Building domain-specific languages with Groovy

Yaroslav Yermilov
Senior Software Engineer, EPAM Systems

About me

Yaroslav Yermilov

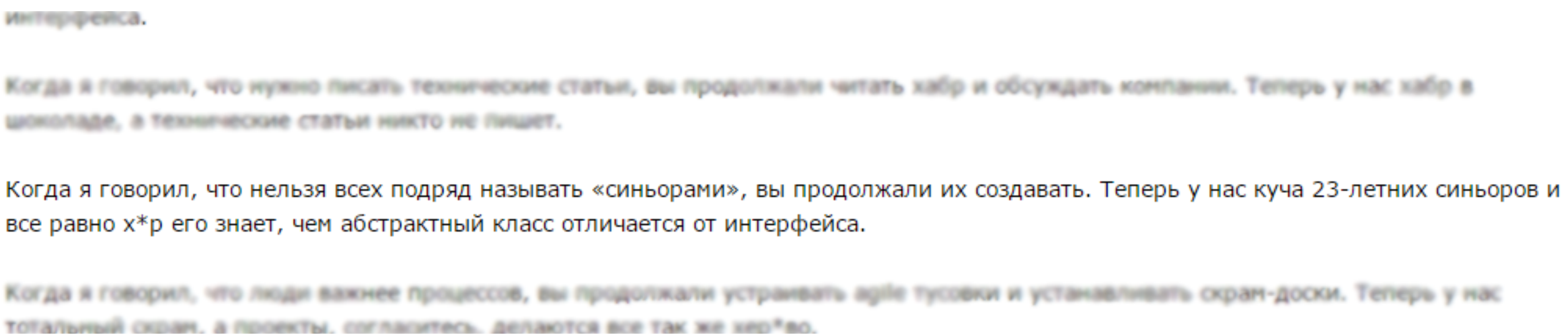
Senior Software Engineer, student

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?

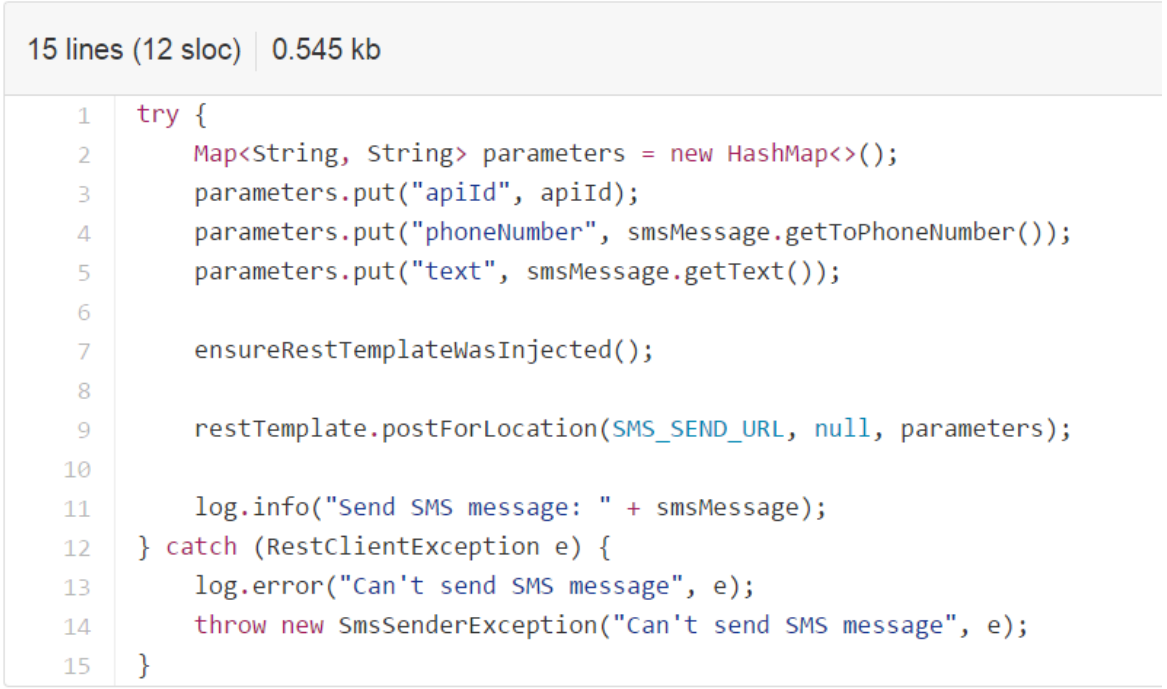
A domain-specific language (DSL) is a computer language specialized to a particular application domain. This is in contrast to a general-purpose language (GPL), which is broadly applicable across domains, and lacks specialized features for a particular domain.

http://en.wikipedia.org/wiki/Domain-specific_language

Quiz!

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

Quiz!

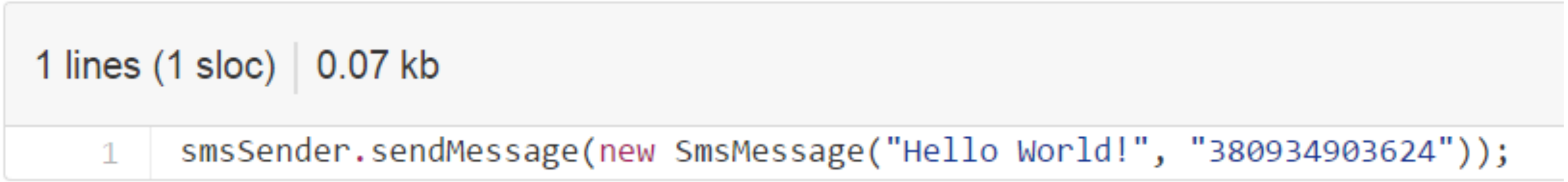


15 lines (12 sloc) | 0.545 kb

```java
try {
    Map<String, String> parameters = new HashMap<>();
    parameters.put("apiId", apiId);
    parameters.put("phoneNumber", smsMessage.getToPhoneNumber());
    parameters.put("text", smsMessage.getText());

    ensureRestTemplateWasInjected();

    restTemplate.postForLocation(SMS_SEND_URL, null, parameters);

    log.info("Send SMS message: " + smsMessage);
} catch (RestClientException e) {
    log.error("Can't send SMS message", e);
    throw new SmsSenderException("Can't send SMS message", e);
}
```

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

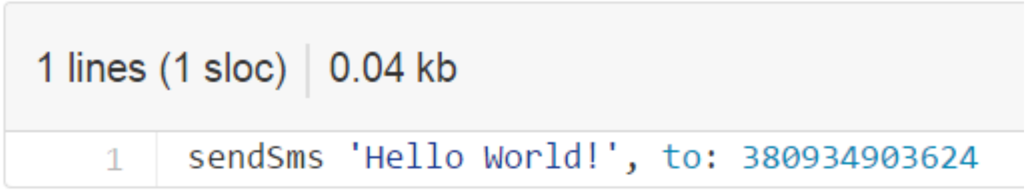
Quiz!


