

---

## Содержание

<b>Предисловие</b>	<b>17</b>
<b>Благодарности</b>	<b>24</b>
<b>Часть I. Описание</b>	<b>27</b>
<b>Глава 1. Вводный пример</b>	<b>29</b>
1.1. Готическая безопасность	29
1.1.1. Контроллер мисс Грант	30
1.2. Модель конечного автомата	31
1.3. Программирование контроллера мисс Грант	34
1.4. Языки и семантическая модель	41
1.5. Использование генерации кода	43
1.6. Использование языковых инструментальных средств	46
1.7. Визуализация	48
<b>Глава 2. Использование предметно-ориентированных языков</b>	<b>49</b>
2.1. Определение предметно-ориентированных языков	49
2.1.1. Границы DSL	51
2.1.2. Фрагментарные и автономные DSL	53
2.2. Зачем используется DSL	54
2.2.1. Повышение производительности разработки	54
2.2.2. Общение с экспертами в предметной области	55
2.2.3. Изменения в контексте выполнения	56
2.2.4. Альтернативные вычислительные модели	57
2.3. Проблемы с DSL	57
2.3.1. Языковая какофония	58
2.3.2. Стоимость построения	58
2.3.3. Язык гетто	59
2.3.4. Ограниченная абстракция	59
2.4. Более широкая обработка языка	60
2.5. Жизненный цикл DSL	60
2.6. Что такое хороший дизайн DSL	62
<b>Глава 3. Реализация предметно-ориентированных языков</b>	<b>63</b>
3.1. Архитектура обработки DSL	63
3.2. Работа синтаксического анализатора	67
3.3. Грамматики, синтаксис и семантики	68
3.4. Анализ данных	69
3.5. Макросы	72
3.6. Тестирование DSL	72
3.6.1. Тестирование семантической модели	73
3.6.2. Тестирование синтаксического анализатора	76
3.6.3. Тестирование сценариев	80

3.7. Обработка ошибок	80
3.8. Миграция DSL	82
<b>Глава 4. Реализация внутреннего DSL</b>	<b>85</b>
4.1. Свободные API и API командных запросов	85
4.2. Необходимость в слое синтаксического анализа	88
4.3. Использование функций	89
4.4. Коллекции литералов	93
4.5. Использование грамматик для выбора внутренних элементов	95
4.6. Замыкания	96
4.7. Работа с синтаксическим деревом	98
4.8. Аннотации	100
4.9. Расширение литералов	101
4.10. Снижение синтаксического шума	101
4.11. Динамический отклик	102
4.12. Проверка типов	103
<b>Глава 5. Реализация внешнего DSL</b>	<b>105</b>
5.1. Стратегия синтаксического анализа	105
5.2. Стратегия получения вывода	108
5.3. Концепции синтаксического анализа	110
5.3.1. Лексический анализ	110
5.3.2. Грамматики и языки	111
5.3.3. Регулярные, контекстно-свободные и контекстно-зависимые грамматики	112
5.3.4. Нисходящий и восходящий синтаксический анализ	114
5.4. Смешивание с другим языком	115
5.5. XML DSL	117
<b>Глава 6. Выбор между внутренними и внешними DSL</b>	<b>119</b>
6.1. Обучение	119
6.2. Стоимость построения	120
6.3. Осведомленность программистов	121
6.4. Общение с экспертами предметной области	121
6.5. Смешивание в базовом языке	121
6.6. Границы строгой выразительности	122
6.7. Настройка времени выполнения	123
6.8. Скатывание в обобщенность	123
6.9. Соединение DSL	124
6.10. Подведение итогов	124
<b>Глава 7. Альтернативные вычислительные модели</b>	<b>127</b>
7.1. Несколько альтернативных моделей	130
7.1.1. Таблицы решений	130
7.1.2. Система правил вывода	130
7.1.3. Конечный автомат	132
7.1.4. Сеть зависимостей	132
7.1.5. Выбор модели	133

