DSL Groovy.
Gradle под микроскопом

Алексей Добрынин
lexa@trifle.one
@mad_lexa

Руслан Михалёв
mikhalev.ruslan@gmail.com
@CryonixMe

Что будем делать

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
   mavenCentral()
   maven { url 'http://custom-repository.com' }
}

dependencies {
   compile 'one.util:streamex:0.6.5',
           'org.yaml:snakeyaml:1.17'
   testCompile name: 'junit', group: 'junit', version: '4.12'
}

task hello {
   println 'hello'
}
```

DSL

Domain-specific language (Предметно-ориентированный язык) - язык программирования, специализированный для конкретной области применения (в противоположность языку общего назначения, применимому к широкому спектру областей и не учитывающему особенности конкретных сфер знаний). Построение такого языка и/или его структура данных отражают специфику решаемых с его помощью задач. Является ключевым понятием языково-ориентированного программирования.

DSL

Domain-specific language (П… язык программирования, специал… бласти применения (в противоположн… мому к широкому спектру областей … ретных сфер знаний). Построение … данных отражают специфику решае… ючевым понятием языково-ориентиров…

XML

```
<joker year='2017'>
  <lecture name='Groovy DSL. Gradle под микроскопом' lang='ru'>
    <speaker>Алексей Добрынин</speaker>
    <speaker>Руслан Михалёв</speaker>
  </lecture>
  <lecture name='Shenandoah: сборщик мусора, который смог (часть 2)' lang='ru'>
    <speaker>Алексей Шипилёв</speaker>
  </lecture>
  <lecture name='Cloud native Java EE' lang='en'>
    <speaker>Sebastian Daschner</speaker>
  </lecture>
  <lecture name='Using Kubernetes for Continuous Integration and Continuous Delivery' lang='en'>
    <speaker>Carlos Sanchez</speaker>
  </lecture>
</joker>
```

JAVA: xml builder

```
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
DocumentBuilder builder = factory.newDocumentBuilder();
Document doc = builder.newDocument();
    Element joker = doc.createElement("joker");
    joker.setAttribute("year", "2017");

        Element lecture = doc.createElement("lecture");
        lecture.setAttribute("name", "Groovy DSL. Gradle под микроскопом");
        lecture.setAttribute("lang", "ru");

        Element[] speakers = new Element[]{doc.createElement("speaker"), doc.createElement("speaker")};
        speakers[0].appendChild(doc.createTextNode("Алексей Добрынин"));
        speakers[1].appendChild(doc.createTextNode("Руслан Михалёв"));
        lecture.appendChild(speakers[0]);
        lecture.appendChild(speakers[1]);
        joker.appendChild(lecture);
```

JAVA: xml builder

```
Element lecture = doc.createElement("lecture");
lecture.setAttribute("name", "Shenandoah: сборщик мусора, который смог (часть 2)");
lecture.setAttribute("lang", "ru");

Element speaker = doc.createElement("speaker");
speaker.appendChild(doc.createTextNode("Алексей Шипилёв"));
lecture.appendChild(speaker);
joker.appendChild(lecture);


Element lecture = doc.createElement("lecture");
lecture.setAttribute("name", "Cloud native Java EE");
lecture.setAttribute("lang", "en");

Element speaker = doc.createElement("speaker");
speaker.appendChild(doc.createTextNode("Sebastian Daschner"));
lecture.appendChild(speaker);
joker.appendChild(lecture);
```

JAVA: xml builder

```
        Element lecture = doc.createElement("lecture");
        lecture.setAttribute("name", "Using Kubernetes for Continuous Integration and Continuous
Delivery");
        lecture.setAttribute("lang", "en");

        Element speaker = doc.createElement("speaker");
        speaker.appendChild(doc.createTextNode("Carlos Sanchez"));
        lecture.appendChild(speaker);
        joker.appendChild(lecture);

    doc.appendChild(joker);

TransformerFactory.newInstance()
       .newTransformer()
       .transform(new DOMSource(doc), new StreamResult(System.out));
```

JAVA: xml builder

```
try {
   DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
   DocumentBuilder builder = factory.newDocumentBuilder();
   Document doc = builder.newDocument();
……
……
……
……
   TransformerFactory.newInstance()
          .newTransformer()
          .transform(new DOMSource(doc), new StreamResult(System.out));
} catch (ParserConfigurationException e) {
   ……
} catch (TransformerConfigurationException e) {
   ……
} catch (TransformerException e) {
   ……
}
```

JAVA: jaxb

```
public class Joker {
   private int year;
   private List<Lecture> lectures;
}
```

JAVA: jaxb

```
public class Joker {
   private int year;
   private List<Lecture> lectures;

   public int getYear() { return year; }
   public void setYear(int year) { this.year = year; }
   public List<Lecture> getLectures() { return lectures; }
   public void setLectures(List<Lecture> lectures) {
      this.lectures = lectures;
   }




}
```

JAVA: jaxb

```
@XmlRootElement
public class Joker {
   private int year;
   private List<Lecture> lectures;

   public int getYear() { return year; }
   @XmlAttribute public void setYear(int year) { this.year = year; }
   public List<Lecture> getLectures() { return lectures; }
   @XmlElement(name = "lecture") public void setLectures(List<Lecture> lectures) {
      this.lectures = lectures;
   }