1 lines (1 sloc) | 0.07 kb

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

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

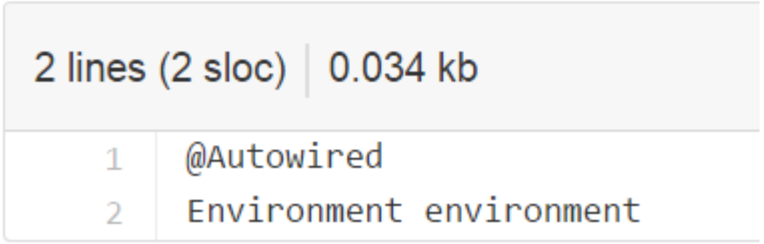
Quiz!


1 lines (1 sloc) | 0.04 kb

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

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

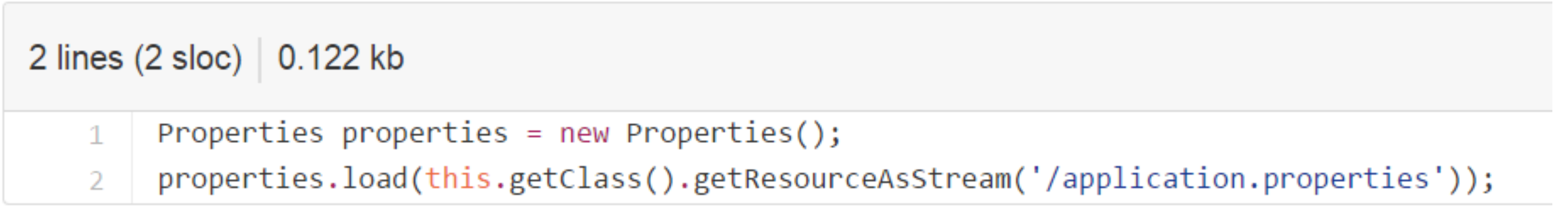
Quiz!


2 lines (2 sloc) | 0.034 kb

```
@Autowired
Environment environment
```

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

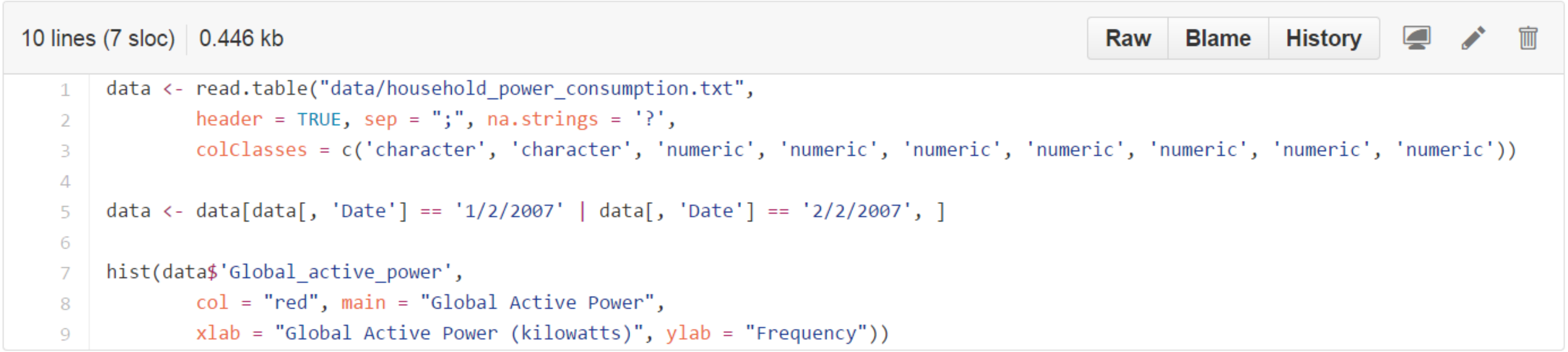
Quiz!


2 lines (2 sloc) | 0.122 kb

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

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

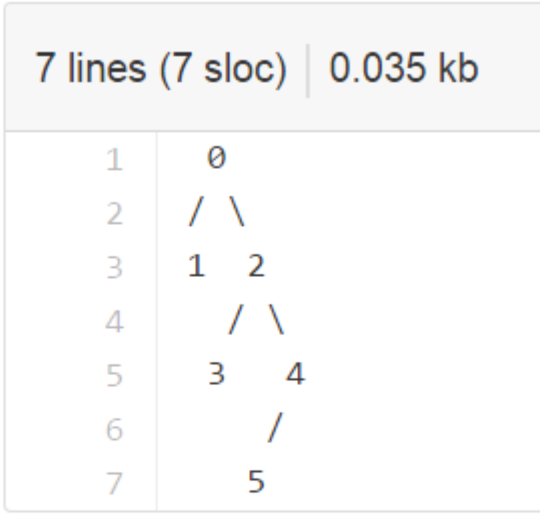
Quiz!



```r
data <- read.table("data/household_power_consumption.txt",
        header = TRUE, sep = ";", na.strings = '?',
        colClasses = c('character', 'character', 'numeric', 'numeric', 'numeric', 'numeric', 'numeric', 'numeric', 'numeric'))

data <- data[data[, 'Date'] == '1/2/2007' | data[, 'Date'] == '2/2/2007', ]

hist(data$'Global_active_power',
        col = "red", main = "Global Active Power",
        xlab = "Global Active Power (kilowatts)", ylab = "Frequency"))
```

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

Quiz!

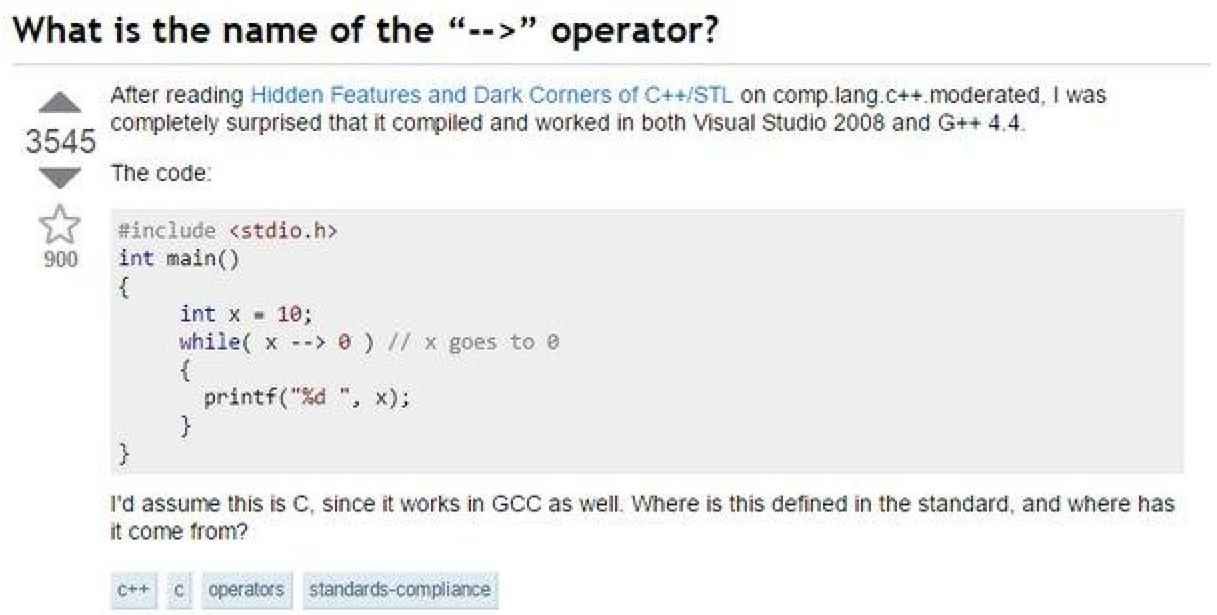


7 lines (7 sloc) | 0.035 kb

```
  0
 / \
 1   2
    / \
  3    4
      /
     5
```

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

Quiz!



