое слово <code>defer</code>	208
Разработка собственных пакетов	210
Функция <code>init()</code>	211
Порядок исполнения	212
Использование GitHub для хранения Go-пакетов	213
Пакет для работы с базой данных	215
Знакомство с базой данных	216
Хранение Go-пакета	220
Дизайн Go-пакета	221
Реализация Go-пакета	223
Тестирование Go-пакета	230
Модули	233
Создание более качественных пакетов	234

Создание документации	235
GitLab Runners и Go	242
Начальная версия файла конфигурации	243
Окончательная версия конфигурационного файла	245
GitHub Actions и Go	246
Хранение секретов в GitHub	248
Окончательная версия конфигурационного файла	248
Утилиты управления версиями	250
Упражнения	252
Резюме	252
Дополнительные ресурсы	253
Глава 6. Даем указания системе UNIX	254
stdin, stdout и stderr	255
Процессы UNIX	256
Обработка сигналов UNIX	256
Файловый ввод-вывод	260
Интерфейсы io.Reader и io.Writer	260
Правильное и неправильное использование io.Reader и io.Writer	261
Буферизованный и небуферизованный файловый ввод-вывод	265
Чтение текстовых файлов	266
Чтение текстового файла построчно	266
Чтение текстового файла слово за словом	267
Чтение текстового файла символ за символом	269
Чтение из /dev/random	270
Считывание определенного объема данных из файла	271
Запись в файл	272
Работа с JSON	275
Использование Marshal() и Unmarshal()	275
Структуры и JSON	277
Чтение и запись данных JSON в виде потоков	278
Структурный вывод записи JSON	279
Работа с XML	281
Преобразование JSON в XML и обратно	282

Работа с YAML	283
Пакет <code>irreg</code>	285
Использование флагов командной строки	286
Чтение конфигурационных файлов JSON	289
Пакет <code>sobra</code>	292
Утилита с тремя командами	294
Добавление флагов командной строки	294
Создание псевдонимов команд	295
Создание подкоманд	296
Поиск циклов в файловой системе UNIX	297
Новое в Go 1.16	300
Встраивание файлов	300
<code>ReadDir</code> и <code>DirEntry</code>	303
Пакет <code>io/fs</code>	304
Обновление приложения телефонной книги	306
Использование <code>sobra</code>	307
Хранение и загрузка данных в формате JSON	308
Реализация команды <code>delete</code>	308
Реализация команды <code>insert</code>	309
Реализация команды <code>list</code>	309
Реализация команды <code>search</code>	310
Упражнения	311
Резюме	312
Дополнительные ресурсы	312
Глава 7. Параллельное выполнение в Go	313
Процессы, потоки и горутины	314
Планировщик Go	315
Переменная среды <code>GOMAXPROCS</code>	317
Параллелизм и распараллеливание	319
Горутины	319
Создание горутины	320
Создание нескольких горутин	321

Ожидание завершения горутин	321
Что делать, если количество вызовов <code>Add()</code> и <code>Done()</code> разное	323
Создание нескольких файлов с помощью горутин	325
Каналы	326
Запись в канал и чтение из него	326
Прием из закрытого канала	329
Каналы как параметры функций	330
Состояния гонки	331
Ключевое слово <code>select</code>	333
Установка тайм-аута горутины	335
Ограничение времени выполнения горутины — внутри <code>main()</code>	335
Ограничение времени выполнения горутины — вне <code>main()</code>	337
Еще раз о каналах в <code>Go</code>	338
Буферизованные каналы	339
<code>Nil</code> -каналы	340
Пулы рабочих процессов	342
Сигнальные каналы	345
Общая память и общие переменные	348
Тип <code>sync.Mutex</code>	349
Тип <code>sync.RWMutex</code>	351
Пакет <code>atomic</code>	354
Совместное использование памяти с помощью горутин	356
Закрытые переменные и оператор <code>go</code>	358
Пакет <code>context</code>	360
Пакет <code>semaphore</code>	365
Упражнения	368
Резюме	368
Дополнительные ресурсы	369
Глава 8. Создание веб-сервисов	370
Пакет <code>net/http</code>	371
Тип <code>http.Response</code>	371
Тип <code>http.Request</code>	372
Тип <code>http.Transport</code>	372