}
```

JAVA: jaxb + lombok

```
@XmlRootElement
@NoArgsConstructor @AllArgsConstructor
public class Joker {
   @Getter @Setter(onMethod = @__(@XmlAttribute)) private int year;
   @Getter @Setter(onMethod = @__(@XmlElement(name = "lecture"))) private List<Lecture> lectures;
}
```

JAVA: jaxb + lombok

```java
@XmlRootElement
@NoArgsConstructor @AllArgsConstructor
public class Joker {
  @Getter @Setter(onMethod = @__(@XmlAttribute)) private int year;
  @Getter @Setter(onMethod = @__(@XmlElement(name = "lecture"))) private List<Lecture> lectures;
}

@XmlRootElement
@NoArgsConstructor @AllArgsConstructor
public class Lecture {
  @Getter @Setter(onMethod = @__(@XmlAttribute)) private String name, lang;
  @Getter @Setter(onMethod = @__(@XmlElement(name = "speaker"))) private List<String> speakers;
}
```

JAVA: jaxb + lombok

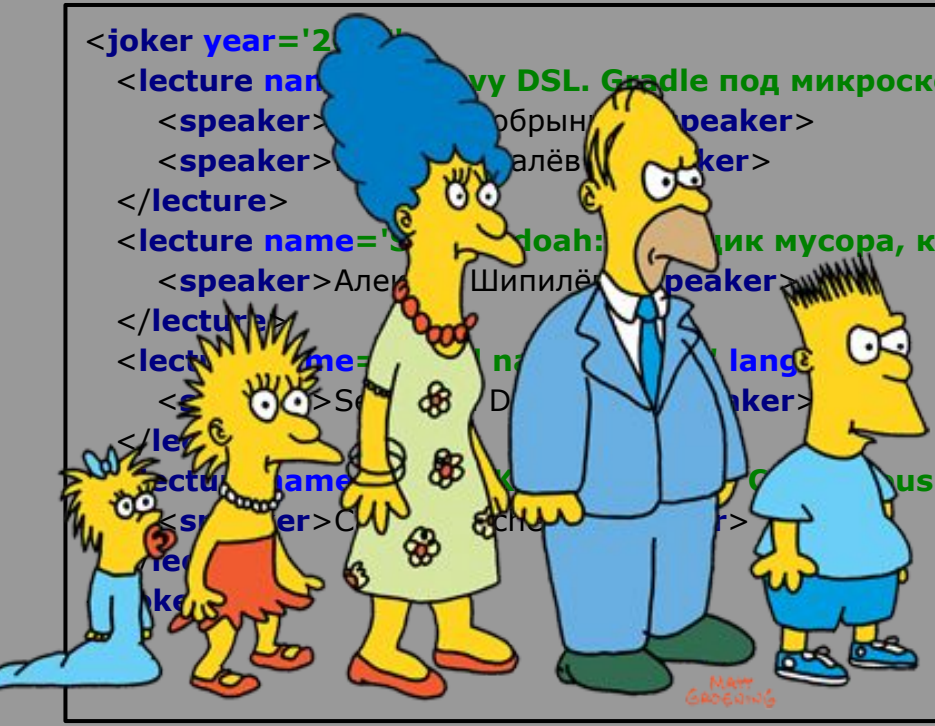
```
JAXBContext context = JAXBContext.newInstance(Joker.class);
Marshaller marshaller = context.createMarshaller();
marshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, Boolean.TRUE);
marshaller.marshal(new Joker(2017, Arrays.asList(
        new Lecture("Groovy DSL. Gradle под микроскопом", "ru",
                Arrays.asList("Алексей Добрынин", "Руслан Михалёв")
        ),
        new Lecture("Shenandoah: сборщик мусора, который смог (часть 2)", "ru",
                Collections.singletonList("Алексей Шипилёв")
        ),
        new Lecture("Cloud native Java EE", "en",
                Collections.singletonList("Sebastian Daschner")
        ),
        new Lecture("Using Kubernetes for Continuous Integration and Continuous Delivery", "en",
                Collections.singletonList("Carlos Sanchez")
        )
)), System.out);
```

Groovy: xml builder

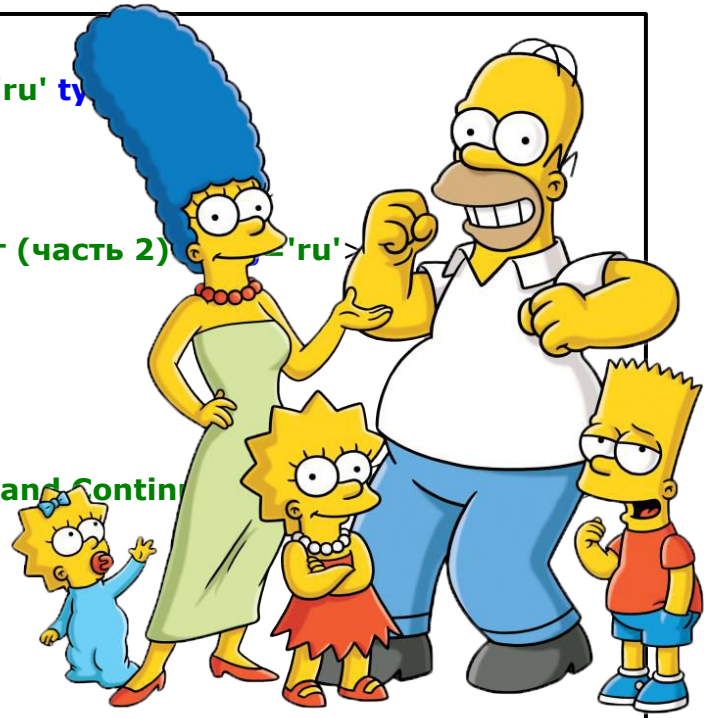
```
def writer = new StringWriter()
new groovy.xml.MarkupBuilder(writer).joker(year: 2017) {
    lecture(name: 'Groovy DSL. Gradle под микроскопом', lang: 'ru') {
        speaker("Алексей Добрынин")
        speaker("Руслан Михалёв")
    }
    lecture(name: 'Shenandoah: сборщик мусора, который смог (часть 2)', lang: 'ru') {
        speaker("Алексей Шипилёв")
    }
    lecture(name: 'Cloud native Java EE', lang: 'en') {
        speaker("Sebastian Daschner")
   }
   lecture(name: 'Using Kubernetes for Continuous Integration and Continuous Delivery', lang: 'en') {
        speaker("Carlos Sanchez")
    }
}
println(writer.toString())
```

JAVA

<joker year='2
  <lecture nam vy DSL. Gradle под микроскопом' lang='ru' ty
    <speaker> обрын peaker>
    <speaker> алёв ker>
  </lecture>
  <lecture name='S doah: ик мусора, который смог (часть 2) ='ru'>
    <speaker>Алек Шипилё peaker>
  </lecture>
  <lect me= n lang
    <s >Se D ker>
  </le
  <lectu ame= us Integration and Contin
    <sp er>C ch r>
  </le
  </ke



Groovy



Примеры DSL

Примеры DSL

Примеры DSL

Примеры DSL

