

# **Building domain-specific languages with Groovy**

Yaroslav Yermilov

Senior Software Engineer, EPAM Systems

# About me

Yaroslav Yermilov <epam>



Senior Software Engineer, student

<https://ua.linkedin.com/pub/yaroslav-yermilov/58/682/506>

<https://www.facebook.com/yaroslav.yermilov>

<https://twitter.com/yermilov17>

fond of Java, complete code, big data, data science, Groovy

интерфейса.

Когда я говорил, что нужно писать технические статьи, вы продолжали читать хабр и обсуждать компании. Теперь у нас хабр в шоколаде, а технические статьи никто не пишет.

Когда я говорил, что нельзя всех подряд называть «синьорами», вы продолжали их создавать. Теперь у нас куча 23-летних синьоров и все равно х\*р его знает, чем абстрактный класс отличается от интерфейса.

Когда я говорил, что люди важнее процессов, вы продолжали устраивать agile тусовки и устанавливать скрам-доски. Теперь у нас тотальный скрам, а проекты, согласитесь, делаются все так же хер\*во.

<http://megamozg.ru/post/7256/>

# DSL?

A domain-specific language (DSL) is a computer language specialized to a particular application domain. This is in contrast to a general-purpose language (GPL), which is broadly applicable across domains, and lacks specialized features for a particular domain.



# Quiz!



<https://github.com/yermilov/groovy-dsl/tree/develop/examples/dsl-or-not-a-dsl>

# Quiz!

```
1 try {
2     Map<String, String> parameters = new HashMap<>();
3     parameters.put("apiId", apiId);
4     parameters.put("phoneNumber", smsMessage.getToPhoneNumber());
5     parameters.put("text", smsMessage.getText());
6
7     ensureRestTemplateWasInjected();
8
9     restTemplate.postForLocation(SMS_SEND_URL, null, parameters);
10
11     log.info("Send SMS message: " + smsMessage);
12 } catch (RestClientException e) {
13     log.error("Can't send SMS message", e);
14     throw new SmsSenderException("Can't send SMS message", e);
15 }
```



<https://github.com/yermilov/groovy-dsl/tree/develop/examples/dsl-or-not-a-dsl>

# Quiz!



```
1 smsSender.sendMessage(new SmsMessage("Hello World!", "380934903624"));
```

# Quiz!



```
1 sendSms 'Hello World!', to: 380934903624
```

# Quiz!

```
1 @Autowired  
2 Environment environment
```



<https://github.com/yermilov/groovy-dsl/tree/develop/examples/dsl-or-not-a-dsl>



# Quiz!



```
1 Properties properties = new Properties();  
2 properties.load(this.getClass().getResourceAsStream('/application.properties'));
```

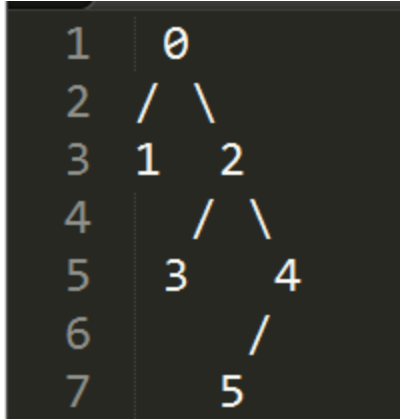
# Quiz!



```
1 data <- read.table("data/household_power_consumption.txt",
2   header = TRUE, sep = ";", na.strings = '?',
3   colClasses = c('character', 'character', 'numeric', 'numeric', 'numeric', 'numeric', 'numeric', 'numeric', 'numeric')
4
5 data <- data[data[, 'Date'] == '1/2/2007' | data[, 'Date'] == '2/2/2007', ]
6
7 hist(data$'Global_active_power',
8   col = "red", main = "Global Active Power",
9   xlab = "Global Active Power (kilowatts)", ylab = "Frequency"))
```

<https://github.com/yermilov/groovy-dsl/tree/develop/examples/dsl-or-not-a-dsl>

# Quiz!



<https://github.com/yermilov/groovy-dsl/tree/develop/examples/dsl-or-not-a-dsl>

# Quiz!

## What is the name of the “-->” operator?

3545 ▲ After reading [Hidden Features and Dark Corners of C++/STL](#) on [comp.lang.c++.moderated](#), I was completely surprised that it compiled and worked in both Visual Studio 2008 and G++ 4.4.