What is the name of the "-->" operator?

3545

900

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:

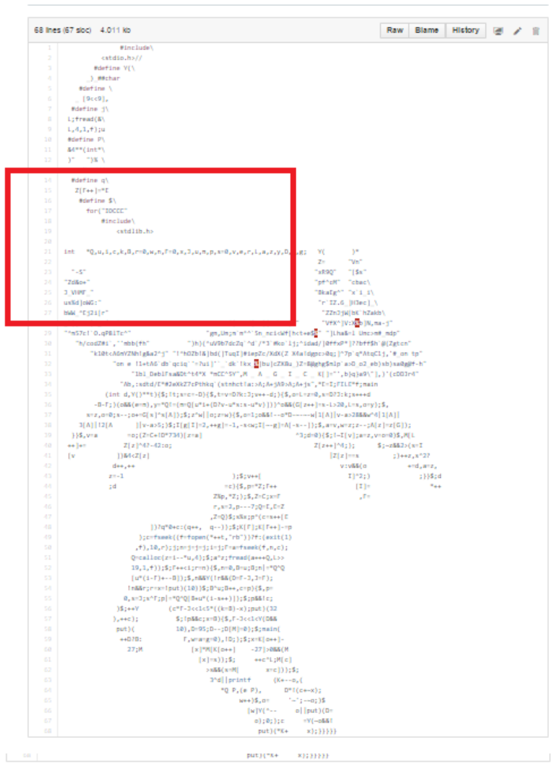
```
#include <stdio.h>
int main()
{
     int x = 10;
     while( x --> 0 ) // x goes to 0
     {
       printf("%d ", x);
     }
}
```

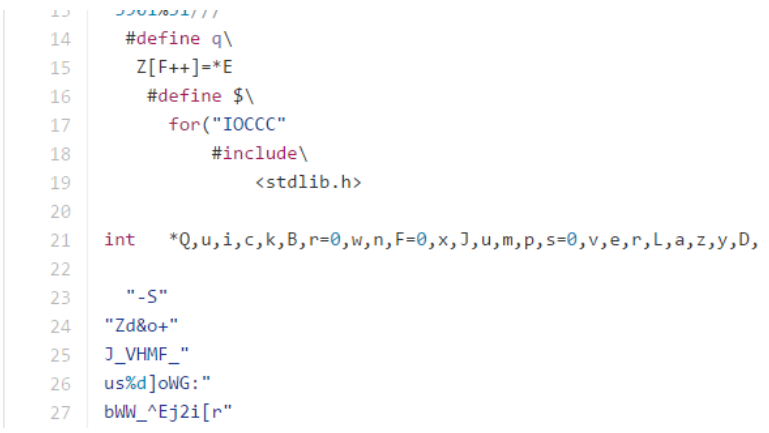
I'd assume this is C, since it works in GCC as well. Where is this defined in the standard, and where has it come from?

c++ c operators standards-compliance

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

Quiz!





```
     #define q\
      Z[F++]=*E
       #define $\
         for("IOCCC"
             #include\
                 <stdlib.h>

int   *Q,u,i,c,k,B,r=0,w,n,F=0,x,J,u,m,p,s=0,v,e,r,L,a,z,y,D,

   "-S"
"Zd&o+"
J_VHMF_"
us%d]oWG:"
bWW_^Ej2i[r"
```

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

How to build DSL?

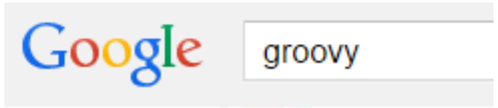
lex & yacc
Компиляторы
Принципы, технологии, инструменты
Альфред Ахо
Рави Сети
Джеффри Ульман

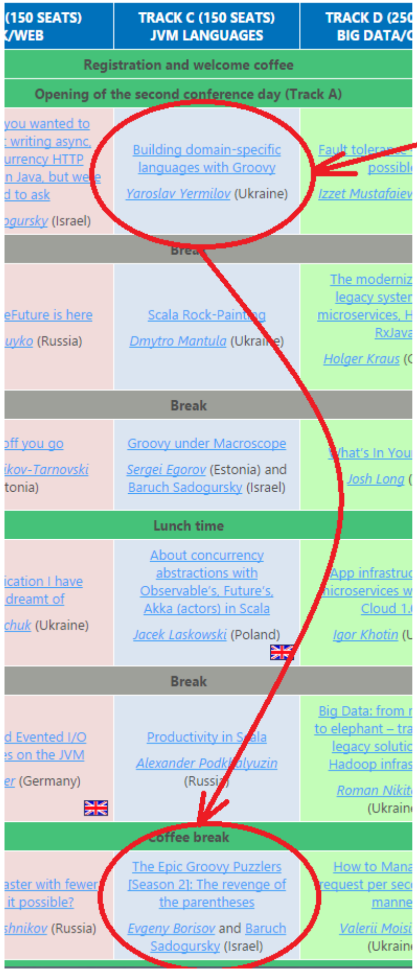
Groovy?

Groovy is a **powerful**, **optionally typed** and **dynamic** language, with **static-typing and static compilation** capabilities, for the Java platform aimed at multiplying developers' 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.