```
def "Деление на 0"() {
    when:
        BigDecimal.ONE.divide(BigDecimal.ZERO)
    then:
        thrown(ArithmeticException)
}
```

Примеры DSL

```
def "Деление на 0"() {
    when:
        BigDecimal.ONE.divide(BigDecimal.ZERO)
    then:
        thrown(ArithmeticException)
}
```

```
def "Модуль числа"() {
    expect:
        Math.abs(num) == abs
    where:
        num   || abs
        -1    || 1
        -5.2  || 5.2
        7L    || 7L
}
```

Примеры DSL

Примеры DSL

```
GParsPool.withPool {
    final AtomicInteger result = new AtomicInteger(0)
    [1, 2, 3, 4, 5].eachParallel {result.addAndGet(it)}
    assert 15 == result
}
```

Примеры DSL

GPars

GRAILS

Примеры DSL

```
class BookController {
    def list() {
        [ books: Book.findAll() ]
    }
}
```

GPars

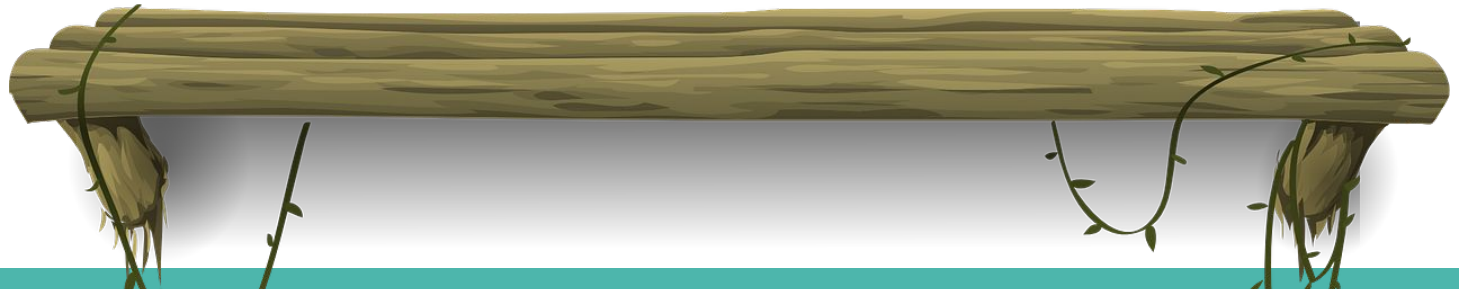
GRAILS

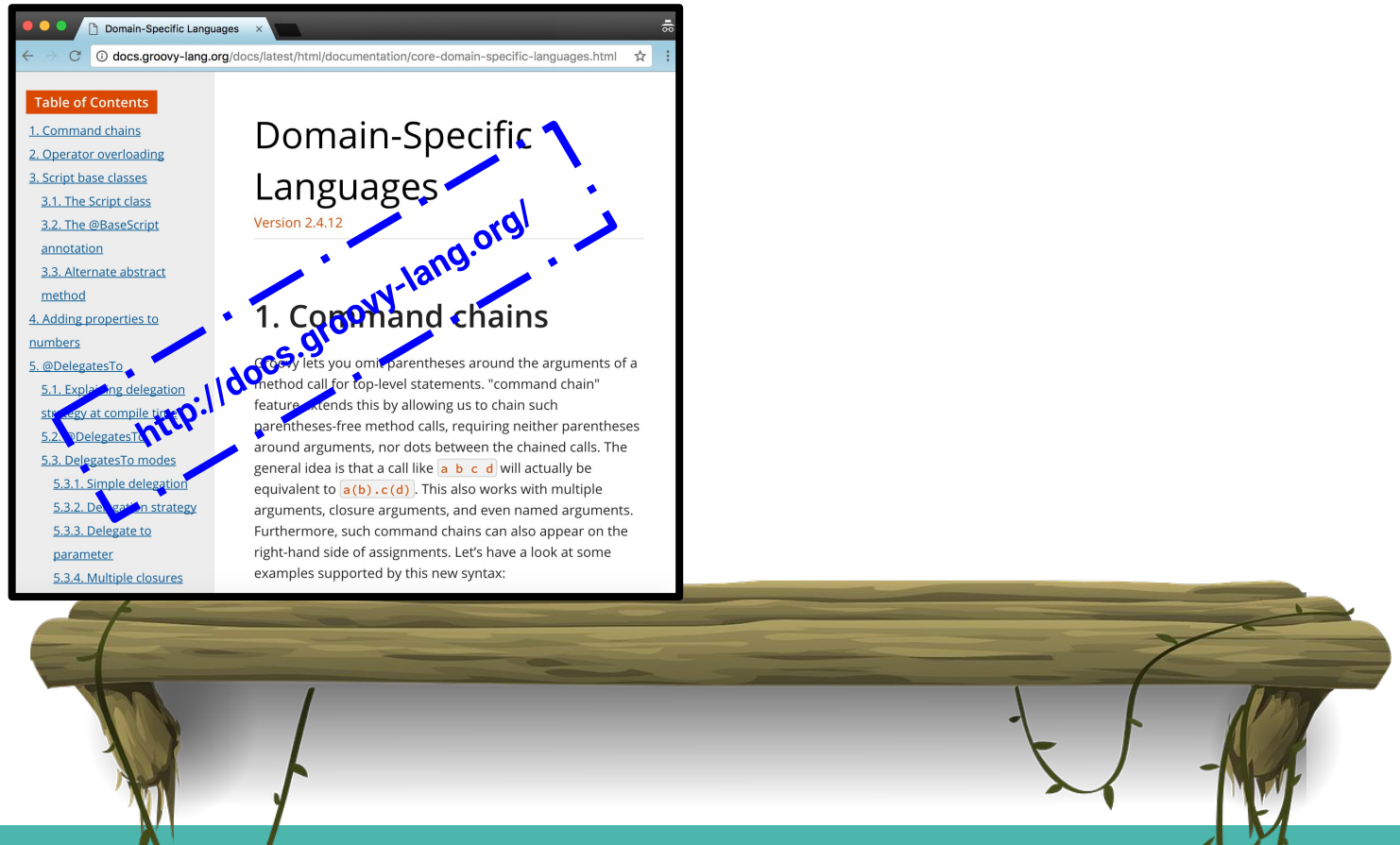
Примеры DSL

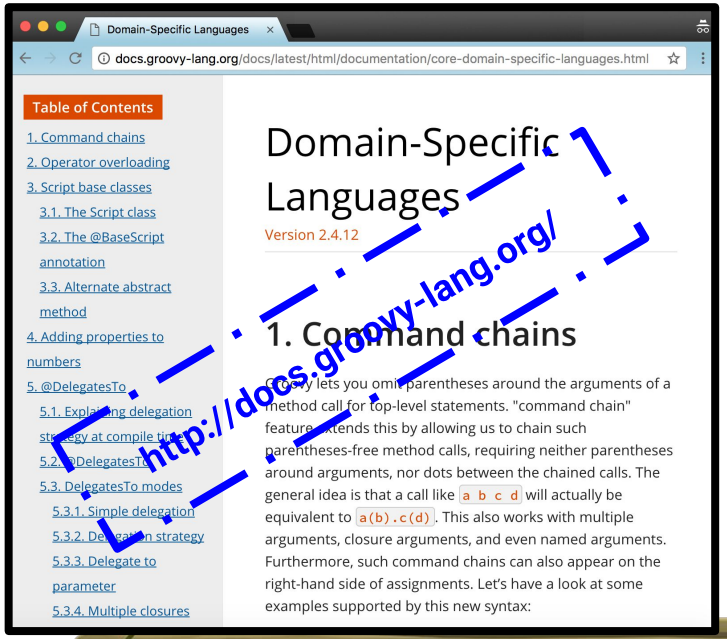
Gradle

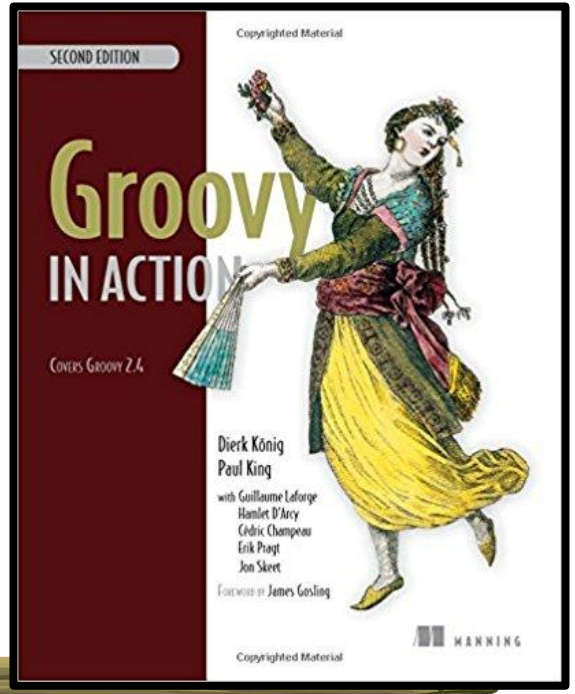
GPars

GRAILS




Domain-Specific Languages
docs.groovy-lang.org/docs/latest/html/documentation/core-domain-specific-languages.html
Table of Contents
1. Command chains
2. Operator overloading
3. Script base classes
3.1. The Script class
3.2. The @BaseScript 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
Domain-Specific Languages
Version 2.4.12
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:
http://docs.groovy-lang.org/


Domain-Specific Languages
docs.groovy-lang.org/docs/latest/html/documentation/core-domain-specific-languages.html
Table of Contents
1. Command chains
2. Operator overloading
3. Script base classes
3.1. The Script class