▼ The code:



```
#include <stdio.h>
int main()
{
    int x = 10;
    while( x --> 0 ) // x goes to 0
    {
        printf("%d ", x);
    }
}
```

I'd assume this is C, since it works in GCC as well. Where is this defined in the standard, and where has it come from?

c++

c

operators

standards-compliance



<http://stackoverflow.com/questions/1642028/what-is-the-name-of-the-operator>

# Quiz!

```

1  #include
2  #stdio.h
3  #define V()
4  #define
5  #define
6  #define
7  #define
8  #define
9  #define
10 #define
11 #define
12 #define
13
14 #include
15 #stdio.h
16
17 int
18 "q,u,v,i,c,b,r,f,w,m,n,f,w,x,y,z,u,p,s,w,v,r,e,r,f,u,b,z,d,y,g,s" V()
19
20 "s"
21 "t00a"
22
23 "s"
24 "t00a"
25
26 "s"
27 "t00a"
28
29 "s"
30 "t00a"
31
32 "s"
33 "t00a"
34
35 "s"
36 "t00a"
37
38 "s"
39 "t00a"
40
41 "s"
42 "t00a"
43
44 "s"
45 "t00a"
46
47 "s"
48 "t00a"
49
50 "s"
51 "t00a"
52
53 "s"
54 "t00a"
55
56 "s"
57 "t00a"
58
59 "s"
60 "t00a"
61
62 "s"
63 "t00a"
64
65 "s"
66 "t00a"
67
68 "s"
69 "t00a"
70
71 "s"
72 "t00a"
73
74 "s"
75 "t00a"
76
77 "s"
78 "t00a"
79
80 "s"
81 "t00a"
82
83 "s"
84 "t00a"
85
86 "s"
87 "t00a"
88
89 "s"
90 "t00a"
91
92 "s"
93 "t00a"
94
95 "s"
96 "t00a"
97
98 "s"
99 "t00a"
100
101 "s"
102 "t00a"
103
104 "s"
105 "t00a"
106
107 "s"
108 "t00a"
109
110 "s"
111 "t00a"
112
113 "s"
114 "t00a"
115
116 "s"
117 "t00a"
118
119 "s"
120 "t00a"
121
122 "s"
123 "t00a"
124
125 "s"
126 "t00a"
127
128 "s"
129 "t00a"
130
131 "s"
132 "t00a"
133
134 "s"
135 "t00a"
136
137 "s"
138 "t00a"
139
140 "s"
141 "t00a"
142
143 "s"
144 "t00a"
145
146 "s"
147 "t00a"
148
149 "s"
150 "t00a"
151
152 "s"
153 "t00a"
154
155 "s"
156 "t00a"
157
158 "s"
159 "t00a"
160
161 "s"
162 "t00a"
163
164 "s"
165 "t00a"
166
167 "s"
168 "t00a"
169
170 "s"
171 "t00a"
172
173 "s"