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

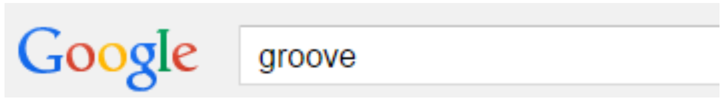
Groovy

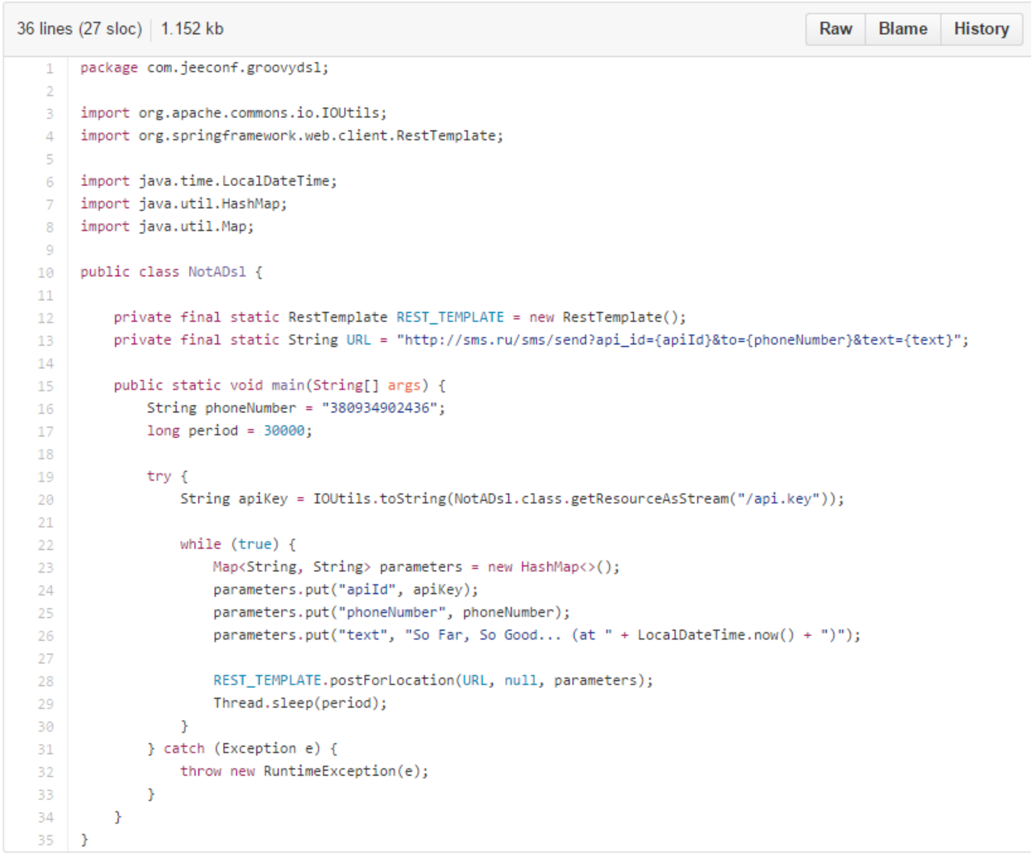




(150 SEATS) /WEB	TRACK C (150 SEATS) JVM LANGUAGES	TRACK D (250 BIG DATA/C
Registration and welcome coffee		
Opening of the second conference day (Track A)		
you wanted to writing async urrency HTTP n Java, but we e d to ask ogursky (Israel)	Building domain-specific languages with Groovy Yaroslav Yermilov (Ukraine)	Fault tolera possibl Izzet Mustafaiev
Brea		
eFuture is here uyko (Russia)	Scala Rock-Painting Dmytro Mantula (Ukraine)	The moderniz legacy syster microservices, H RxJava Holger Kraus (C
Break		
off you go ikov-Tarnovski tonia)	Groovy under Macroscope Sergei Egorov (Estonia) and Baruch Sadogursky (Israel)	What's In You Josh Long (
Lunch time		
ication I have dreamt of chuk (Ukraine)	About concurrency abstractions with Observable's, Future's, Akka (actors) in Scala Jacek Laskowski (Poland) 🇬🇧	App infrastru icroservices w Cloud 1. Igor Khotin (U
Break		
d Evented I/O s on the JVM er (Germany) 🇬🇧	Productivity in Scala Alexander Podkhalyuzin (Russia)	Big Data: from r to elephant – tra legacy solutic Hadoop infras Roman Nikit (Ukrain
Coffee break		
aster with fewer it possible? shnikov (Russia)	The Epic Groovy Puzzlers [Season 2]: The revenge of the parentheses Evgeny Borisov and Baruch Sadogursky (Israel)	How to Mana request per sec manne Valerii Moisi (Ukrain

YOU ARE HERE



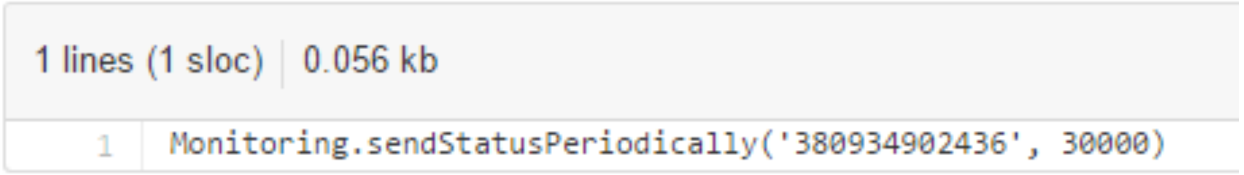


```java
package com.jeeconf.groovydsl;

import org.apache.commons.io.IOUtils;
import org.springframework.web.client.RestTemplate;

import java.time.LocalDateTime;
import java.util.HashMap;
import java.util.Map;

public class NotADsl {

    private final static RestTemplate REST_TEMPLATE = new RestTemplate();
    private final static String URL = "http://sms.ru/sms/send?api_id={apiId}&to={phoneNumber}&text={text}";

    public static void main(String[] args) {
        String phoneNumber = "380934902436";
        long period = 30000;

        try {
            String apiKey = IOUtils.toString(NotADsl.class.getResourceAsStream("/api.key"));

            while (true) {
                Map<String, String> parameters = new HashMap<>();
                parameters.put("apiId", apiKey);
                parameters.put("phoneNumber", phoneNumber);
                parameters.put("text", "So Far, So Good... (at " + LocalDateTime.now() + ")");

                REST_TEMPLATE.postForLocation(URL, null, parameters);
                Thread.sleep(period);
            }
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }
}
```

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

Notes:
concentrate more on possibilities
concentrate less on technical details
something is unclear - ask
want to discuss - you are welcome after the talk

