

# Оглавление

<b>Вступление</b> . . . . .	23
<b>Предисловие</b> . . . . .	25
<b>Благодарности</b> . . . . .	27
Лондонское Java-сообщество . . . . .	27
www.coderanch.com . . . . .	28
Manning publications . . . . .	28
Особые благодарности . . . . .	29
Благодарности Бена Эванса . . . . .	29
Благодарности Мартина Вербурга . . . . .	30
<b>Об этой книге</b> . . . . .	31
Как работать с этой книгой . . . . .	32
Для кого предназначена книга . . . . .	33
Дорожная карта . . . . .	33
Соглашения в коде и материал для скачивания . . . . .	34
Требования к программному обеспечению . . . . .	35
<b>Об авторах</b> . . . . .	36
<b>Иллюстрация на обложке</b> . . . . .	37
<b>От издательства</b> . . . . .	38

## Часть 1. Разработка на Java 7

<b>Глава 1. Введение в Java 7</b> . . . . .	41
1.1. Язык и платформа . . . . .	42
1.2. Малое прекрасно — расширения языка Java, или Проект «Монета» . . . . .	44
1.3. Изменения в рамках проекта «Монета» . . . . .	47
1.3.1. Строки в конструкции switch . . . . .	48
1.3.2. Усовершенствованный синтаксис для числовых литералов . . . . .	48
1.3.3. Усовершенствованная обработка исключений . . . . .	51
1.3.4. Использование ресурсов в блоке try (try-with-resources). . . . .	53
1.3.5. Ромбовидный синтаксис. . . . .	56
1.3.6. Упрощенный вызов методов с переменным количеством аргументов. . . . .	57
1.4. Резюме . . . . .	59
<b>Глава 2. Новый ввод-вывод.</b> . . . . .	60
2.1. История ввода-вывода в Java. . . . .	62
2.1.1. Java 1.0–1.3. . . . .	62
2.1.2. Java 1.4 и неблокирующий ввод-вывод . . . . .	63
2.1.3. Введение в NIO.2 . . . . .	64
2.2. Path — основа файлового ввода-вывода . . . . .	64
2.2.1. Создание пути. . . . .	67
2.2.2. Получение информации о пути . . . . .	68
2.2.3. Избавление от избыточности. . . . .	69
2.2.4. Преобразование путей. . . . .	70
2.2.5. Пути NIO.2 и класс File, существующий в Java . . . . .	71
2.3. Работа с каталогами и деревьями каталогов . . . . .	71
2.3.1. Поиск файлов в каталоге. . . . .	72
2.3.2. Движение по дереву каталогов . . . . .	72

2.4. Ввод-вывод файловой системы при работе с NIO.2 . . . . .	74
2.4.1. Создание и удаление файлов. . . . .	75
2.4.2. Копирование и перемещение файлов . . . . .	76
2.4.3. Атрибуты файлов . . . . .	77
2.4.4. Быстрое считывание и запись данных . . . . .	82
2.4.5. Уведомление об изменении файлов. . . . .	83
2.4.6. SeekableByteChannel . . . . .	85
2.5. Асинхронные операции ввода-вывода . . . . .	85
2.5.1. Стиль с ожиданием . . . . .	86
2.5.2. Стиль с применением обратных вызовов . . . . .	89
2.6. Окончательная шлифовка технологии сокет — канал. . . . .	91
2.6.1. NetworkChannel . . . . .	92
2.6.2. MulticastChannel. . . . .	93
2.7. Резюме . . . . .	94

## Часть 2. Необходимые технологии

<b>Глава 3. Внедрение зависимостей</b> . . . . .	98
3.1. Дополнительные знания: понятие об инверсии управления и внедрении зависимостей. . . . .	99
3.1.1. Инверсия управления . . . . .	99
3.1.2. Внедрение зависимостей . . . . .	100
3.1.3. Переход к внедрению зависимостей . . . . .	102
3.2. Стандартизированное внедрение зависимостей в Java . . . . .	107
3.2.1. Аннотация @Inject. . . . .	109
3.2.2. Аннотация @Qualifier. . . . .	110
3.2.3. Аннотация @Named. . . . .	111
3.2.4. Аннотация @Scope . . . . .	112
3.2.5. Аннотация @Singleton . . . . .	113
3.2.6. Интерфейс Provider<T>. . . . .	113