174 "t00a"
175
176 "s"
177 "t00a"
178
179 "s"
180 "t00a"
181
182 "s"
183 "t00a"
184
185 "s"
186 "t00a"
187
188 "s"
189 "t00a"
190
191 "s"
192 "t00a"
193
194 "s"
195 "t00a"
196
197 "s"
198 "t00a"
199
200 "s"
201 "t00a"
202
203 "s"
204 "t00a"
205
206 "s"
207 "t00a"
208
209 "s"
210 "t00a"
211
212 "s"
213 "t00a"
214
215 "s"
216 "t00a"
217
218 "s"
219 "t00a"
220
221 "s"
222 "t00a"
223
224 "s"
225 "t00a"
226
227 "s"
228 "t00a"
229
230 "s"
231 "t00a"
232
233 "s"
234 "t00a"
235
236 "s"
237 "t00a"
238
239 "s"
240 "t00a"
241
242 "s"
243 "t00a"
244
245 "s"
246 "t00a"
247
248 "s"
249 "t00a"
250
251 "s"
252 "t00a"
253
254 "s"
255 "t00a"
256
257 "s"
258 "t00a"
259
260 "s"
261 "t00a"
262
263 "s"
264 "t00a"
265
266 "s"
267 "t00a"
268
269 "s"
270 "t00a"
271
272 "s"
273 "t00a"
274
275 "s"
276 "t00a"
277
278 "s"
279 "t00a"
280
281 "s"
282 "t00a"
283
284 "s"
285 "t00a"
286
287 "s"
288 "t00a"
289
290 "s"
291 "t00a"
292
293 "s"
294 "t00a"
295
296 "s"
297 "t00a"
298
299 "s"
300 "t00a"
301
302 "s"
303 "t00a"
304
305 "s"
306 "t00a"
307
308 "s"
309 "t00a"
310
311 "s"
312 "t00a"
313
314 "s"
315 "t00a"
316
317 "s"
318 "t00a"
319
320 "s"
321 "t00a"
322
323 "s"
324 "t00a"
325
326 "s"
327 "t00a"
328
329 "s"
330 "t00a"
331
332 "s"
333 "t00a"
334
335 "s"
336 "t00a"
337
338 "s"
339 "t00a"
340
341 "s"
342 "t00a"
343
344 "s"
345 "t00a"
346
347 "s"
348 "t00a"
349
350 "s"
351 "t00a"
352
353 "s"
354 "t00a"
355
356 "s"
357 "t00a"
358
359 "s"
360 "t00a"
361
362 "s"
363 "t00a"
364
365 "s"
366 "t00a"
367
368 "s"
369 "t00a"
370
371 "s"
372 "t00a"
373
374 "s"
375 "t00a"
376
377 "s"
378 "t00a"
379
380 "s"
381 "t00a"
382
383 "s"
384 "t00a"
385
386 "s"
387 "t00a"
388
389 "s"
390 "t00a"
391
392 "s"
393 "t00a"
394
395 "s"
396 "t00a"
397
398 "s"
399 "t00a"
400
401 "s"
402 "t00a"
403
