

Оглавление

Предисловие.....	12
Вступление	13
Благодарности.....	14
Об этой книге.....	15
Об авторах	19
Об изображении на обложке	19
Часть 1. Введение в Kotlin.....	21
Глава 1. Kotlin: что это и зачем	22
1.1. Знакомство с Kotlin.....	22
1.2. Основные черты языка Kotlin.....	23
1.2.1. Целевые платформы: серверные приложения, Android и везде, где запускается Java.....	23
1.2.2. Статическая типизация.....	24
1.2.3. Функциональное и объектно-ориентированное программирование.....	25
1.2.4. Бесплатный язык с открытым исходным кодом.....	27
1.3. Приложения на Kotlin	27
1.3.1. Kotlin на сервере.....	27
1.3.2. Kotlin в Android	29
1.4. Философия Kotlin.....	30
1.4.1. Прагматичность.....	31
1.4.2. Лаконичность.....	31
1.4.3. Безопасность	32
1.4.4. Совместимость.....	33
1.5. Инструментарий Kotlin	34
1.5.1. Компиляция кода на Kotlin.....	35
1.5.2. Плагин для IntelliJ IDEA и Android Studio	36
1.5.3. Интерактивная оболочка.....	36
1.5.4. Плагин для Eclipse	36
1.5.5. Онлайн-полигон	36
1.5.6. Конвертер кода из Java в Kotlin	37
1.6. Резюме.....	37
Глава 2. Основы Kotlin.....	39
2.1. Основные элементы: переменные и функции	39
2.1.1. Привет, мир!.....	40
2.1.2. Функции.....	40
2.1.3. Переменные	42
2.1.4. Простое форматирование строк: шаблоны	44
2.2. Классы и свойства	45
2.2.1 Свойства.....	46
2.2.2. Собственные методы доступа	48

2.2.3. Размещение исходного кода на Kotlin: пакеты и каталоги	49
2.3. Представление и обработка выбора: перечисления и конструкция «when»	51
2.3.1. Объявление классов перечислений.....	51
2.3.2. Использование оператора «when» с классами перечислений.....	52
2.3.3. Использование оператора «when» с произвольными объектами.....	54
2.3.4. Выражение «when» без аргументов.....	55
2.3.5. Автоматическое приведение типов: совмещение проверки и приведения типа.....	55
2.3.6. Рефакторинг: замена «if» на «when».....	58
2.3.7. Блоки в выражениях «if» и «when».....	59
2.4. Итерации: циклы «while» и «for»	60
2.4.1. Цикл «while».....	60
2.4.2. Итерации по последовательности чисел: диапазоны и прогрессии	61
2.4.3. Итерации по элементам словарей.....	62
2.4.4. Использование «in» для проверки вхождения в диапазон или коллекцию	64
2.5. Исключения в Kotlin	65
2.5.1. «try», «catch» и «finally»	66
2.5.2. «try» как выражение	67
2.6. Резюме.....	68
Глава 3. Определение и вызов функций	70
3.1. Создание коллекций в Kotlin	70
3.2. Упрощение вызова функций	72
3.2.1. Именованные аргументы	73
3.2.2. Значения параметров по умолчанию	74
3.2.3. Избавление от статических вспомогательных классов: свойства и функции верхнего уровня	76
3.3. Добавление методов в сторонние классы: функции-расширения и свойства-расширения.....	78
3.3.1. Директива импорта и функции-расширения.....	80
3.3.2. Вызов функций-расширений из Java	80
3.3.3. Вспомогательные функции как расширения	81
3.3.4. Функции-расширения не переопределяются.....	82
3.3.5. Свойства-расширения	84
3.4. Работа с коллекциями: переменное число аргументов, инфиксная форма записи вызова и поддержка в библиотеке	85
3.4.1. Расширение API коллекций Java	85
3.4.2. Функции, принимающие произвольное число аргументов	86
3.4.3. Работа с парами: инфиксные вызовы и мультидекларации.....	87
3.5. Работа со строками и регулярными выражениями	88
3.5.1. Разбиение строк.....	89
3.5.2. Регулярные выражения и строки в тройных кавычках.....	89
3.5.3. Многострочные литералы в тройных кавычках.....	91
3.6. Чистим код: локальные функции и расширения.....	93
3.7. Резюме	96