3.3. Guice 3 — эталонная реализация внедрения зависимостей в Java . . . . .	114
3.3.1. Знакомство с Guice . . . . .	115
3.3.2. Морские узлы — различные связи в Guice . . . . .	118
3.3.3. Задание области видимости для внедренных объектов в Guice . . . . .	121
3.4. Резюме . . . . .	123
<b>Глава 4. Современная параллельная обработка. . . . .</b>	<b>124</b>
4.1. Теория параллелизма — базовый пример . . . . .	125
4.1.1. Рассмотрение модели потоков в Java . . . . .	125
4.1.2. Структурные концепции . . . . .	127
4.1.3. Как и в каких случаях возникает конфликт . . . . .	128
4.1.4. Источники издержек . . . . .	129
4.1.5. Пример обработчика транзакций . . . . .	130
4.2. Параллельная обработка с блочной структурой (до Java 5) . . . . .	131
4.2.1. Синхронизация и блокировки . . . . .	132
4.2.2. Модель состояния для потока . . . . .	133
4.2.3. Полностью синхронизированные объекты . . . . .	134
4.2.4. Взаимные блокировки . . . . .	136
4.2.5. Почему synchronized?. . . . .	138
4.2.6. Ключевое слово volatile . . . . .	139
4.2.7. Неизменяемость . . . . .	140
4.3. Составные элементы современных параллельных приложений . . . . .	142
4.3.1. Атомарные классы — java.util.concurrent.atomic . . . . .	142
4.3.2. Блокировки — java.util.concurrent.locks. . . . .	143
4.3.3. CountdownLatch . . . . .	147
4.3.4. ConcurrentHashMap . . . . .	149
4.3.5. CopyOnWriteArrayList . . . . .	150
4.3.6. Очереди . . . . .	153

4.4. Контроль исполнения . . . . .	160
4.4.1. Моделирование задач . . . . .	161
4.4.2. ScheduledThreadPoolExecutor . . . . .	163
4.5. Фреймворк fork/join (ветвление/слияние). . . . .	164
4.5.1. Простой пример fork/join . . . . .	165
4.5.2. ForkJoinTask и захват работы . . . . .	168
4.5.3. Параллелизация проблем . . . . .	168
4.6. Модель памяти языка Java (JMM). . . . .	169
4.7. Резюме . . . . .	172

## **Глава 5. Файлы классов и байт-код. . . . .** 173

5.1. Загрузка классов и объекты классов . . . . .	174
5.1.1. Обзор — загрузка и связывание. . . . .	174
5.1.2. Объекты классов . . . . .	177
5.1.3. Загрузчики классов . . . . .	177
5.1.4. Пример — загрузчики классов при внедрении зависимостей . . . . .	179
5.2. Использование дескрипторов методов. . . . .	181
5.2.1. MethodHandle . . . . .	182
5.2.2. MethodType . . . . .	182
5.2.3. Поиск дескрипторов методов. . . . .	183
5.2.4. Пример: сравнение рефлексии, использования посредников и дескрипторов методов. . . . .	184
5.2.5. Почему стоит выбирать дескрипторы методов . . . . .	187
5.3. Исследование файлов классов. . . . .	188
5.3.1. Знакомство с javap . . . . .	189
5.3.2. Внутренняя форма сигнатур методов. . . . .	189
5.3.3. Пул констант. . . . .	191
5.4. Байт-код . . . . .	193
5.4.1. Пример: дизассемблирование класса. . . . .	194
5.4.2. Среда времени исполнения . . . . .	196