404 "s"
405 "t00a"
406
407 "s"
408 "t00a"
409
410 "s"
411 "t00a"
412
413 "s"
414 "t00a"
415
416 "s"
417 "t00a"
418
419 "s"
420 "t00a"
421
422 "s"
423 "t00a"
424
425 "s"
426 "t00a"
427
428 "s"
429 "t00a"
430
431 "s"
432 "t00a"
433
434 "s"
435 "t00a"
436
437 "s"
438 "t00a"
439
440 "s"
441 "t00a"
442
443 "s"
444 "t00a"
445
446 "s"
447 "t00a"
448
449 "s"
450 "t00a"
451
452 "s"
453 "t00a"
454
455 "s"
456 "t00a"
457
458 "s"
459 "t00a"
460
461 "s"
462 "t00a"
463
464 "s"
465 "t00a"
466
467 "s"
468 "t00a"
469
470 "s"
471 "t00a"
472
473 "s"
474 "t00a"
475
476 "s"
477 "t00a"
478
479 "s"
480 "t00a"
481
482 "s"
483 "t00a"
484
485 "s"
486 "t00a"
487
488 "s"
489 "t00a"
490
491 "s"
492 "t00a"
493
494 "s"
495 "t00a"
496
497 "s"
498 "t00a"
499
500 "s"
501 "t00a"
502
503 "s"
504 "t00a"
505
506 "s"
507 "t00a"
508
509 "s"
510 "t00a"
511
512 "s"
513 "t00a"
514
515 "s"
516 "t00a"
517
518 "s"
519 "t00a"
520
521 "s"
522 "t00a"
523
524 "s"
525 "t00a"
526
527 "s"
528 "t00a"
529
530 "s"
531 "t00a"
532
533 "s"
534 "t00a"
535
536 "s"
537 "t00a"
538
539 "s"
540 "t00a"
541
542 "s"
543 "t00a"
544
545 "s"
546 "t00a"
547
548 "s"
549 "t00a"
550
551 "s"
552 "t00a"
553
554 "s"
555 "t00a"
556
557 "s"
558 "t00a"
559
560 "s"
561 "t00a"
562
563 "s"
564 "t00a"
565
566 "s"
567 "t00a"
568
569 "s"
570 "t00a"
571
572 "s"
573 "t00a"
574
575 "s"
576 "t00a"
577
578 "s"
579 "t00a"
580
581 "s"
582 "t00a"
583
584 "s"
585 "t00a"
586
587 "s"
588 "t00a"
589
590 "s"
591 "t00a"
592
593 "s"
594 "t00a"
595
596 "s"
597 "t00a
```



```

14 #define q\  

15 Z[F++]=*E  

16 #define $\  

17 for("IOCCC"  

18 #include\  

19 <stdlib.h>  

20  

21 int *Q,u,i,c,k,B,r=0,w,n,F=0,x,J,u,  

22  

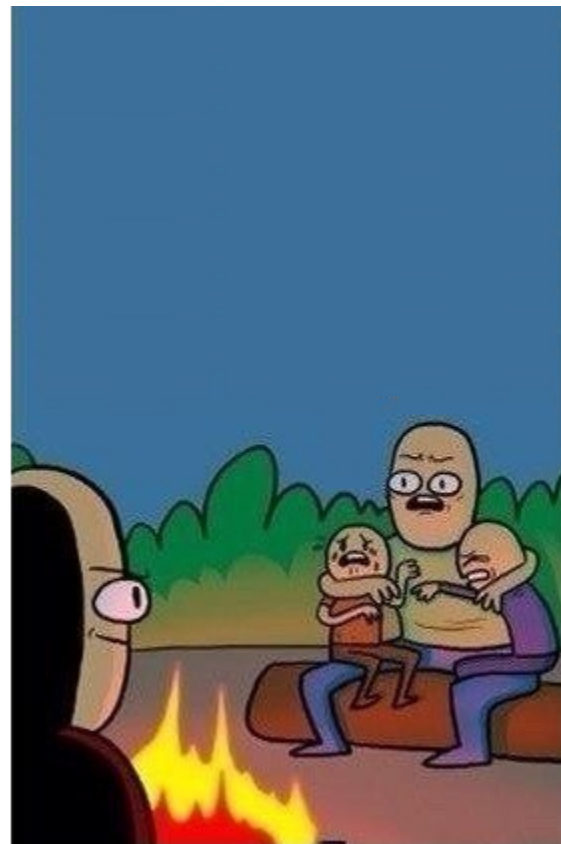
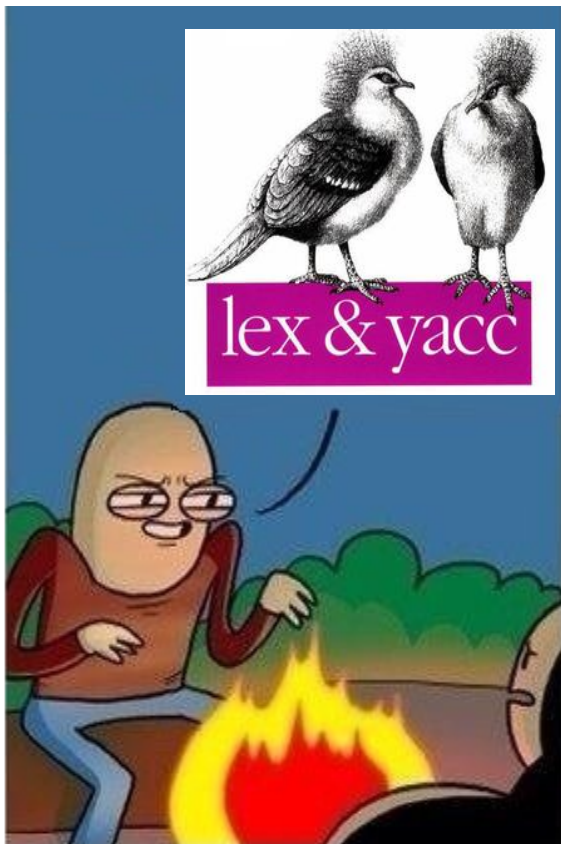
23 "-S"  

24 "Zd&o+"