Глава 4. Классы, объекты и интерфейсы.....	97
4.1. Создание иерархий классов.....	98
4.1.1. Интерфейсы в Kotlin	98
4.1.2. Модификаторы <code>open</code> , <code>final</code> и <code>abstract</code> : по умолчанию <code>final</code>	101
4.1.3. Модификаторы видимости: по умолчанию <code>public</code>	103
4.1.4. Внутренние и вложенные классы: по умолчанию вложенные	105
4.1.5. Запечатанные классы: определение жестко заданных иерархий	108
4.2. Объявление классов с нетривиальными конструкторами или свойствами.....	110
4.2.1. Инициализация классов: основной конструктор и блоки инициализации.....	110
4.2.2. Вторичные конструкторы: различные способы инициализации суперкласса.....	113
4.2.3. Реализация свойств, объявленных в интерфейсах.....	115
4.2.4. Обращение к полю из методов доступа	117
4.2.5. Изменение видимости методов доступа	118
4.3. Методы, сгенерированные компилятором: классы данных и делегирование.....	119
4.3.1. Универсальные методы объектов	120
4.3.2. Классы данных: автоматическая генерация универсальных методов.....	123
4.3.3. Делегирование в классах. Ключевое слово <code>by</code>	124
4.4. Ключевое слово <code>object</code> : совместное объявление класса и его экземпляра	127
4.4.1. Объявление объекта: простая реализация шаблона «Одиночка».....	127
4.4.2. Объекты-компаньоны: место для фабричных методов и статических членов класса	130
4.4.3. Объекты-компаньоны как обычные объекты.....	132
4.4.4. Объекты-выражения: другой способ реализации анонимных внутренних классов	135
4.5. Резюме.....	136
Глава 5. Лямбда-выражения	138
5.1. Лямбда-выражения и ссылки на члены класса	138
5.1.1. Введение в лямбда-выражения: фрагменты кода как параметры функций	139
5.1.2. Лямбда-выражения и коллекции	140
5.1.3. Синтаксис лямбда-выражений.....	141
5.1.4. Доступ к переменным из контекста	145
5.1.5. Ссылки на члены класса	148
5.2. Функциональный API для работы с коллекциями.....	150
5.2.1. Основы: <code>filter</code> и <code>map</code>	150
5.2.2. Применение предикатов к коллекциям: функции <code>«all»</code> , <code>«any»</code> , <code>«count»</code> и <code>«find»</code>	152
5.2.3. Группировка значений в списке с функцией <code>groupBy</code>	154
5.2.4. Обработка элементов вложенных коллекций: функции <code>flatMap</code> и <code>flatten</code>	154
5.3. Отложенные операции над коллекциями: последовательности	156
5.3.1. Выполнение операций над последовательностями: промежуточная и завершающая операции	157
5.3.2. Создание последовательностей	160
5.4. Использование функциональных интерфейсов Java	161
5.4.1. Передача лямбда-выражения в Java-метод	162

5.4.2. SAM-конструкторы: явное преобразование лямбда-выражений в функциональные интерфейсы	164
5.5. Лямбда-выражения с получателями: функции «with» и «apply»	166
5.5.1. Функция «with»	166
5.5.2. Функция «apply»	169
5.6. Резюме	171
Глава 6. Система типов Kotlin.....	172
6.1. Поддержка значения null.....	172
6.1.1. Типы с поддержкой значения null	173
6.1.2. Зачем нужны типы	175
6.1.3. Оператор безопасного вызова: «?»	177
6.1.4. Оператор «Элвис»: «?:».....	178
6.1.5. Безопасное приведение типов: оператор «as?».....	180
6.1.6. Проверка на null: утверждение «!!»	182
6.1.7. Функция let.....	184
6.1.8. Свойства с отложенной инициализацией	186
6.1.9. Расширение типов с поддержкой null.....	188
6.1.10. Типовые параметры с поддержкой null.....	189
6.1.11. Допустимость значения null и Java.....	190
6.2. Примитивные и другие базовые типы	195
6.2.1. Примитивные типы: Int, Boolean и другие	195
6.2.2. Примитивные типы с поддержкой null: Int?, Boolean? и прочие	197
6.2.3. Числовые преобразования.....	198
6.2.4. Корневые типы Any и Any?	200
6.2.5. Тип Unit: тип «отсутствующего» значения.....	201
6.2.6. Тип Nothing: функция, которая не завершается	202
6.3. Массивы и коллекции.....	203
6.3.1 Коллекции и допустимость значения null.....	203
6.3.2. Изменяемые и неизменяемые коллекции	206
6.3.3. Коллекции Kotlin и язык Java	208
6.3.4. Коллекции как платформенные типы	210
6.3.5. Массивы объектов и примитивных типов	213
6.4. Резюме	215
Часть 2. Непростой Kotlin	217
Глава 7. Перегрузка операторов и другие соглашения.....	218
7.1. Перегрузка арифметических операторов	219
7.1.1. Перегрузка бинарных арифметических операций.....	219
7.1.2. Перегрузка составных операторов присваивания	222
7.1.3. Перегрузка унарных операторов	224
7.2. Перегрузка операторов сравнения.....	225
7.2.1. Операторы равенства: «equals».....	225
7.2.2. Операторы отношения: compareTo	227
7.3. Соглашения для коллекций и диапазонов.....	228
7.3.1. Обращение к элементам по индексам: «get» и «set»	228
7.3.2. Соглашение «in»	230