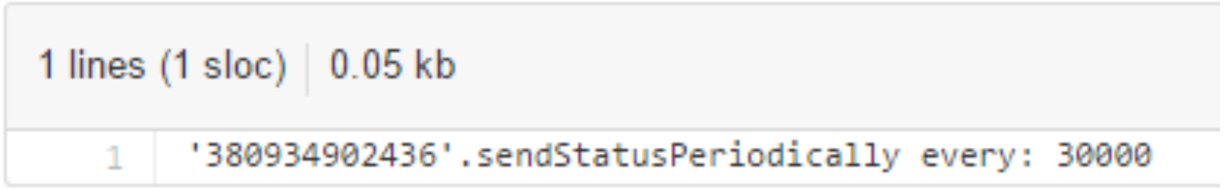
https://github.com/yermilov/groovy-dsl



1 lines (1 sloc) | 0.056 kb

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

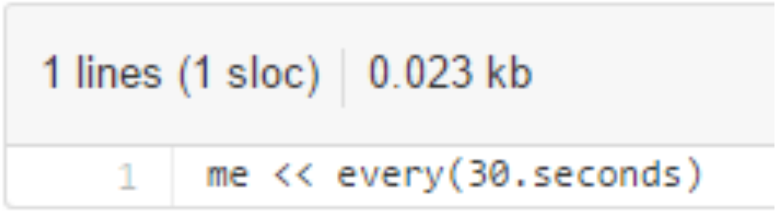
https://github.com/yermilov/groovy-dsl



1 lines (1 sloc) | 0.05 kb

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

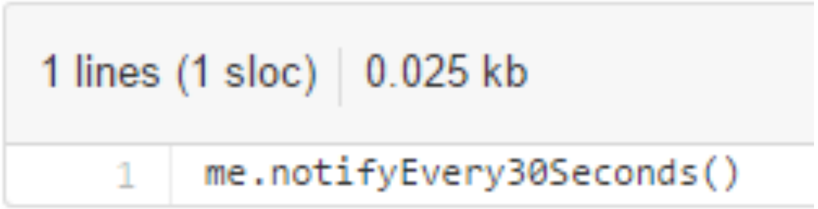
https://github.com/yermilov/groovy-dsl



1 lines (1 sloc) | 0.023 kb

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

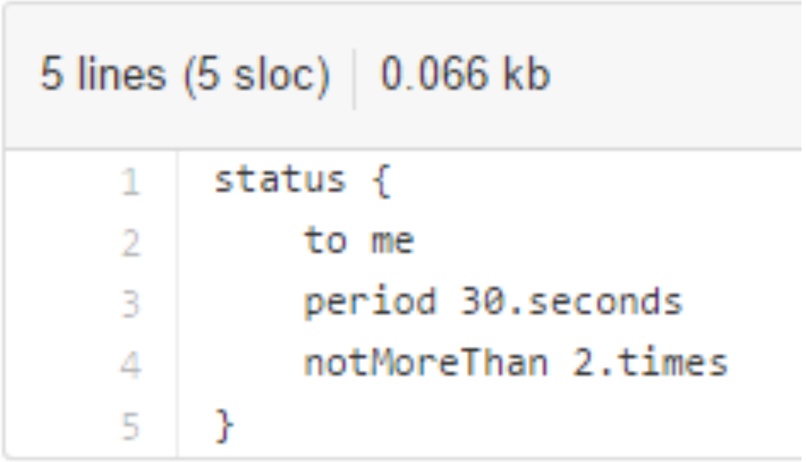
https://github.com/yermilov/groovy-dsl



1 lines (1 sloc) | 0.025 kb

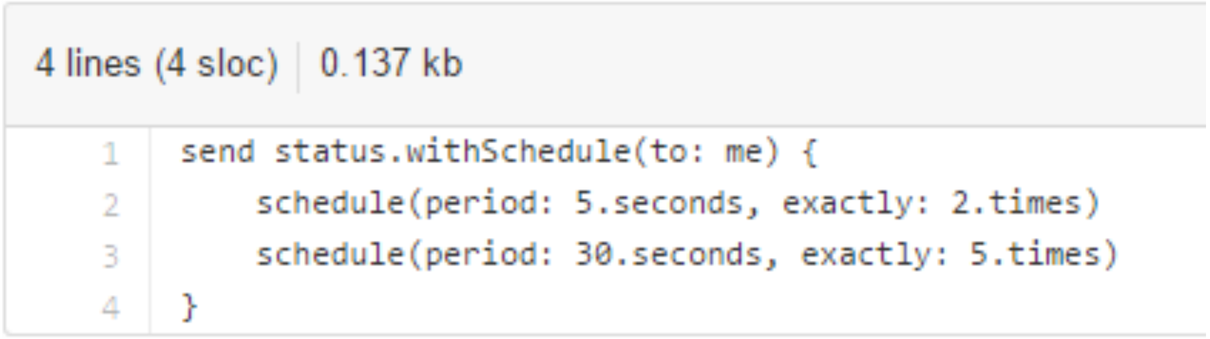
```
me.notifyEvery30Seconds()
```

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


5 lines (5 sloc) | 0.066 kb

```
status {
    to me
    period 30.seconds
    notMoreThan 2.times
}
```

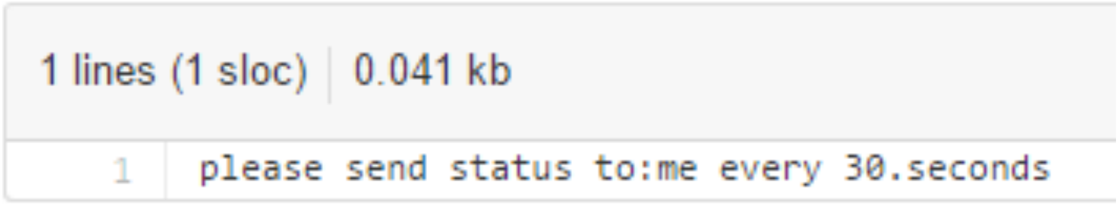
https://github.com/yermilov/groovy-dsl



4 lines (4 sloc) | 0.137 kb

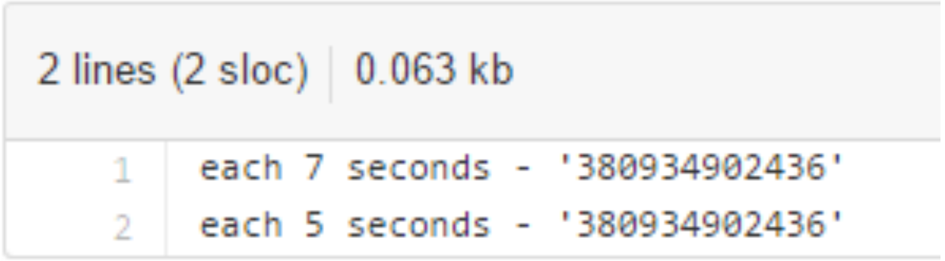
```
send status.withSchedule(to: me) {
    schedule(period: 5.seconds, exactly: 2.times)
    schedule(period: 30.seconds, exactly: 5.times)
}
```