Создание веб-сервера	373
Обновление приложения телефонной книги	377
Определение API	377
Реализация обработчиков	378
Предоставление метрик для Prometheus	386
Пакет <code>runtime/metrics</code>	387
Предоставление метрик	389
Чтение метрик	396
Ввод метрик в Prometheus	397
Визуализация метрик Prometheus в Grafana	401
Разработка веб-клиентов	403
Использование <code>http.NewRequest()</code> для улучшения работы клиента	404
Создание клиента для сервиса телефонной книги	407
Создание файловых серверов	412
Загрузка содержимого приложения телефонной книги	414
Время ожидания HTTP-соединений	416
Использование функции <code>SetDeadline()</code>	416
Установка периода ожидания на стороне клиента	417
Установка времени ожидания на стороне сервера	420
Упражнения	421
Резюме	421
Дополнительные ресурсы	422
Глава 9. Работа с TCP/IP и WebSocket	423
TCP/IP	424
Пакет <code>net</code>	426
Разработка TCP-клиента	426
Разработка TCP-клиента с помощью <code>net.Dial()</code>	426
Разработка TCP-клиента, использующего <code>net.DialTCP()</code>	428
Разработка TCP-сервера	430
Разработка TCP-сервера с помощью <code>net.Listen()</code>	430
Разработка TCP-сервера, использующего <code>net.ListenTCP()</code>	433
Разработка UDP-клиента	435

Разработка UDP-сервера	437
Разработка параллельных TCP-серверов	440
Работа с доменными сокетами UNIX	442
Сервер на сокетах домена UNIX	442
Клиент сокета домена UNIX	444
Создание сервера WebSocket	447
Реализация сервера	448
Создание клиента WebSocket	455
Упражнения	459
Резюме	460
Дополнительные ресурсы	460
Глава 10. Работа с REST API	461
Введение в REST	462
Разработка серверов и клиентов RESTful	464
Сервер RESTful	465
Клиент RESTful	473
Создание функционального сервера RESTful	481
REST API	482
Использование пакета gorilla/mux	483
Использование подмаршрутизаторов	484
Работа с базой данных	484
Тестирование пакета restdb	490
Реализация сервера RESTful	491
Тестирование сервера RESTful	495
Создание клиента RESTful	499
Создание структуры клиента командной строки	500
Реализация клиентских команд RESTful	501
Использование клиента RESTful	505
Работа с несколькими версиями REST API	507
Загрузка и скачивание двоичных файлов	507
Использование Swagger для документации REST API	512
Документирование REST API	514

Создание файла документации	517
Обслуживание файла документации	518
Упражнения	521
Резюме	521
Дополнительные ресурсы	521
Глава 11. Тестирование и профилирование кода	523
Оптимизация кода	524
Оценка производительности	525
Переписывание функции <code>main()</code> для более качественного тестирования . . .	526
Анализ производительности буферизованной записи и чтения	527
Утилита <code>benchstat</code>	531
Неправильно определенные бенчмарк-функции	532
Профилирование кода	533
Профилирование приложения командной строки	534
Профилирование HTTP-сервера	537
Веб-интерфейс профилировщика <code>Go</code>	539
Утилита <code>go tool trace</code>	540
Трассировка веб-сервера со стороны клиента	542
Посещение всех маршрутов веб-сервера	544
Тестирование <code>Go</code> -кода	548
Написание тестов для <code>./ch03/intRE.go</code>	549
Функция <code>TempDir()</code>	551
Функция <code>Cleanup()</code>	551
Пакет <code>testing/quick</code>	553
Тайм-аут тестов	555
Покрытие тестового кода	556
Поиск недостижимого <code>Go</code> -кода	559
Тестирование HTTP-сервера с помощью серверной части базы данных	561
Фаззинг	566
Кросс-компиляция	567
Использование директивы <code>go:generate</code>	569
Создание примеров функций	571