7.3.3. Соглашение <code>rangeTo</code>	231
7.3.4. Соглашение « <code>iterator</code> » для цикла « <code>for</code> »	232
7.4. Мультидекларации и функции <code>component</code>	233
7.4.1. Мультидекларации и циклы	235
7.5. Повторное использование логики обращения к свойству: делегирование свойств.....	236
7.5.1. Делегирование свойств: основы.....	237
7.5.2. Использование делегирования свойств: отложенная инициализация и <code>«by lazy()»</code>	238
7.5.3. Реализация делегирования свойств	240
7.5.4. Правила трансляции делегированных свойств.....	244
7.5.5. Сохранение значений свойств в словаре.....	245
7.5.6. Делегирование свойств в фреймворках	246
7.6. Резюме	248
Глава 8. Функции высшего порядка: лямбда-выражения как параметры и возвращаемые значения.....	249
8.1. Объявление функций высшего порядка.....	250
8.1.1. Типы функций.....	250
8.1.2. Вызов функций, переданных в аргументах	251
8.1.3. Использование типов функций в коде на Java	253
8.1.4. Значения по умолчанию и пустые значения для параметров типов функций	254
8.1.5. Возврат функций из функций.....	257
8.1.6. Устранение повторяющихся фрагментов с помощью лямбда-выражений....	259
8.2. Встраиваемые функции: устранение накладных расходов лямбда-выражений	262
8.2.1. Как работает встраивание функций	262
8.2.2. Ограничения встраиваемых функций.....	264
8.2.3. Встраивание операций с коллекциями.....	265
8.2.4. Когда следует объявлять функции встраиваемыми	267
8.2.5. Использование встраиваемых лямбда-выражений для управления ресурсами	268
8.3. Порядок выполнения функций высшего порядка.....	269
8.3.1. Инструкции <code>return</code> в лямбда-выражениях:	
выход из вмещающей функции.....	270
8.3.2. Возврат из лямбда-выражений:	
возврат с помощью меток.....	271
8.3.3. Анонимные функции: по умолчанию возврат выполняется локально	273
8.4. Резюме.....	274
Глава 9. Обобщенные типы	276
9.1. Параметры обобщенных типов	277
9.1.1. Обобщенные функции и свойства.....	278
9.1.2. Объявление обобщенных классов	279
9.1.3. Ограничения типовых параметров	281
9.1.4. Ограничение поддержки <code>null</code> в типовом параметре	283

9.2. Обобщенные типы во время выполнения: стирание и овеществление параметров типов	284
9.2.1. Обобщенные типы во время выполнения: проверка и приведение типов	284
9.2.2. Объявление функций с овеществляемыми типовыми параметрами	287
9.2.3. Замена ссылок на классы овеществляемыми типовыми параметрами	290
9.2.4. Ограничения овеществляемых типовых параметров	291
9.3. Вариантность: обобщенные типы и подтипы	292
9.3.1. Зачем нужна вариантность: передача аргумента в функцию.....	292
9.3.2. Классы, типы и подтипы.....	293
9.3.3. Ковариантность: направление отношения тип–подтип сохраняется.....	296
9.3.4. Контравариантность: направление отношения тип–подтип изменяется на противоположное	300
9.3.5. Определение вариантности в месте использования: определение варианты для вхождений типов.....	303
9.3.6. Проекция со звездочкой: использование * вместо типового аргумента.....	306
9.4. Резюме.....	311
Глава 10. Аннотации и механизм рефлексии	313
10.1. Объявление и применение аннотаций.....	314
10.1.1. Применение аннотаций	314
10.1.2. Целевые элементы аннотаций	315
10.1.3. Использование аннотаций для настройки сериализации JSON.....	318
10.1.4. Объявление аннотаций.....	320
10.1.5. Метааннотации: управление обработкой аннотаций.....	321
10.1.6. Классы как параметры аннотаций.....	322
10.1.7. Обобщенные классы в параметрах аннотаций	323
10.2. Рефлексия: интроспекция объектов Kotlin во время выполнения	325
10.2.1. Механизм рефлексии в Kotlin: KClass, KCallable, KFunction и KProperty	326
10.2.2. Сериализация объектов с использованием механизма рефлексии	330
10.2.3. Настройка сериализации с помощью аннотаций.....	332
10.2.4. Парсинг формата JSON и десериализация объектов	336
10.2.5. Заключительный этап десериализации: callBy() и создание объектов с использованием рефлексии.....	340
10.3. Резюме	345
Глава 11. Конструирование DSL	346
11.1. От API к DSL.....	346
11.1.1. Понятие предметно-ориентированного языка.....	348
11.1.2. Внутренние предметно-ориентированные языки	349
11.1.3. Структура предметно-ориентированных языков	351
11.1.4. Создание разметки HTML с помощью внутреннего DSL.....	352
11.2. Создание структурированных API: лямбда-выражения с получателями в DSL.....	354
11.2.1. Лямбда-выражения с получателями и типы функций-расширений	354
11.2.2. Использование лямбда-выражений с получателями в построителях разметки HTML.....	358
11.2.3. Построители на Kotlin: поддержка абстракций и многократного использования	363