5.4.3. Введение в коды операций . . . . .	197
5.4.4. Коды операций для загрузки и сохранения . . . . .	198
5.4.5. Арифметические коды операций . . . . .	199
5.4.6. Коды операций для контроля исполнения . . . . .	200
5.4.7. Коды операций для активизации . . . . .	201
5.4.8. Коды операций для работы с платформой . . . . .	201
5.4.9. Сокращенные формы записи кодов операций . . . . .	202
5.4.10. Пример: сцепление (конкатенация) строк . . . . .	202
5.5. <code>invokedynamic</code> . . . . .	204
5.5.1. Как работает <code>invokedynamic</code> . . . . .	205
5.5.2. Пример: дизассемблирование <code>invokedynamic</code> -вызова . . . . .	206
5.6. Резюме . . . . .	207
<b>Глава 6. Понятие о повышении производительности . . . . .</b>	<b>209</b>
6.1. Терминологическое описание производительности — базовые определения . . . . .	211
6.1.1. Ожидание . . . . .	211
6.1.2. Пропускная способность . . . . .	212
6.1.3. Коэффициент использования. . . . .	212
6.1.4. Эффективность . . . . .	213
6.1.5. Мощность . . . . .	213
6.1.6. Масштабируемость . . . . .	213
6.1.7. Деградация . . . . .	213
6.2. Прагматический подход к анализу производительности . . . . .	214
6.2.1. Знайте, что именно вы измеряете . . . . .	215
6.2.2. Умейте проводить измерения . . . . .	216
6.2.3. Знайте, какого уровня производительности вы хотите достичь . . . . .	217
6.2.4. Знайте, когда следует прекратить оптимизацию . . . . .	218

6.2.5. Знайте, какой ценой дается повышение производительности . . . . .	218
6.2.6. Знайте об опасности поспешной оптимизации . . . . .	219
6.3. Что пошло не так? И почему нас это должно волновать? . . . . .	220
6.3.1. Закон Мура: прошлые и будущие тенденции изменения производительности . . . . .	221
6.3.2. Понятие об иерархии латентности памяти . . . . .	222
6.3.3. Почему так сложно выполнять оптимизацию производительности в Java . . . . .	224
6.4. Вопрос времени — от железа и вверх . . . . .	225
6.4.1. Аппаратные часы . . . . .	225
6.4.2. Проблема с <code>nanotime()</code> . . . . .	226
6.4.3. Роль времени при повышении производительности . . . . .	229
6.4.4. Практический пример: понятие о кэш-промахах. . . . .	230
6.5. Сборка мусора. . . . .	232
6.5.1. Основы . . . . .	233
6.5.2. Отслеживание и очистка . . . . .	234
6.5.3. <code>jmap</code> . . . . .	236
6.5.4. Полезные переключатели виртуальной машины Java . . . . .	241
6.5.5. Чтение журналов сборщика мусора . . . . .	242
6.5.6. Визуализация использования памяти с помощью <code>VisualVM</code> . . . . .	243
6.5.7. Анализ локальности. . . . .	246
6.5.8. Параллельное отслеживание и очистка . . . . .	248
6.5.9. G1 — новый сборщик мусора для Java . . . . .	249
6.6. Динамическая компиляция с применением <code>HotSpot</code> . . . . .	250
6.6.1. Знакомство с <code>HotSpot</code> . . . . .	252
6.6.2. Встраиваемая подстановка методов. . . . .	254
6.6.3. Динамическая компиляция и мономорфные вызовы. . . . .	255
6.6.4. Чтение журналов компиляции . . . . .	255
6.7. Резюме . . . . .	257

## Часть 3. Многоязычное программирование на виртуальной машине Java

<b>Глава 7. Альтернативные языки для виртуальной машины Java</b> . . . . .	262
7.1. Языку Java не хватает гибкости? Это провокация! . . . . .	263
7.1.1. Система согласования . . . . .	263
7.1.2. Концептуальные основы функционального программирования . . . . .	265
7.1.3. Идиомы словаря и фильтра . . . . .	266
7.2. Языковой зоопарк . . . . .	268
7.2.1. Сравнение интерпретируемых и компилируемых языков . . . . .	269
7.2.2. Сравнение динамической и статической типизации . . . . .	269
7.2.3. Сравнение императивных и функциональных языков . . . . .	270
7.2.4. Сравнение повторной реализации и оригинала . . . . .	271
7.3. Многоязычное программирование на виртуальной машине Java . . . . .	272
7.3.1. Зачем использовать другой язык вместо Java . . . . .	274
7.3.2. Многообещающие языки . . . . .	275
7.4. Как подобрать для проекта другой язык вместо Java . . . . .	276
7.4.1. Высоки ли риски в области проекта . . . . .	277
7.4.2. Насколько хорошо язык взаимодействует с Java . . . . .	278
7.4.3. Имеется ли хороший инструментарий и поддержка данного языка на уровне тестов . . . . .	278
7.4.4. Насколько сложно выучить данный язык . . . . .	279
7.4.5. Насколько много разработчиков использует данный язык . . . . .	279
7.5. Как виртуальная машина Java поддерживает альтернативные языки . . . . .	280
7.5.1. Среда времени исполнения для не Java-языков . . . . .	281
7.5.2. Фикции компилятора . . . . .	281
7.6. Резюме . . . . .	284