<b>Глава 8. Генерация кода</b>	<b>135</b>
8.1. Выбор объекта генерации	136
8.2. Как генерировать код	138
8.3. Смешивание сгенерированного и рукописного кодов	139
8.4. Генерация удобочитаемого кода	140
8.5. Предварительный анализ генерации кода	141
8.6. Источники дополнительной информации	141
<b>Глава 9. Языковые инструментальные средства</b>	<b>143</b>
9.1. Элементы языковых инструментальных средств	143
9.2. Языки определения схем и метамодели	144
9.3. Редактирование исходного текста и проекционное редактирование	149
9.3.1. Множественные представления	151
9.4. Иллюстративное программирование	151
9.5. Тур по инструментам	153
9.6. Языковые инструментальные средства и CASE-инструменты	154
9.7. Следует ли использовать языковые инструментальные средства	155
<b>Часть II. Общие вопросы</b>	<b>157</b>
<b>Глава 10. Зоопарк DSL</b>	<b>159</b>
10.1. Graphviz	159
10.2. JMock	160
10.3. CSS	162
10.4. Hibernate Query Language (HQL )	163
10.5. XAML	164
10.6. FIT	166
10.7. Make и другие	167
<b>Глава 11. Семантическая модель</b>	<b>171</b>
11.1. Как это работает	171
11.2. Когда это использовать	173
11.3. Вводный пример (Java )	174
<b>Глава 12. Таблица символов</b>	<b>177</b>
12.1. Как это работает	177
12.1.1. Статически типизированные символы	179
12.2. Когда это использовать	180
12.3. Дополнительная литература	180
12.4. Сеть зависимостей во внешнем DSL (Java и ANTLR )	180
12.5. Использование символьных ключей во внутреннем DSL (Ruby )	182
12.6. Использование перечислений для статически типизированных символов (Java )	183
<b>Глава 13. Переменная контекста</b>	<b>187</b>
13.1. Как это работает	187
13.2. Когда это использовать	188
13.3. Чтение INI-файла (C#)	188

<b>Глава 14. Построитель конструкции</b>	<b>191</b>
14.1. Как это работает	191
14.2. Когда это использовать	192
14.3. Построение простых полетных данных (C#)	192
<b>Глава 15. Макрос</b>	<b>195</b>
15.1. Как это работает	195
15.1.1. Текстовые макросы	196
15.1.2. Синтаксические макросы	199
15.2. Когда это использовать	202
<b>Глава 16. Уведомление</b>	<b>205</b>
16.1. Как это работает	205
16.2. Когда это использовать	206
16.3. Очень простое уведомление (C#)	206
16.4. Уведомление анализа (Java )	207
<b>Часть III. Вопросы создания внешних DSL</b>	<b>211</b>
<b>Глава 17. Трансляция, управляемая разделителями</b>	<b>213</b>
17.1. Как это работает	213
17.2. Когда это использовать	216
17.3. Карты постоянных клиентов (C#)	216
17.3.1. Семантическая модель	217
17.3.2. Синтаксический анализатор	219
17.4. Синтаксический анализ неавтономных инструкций контроллера мисс Грант (Java )	221
<b>Глава 18. Синтаксически управляемая трансляция</b>	<b>229</b>
18.1. Как это работает	230
18.1.1. Лексический анализатор	231
18.1.2. Синтаксический анализатор	233
18.1.3. Генерация вывода	235
18.1.4. Семантические предикаты	236
18.2. Когда это использовать	236
18.3. Дополнительная литература	236
<b>Глава 19. Форма Бэкуса–Наура</b>	<b>237</b>
19.1. Как это работает	237
19.1.1. Символы множественности (операторы Клини)	239
19.1.2. Другие полезные операторы	240
19.1.3. Грамматики, разбирающие выражения	240
19.1.4. Преобразование РБНФ в БНФ	241
19.1.5. Действия	243
19.2. Когда это использовать	245