11.3. Гибкое вложение блоков с использованием соглашения « <i>invoke</i> ».....	366
11.3.1. Соглашение « <i>invoke</i> »: объекты, вызываемые как функции.....	367
11.3.2. Соглашение « <i>invoke</i> » и типы функций.....	367
11.3.3. Соглашение « <i>invoke</i> » в предметно-ориентированных языках: объявление зависимостей в Gradle	369
11.4. Предметно-ориентированные языки Kotlin на практике.....	371
11.4.1. Цепочки инфиксных вызовов: « <i>should</i> » в фреймворках тестирования	371
11.4.2. Определение расширений для простых типов: обработка дат	374
11.4.3. Члены-расширения: внутренний DSL для SQL	375
11.4.4. Anko: динамическое создание пользовательских интерфейсов в Android	378
11.5. Резюме	380
Приложение А. Сборка проектов на Kotlin.....	382
A.1. Сборка кода на Kotlin с помощью Gradle.....	382
A.1.1. Сборка Kotlin-приложений для Android с помощью Gradle.....	383
A.1.2. Сборка проектов с обработкой аннотаций	384
A.2. Сборка проектов на Kotlin с помощью Maven	384
A.3. Сборка кода на Kotlin с помощью Ant.....	385
Приложение В. Документирование кода на Kotlin	387
B.1. Документирующие комментарии в Kotlin	387
B.2. Создание документации с описанием API	389
Приложение С. Экосистема Kotlin.....	390
C.1. Тестирование	390
C.2. Внедрение зависимостей	391
C.3. Сериализация JSON.....	391
C.4. Клиенты HTTP	391
C.5. Веб-приложения.....	391
C.6. Доступ к базам данных.....	392
C.7. Утилиты и структуры данных	392
C.8. Настольные приложения.....	393
Предметный указатель	394

Предисловие

Впервые оказавшись в JetBrains весной 2010 года, я был абсолютно уверен, что миру не нужен еще один язык программирования общего назначения. Я полагал, что существующие JVM-языки достаточно хороши, да и кто в здравом уме станет создавать новый язык? Примерно после часа обсуждения проблем разработки крупномасштабных программных продуктов мое мнение изменилось и я набросал на доске первые идеи, которые позже стали частью языка Kotlin. Вскоре после этого я присоединился к JetBrains для проектирования языка и работы над компилятором.

Сегодня, спустя шесть лет, мы приближаемся к выпуску второй версии. Сейчас в команде работает более 30 человек, а у языка появились тысячи активных пользователей, и у нас осталось еще множество потрясающих идей, которые с трудом укладываются в моей голове. Но не волнуйтесь: прежде чем стать частью языка, эти идеи подвергнутся тщательной проверке. Мы хотим, чтобы в будущем описание языка Kotlin по-прежнему могло уместиться в одну не слишком большую книгу.

Изучение языка программирования – это захватывающее и часто очень полезное занятие. Если это ваш первый язык, с ним вы откроете целый новый мир программирования. Если нет, он заставит вас по-другому думать о знакомых вещах и глубже осознать их на более высоком уровне абстракции. Эта книга предназначена в основном для последней категории читателей, уже знакомых с Java.

Проектирование языка с нуля – это сама по себе сложная задача, но обеспечение взаимодействия с другим языком – это совсем другая история, полная злых огров и мрачных подземелий. (Если не верите мне – спросите Бъярне Страуструпа (Bjarne Stroustrup), создателя C++). Совместимость с Java (то есть возможность смешивать код на Java и Kotlin и вызывать один из другого) стала одним из краеугольных камней Kotlin, и эта книга уделяет большое внимание данному аспекту. Взаимодействие с Java очень важно для постепенного внедрения Kotlin в существующие проекты на Java. Даже создавая проект с нуля, важно учитывать, как язык вписывается в общую картину платформы со всем многообразием библиотек, написанных на Java.

В данный момент, когда я пишу эти строки, разрабатываются две новые платформы: Kotlin уже запускается на виртуальных машинах JavaScript, обеспечивая поддержку разработки всех уровней веб-приложения, и скоро его можно будет компилировать прямо в машинный код и запускать без виртуальной машины. Хотя эта книга ориентирована на JVM, многое из того, что вы узнаете, может быть использовано в других средах выполнения.

Авторы присоединились к команде Kotlin с самых первых дней, поэтому они хорошо знакомы с языком и его внутренним устройством. Благодаря опыту выступления на конференциях и проведению семинаров и курсов о Kotlin, авторы смогли сформировать хорошие объяснения, предвосхищающие самые распространенные вопросы и возможные подводные камни. Книга объясняет высокоуровневые понятия языка во всех необходимых подробностях.

Надеюсь, вы хорошо проведете время с книгой, изучая наш язык. Как я часто говорю в сообщениях нашего сообщества: «Хорошего Kotlin!».

Андрей Бреслав

Ведущий разработчик языка Kotlin в JetBrains

Вступление

Идея создания Kotlin зародилась в JetBrains в 2010 году. К тому времени компания уже была признанным производителем инструментов разработки для множества языков, включая Java, C#, JavaScript, Python, Ruby и PHP. IntelliJ IDEA – наш флагманский продукт, интегрированная среда разработки (IDE) для Java – также включала плагины для Groovy и Scala.