Упражнения	573
Резюме	573
Дополнительные ресурсы	574
Глава 12. Работа с gRPC	575
Введение в gRPC	575
Буферы протокола	576
Определение файла языка определения интерфейса	577
Разработка сервера gRPC	581
Разработка клиента gRPC	584
Упражнения	588
Резюме	588
Дополнительные ресурсы	589
Глава 13. Дженерики Go	590
Введение в дженерики	591
Ограничения	593
Определение новых типов данных с помощью дженериков	596
Интерфейсы и дженерики	600
Рефлексия и дженерики	602
Упражнения	604
Резюме	605
Дополнительные ресурсы	605
Приложение. Сборщик мусора Go	607
Куча и стек	607
Сборка мусора	611
Алгоритм трех цветов	614
Подробнее о работе сборщика мусора в Go	617
Карты, срезы и сборщик мусора Go	618
Сравнение эффективности представленных техник	621
Дополнительные ресурсы	622

Об авторе

Михалис Цукалос — системный инженер UNIX, который увлекается написанием технических текстов. Автор книг *Go Systems Programming* и *Mastering Go*¹, как первого, так и второго изданий. Получил степень бакалавра математики в Университете Патр и степень магистра информационных технологий в Университетском колледже Лондона. Написал более 300 технических статей для различных журналов, включая *Sys Admin*, *MacTech*, *Linux User and Developer*, *Usenix ;login.*, *Linux Format* и *Linux Journal*. В круг научных интересов Михалиса входят временные ряды, базы данных и индексирование.

Вы можете связаться с автором по адресу <https://www.mtsoukalos.eu/> и @mactsouk.

*Поскольку написание книги — командная работа, я хотел бы поблагодарить за помощь сотрудников издательства Packt Publishing. Спасибо Амиту Рамадасу за ответы на все мои вопросы, Шайлешу Джайну — за то, что убедил начать работу над третьим изданием *Mastering Go*, Эдварду Докси — за полезные комментарии и предложения и Дереку Паркеру — за его хорошую работу.*

Наконец, я хотел бы поблагодарить вас, читателя, за выбор этой книги. Надеюсь, она окажется вам полезной.

¹ Цукалос М. *GoLang* для профи. Работа с сетью, многопоточность, структуры данных и машинное обучение с Go. — СПб.: Питер, 2020.

О научном редакторе

Дерек Паркер — инженер-программист в Red Hat. Создатель отладчика Delve для Go и автор компилятора Go, компоновщика и стандартной библиотеки. Является автором проектов с открытым исходным кодом и специалистом по сопровождению ПО, работал над множеством проектов — от интерфейсного JavaScript до низкоуровневого ассемблерного кода.

Я хотел бы поблагодарить мою жену Эрику и двух наших замечательных детей за то, что они дали мне возможность работать над этим проектом.

Предисловие

Книга, которую вы сейчас читаете, называется «Golang для профи: Создаем профессиональные утилиты, параллельные серверы и сервисы» (3-е издание), и она вся о том, как стать лучшим разработчиком на Go! Если у вас есть второе издание, то не выбрасывайте его — Go не так уж сильно изменился, и второе издание по-прежнему полезно. Однако третье издание во многих аспектах лучше!

Добавлено много интересных новых тем, включая написание сервисов RESTful, работу с протоколом WebSocket и использование GitHub Actions и GitLab Actions для проектов Go, а также совершенно новая глава о дженериках и разработке множества полезных утилит. Кроме того, я постарался сделать это издание меньше, чем второе, и улучшить структуру книги, чтобы облегчить вам ее чтение и ускорить его.

Я также постарался включить нужное количество теории и практики — но только вы, читатель, можете сказать, насколько мне это удалось! Попробуйте выполнить упражнения в конце каждой главы и не стесняйтесь обращаться ко мне с идеями, которые могут еще больше улучшить будущие издания этой книги!