<b>Глава 20. Лексический анализатор на основе таблицы регулярных выражений</b>	<b>247</b>
20.1. Как это работает	248
20.2. Когда это использовать	249
20.3. Лексический анализатор контроллера мисс Грант (Java)	249
<b>Глава 21. Синтаксический анализатор на основе рекурсивного спуска</b>	<b>253</b>
21.1. Как это работает	254
21.2. Когда это использовать	257
21.3. Дополнительная литература	257
21.4. Рекурсивный спуск и контроллер мисс Грант (Java )	257
<b>Глава 22. Комбинатор синтаксических анализаторов</b>	<b>263</b>
22.1. Как это работает	264
22.1.1. Выполнение действий	267
22.1.2. Функциональный стиль комбинаторов	268
22.2. Когда это использовать	268
22.3. Комбинаторы синтаксических анализаторов и контроллер мисс Грант (Java )	269
<b>Глава 23. Генератор синтаксических анализаторов</b>	<b>277</b>
23.1. Как это работает	277
23.1.1. Встроенные действия	278
23.2. Когда это использовать	280
23.3. Hello World (Java и ANTLR )	280
23.3.1. Написание базовой грамматики	281
23.3.2. Построение синтаксического анализатора	282
23.3.3. Добавление кода действий в грамматику	284
23.3.4. Применение шаблона Generation Gap	286
<b>Глава 24. Построение дерева</b>	<b>289</b>
24.1. Как это работает	289
24.2. Когда это использовать	291
24.3. Использование синтаксиса построения дерева ANTLR (Java и ANTLR )	292
24.3.1. Токенизация	293
24.3.2. Синтаксический анализ	294
24.3.3. Наполнение семантической модели	296
24.4. Построение дерева с использованием кода действий (Java и ANTLR )	299
<b>Глава 25. Встроенная трансляция</b>	<b>305</b>
25.1. Как это работает	305
25.2. Когда это использовать	306
25.3. Контроллер мисс Грант (Java и ANTLR)	306
<b>Глава 26. Встроенная интерпретация</b>	<b>311</b>
26.1. Как это работает	311
26.2. Когда это использовать	311
26.3. Калькулятор (ANTLR и Java )	312

<b>Глава 27. Внешний код</b>	<b>315</b>
27.1. Как это работает	315
27.2. Когда это использовать	317
27.3. Встраивание динамического кода (ANTLR, Java и Javascript)	317
27.3.1. Семантическая модель	318
27.3.2. Синтаксический анализатор	320
<b>Глава 28. Альтернативная токенизация</b>	<b>325</b>
28.1. Как это работает	325
28.1.1. Кавычки	326
28.1.2. Лексическое состояние	328
28.1.3. Изменение типа токена	330
28.1.4. Игнорирование типов токенов	331
28.2. Когда это использовать	332
<b>Глава 29. Вложенные операторные выражения</b>	<b>333</b>
29.1. Как это работает	333
29.1.1. Применение восходящих синтаксических анализаторов	334
29.1.2. Нисходящие синтаксические анализаторы	335
29.2. Когда это использовать	337
<b>Глава 30. Символ новой строки в качестве разделителя</b>	<b>339</b>
30.1. Как это работает	339
30.2. Когда это использовать	341
<b>Глава 31. Прочие вопросы</b>	<b>343</b>
31.1. Синтаксические отступы	343
31.2. Модулярная грамматика	345
<b>Часть IV. Вопросы создания внутренних DSL</b>	<b>347</b>
<b>Глава 32. Построитель выражений</b>	<b>349</b>
32.1. Как это работает	350
32.2. Когда это использовать	350
32.3. Свободный календарь с строителем и без него (Java )	351
32.4. Использование для календаря нескольких строителей (Java )	353
<b>Глава 33. Последовательность функций</b>	<b>357</b>
33.1. Как это работает	357
33.2. Когда это использовать	358
33.3. Простая конфигурация компьютера (Java)	358
<b>Глава 34. Вложенные функции</b>	<b>361</b>
34.1. Как это работает	361
34.2. Когда это использовать	363
34.3. Простой пример конфигурации компьютера (Java )	363
34.4. Обработка различных аргументов с помощью токенов (C#)	365