<b>Глава 8. Groovy — динамический приятель Java</b> . . . . .	285
8.1. Знакомство с Groovy . . . . .	287
8.1.1. Компиляция и запуск. . . . .	288
8.1.2. Консоль Groovy . . . . .	289
8.2. Groovy 101 — синтаксис и семантика . . . . .	290
8.2.1. Стандартный импорт . . . . .	291
8.2.2. Числовая обработка . . . . .	292
8.2.3. Переменные, сравнение динамических и статических типов, а также контекст . . . . .	293
8.2.4. Синтаксис списков и словарей . . . . .	295
8.3. Отличия от Java — ловушки для новичков . . . . .	296
8.3.1. Опциональные точки с запятой и операторы возврата . . . . .	297
8.3.2. Опциональные скобки для параметров методов . . . . .	297
8.3.3. Модификаторы доступа . . . . .	298
8.3.4. Обработка исключений . . . . .	298
8.3.5. Оператор равенства в Groovy . . . . .	299
8.3.6. Внутренние классы . . . . .	299
8.4. Функции Groovy, пока отсутствующие в Java . . . . .	300
8.4.1. GroovyBeans . . . . .	300
8.4.2. Оператор безопасного разыменования . . . . .	301
8.4.3. Оператор Элвис. . . . .	302
8.4.4. Улучшенные строки. . . . .	303
8.4.5. Функциональные литералы . . . . .	304
8.4.6. Первоклассная поддержка для операций с коллекциями . . . . .	305
8.4.7. Первоклассная поддержка работы с регулярными выражениями . . . . .	307
8.4.8. Простая XML-обработка . . . . .	308
8.5. Взаимодействие между Groovy и Java. . . . .	310
8.5.1. Вызов Java из Groovy. . . . .	311
8.5.2. Вызов Groovy из Java . . . . .	311
8.6. Резюме . . . . .	315

<b>Глава 9. Язык Scala — мощный и лаконичный</b> . . . . .	316
9.1. Быстрый обзор Scala . . . . .	317
9.1.1. Scala — лаконичный язык . . . . .	317
9.1.2. Сопоставимые выражения . . . . .	320
9.1.3. Case-классы . . . . .	322
9.1.4. Акторы . . . . .	324
9.2. Подходит ли Scala для моего проекта? . . . . .	325
9.2.1. Сравнение Scala и Java . . . . .	325
9.2.2. Когда и каким образом приступить к использованию Scala. . . . .	326
9.2.3. Признаки, указывающие, что Scala может не подойти для вашего проекта . . . . .	327
9.3. Как вновь сделать код красивым с помощью Scala . . . . .	327
9.3.1. Использование компилятора и REPL. . . . .	328
9.3.2. Выведение типов. . . . .	329
9.3.3. Методы . . . . .	330
9.3.4. Импорт . . . . .	331
9.3.5. Циклы и управляющие структуры . . . . .	332
9.3.6. Функциональное программирование на Scala. . . . .	333
9.4. Объектная модель Scala — знакомая, но своеобразная . . . . .	334
9.4.1. Любая сущность — это объект. . . . .	335
9.4.2. Конструкторы . . . . .	336
9.4.3. Типажи . . . . .	337
9.4.4. Одиночка и объект-спутник . . . . .	339
9.4.5. Case-классы и сопоставимые выражения . . . . .	342
9.4.6. Предостережение . . . . .	344
9.5. Структуры данных и коллекции . . . . .	345
9.5.1. Список . . . . .	346
9.5.2. Словарь . . . . .	350
9.5.3. Обобщенные типы. . . . .	351

9.6. Знакомство с актерами . . . . .	354
9.6.1. Весь код — театр. . . . .	355
9.6.2. Обмен информацией с актерами через почтовый ящик . . .	356
9.7. Резюме . . . . .	358