```

<https://github.com/vermilov/groovy-dsl/tree/develop/examples/dsl-or-not-a-dsl>

# How to build DSL?



# Groovy?

Groovy

powerful, optionally typed

dynamic

static-typing and static compilation

familiar and easy to learn syntax.

Domain-Specific Language

meta-programming

functional

<http://www.groovy-lang.org/>



# Groovy

Google groovy



(150 SEATS) WEB	TRACK C (150 SEATS) JVM LANGUAGES	TRACK D (250 SEATS) BIG DATA/CLOUD
Registration and welcome coffee		
Opening of the second conference day (Track A)		
you wanted to writing async urrency HTTP n Java, but we d to ask agursky (Israel)	<a href="#">Building domain-specific languages with Groovy</a> <a href="#">Yaroslav Yermilov</a> (Ukraine)	<a href="#">Fault tolerance possibilities</a> <a href="#">Izzet Mustafayev</a>
Break		
<a href="#">eFuture is here</a> <a href="#">uyko</a> (Russia)	<a href="#">Scala Rock-Painting</a> <a href="#">Dmytro Mantula</a> (Ukraine)	<a href="#">The moderniz legacy system microservices, b</a> <a href="#">Rxlava</a> <a href="#">Holger Kraus</a> (C)
Break		
<a href="#">ff you go</a> <a href="#">ikov-Tarnovski</a> tonia)	<a href="#">Groovy under MacroScope</a> <a href="#">Sergei Egorov</a> (Estonia) and <a href="#">Baruch Sadogursky</a> (Israel)	<a href="#">hat's in You</a> <a href="#">Josh Long</a> (C)
Lunch time		
<a href="#">ication I have dreamt of</a> <a href="#">chuk</a> (Ukraine)	<a href="#">About concurrency abstractions with Observable's Future's, Akka (actors) in Scala</a> <a href="#">Jacek Laskowski</a> (Poland)	<a href="#">App infrastru microservices w Cloud 1.1</a> <a href="#">Igor Khotin</a> (U)
Break		
<a href="#">d Evented I/O s on the JVM</a> <a href="#">er</a> (Germany)	<a href="#">Productivity in Scala</a> <a href="#">Alexander Podkajuzin</a> (Russia)	<a href="#">Big Data: from i to elephant – tra legacy solutio Hadoop infras</a> <a href="#">Roman Nikit</a> (Ukraine)
Coffee break		
<a href="#">aster with fewer it possible?</a> <a href="#">shnikov</a> (Russia)	<a href="#">The Epic Groovy Puzzlers (Season 2): The revenge of the parentheses</a> <a href="#">Evgeny Borisov</a> and <a href="#">Baruch Sadogursky</a> (Israel)	<a href="#">How to Mani request per seq manne</a> <a href="#">Valerii Moisi</a> (Ukraine)