Опыт разработки инструментария для такого разнообразного набора языков позволил нам получить уникальное понимание процесса проектирования языков целом и взглянуть на него под другим углом. И все же интегрированные среды разработки на платформе IntelliJ, включая IntelliJ IDEA, по-прежнему разрабатывались на Java. Мы немного завидовали нашим коллегам из команды .NET, которые вели разработку на C# – современном, мощном, быстро развивающемся языке. Но у нас не было языка, который мы могли бы использовать вместо Java.

Какими должны быть требования к такому языку? Первое и самое очевидное – статическая типизация. Мы не знаем другого способа разрабатывать проекты с миллионами строк кода на протяжении многих лет, не сходя при этом с ума. Второе – полная совместимость с существующим кодом на Java. Этот код является чрезвычайно ценным активом компании JetBrains, и мы не могли себе позволить потерять или обесценить его из-за проблем совместимости. В-третьих, мы не хотели идти на компромиссы с точки зрения качества инструментария. Производительность разработчиков – самая главная ценность компании JetBrains, а для её достижения нужен хороший инструментарий. Наконец, нам был нужен язык, простой в изучении и применении.

Когда мы видим в своей компании неудовлетворенную потребность, мы знаем, что есть и другие компании, оказавшиеся в подобной ситуации, и что наше решение найдет много пользователей за пределами JetBrains. Учитывая это, мы решили начать проект разработки нового языка: Kotlin. Как это часто бывает, проект занял больше времени, чем мы ожидали, и Kotlin 1.0 вышел больше чем через пять лет после того, как в репозитории появился первый фрагмент его реализации; но теперь мы уверены, что язык нашел своих пользователей и будет использоваться в дальнейшем.

Язык Kotlin назван в честь острова Котлин неподалеку от Санкт-Петербурга в России, где живет большинство разработчиков Kotlin. Выбрав для названия языка имя острова, мы последовали прецеденту, созданному языками Java и Ceylon, но решили найти что-то ближе к нашему дому.

По мере приближения к выпуску первой версии языка мы поняли, что нам не помешала бы книга про Kotlin, написанная людьми, которые участвовали в принятии проектных решений и могут уверенно объяснить, почему Kotlin устроен так, а не иначе. Данная книга – результат совместных усилий этих людей, и мы надеемся, что она поможет вам узнать и понять язык Kotlin. Удачи вам, и программируйте с удовольствием!

Благодарности

Прежде всего мы хотим поблагодарить Сергея Дмитриева и Максима Шафирова за веру в идею нового языка и решение вложить средства JetBrains в его разработку. Без них не было бы ни языка, ни этой книги.

Мы хотели бы особо поблагодарить Андрея Бреслава – главного виновника, что язык спроектирован так, что писать про него одно удовольствие (впрочем, как и программировать на нем). Несмотря на занятость управлением постоянно растущей командой Kotlin, Андрей смог сделать много полезных замечаний, что мы очень высоко ценим. Более того, вы можете быть уверены, что книга получила одобрение от ведущего разработчика языка в виде предисловия, которое он любезно написал.

Мы благодарны команде издательства Manning, которая помогла нам написать книгу и сделать ее легко читаемой и хорошо структурированной. В частности, мы хотим поблагодарить редактора-консультанта Дэна Махари (Dan Maharry), который всегда стремился найти время для обсуждения, несмотря на наш напряженный график, а также Майкла Стивенса (Michael Stephens), Хелен Стергиус (Helen Stergius), Кевина Салливана (Kevin Sullivan), Тиффани Тейлор (Tiffany Taylor), Элизабет Мартин (Elizabeth Martin) и Марию Тюдор (Marija Tudor). Отзывы наших технических редакторов Брента Уотсона (Brent Watson) и Игоря Войды (Igor Wojda) оказались просто бесценны, так же как замечания рецензентов, читавших рукопись в процессе работы: Александро Кампей (Alessandro Campeis), Амита Ламба (Amit Lamba), Анджело Кости (Angelo Costa), Бориса Василе (Boris Vasile), Брендана Грейнджера (Brendan Grainger), Кальвина Фернандеса (Calvin Fernandes), Кристофера Бейли (Christopher Bailey), Кристофера Борца (Christopher Bortz), Конора Редмонда (Conor Redmond), Дилана Скотта (Dylan Scott), Филипа Правика (Filip Pravica), Джейсона Ли (Jason Lee), Джастина Ли (Justin Lee), Кевина Орра (Kevin Orr), Николаса Франкеля (Nicolas Frankel), Павла Гайды (Paweł Gajda), Рональда Тишлера (Ronald Tischliar) и Тима Лаверса (Tim Laver).

Также благодарим всех, кто отправил свои предложения в рамках МЕАР (Manning Early Access Program – программы раннего доступа Manning) на форуме книги; ваши комментарии помогли улучшить текст книги.