## **Глава 10. Clojure: программирование повышенной надежности . . . . .**

10.1. Введение в Clojure . . . . .	360
10.1.1. Hello World на языке Clojure . . . . .	361
10.1.2. Знакомство с REPL . . . . .	362
10.1.3. Как делаются ошибки . . . . .	363
10.1.4. Учимся любить скобки . . . . .	363
10.2. Поиск Clojure — синтаксис и семантика . . . . .	364
10.2.1. Базовый курс по работе со специальными формами . . . . .	365
10.2.2. Списки, векторы, словари и множества . . . . .	366
10.2.3. Арифметика, проверка на равенство и другие операции .	369
10.3. Работа с функциями и циклами в Clojure . . . . .	370
10.3.1. Простые функции Clojure . . . . .	370
10.3.2. Макросы чтения и диспетчеризация. . . . .	373
10.3.3. Функциональное программирование и замыкания . . . . .	375
10.4. Введение в последовательности Clojure . . . . .	377
10.4.1. Ленивые последовательности . . . . .	379
10.4.2. Последовательности и функции с переменным количеством аргументов . . . . .	380
10.5. Взаимодействие между Clojure и Java. . . . .	382
10.5.1. Вызов Java из Clojure . . . . .	382
10.5.2. Тип Java у значений Clojure . . . . .	383
10.5.3. Использование посредников Clojure . . . . .	384
10.5.4. Исследовательское программирование в среде REPL . . . . .	385
10.5.5. Использование Clojure из Java . . . . .	386

10.6. Параллелизм в Clojure . . . . .	386
10.6.1. Функции future и pcall . . . . .	387
10.6.2. Ссылки . . . . .	389
10.6.3. Агенты . . . . .	393
10.7. Резюме . . . . .	394

## **Часть 4. Создание многоязычного проекта**

<b>Глава 11.</b> Разработка через тестирование . . . . .	397
11.1. Суть разработки через тестирование . . . . .	399
11.1.1. Образец разработки через тестирование с одним случаем использования . . . . .	400
11.1.2. Образец разработки через тестирование с несколькими случаями использования . . . . .	405
11.1.3. Дальнейшие размышления о цикле «красный — зеленый — рефакторинг» . . . . .	408
11.1.4. JUnit . . . . .	410
11.2. Тестовые двойники . . . . .	412
11.2.1. Пустой объект . . . . .	413
11.2.2. Объект-заглушка . . . . .	415
11.2.3. Поддельный объект . . . . .	419
11.2.4. Подставной объект . . . . .	425
11.3. Знакомство со ScalaTest . . . . .	427
11.4. Резюме . . . . .	429
<b>Глава 12.</b> Сборка и непрерывная интеграция . . . . .	431
12.1. Знакомство с Maven 3 . . . . .	434
12.2. Экспресс-проект с Maven 3 . . . . .	435
12.3. Maven 3 — сборка java7developer . . . . .	438
12.3.1. Файл POM . . . . .	438
12.3.2. Запуск примеров . . . . .	445

12.4. Jenkins — обеспечение непрерывной интеграции . . . . .	448
12.4.1. Базовая конфигурация. . . . .	450
12.4.2. Настройка задачи . . . . .	452
12.4.3. Выполнение задачи . . . . .	455
12.5. Параметры кода в Maven и Jenkins. . . . .	457
12.5.1. Установка плагинов Jenkins . . . . .	458
12.5.2. Обеспечение согласованности кода с помощью плагина Checkstyle . . . . .	459
12.5.3. Обеспечение качества кода с помощью FindBugs . . . . .	461
12.6. Leiningen . . . . .	464
12.6.1. Знакомство с Leiningen. . . . .	465
12.6.2. Архитектура Leiningen . . . . .	465
12.6.3. Пример: Hello Lein . . . . .	466
12.6.4. REPL-ориентированная разработка через тестирование с применением Leiningen . . . . .	469
12.6.5. Упаковка и развертывание кода с помощью Leiningen . . .	471
12.7. Резюме . . . . .	472