YOU ARE HERE

Google groove





```

1 package com.jeeconf.groovydsl;
2
3 import org.apache.commons.io.IOUtils;
4 import org.springframework.web.client.RestTemplate;
5
6 import java.time.LocalDateTime;
7 import java.util.HashMap;
8 import java.util.Map;
9
10 public class NotADsl {
11
12     private final static RestTemplate REST_TEMPLATE = new RestTemplate();
13     private final static String URL = "http://sms.ru/sms/send?api_id={apiId}&to={phoneNumber}&text={text}";
14
15     public static void main(String[] args) {
16         String phoneNumber = "380934902436";
17         Long period = 30000;
18
19         try {
20             String apiKey = IOUtils.toString(NotADsl.class.getResourceAsStream("/api.key"));
21
22             while (true) {
23                 Map<String, String> parameters = new HashMap<>();
24                 parameters.put("apiId", apiKey);
25                 parameters.put("phoneNumber", phoneNumber);
26                 parameters.put("text", "So Far, So Good... (at " + LocalDateTime.now());
27
28                 REST_TEMPLATE.postForLocation(URL, null, parameters);
29                 Thread.sleep(period);
30             }
31         } catch (Exception e) {
32             throw new RuntimeException(e);
33         }
34     }
35 }
36

```



```

1 package com.jeeconf.groovydsl
2
3 import com.jeeconf.groovydsl.Monitoring;
4
5 class JavaDsl {
6
7     public static void main(String[] args) {
8         Monitoring.sendStatusPeriodically("380934902436", 30000);
9     }
10 }

```

<https://github.com/yermilov/groovy-dsl>

# LET'S



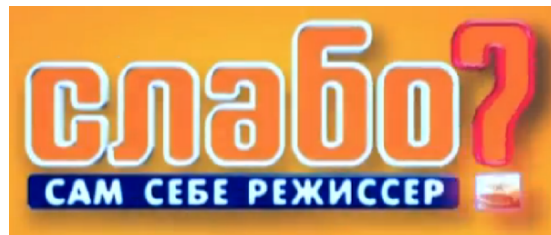
Notes:

concentrate more on possibilities

concentrate less on technical details

something is unclear - ask

want to discuss - you are welcome after the talk



<https://github.com/yeramilov/groovy-dsl>

```
1 Monitoring.sendStatusPeriodically('380934902436', 30000)
```



<https://github.com/yermilov/groovy-dsl>

```
1 '380934902436'.sendStatusPeriodically every: 30000
```



<https://github.com/yermilov/groovy-dsl>

```
1 me << every(30.seconds)
```



<https://github.com/yermilov/groovy-dsl>

```
1 me.notifyEvery30Seconds()
```



<https://github.com/yermilov/groovy-dsl>

```
1 status {  
2     to me  
3     period 30.seconds  
4     notMoreThan 2.times  
5 }
```



<https://github.com/yermilov/groovy-dsl>



```
1 send status.withSchedule(to: me) {  
2     schedule(period: 5.seconds, exactly: 2.times)  
3     schedule(period: 30.seconds, exactly: 5.times)  
4 }
```



<https://github.com/yermilov/groovy-dsl>

```
1 please send status to:me every 30.seconds
```



<https://github.com/yermilov/groovy-dsl>

```
1 each 7 seconds - '380934902436'
```



<https://github.com/yermilov/groovy-dsl>

# Groovy existing DSLs

```
1 apply plugin: 'java'
2 apply plugin: 'groovy'
3
4 repositories {
5     jcenter()
6 }
7
8 dependencies {
9     compile 'org.codehaus.groovy:groovy-all:2.3.6'
10
11     compile 'commons-io:commons-io:2.4'
12
13     compile 'org.springframework:spring-web:4.1.5.RELEASE'
14
15     testCompile 'org.spockframework:spock-core:0.7-groovy-2.0'
16     testCompile 'cglib:cglib-nodep:2.2'
17     testCompile 'org.objenesis:objenesis:1.2'
18 }
```