Для кого эта книга

Книга предназначена для программистов на Go среднего уровня, которые хотят улучшить свои навыки и перевести их на новый уровень. Она также будет полезна опытным разработчикам на других языках программирования, которые хотят изучить Go, не углубляясь в основы программирования.

Структура издания

Глава 1 начинается с рассказа об истории возникновения, важных свойствах и преимуществах Go. После этого описываются утилиты `godoc` и `go doc` и объясняется, как компилировать и исполнять программы на Go. Далее в главе рассказывается о выводе данных и получении пользовательского ввода, работе

с аргументами командной строки и использовании файлов журнала. Наконец, мы разработаем базовую версию приложения для телефонной книги, которую будем совершенствовать в следующих главах.

В главе 2 рассматриваются основные типы данных Go, как числовые, так и нечисловые, а также массивы и срезы, позволяющие группировать данные одного типа. В ней также рассматриваются указатели Go, константы и работа с датами и временем. Последняя часть главы посвящена генерации случайных чисел и заполнению приложения телефонной книги случайными данными.

Глава 3 начинается с карт, после чего переходит к структурам и ключевому слову `struct`. Кроме того, в ней вы найдете информацию о регулярных выражениях, сопоставлении с образцами и работе с CSV-файлами. Наконец, мы улучшим приложение телефонной книги, добавив в него постоянное хранение данных.

Глава 4 посвящена рефлексии, интерфейсам и методам типов — функциям, прикрепленным к типам данных. В ней также описывается использование интерфейса `sort.Interface` для сортировки фрагментов, использование пустого интерфейса, а также представлены утверждения типа, переключатели типа и тип данных `error`. Дополнительно, прежде чем улучшать приложение телефонной книги, мы обсудим, как Go может имитировать некоторые объектно-ориентированные концепции.

Глава 5 посвящена пакетам, модулям и функциям, которые являются основными элементами пакетов. Среди прочего мы создадим пакет Go, позволяющий взаимодействовать с базой данных PostgreSQL, документацию для него, а также объясним не всегда простое использование ключевого слова `defer`. В этой главе также содержится информация о том, как использовать GitLab Runners и GitHub Actions для автоматизации, и о том, как создать образ Docker для двоичного файла Go.

Глава 6 посвящена системному программированию и содержит такие темы, как работа с аргументами командной строки, обработка сигналов UNIX, ввод и вывод файлов, интерфейсы `io.Reader` и `io.Writer` и использование пакетов `viper` и `cobra`. Кроме того, мы поговорим о работе с файлами JSON, XML и YAML, создадим удобную утилиту командной строки, позволяющую обнаруживать циклы в файловой системе UNIX, и обсудим встраивание файлов в двоичные файлы Go, а также функцию `os.ReadDir()`, тип `os.DirEntry` и пакет `io/fs`. Наконец, мы обновим приложение телефонной книги, чтобы можно было использовать данные JSON, и преобразуем его в соответствующую утилиту командной строки с помощью пакета `cobra`.

В главе 7 обсуждаются горутины, каналы и конвейеры. Мы рассмотрим различия между процессами, потоками и горутинами, пакет `sync` и то, как работает

планировщик Go. Кроме того, мы исследуем использование ключевого слова `select` и обсудим различные «типы» каналов Go, а также общую память, мьютексы, типы `sync.Mutex` и `sync.RWMutex`. В остальной части главы мы рассмотрим пакеты `context` и `semaphore`, рабочие пулы и выясним, как устанавливать тайм-аут для горутин и выявлять состояние гонки.

В главе 8 обсуждаются пакет `net/http`, разработка веб-серверов и веб-сервисов, предоставление метрик в Prometheus, визуализация метрик в Grafana, создание веб-клиентов и файловых серверов. Мы также преобразуем приложение телефонной книги в веб-сервис и создадим для него клиент командной строки.

Глава 9 посвящена пакету `net`, TCP/IP и протоколам TCP и UDP, а также сокетам UNIX и протоколу WebSocket. В этой главе мы разработаем множество сетевых серверов и клиентов.