3.2. The @BaseScript 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
Domain-Specific Languages
Version 2.4.12
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:
http://docs.groovy-lang.org/


Copyrighted Material
SECOND EDITION
Groovy
IN ACTION
Covers Groovy 2.4
Dierk König
Paul King
with Guillaume Laforge
Hamlet D'Arcy
Cédric Champeau
Erik Pragt
Jon Skeet
Foreword by James Gosling
MANNING
Copyrighted Material


Domain-Specific Languages
docs.groovy-lang.org/docs/latest/html/documentation/core-domain-specific-languages.html
Table of Contents
1. Command chains
2. Operator overloading
3. Script base classes
3.1. The Script class
3.2. The @BaseScript 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
Domain-Specific Languages
Version 2.4.12
1. Command
http://docs.groovy-lang
JUG.RU
SECOND EDITION
Copyrighted Material
Groovy
ACTION
Dierk König
Paul King
with Guillaume Laforge
Hamlet D'Arcy
Cédric Champeau
Erik Pragt
Jon Skeet
Foreword by James Gosling
MANNING
Copyrighted Material


Domain-Specific Languages
docs.groovy-lang.org/docs/latest/html/documentation/core-domain-specific-languages.html
Table of Contents
1. Command chains
2. Operator overloading
3. Script base classes
3.1. The Script class
3.2. The @BaseScript 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
Domain-Specific Languages
Version 2.4.12
1. Command
http://docs.groovy-lang
SECOND EDITION
Copyrighted Material
Groovy
ACTION
Dierk König
Paul King
with Guillaume Laforge
Hamlet D'Arcy
Cédric Champeau
Erik Pragt
Jon Skeet
Foreword by James Gosling
MANNING
Copyrighted Material


Domain-Specific Languages