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


1 lines (1 sloc) | 0.041 kb

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

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

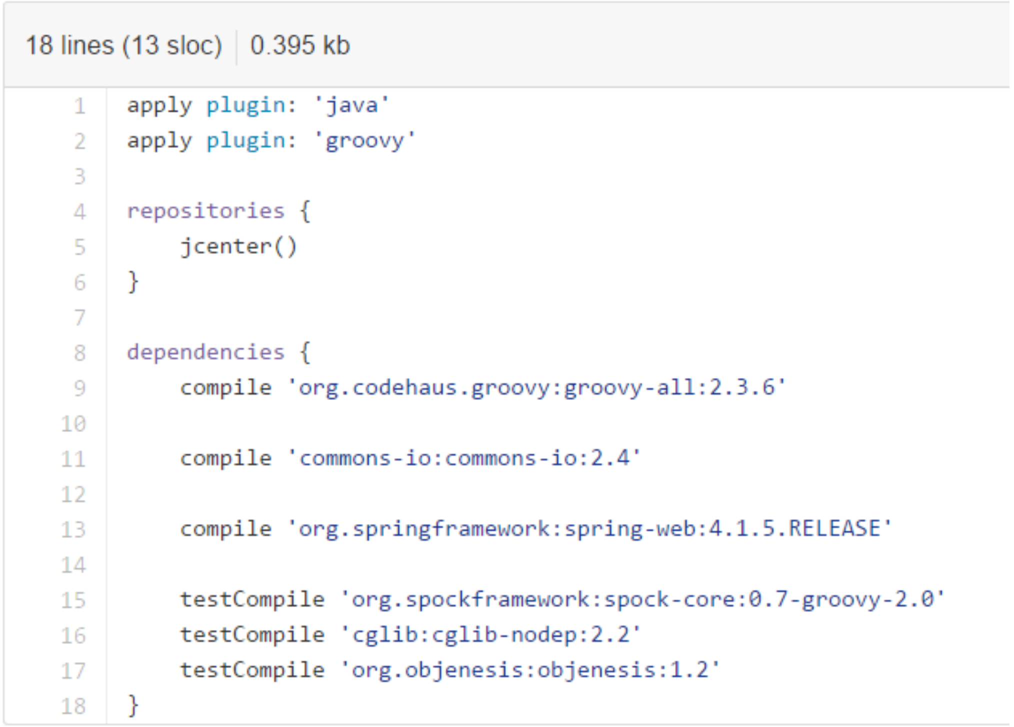


2 lines (2 sloc) | 0.063 kb

```
each 7 seconds - '380934902436'
each 5 seconds - '380934902436'
```

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

Groovy existing DSLs



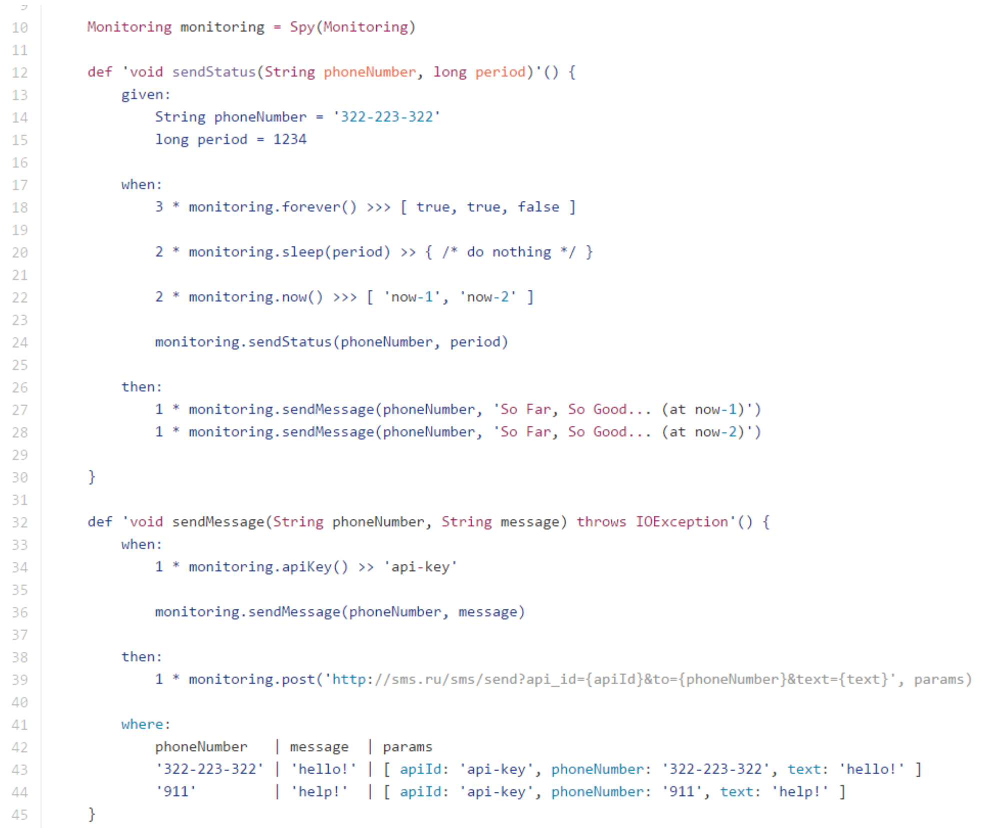
18 lines (13 sloc) | 0.395 kb

```
apply plugin: 'java'
apply plugin: 'groovy'

repositories {
    jcenter()
}

dependencies {
    compile 'org.codehaus.groovy:groovy-all:2.3.6'

    compile 'commons-io:commons-io:2.4'

    compile 'org.springframework:spring-web:4.1.5.RELEASE'

    testCompile 'org.spockframework:spock-core:0.7-groovy-2.0'
    testCompile 'cglib:cglib-nodep:2.2'
    testCompile 'org.objenesis:objenesis:1.2'
}
```

gradle

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

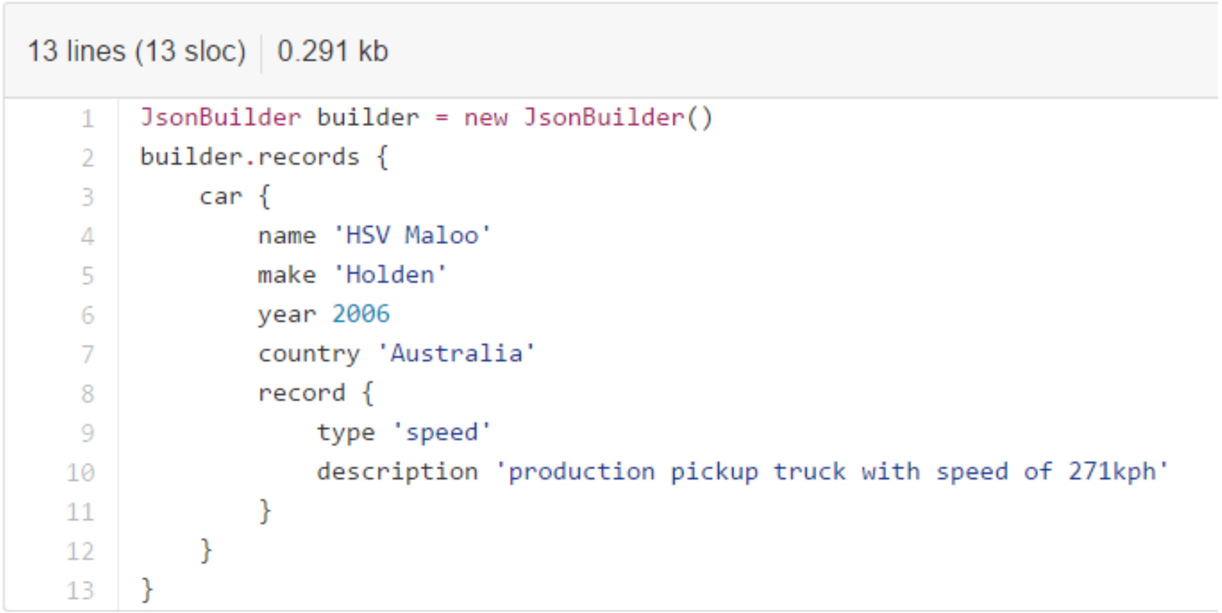
Groovy existing DSLs



```
    Monitoring monitoring = Spy(Monitoring)

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

        when:
            3 * monitoring.forever() >>> [ true, true, false ]

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

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

            monitoring.sendStatus(phoneNumber, period)

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

    }

    def 'void sendMessage(String phoneNumber, String message) throws IOException'() {
        when:
            1 * monitoring.apiKey() >> 'api-key'

            monitoring.sendMessage(phoneNumber, message)

        then:
            1 * monitoring.post('http://sms.ru/sms/send?api_id={apiId}&to={phoneNumber}&text={text}', params)

        where:
            phoneNumber   | message  | params
            '322-223-322' | 'hello!' | [ apiId: 'api-key', phoneNumber: '322-223-322', text: 'hello!' ]
            '911'         | 'help!'  | [ apiId: 'api-key', phoneNumber: '911', text: 'help!' ]
    }
```