<https://github.com/yermilov/groovy-dsl>

# Groovy existing DSLs

```
Monitoring monitoring = Spy(Monitoring)

def 'void sendStatus(String phoneNumber, long period)'() {
  given:
    String phoneNumber = '322-223-322'
    Long period = 1234

  when:
    2 * monitoring.sleep(period) >> { /* do nothing */ }

    2 * monitoring.now() >>> [ 'now-1', 'now-2' ]

    monitoring.sendStatus(phoneNumber, period, 2)

  then:
    1 * monitoring.sendMessage(phoneNumber, 'So Far, So Good... (at now-1)')
    1 * monitoring.sendMessage(phoneNumber, 'So Far, So Good... (at now-2)')
}
```



<https://github.com/yermilov/groovy-dsl>

# Groovy existing DSLs

```
1  JsonBuilder builder = new JsonBuilder()
2  builder.records {
3      car {
4          name 'HSV Maloo'
5          make 'Holden'
6          year 2006
7          country 'Australia'
8          record {
9              type 'speed'
10             description 'production pickup truck with speed of 271kph'
11         }
12     }
13 }
```

# Groovy existing DSLs

```
1 def writer = new StringWriter()
2 def xml = new MarkupBuilder(writer)
3
4 xml.records() {
5     car(name:'HSV Maloo', make:'Holden', year:2006) {
6         country('Australia')
7         record(type:'speed', 'Production Pickup Truck with speed of 271kph')
8     }
9     car(name:'Royale', make:'Bugatti', year:1931) {
10        country('France')
11        record(type:'price', 'Most Valuable Car at $15 million')
12    }
13 }
14
```

# Groovy existing DSLs

```
1 def method = request.method
2
3 if (!session) {
4     session = request.getSession(true)
5 }
6
7 if (!session.groovlet) {
8     session.groovlet = 'Groovlets rock!'
9 }
10
11 html.html {
12     head {
13         title 'Groovlet info'
14     }
15     body {
16         h1 'General info'
17         ul {
18             li "Method: ${method}"
19             li "RequestURI: ${request.requestURI}"
20             li "session.groovlet: ${session.groovlet}"
21             li "application.version: ${context.version}"
22         }
23
24         h1 'Headers'
25         ul {
26             headers.each {
27                 li "${it.key} = ${it.value}"
28             }
29         }
30     }
31 }
```

<https://github.com/yermilov/groovy-dsl/tree/develop/examples/existing-dsls>



# Groovy existing DSLs

```
1 import groovy.swing.SwingBuilder
2 import javax.swing.*
3
4 def swing = new SwingBuilder()
5
6 def sharedPanel = {
7     swing.panel() {
8         label("Shared Panel")
9     }
10 }
11
12 count = 0
13 swing.edt {
14     frame(title: 'Frame', defaultCloseOperation: JFrame.EXIT_ON_CLOSE, pack: true, show: true) {
15         vbox {
16             textlabel = label('Click the button!')
17             button(
18                 text: 'Click Me',
19                 actionPerformed: {
20                     count++
21                     textlabel.text = "Clicked ${count} time(s)."
22                     println "Clicked!"
23                 }
24             )
25             widget(sharedPanel())
26             widget(sharedPanel())
27         }
28     }
29 }
```