docs.groovy-lang.org/docs/latest/html/documentation/core-domain-specific-languages.html
Table of Contents
1. Command chains
2. Operator overloading
3. Script base classes
3.1. The Script class
3.2. The @BaseScript 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
Domain-Specific
Languages
Version 2.4.12
1. Command
http://docs.groovy-lang
JUG.RU
SECOND EDITION
Dierk König
Paul King
with Guillaume Laforge
Hamlet D'Arcy
Cédric Champeau
Erik Pragt
Jon Skeet
Foreword by James Gosling
MANNING
Copyrighted Material
hazelcast
DINS


latest/html/documentation/core-domain-specific-languages.html
http://docs.groovy-lang
5. @DelegatesTo
5.1. Explaining delegation strategy at compile time
5.3. DelegatesTo modes
5.3.1. Simple delegation
5.3.3. Delegate to parameter
5.3.4. Multiple closures
SECOND EDITION
Dierk König
Paul King
with Guillaume Laforge
Hamlet D'Arcy
Cédric Champeau
Erik Pragt
Jon Skeet
MANNING
Copyrighted Material
hazelcast
DINS


latest/html/documentation/core-domain-specific-languages.html
http://docs.groovy-lang
SECOND EDITION
hazelcast
DINS
MANNING


latest/html/documentation/core-domain-specif
omain-Specifi
anguages
n 2.4.12
1. Command
http://docs.groovy-lang
5. @DelegatesTo
5.1. Explaining delegation
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
SECOND EDITION
hazelcast
MANNING


16+

Что будем делать

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
   mavenCentral()
   maven { url 'http://custom-repository.com' }
}

dependencies {
   compile 'one.util:streamex:0.6.5',
           'org.yaml:snakeyaml:1.17'
   testCompile name: 'junit', group: 'junit', version: '4.12'
}

task hello {
   println 'hello'
}
```