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

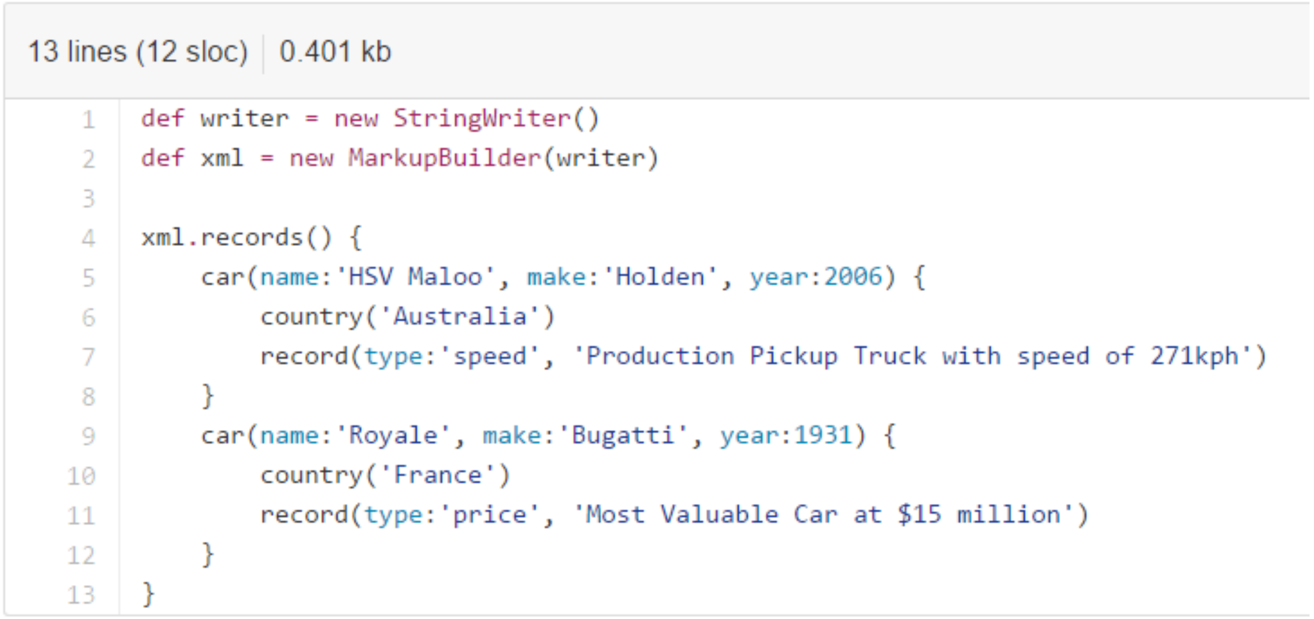
Groovy existing DSLs


13 lines (13 sloc) | 0.291 kb

```
JsonBuilder builder = new JsonBuilder()
builder.records {
    car {
        name 'HSV Maloo'
        make 'Holden'
        year 2006
        country 'Australia'
        record {
            type 'speed'
            description 'production pickup truck with speed of 271kph'
        }
    }
}
```

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

Groovy existing DSLs

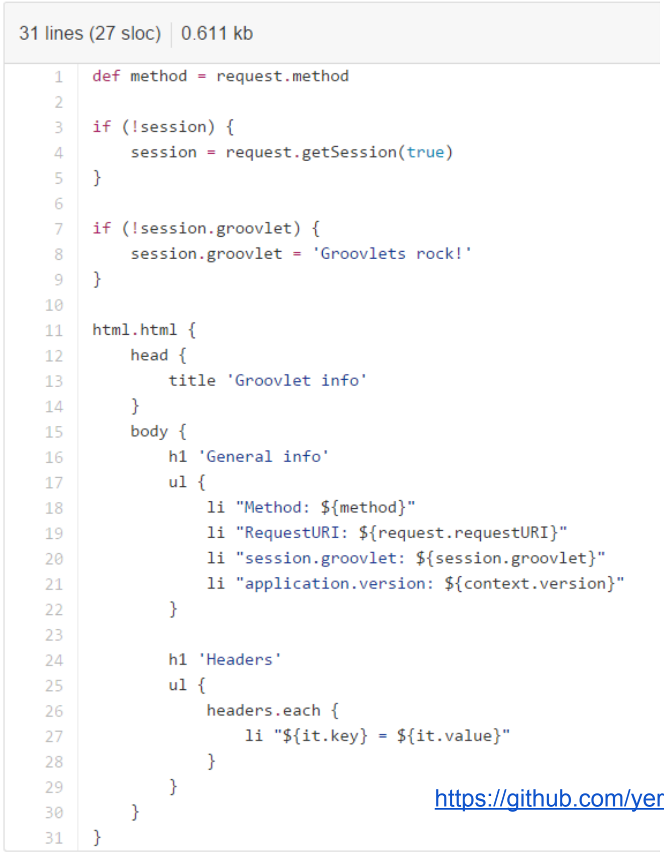


13 lines (12 sloc) | 0.401 kb

```groovy
def writer = new StringWriter()
def xml = new MarkupBuilder(writer)

xml.records() {
    car(name:'HSV Maloo', make:'Holden', year:2006) {
        country('Australia')
        record(type:'speed', 'Production Pickup Truck with speed of 271kph')
    }
    car(name:'Royale', make:'Bugatti', year:1931) {
        country('France')
        record(type:'price', 'Most Valuable Car at $15 million')
    }
}
```