Мы благодарны всем участникам команды Kotlin, которым приходилось выслушивать ежедневные заявления вроде: «Еще один раздел закончен!» – на протяжении всего периода написания этой книги. Мы хотим поблагодарить наших коллег, которые помогли составить план книги и оставляли отзывы о её ранних вариантах, особенно Илью Рыженкова, Хади Харiri (Hadi Hariri), Михаила Глухих и Илью Горбунова. Мы также хотим поблагодарить друзей, которые не только поддерживали нас, но и читали текст книги и оставляли отзывы о ней (иногда на горнолыжных курортах во время отпуска): Льва Серебрякова, Павла Николаева и Алису Афонину.

Наконец, мы хотели бы поблагодарить наши семьи и котов за то, что они делают этот мир лучше.

Об этой книге

Книга «*Kotlin в действии*» расскажет о языке Kotlin и как писать на нем приложения для виртуальной машины Java и Android. Она начинается с обзора основных особенностей языка Kotlin, постепенно раскрывая наиболее отличительные аспекты, такие как поддержка создания высокоуровневых абстракций и предметно-ориентированных языков (Domain-Specific Languages, DSL). Книга уделяет большое внимание интеграции Kotlin с существующими проектами на языке Java и поможет вам внедрить Kotlin в текущую рабочую среду.

Книга описывает версию языка Kotlin 1.0. Версия Kotlin 1.1 разрабатывалась параллельно с написанием книги, и, когда это было возможно, мы упоминали об изменениях в версии 1.1. Но поскольку на момент написания книги новая версия еще не была готова, мы не могли полностью охватить все нововведения. За более подробной информацией о новых возможностях и изменениях обращайтесь к документации по адресу: <https://kotlinlang.org>.

Кому адресована эта книга

Книга «*Kotlin в действии*» адресована в первую очередь разработчикам с опытом программирования на языке Java. Kotlin во многом основан на понятиях и приёмах языка Java, и с помощью этой книги вы быстро освоите его, используя имеющиеся знания. Если вы только начали изучать Java или владеете другими языками программирования, такими как C# или JavaScript, вам может понадобиться обратиться к другим источникам информации, чтобы понять наиболее сложные аспекты взаимодействия Kotlin с JVM, но вы все равно сможете изучить Kotlin, читая эту книгу. Мы описываем язык Kotlin в целом, не привязываясь к конкретной предметной области, поэтому книга должна быть одинаково полезна и для разработчиков серверных приложений, и для разработчиков на Android, и для всех, кто создает проекты для JVM.

Как организована эта книга

Книга делится на две части. *Часть 1* объясняет, как начать использовать Kotlin вместе с существующими библиотеками и API:

- *глава 1* рассказывает о ключевых целях, ценностях и областях применения языка и показывает различные способы запуска кода на Kotlin;
- *глава 2* демонстрирует важные элементы любой программы на языке Kotlin, включая управляющие структуры, переменные и функции;

- в *главе 3* подробно рассматриваются объявления функций в Kotlin, а также вводятся понятия функций-расширений (extension functions) и свойств-расширений (extension properties);
- *глава 4* посвящена объявлению классов и знакомит с понятиями классов данных (data classes) и объектов-компаньонов (companion objects);
- *глава 5* знакомит с лямбда-выражениями и демонстрирует ряд примеров их использования в стандартной библиотеке Kotlin;
- *глава 6* описывает систему типов Kotlin, обращая особое внимание на работу с типами, допускающими значения null, и с коллекциями.

Часть 2 научит вас создавать собственные API и абстракции на языке Kotlin и охватывает некоторые более продвинутые особенности языка:

- *глава 7* рассказывает о соглашениях, придающих особый смысл методам и свойствам с определенными именами, и вводит понятие делегируемых свойств (delegated properties);
- *глава 8* показывает, как объявлять функции высшего порядка – функции, принимающие другие функции или возвращающие их. Здесь также вводится понятие встраиваемых функций (inline functions);
- *глава 9* глубоко погружается в тему обобщенных типов (generics), начиная с базового синтаксиса и переходя к более продвинутым темам, таким как овеществляемые типовые параметры (reified type parameters) и вариантность;
- *глава 10* посвящена использованию аннотаций и механизма рефлексии (reflection) и организована вокруг JKid – простой библиотеки сериализации в формат JSON, которая интенсивно использует эти понятия;
- *глава 11* вводит понятие предметно-ориентированного языка (DSL), описывает инструменты Kotlin для их создания и демонстрирует множество примеров DSL.

В книге есть три приложения. Приложение А объясняет, как выполнять сборку проектов на Kotlin с помощью Gradle, Maven и Ant. Приложение В фокусируется на документирующих комментариях и создании документации с описанием API модулей. Приложение С является руководством по экосистеме Kotlin и поиску актуальной информации в Интернете.

Книгу лучше читать последовательно, от начала до конца, но также можно обращаться к отдельным главам, посвященным интересующим вам конкретным темам, и переходить по перекрестным ссылкам для уточнения незнакомых понятий.