Что будем делать

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
   jcenter()
   maven { url 'http://custom-repository.com' }
}

dependencies {
   compile 'one.util:streamex:0.6.5',
           'org.yaml:snakeyaml:1.17'
   testCompile name: 'junit', group: 'junit', version: '4.12'
}

task hello {
   println 'hello'
}
```

Что будем делать

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
   jcenter()
   maven { url 'http://custom-repository.com' }
}

dependencies {
   compile 'one.util:streamex:0.6.5',
           'org.yaml:snakeyaml:1.17'
   testCompile name: 'junit', group: 'junit', version: '4.12'
}

task hello {
   println 'hello'
}
```

Granny

1.6+

Доклад является ознакомительным, с инструментарием для разработки DSL на языке Groovy, и ни в коем случае не претендует на 100% совпадение с подходами разработчиков build tool Gradle. По этой причине, чтобы не вводить в заблуждение слушателей, в дальнейшем мы будем говорить, что разрабатываем DSL для скриптов Granny

С чего начать

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
   jcenter()
   maven { url 'http://custom-repository.com' }
}

dependencies {
   compile 'one.util:streamex:0.6.5',
           'org.yaml:snakeyaml:1.17'
   testCompile name: 'junit', group: 'junit', version: '4.12'
}

task hello {
   println 'hello'
}
```

С чего начать

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
   jcenter()
   maven { url 'http://custom-repository.com' }
}

dependencies {
   compile 'one.util:streamex:0.6.5',
           'org.yaml:snakeyaml:1.17'
   testCompile name: 'junit', group: 'junit', version: '4.12'
}

task hello {
   println 'hello'
}
```

Рабочая область

Запускаем пустой скрипт granny

```
public class GrannyInternal {



}
```

Запускаем пустой скрипт granny

```
public class GrannyInternal {
   File buildScript;


   public GrannyInternal(File buildScript) {


      this.buildScript = buildScript;
   }




}
```

Запускаем пустой скрипт granny

```
public class GrannyInternal {
   File buildScript;


   public GrannyInternal(File buildScript) {


      this.buildScript = buildScript;
   }

   public void build() {


   }
}
```

Запускаем пустой скрипт granny

```
public class GrannyInternal {
   File buildScript;


   public GrannyInternal(File buildScript) {


      this.buildScript = buildScript;
   }

   public void build() {
      GroovyShell shell = new GroovyShell();

   }
}
```

Запускаем пустой скрипт granny

```
public class GrannyInternal {
   File buildScript;


   public GrannyInternal(File buildScript) {


      this.buildScript = buildScript;
   }

   public void build() throws IOException {
      GroovyShell shell = new GroovyShell();
      shell.evaluate(buildScript);
   }
}
```

Запускаем пустой скрипт granny

```
public class GrannyInternal {
   File buildScript;
   CompilerConfiguration config;

   public GrannyInternal(File buildScript) {


      this.buildScript = buildScript;
   }

   public void build() throws IOException {
      GroovyShell shell = new GroovyShell();
      shell.evaluate(buildScript);
   }
}
```

Запускаем пустой скрипт granny

```
public class GrannyInternal {
   File buildScript;
   CompilerConfiguration config;

   public GrannyInternal(File buildScript) {
      config = new CompilerConfiguration();

      this.buildScript = buildScript;
   }

   public void build() throws IOException {
      GroovyShell shell = new GroovyShell(config);
      shell.evaluate(buildScript);
   }
}
```

Запускаем пустой скрипт granny

```
public class GrannyInternal {
   File buildScript;
   CompilerConfiguration config;

   public GrannyInternal(File buildScript) {
      config = new CompilerConfiguration();
      config.setDefaultScriptExtension(".granny");
      this.buildScript = buildScript;
   }

   public void build() throws IOException {
      GroovyShell shell = new GroovyShell(config);
      shell.evaluate(buildScript);
   }
}
```

Запускаем пустой скрипт granny

```
println(lang: "groovy", version: "2.5.0")
```

```
public class GrannyInternal {
    File buildScript;
    CompilerConfiguration config;

    public GrannyInternal(File buildScript) {
        config = new CompilerConfiguration();
        config.setDefaultScriptExtension(".granny");
        this.buildScript = buildScript;
    }

    public void build() throws IOException {
        GroovyShell shell = new GroovyShell(config);
        shell.evaluate(buildScript);
    }
}
```

Запускаем пустой скрипт granny

```
public class GrannyInternal {
    File buildScript;
    CompilerConfiguration config;

    public GrannyInternal(File buildScript) {
        config = new CompilerConfiguration();
        config.setDefaultScriptExtension(".granny");
        this.buildScript = buildScript;
    }

    public void build() throws IOException {
        GroovyShell shell = new GroovyShell(config);
        shell.evaluate(buildScript);
    }
}
```

```
println(lang: "groovy", version: "2.5.0" )
```

```
(1..100).findAll{ it % 2 }.sum()
```

Запускаем пустой скрипт granny

```groovy
public class GrannyInternal {
    File buildScript;
    CompilerConfiguration config;

    public GrannyInternal(File buildScript) {
        config = new CompilerConfiguration();
        config.setDefaultScriptExtension(".granny");
        this.buildScript = buildScript;
    }

    public void build() throws IOException {
        GroovyShell shell = new GroovyShell(config);
        shell.evaluate(buildScript);
    }
}
```

```groovy
println(lang: "groovy", version: "2.5.0")
```

```groovy
["groovy","java","scala","kotlin"]*.size()
```

```groovy
(1..100).findAll{ it % 2 }.sum()
```

Запускаем пустой скрипт granny

```
public class GrannyInternal {
    File buildScript;
    CompilerConfiguration config;

    public GrannyInternal(File buildScript) {
        config = new CompilerConfiguration();
        config.setDefaultScriptExtension(".granny");
        this.buildScript = buildScript;
    }

    public void build() throws IOException {
        GroovyShell shell = new GroovyShell(config);
        shell.evaluate(buildScript);
    }
}
```

```
println(lang: "groovy", version: "2.5.0")
```

```
["groovy", "java", "scala", "kotlin"]*.size()
```

```
(1..100).findAll{ it % 2 }.sum()
```

```
def jvm = ["java", "groovy", "scala", "kotlin"]
def lang = ["c++", *jvm, "php", "js"]
```