В главе 10 описана работа с REST API и сервисами RESTful. Мы выясним, как определять REST API и разрабатывать мощные параллельные серверы RESTful, а также утилиты командной строки, которые действуют как клиенты сервисов RESTful. Наконец, мы познакомимся со Swagger, позволяющим создавать документацию для REST API, и выясним, как загружать двоичные файлы.

В главе 11 обсуждаются тестирование, оптимизация и профилирование кода, а также кросс-компиляция, сравнительный анализ кода Go, создание примеров функций, использование директивы `go:generate` и поиск недоступного кода Go.

Глава 12 посвящена работе с gRPC в Go. gRPC — это альтернатива сервисам RESTful, разработанная Google. В этой главе вы узнаете, как определить методы и сообщения сервиса gRPC, как перевести их в код Go и как разработать сервер и клиент для этого сервиса gRPC.

Глава 13 посвящена дженерикам и тому, как использовать новый синтаксис для написания универсальных функций и определения универсальных типов данных. Дженерики появились в версии Go 1.18, которая, согласно циклу разработки Go, была официально выпущена в марте 2022 года.

В приложении рассказывается о работе сборщика мусора Go и показано, как этот компонент Go может повлиять на производительность вашего кода.

Как сделать книгу максимально полезной

Для работы с книгой требуется компьютер UNIX с относительно недавно установленной версией Go, то есть это может быть любая машина, работающая под управлением macOS X, macOS или Linux. Большая часть представленного

кода также будет выполняться на компьютерах с Microsoft Windows без каких-либо изменений.

Чтобы извлечь из книги максимальную пользу, попробуйте как можно скорее применить знания из каждой главы в собственных программах и посмотреть, что работает, а что нет! Как я уже говорил, попытайтесь выполнить упражнения, приведенные в конце каждой главы, или реализуйте свои программные задачи.

Файлы с примерами кода

Пакет кода для книги размещен на GitHub по адресу <https://github.com/mactsouk/mastering-Go-3rd>. У нас есть и другие пакеты кода из нашего богатого каталога книг и видео, доступные по адресу <https://github.com/PacktPublishing/>. Не забудьте в них заглянуть!

Цветные иллюстрации

Мы также предоставляем PDF-файл с цветными оригинальными снимками экрана и схемами из этой книги. Вы можете скачать его здесь: https://static.packt-cdn.com/downloads/9781801079310_ColorImages.pdf.

Условные обозначения

В книге используются следующие условные обозначения.

Кодовые слова в тексте, имена папок, имена файлов и их расширения, пути и пользовательский ввод оформляются **моноширинным шрифтом**. Например: «Звезда этой главы — пакет `net/http`, который содержит функции, позволяющие разрабатывать мощные веб-серверы и веб-клиенты».

Блок кода оформляется следующим образом:

```
package main

import (
    "fmt"
    "math/rand"
    "os"
    "path/filepath"
    "strconv"
    "time"
)
```

Когда мы хотим привлечь ваше внимание к определенной части блока кода, соответствующие строки или элементы выделяются **моноширинным жирным шрифтом**:

```
package main

import (
    "fmt"
    "math/rand"
    "os"
    "path/filepath"
    "strconv"
    "time"
)
```

Любой ввод или вывод из командной строки оформляется следующим образом:

```
$ go run www.go
Using default port number: :8001
Served: localhost:8001
```

Шрифтом без засечек оформляются URL или слова, которые вы видите на экране, например в меню или диалоговых окнах.

Новые термины и важные слова выделены *курсивом*.



Этот рисунок указывает на предупреждения или важные примечания.



Этот рисунок указывает на советы и рекомендации.

От издательства

Ваши замечания, предложения, вопросы отправляйте по адресу comp@piter.com (издательство «Питер», компьютерная редакция).

Мы будем рады узнать ваше мнение!

На веб-сайте издательства www.piter.com вы найдете подробную информацию о наших книгах.