34.5. Использование токенов подтипов для поддержки интегрированной среды разработки (Java)	366
34.6. Использование инициализаторов объектов (C#)	368
34.7. Повторяющиеся события (C#)	369
34.7.1. Семантическая модель	369
34.7.2. DSL	372
<b>Глава 35. Соединение методов в цепочки</b>	<b>375</b>
35.1. Как это работает	375
35.1.1. Построители или значения	377
35.1.2. Проблема окончания	377
35.1.3. Иерархическая структура	378
35.1.4. Последовательные интерфейсы	378
35.2. Когда это использовать	379
35.3. Простой пример конфигурации компьютера (Java )	379
35.4. Соединение методов в цепочки со свойствами (C#)	383
35.5. Последовательные интерфейсы (C#)	383
<b>Глава 36. Перенос области видимости в объект</b>	<b>387</b>
36.1. Как это работает	388
36.2. Когда это использовать	388
36.3. Коды безопасности (C#)	389
36.3.1. Семантическая модель	389
36.3.2. DSL	391
36.4. Использование вычисления экземпляра (Ruby )	393
36.5. Использование инициализатора экземпляра (Java )	395
<b>Глава 37. Замыкание</b>	<b>397</b>
37.1. Как это работает	397
37.2. Когда это использовать	401
<b>Глава 38. Вложенные замыкания</b>	<b>403</b>
38.1. Как это работает	403
38.2. Когда это использовать	405
38.3. Заворачивание последовательности функций во вложенное замыкание (Ruby)	405
38.4. Простой пример (C#)	407
38.5. Применение соединения методов в цепочки (Ruby )	408
38.6. Последовательность функций с явными аргументами замыканий (Ruby )	410
38.7. Применение вычисления экземпляра (Ruby )	411
<b>Глава 39. Список литералов</b>	<b>415</b>
39.1. Как это работает	415
39.2. Когда это использовать	415
<b>Глава 40. Ассоциативные массивы литералов</b>	<b>417</b>
40.1. Как это работает	417
40.2. Когда это использовать	418

40.3. Настройка конфигурации компьютера с помощью списков и отображений (Ruby )	418
40.4. Имитация Lisp (Ruby )	419
<b>Глава 41. Динамический отклик</b>	<b>423</b>
41.1. Как это работает	424
41.2. Когда это использовать	424
41.3. Расчет бонусов с помощью анализа имен методов (Ruby )	426
41.3.1. Модель	426
41.3.2. Построитель	428
41.4. Расчет бонусов с помощью цепочек вызовов (Ruby )	429
41.4.1. Модель	430
41.4.2. Построитель	430
41.5. Уменьшение шума в коде контроллера тайника (JRuby )	433
<b>Глава 42. Аннотации</b>	<b>439</b>
42.1. Как это работает	440
42.1.1. Определение аннотации	440
42.1.2. Обработка аннотаций	441
42.2. Когда это использовать	442
42.3. Пользовательский синтаксис с обработкой времени выполнения (Java )	443
42.4. Использование метода класса (Ruby )	445
42.5. Динамическая генерация кода (Ruby )	446
<b>Глава 43. Работа с синтаксическим деревом</b>	<b>449</b>
43.1. Как это работает	449
43.2. Когда это использовать	450
43.3. Генерация запросов IMAP из условий C# (C#)	451
43.3.1. Семантическая модель	452
43.3.2. Построение из исходного текста C#	454
43.3.3. Отступление	458
<b>Глава 44. Класс таблицы символов</b>	<b>461</b>
44.1. Как это работает	462
44.2. Когда это использовать	462
44.3. Статически типизированный класс таблицы символов (Java )	463
<b>Глава 45. Шлифовка текста</b>	<b>469</b>
45.1. Как это работает	469
45.2. Когда это использовать	470
45.3. Шлифовка правил дисконта (Ruby )	470
<b>Глава 46. Расширение литералов</b>	<b>473</b>
46.1. Как это работает	473
46.2. Когда это использовать	474
46.3. Ингредиенты рецепта (C#)	474