Запускаем пустой скрипт granny

```
public class GrannyInternal {
    File buildScript;
    CompilerConfiguration config;

    public GrannyInternal(File buildScript) {
        config = new CompilerConfiguration();
        config.setDefaultScriptExtension(".granny");
        this.buildScript = buildScript;
    }

    public void build() throws IOException {
        GroovyShell shell = new GroovyShell(config);
        shell.evaluate(buildScript);
    }
}
```

```
println(lang: "groovy", version: "2.5.0")
```

```
["groovy","java","scala","kotlin"]*.size()
```

```
(1..100).findAll{ it % 2 }.sum()
```

```
def jvm = ["java", "groovy", "scala", "kotlin"]
def lang = ["c++", *jvm, "php", "js"]
```

```
["a","b","d","e"] - ["c","d","e"]
```

Запускаем пустой скрипт granny

```
public class GrannyInternal {
    File buildScript;
    CompilerConfiguration config;

    public GrannyInternal(File buildScript) {
        config = new CompilerConfiguration();
        config.setDefaultScriptExtension(".granny");
        this.buildScript = buildScript;
    }

    public void build() throws IOException {
        GroovyShell shell = new GroovyShell(config);
        shell.evaluate(buildScript);
    }
}
```

```
println(lang: "groovy", version: "2.5.0")
```

```
["groovy","java","scala","kotlin"]*.size()
```

```
(1..100).findAll{ it % 2 }.sum()
```

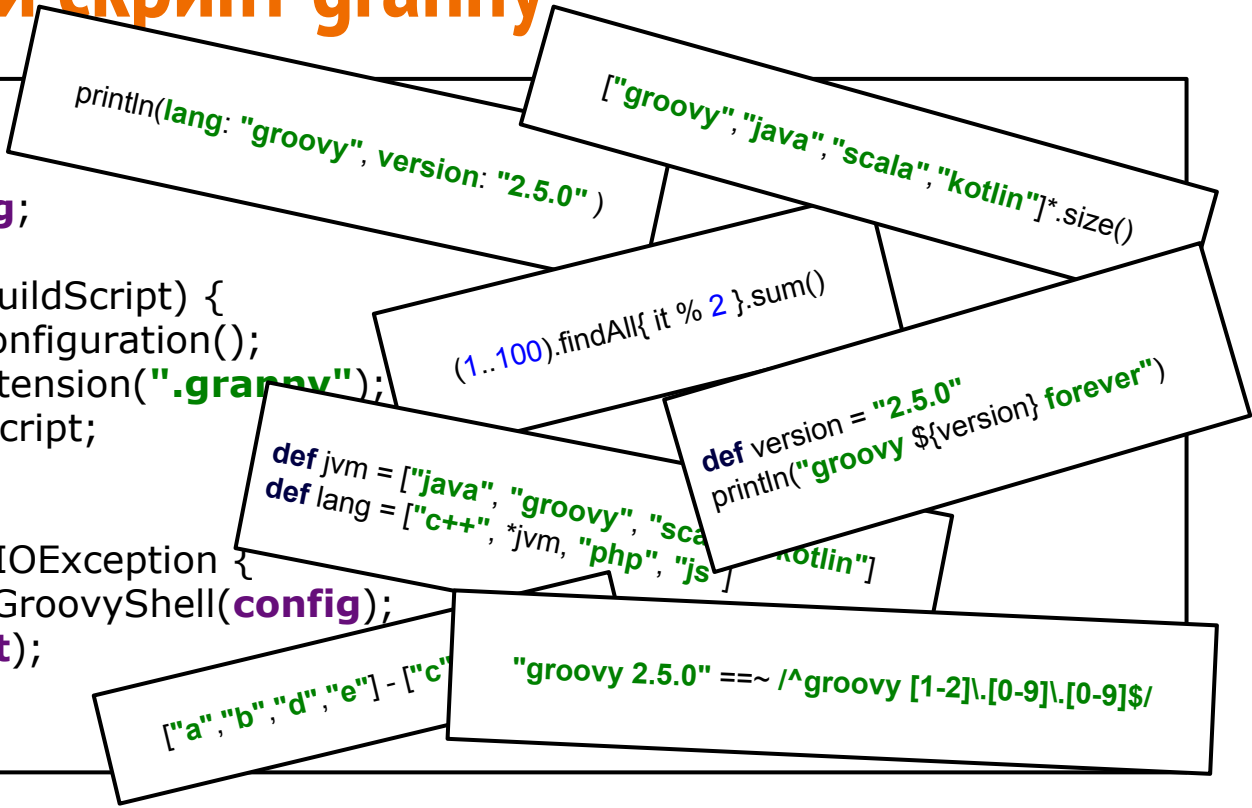
```
def jvm = ["java", "groovy", "scala", "kotlin"]
def lang = ["c++", *jvm, "php", "js"]
```

```
["a","b","d","e"] - ["c"
```

```
"groovy 2.5.0" ==~ /^groovy [1-2]\.[0-9]\.[0-9]$/
```