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

Groovy existing DSLs

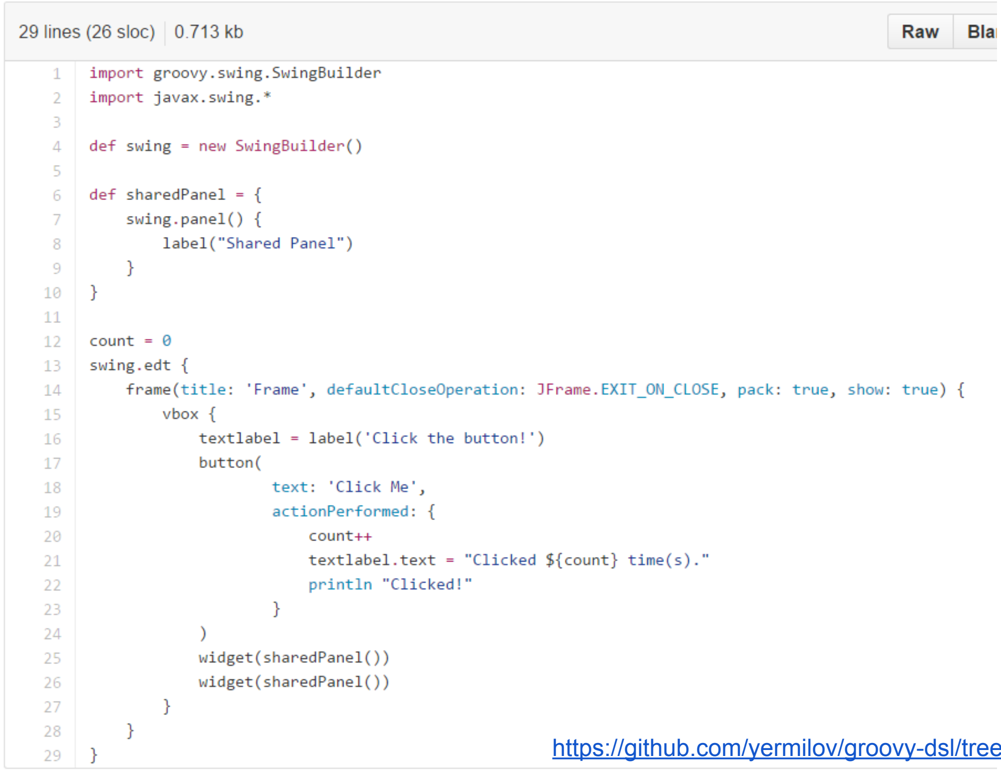


31 lines (27 sloc) | 0.611 kb

```
def method = request.method

if (!session) {
    session = request.getSession(true)
}

if (!session.groovlet) {
    session.groovlet = 'Groovlets rock!'
}

html.html {
    head {
        title 'Groovlet info'
    }
    body {
        h1 'General info'
        ul {
            li "Method: ${method}"
            li "RequestURI: ${request.requestURI}"
            li "session.groovlet: ${session.groovlet}"
            li "application.version: ${context.version}"
        }

        h1 'Headers'
        ul {
            headers.each {
                li "${it.key} = ${it.value}"
            }
        }
    }
}
```

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

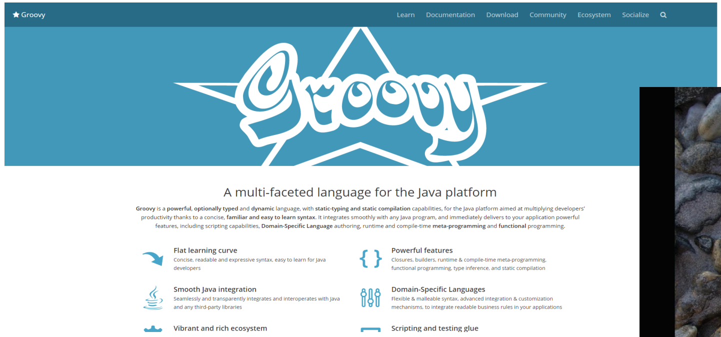
Groovy existing DSLs



```groovy
import groovy.swing.SwingBuilder
import javax.swing.*

def swing = new SwingBuilder()

def sharedPanel = {
    swing.panel() {
        label("Shared Panel")
    }
}

count = 0
swing.edt {
    frame(title: 'Frame', defaultCloseOperation: JFrame.EXIT_ON_CLOSE, pack: true, show: true) {
        vbox {
            textlabel = label('Click the button!')
            button(
                    text: 'Click Me',
                    actionPerformed: {
                        count++
                        textlabel.text = "Clicked ${count} time(s)."
                        println "Clicked!"
                    }
            )
            widget(sharedPanel())
            widget(sharedPanel())
        }
    }
}
```

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**, **optionally typed** and **dynamic** language, with **static-typing and static compilation** capabilities, for the Java platform aimed at multiplying developers' 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

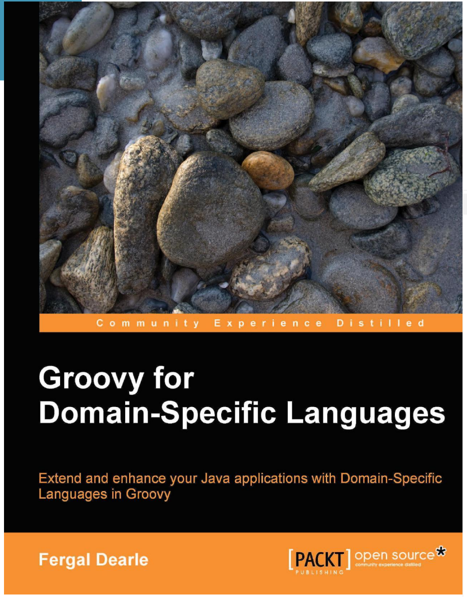
Smooth Java integration
Seamlessly and transparently integrates and interoperates with Java and any third-party libraries

Domain-Specific Languages
Flexible & malleable 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

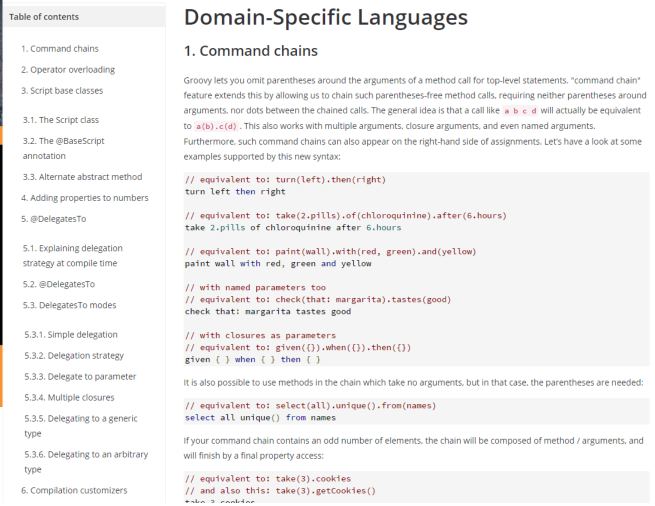


Table of contents

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(chloroquinine).after(6.hours)
take 2.pills of chloroquinine 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({}).when({}).then({})
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()
```

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

Thanks!

It's now safe to turn off
your computer.