## **Глава 13.** Быстрая веб-разработка . . . . . 474

13.1. Проблема с веб-фреймворками на основе Java. . . . .	475
13.1.1. Почему компиляция Java не подходит для быстрой веб-разработки. . . . .	476
13.1.2. Почему статическая типизация не подходит для быстрой веб-разработки. . . . .	477
13.2. Критерии при выборе веб-фреймворка . . . . .	478
13.3. Знакомство с Grails . . . . .	480
13.4. Экспресс-проект с Grails. . . . .	481
13.4.1. Создание объекта предметной области . . . . .	483
13.4.2. Разработка через тестирование . . . . .	483
13.4.3. Сохраняемость объектов предметной области . . . . .	486
13.4.4. Создание тестовых данных . . . . .	487
13.4.5. Контроллеры . . . . .	488

13.4.6. Виды GSP/JSP . . . . .	489
13.4.7. Скаффолдинг и автоматическое создание пользовательского интерфейса . . . . .	491
13.4.8. Быстрая циклическая разработка . . . . .	492
13.5. Дальнейшее исследование Grails . . . . .	492
13.5.1. Логирование . . . . .	493
13.5.2. GORM — объектно-реляционное отображение . . . . .	493
13.5.3. Плагины Grails . . . . .	494
13.6. Знакомство с Comprojure . . . . .	495
13.6.1. Hello World с Comprojure . . . . .	496
13.6.2. Ring и маршруты . . . . .	499
13.6.3. Hiccup . . . . .	500
13.7. Пример проекта с Comprojure: «А не выдра ли я?» . . . . .	500
13.7.1. Настройка программы «А не выдра ли я?» . . . . .	502
13.7.2. Основные функции в программе «А не выдра ли я?» . . . . .	504
13.8. Резюме . . . . .	507
<b>Глава 14. О сохранении основательности . . . . .</b>	<b>509</b>
14.1. Чего ожидать в Java 8 . . . . .	509
14.1.1. Лямбда-выражения (замыкания) . . . . .	510
14.1.2. Модуляризация (проект Jigsaw) . . . . .	512
14.2. Многоязычное программирование . . . . .	514
14.2.1. Межъязыковые взаимодействия и метаобъектные протоколы . . . . .	515
14.2.2. Многоязычная модуляризация . . . . .	516
14.3. Будущие тенденции параллелизма . . . . .	517
14.3.1. Многоядерный мир . . . . .	517
14.3.2. Параллельная обработка, управляемая во время исполнения . . . . .	518

14.4. Новые направления в развитии виртуальной машины Java . . . . .	519
14.4.1. Конвергенция виртуальных машин . . . . .	520
14.4.2. Сопрограммы . . . . .	521
14.4.3. Кортежи . . . . .	522
14.5. Резюме . . . . .	525

## Приложения

### Приложение А. Установка исходного кода

java7developer . . . . .	528
А.1. Структура исходного кода java7developer . . . . .	528
А.2. Скачивание и установка Maven . . . . .	530
А.3. Запуск сборки java7developer . . . . .	532
А.3.1. Однократная подготовка сборки . . . . .	532
А.3.2. Очистка . . . . .	533
А.3.3. Компиляция . . . . .	534
А.3.4. Тестирование . . . . .	535
А.4. Резюме . . . . .	536

### Приложение В. Синтаксис и примеры паттернов

подстановки . . . . .	537
В.1. Синтаксис паттернов подстановки . . . . .	537
В.2. Примеры паттернов подстановки . . . . .	537

### Приложение С. Установка альтернативных языков для виртуальной машины Java . . . . .

С.1. Groovy . . . . .	539
С.1.1. Скачивание Groovy . . . . .	539
С.1.2. Установка Groovy . . . . .	540

С.2. Scala . . . . .	542
С.3. Clojure. . . . .	543
С.4. Grails. . . . .	543
С.4.1. Скачивание Grails . . . . .	543
С.4.2. Установка Grails . . . . .	544
<b>Приложение D. Скачивание и установка Jenkins . . . . .</b>	<b>547</b>
D.1. Загрузка Jenkins . . . . .	547
D.2. Установка Jenkins . . . . .	547
D.2.1. Запуск WAR-файла . . . . .	548
D.2.2. Установка WAR-файла. . . . .	548
D.2.3. Установка специализированного пакета . . . . .	548
D.2.4. Первый запуск Jenkins. . . . .	548
<b>Приложение E. java7developer — Maven POM . . . . .</b>	<b>550</b>
E.1. Конфигурация сборки . . . . .	550
E.2. Управление зависимостями . . . . .	554