Соглашения об оформлении программного кода и загружаемые ресурсы

В книге приняты следующие соглашения:

- *Курсивом* обозначаются новые термины.
- Моноширинным шрифтом выделены фрагменты кода, имена классов и функций и другие идентификаторы.
- Примеры кода сопровождаются многочисленными примечаниями, подчеркивающими важные понятия.

Многие листинги кода в книге показаны вместе с его выводом. В таких случаях строки кода, производящие вывод, начинаются с префикса `>>>`, как показано ниже:

```
>>> println("Hello World")
Hello World
```

Некоторые примеры являются полноценными программами, тогда как другие – лишь фрагменты, демонстрирующие определенные понятия и могущие содержать сокращения (обозначенные как ...) или синтаксические ошибки (описанные в тексте книги или самих примерах). Выполняемые примеры можно загрузить в виде zip-архива на сайте издательства www.manning.com/books/kotlin-in-action. Примеры из книги также выгружаются в онлайн-окружение <http://try.kotlinlang.org>, где вы сможете опробовать любой пример, сделав всего несколько щелчков в окне браузера.

Авторы онлайн

Покупка книги «*Kotlin в действии*» дает право свободного доступа к закрытому веб-форуму издательства Manning Publication, где можно высказать свои замечания о книге, задать технические вопросы и получить помощь от авторов и других пользователей. Чтобы получить доступ к форуму, перейдите по ссылке www.manning.com/books/kotlin-in-action. На этой странице описывается, как попасть на форум после регистрации, какая помощь доступна и какие правила поведения действуют на форуме.

Издательство Manning обязуется предоставить читателям площадку для содержательного диалога не только между читателями, но и между читателями и авторами. Но авторы не обязаны выделять определенное количество времени для участия, т. к. их вклад в работу форума является добровольным (и неоплачиваемым). Мы предлагаем читателям задавать авторам действительно непростые вопросы, чтобы их интерес не угасал!

Прочие онлайн-ресурсы

Kotlin имеет активное сообщество, поэтому имеющие вопросы или желающие пообщаться с другими пользователями Kotlin могут воспользоваться следующими ресурсами:

- официальный форум Kotlin – <https://discuss.kotlinlang.org>;
- чат Slack – <http://kotlinlang.slack.com> (вы можете получить приглашение по ссылке <http://kotlinslackin.herokuapp.com/>);
- вопросы и ответы с тегом Kotlin на Stack Overflow – <http://stackoverflow.com/questions/tagged/kotlin>;
- Kotlin на форуме Reddit – <http://www.reddit.com/r/Kotlin>.

Об авторах

Дмитрий Жемеров работает в компании JetBrains с 2003 года и принимал участие в разработке многих продуктов, в том числе IntelliJ IDEA, PyCharm и WebStorm. Был одним из первых участников команды Kotlin, создал начальную версию генератора байт-кода JVM из кода Kotlin и сделал много презентаций о языке Kotlin на различных встречах по всему миру. Сейчас возглавляет команду, работающую над плагином Kotlin IntelliJ IDEA.

Светлана Исакова вошла в состав команды Kotlin в 2011 году. Работала над механизмом вывода типов (type inference) и подсистемой компилятора по разрешению перегруженных имен. Сейчас она – технический евангелист, рассказывает о Kotlin на конференциях и разрабатывает онлайн-курс по языку Kotlin.

Об изображении на обложке

Иллюстрация на обложке книги «*Kotlin в действии*» называется «Одежда русской женщины на Валдае в 1764 году». Город Валдай находится в Новгородской области, между Москвой и Санкт-Петербургом. Иллюстрация взята из работы Томаса Джеффериса (Thomas Jefferys) «Коллекция платьев разных народов, древних и современных», опубликованной в Лондоне между 1757 и 1772 г.. На титульной странице говорится, что это – раскрашенная вручную гравюра, обработанная гуммиарабиком для повышения яркости. Томаса Джеффериса (1719–1771) называли географом короля Георга III. Он был английским картографом и ведущим поставщиком карт своего времени. Он гравировал и печатал карты для правительства и других официальных органов, выпускал широкий спектр коммерческих карт и атласов, особенно Северной Америки. Работа картографом пробудила интерес к местным традиционным нарядам в землях, которые он исследовал и картографировал; эти наряды великолепно представлены в четырехтомном сборнике.

Очарование дальними странами и путешествия для удовольствия были относительно новым явлением в восемнадцатом веке, и такие коллекции, как эта, были популярны и показывали туристам и любителям книг о путешествиях жителей других стран. Разнообразие рисунков в книгах Джеффериса красноречиво говорит об уникальности и индивидуальности народов мира много веков назад. С тех пор стиль одежды сильно изменился, и разнообразие, характеризующее различные области и страны, исчезло. Сейчас часто трудно отличить даже жителей одного континента от дру-

гого. Возможно, с оптимистической точки зрения, мы обменяли культурное и визуальное разнообразие на более разнообразную частную жизнь или более разнообразную и интересную интеллектуальную и техническую деятельность.