<b>Часть V. Альтернативные вычислительные модели</b>	<b>477</b>
<b>Глава 47. Адаптивная модель</b>	<b>479</b>
47.1. Как это работает	480
47.1.1. Внедрение императивного кода в адаптивную модель	481
47.1.2. Инструментарий	483
47.2. Когда это использовать	483
<b>Глава 48. Таблицы принятия решений</b>	<b>485</b>
48.1. Как это работает	485
48.2. Когда это использовать	487
48.3. Вычисление оплаты заказа (C#)	487
48.3.1. Модель	487
48.3.2. Синтаксический анализатор	491
<b>Глава 49. Сеть зависимостей</b>	<b>495</b>
49.1. Как это работает	496
49.2. Когда это использовать	498
49.3. Анализ зелий (C#)	498
49.3.1. Семантическая модель	499
49.3.2. Синтаксический анализатор	500
<b>Глава 50. Система правил вывода</b>	<b>503</b>
50.1. Как это работает	504
50.1.1. Цепочки выводов	505
50.1.2. Противоречивые выводы	505
50.1.3. Шаблоны в структуре правила	506
50.2. Когда это использовать	507
50.3. Проверка для членства в клубе (C#)	507
50.3.1. Модель	507
50.3.2. Синтаксический анализатор	508
50.3.3. Развитие DSL	509
50.4. Правила избрания: расширение членства в клубе (C#)	511
50.4.1. Модель	512
50.4.2. Синтаксический анализатор	514
<b>Глава 51. Конечный автомат</b>	<b>517</b>
51.1. Как это работает	517
51.2. Когда это использовать	519
51.3. Контроллер тайника (Java)	519
<b>Часть VI. Генерация кода</b>	<b>521</b>
<b>Глава 52. Генерация с помощью преобразователя</b>	<b>523</b>
52.1. Как это работает	523
52.2. Когда это использовать	524
52.3. Контроллер тайника (Java генерирует C )	525

<b>Глава 53. Шаблонная генерация</b>	<b>529</b>
53.1. Как это работает	529
53.2. Когда это использовать	531
53.3. Генерация конечного автомата тайника с помощью вложенных условных конструкций (Velocity и Java, генерирующие C )	531
<b>Глава 54. Встроенный помощник</b>	<b>537</b>
54.1. Как это работает	538
54.2. Когда это использовать	538
54.3. Состояния тайника (Java и ANTLR )	539
54.4. Должен ли помощник генерировать HTML (Java и Velocity)	541
<b>Глава 55. Генерация, осведомленная о модели</b>	<b>545</b>
55.1. Как это работает	546
55.2. Когда это использовать	546
55.3. Конечный автомат тайника (C )	546
55.4. Динамическая загрузка конечного автомата (C )	553
<b>Глава 56. Генерация, игнорирующая модель</b>	<b>557</b>
56.1. Как это работает	557
56.2. Когда это использовать	558
56.3. Конечный автомат тайника как вложенные условные конструкции (C )	558
<b>Глава 57. Отделение генерируемого кода с помощью наследования</b>	<b>561</b>
57.1. Как это работает	562
57.2. Когда это использовать	563
57.3. Генерация классов из схемы данных (Java и немного Ruby)	563
<b>Список литературы</b>	<b>569</b>
<b>Предметный указатель</b>	<b>570</b>