<https://github.com/yermilov/groovy-dsl/tree/develop/examples/existing-dsls>

# Looking further

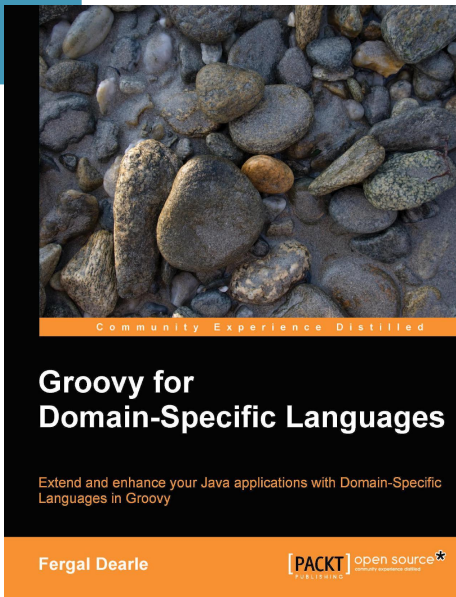


A multi-faceted language for the Java platform

Groovy is a powerful, optimally typed and dynamic language, with static typing and static compilation capabilities, for the Java platform aimed at multiplying developer productivity thanks to a concise, familiar and easy-to-learn syntax. It integrates smoothly with any Java program, and immediately delivers to your application powerful features, including scripting capabilities, Domain-Specific Language authoring, runtime and compile-time meta-programming and functional programming.

- Flat learning curve**  
Concise, readable and expressive syntax, easy to learn for Java developers
- Powerful features**  
Closures, builders, runtime & compile-time meta-programming, functional programming, type inference, and static compilation
- Smooth Java integration**  
Seamlessly and transparently integrates and inter-operates with Java and any third-party libraries
- Domain-Specific Languages**  
Flexible & modifiable syntax, advanced integration & customization mechanisms, to integrate readable business rules in your applications
- Vibrant and rich ecosystem**
- Scripting and testing glue**

<http://www.groovy-lang.org/>



Community Experience Distilled

## Groovy for Domain-Specific Languages

Extend and enhance your Java applications with Domain-Specific Languages in Groovy

Fergal Dearle

**PACKT** open source  
publishing

### Table of contents

1. Command chains
2. Operator overloading
3. Script base classes
- 3.1. The Script class
- 3.2. The @BScript annotation
- 3.3. Alternate abstract method
4. Adding properties to numbers
5. @DelegatesTo
- 5.1. Explaining delegation strategy at compile time
- 5.2. @DelegatesTo
- 5.3. DelegatesTo modes
- 5.3.1. Simple delegation
- 5.3.2. Delegation strategy
- 5.3.3. Delegate to parameter
- 5.3.4. Multiple closures
- 5.3.5. Delegating to a generic type
- 5.3.6. Delegating to an arbitrary type
6. Compilation customizers

## Domain-Specific Languages

### 1. Command chains

Groovy lets you omit parentheses around the arguments of a method call for top-level statements. "command chain" feature extends this by allowing us to chain such parentheses-free method calls, requiring neither parentheses around arguments, nor dots between the chained calls. The general idea is that a call like `a.b.c.d` will actually be equivalent to `a(b).c(d)`. This also works with multiple arguments, closure arguments, and even named arguments. Furthermore, such command chains can also appear on the right-hand side of assignments. Let's have a look at some examples supported by this new syntax:

```
// equivalent to: turn(left), then(right)
turn left then right

// equivalent to: take(2.pills).of(chloroquine).after(6.hours)
take 2.pills of chloroquine after 6.hours

// equivalent to: paint(wall), with(red, green), and(yellow)
paint wall with red, green and yellow

// with named parameters too
// equivalent to: check(that: margarita).tastes(good)
check that: margarita tastes good

// with closures as parameters
// equivalent to: given(i).when(i).then(i)
given { } when { } then { }
```

It is also possible to use methods in the chain which take no arguments, but in that case, the parentheses are needed:

```
// equivalent to: select(all).unique().from(names)
select all unique() from names
```

If your command chain contains an odd number of elements, the chain will be composed of method / arguments, and will finish by a final property access:

```
// equivalent to: take(3).cookies
// and also this: take(3).getCookies()
take 3 cookies
```

<http://www.groovy-lang.org/dsls.html>

# Thanks!

It's now safe to turn off  
your computer.