В наше время, когда трудно отличить одну компьютерную книгу от другой, издательство Manning с инициативой и находчивостью наделяет книги обложками, изображающими богатое разнообразие жизненного уклада народов многовековой давности, давая новую жизнь рисункам Джейффе-риса.

Часть 1

Введение в Kotlin

Цель этой части книги – помочь начать продуктивно писать код на языке Kotlin, используя существующие API. Глава 1 познакомит вас с языком Kotlin в общих чертах. В главах 2–4 вы узнаете, как в Kotlin реализованы основные понятия языка Java – операторы, функции, классы и типы, – и как Kotlin обогащает их, делая программирование более приятным. Вы сможете положиться на имеющиеся знания языка Java, а также на вспомогательные инструменты, входящие в состав интегрированной среды разработки, и конвертер кода на Java в код на Kotlin, чтобы быстро начать писать код. В главе 5 вы узнаете, как лямбда-выражения помогают эффективно решать некоторые из распространенных задач программирования, такие как работа с коллекциями. Наконец, в главе 6 вы познакомитесь с одной из ключевых особенностей Kotlin – поддержкой операций со значениями `null`.

Глава 1

Kotlin: что это и зачем

В этой главе:

- общий обзор языка Kotlin;
- основные особенности;
- возможности разработки для Android и серверных приложений;
- отличие Kotlin от других языков;
- написание и выполнение кода на языке Kotlin.

Что же такое Kotlin? Это новый язык программирования для платформы Java. Kotlin – лаконичный, безопасный и прагматичный язык, совместимый с Java. Его можно использовать практически везде, где применяется Java: для разработки серверных приложений, приложений для Android и многое другое. Kotlin прекрасно работает со всеми существующими библиотеками и фреймворками, написанными на Java, не уступая последнему в производительности. В этой главе мы подробно рассмотрим основные черты языка Kotlin.

1.1. Знакомство с Kotlin

Начнем с небольшого примера для демонстрации языка Kotlin. В этом примере определяется класс `Person`, создается коллекция его экземпляров, выполняется поиск самого старого и выводится результат. Даже в этом маленьком фрагменте кода можно заметить множество интересных особенностей языка Kotlin; мы выделили некоторые из них, чтобы вам проще было отыскать их в книге в будущем. Код пояснен довольно кратко, но не беспокойтесь, если что-то останется непонятным. Позже мы подробно всё обсудим.

Желающие опробовать этот пример могут воспользоваться онлайн-полигоном по адресу: <http://try.kotlin.in>. Введите пример, щелкните на кнопке **Run** (Запустить), и код будет выполнен.

Листинг 1.1. Первое знакомство с Kotlin

```

data class Person(val name: String,           ← Класс «данных»
                val age: Int? = null)   ← Тип, допускающий значение null (Int?);
                                         значение параметра по умолчанию

fun main(args: Array<String>) {           ← Функция верхнего уровня
    val persons = listOf(Person("Alice"),    ← Именованный аргумент
                         Person("Bob", age = 29))

    val oldest = persons.maxBy { it.age ?: 0 }   ← Лямбда-выражение; оператор «Элвис»
    println("The oldest is: $oldest")           ← Стока-шаблон
}

// The oldest is: Person(name=Bob, age=29)      ← Часть вывода автоматически сгенерирована
                                                ← методом toString

```

Здесь объявляется простой класс данных с двумя свойствами: `name` и `age`. Свойству `age` по умолчанию присваивается значение `null` (если оно не задано). При создании списка людей возраст Алисы не указывается, поэтому он принимает значение `null`. Затем, чтобы отыскать самого старого человека в списке, вызывается функция `maxBy`. Лямбда-выражение, которое передается функции, принимает один параметр с именем `it` по умолчанию. *Оператор «Элвис»* (`?:`) возвращает ноль, если возраст имеет значение `null`. Поскольку возраст Алисы не указан, оператор «Элвис» заменит его нулем, поэтому Боб получит приз как самый старый человек.

Вам понравилось? Читайте дальше, чтобы узнать больше и стать экспертом в языке Kotlin. Мы надеемся, что скоро вы увидите такой код в своих проектах, а не только в этой книге.

1.2. Основные черты языка Kotlin

Возможно, у вас уже есть некоторое представление о языке Kotlin. Давайте подробнее рассмотрим его ключевые особенности. Для начала определим типы приложений, которые можно создавать с его помощью.

1.2.1. Целевые платформы: серверные приложения, Android и везде, где запускается Java

Основная цель языка Kotlin – предложить более компактную, производительную и безопасную альтернативу языку Java, пригодную для использования везде, где сегодня применяется Java. Java – чрезвычайно популярный язык, который используется в самых разных окружениях, начиная от смарт-карт (технология Java Card) до крупнейших вычислительных центров таких компаний, как Google, Twitter и LinkedIn. В большинстве таких окружений применение Kotlin способно помочь разработчикам достигать своих целей меньшим объемом кода и избегая многих неприятностей.

Конец ознакомительного фрагмента.
Приобрести книгу можно
в интернет-магазине «Электронный универс»
(e-Univers.ru)