Запускаем пустой скрипт granny

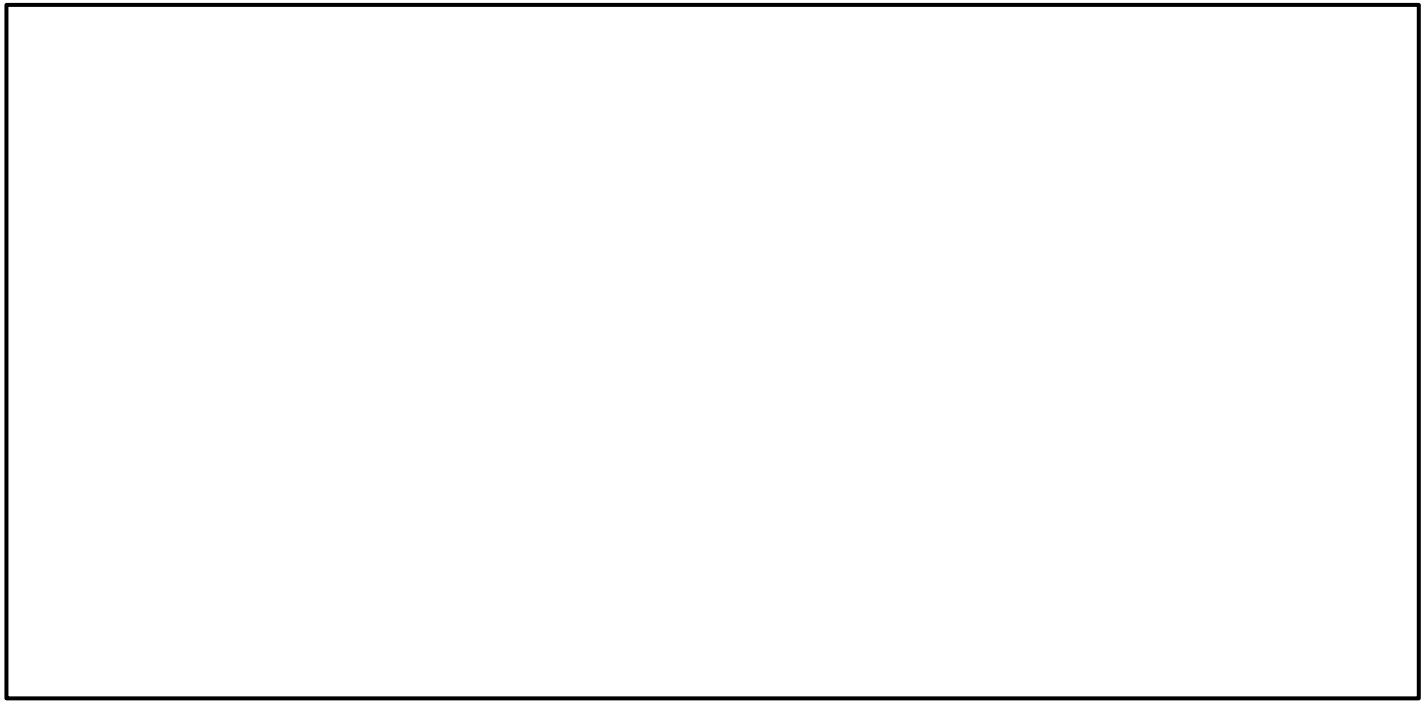
```
public class GrannyInternal {
    File buildScript;
    CompilerConfiguration config;

    public GrannyInternal(File buildScript) {
        config = new CompilerConfiguration();
        config.setDefaultScriptExtension(".granny");
        this.buildScript = buildScript;
    }

    public void build() throws IOException {
        GroovyShell shell = new GroovyShell(config);
        shell.evaluate(buildScript);
    }
}
```



Apply

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
   jcenter()
   maven { url 'http://custom-repository.com' }
}

dependencies {
   compile 'one.util:streamex:0.6.5',
           'org.yaml:snakeyaml:1.17'
   testCompile name: 'junit', group: 'junit', version: '4.12'
}

task hello {
   println 'hello'
}
```

Попробуем запустить



```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
                group: 'junit',
                version: '4.12'
}

task hello {
  println 'hello'
}
```

Попробуем запустить


```
groovy.lang.MissingMethodException: No signa
method: Script1.apply() is applicable for
types: (LinkedHashMap) values: [[plugi

Possible solutions: any(), any(groovy.l
every(), tap(groovy.lang.Closure),
every(groovy.lang.Closure), split(groov      e)
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
              group: 'junit',
              version: '4.12'
}

task hello {
  println 'hello'
}
```

Попробуем запустить

groovy.lang.MissingMethodException: No signature of method: Script1.apply() is applicable for argument types: (LinkedHashMap) values: [[plugin:java]]

Possible solutions: any(), any(groovy.lang.Closure), every(), tap(groovy.lang.Closure), every(groovy.lang.Closure), split(groovy.lang.Closure)

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
                group: 'junit',
                version: '4.12'
}

task hello {
  println 'hello'
}
```

Попробуем запустить

groovy.lang.MissingMethodException: No signature of method: Script1.apply() is applicable for argument types: (LinkedHashMap) values: [[plugin:java]]

Possible solutions: any(), any(groovy.lang.Closure), every(), tap(groovy.lang.Closure), every(groovy.lang.Closure), split(groovy.lang.Closure)

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
                group: 'junit',
                version: '4.12'
}

task hello {
  println 'hello'
}
```

OK-OK Groovy, я сделаю как ты хочешь

```
public class ProjectScript {




}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
                  group: 'junit',
                  version: '4.12'
}

task hello {
  println 'hello'
}
```

OK-OK Groovy, я сделаю как ты хочешь

```
public abstract class ProjectScript extends Script {




}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
                  group: 'junit',
                  version: '4.12'
}

task hello {
  println 'hello'
}
```

OK-OK Groovy, я сделаю как ты хочешь

```
public abstract class ProjectScript extends Script {



}
```

```
public class GrannyInternal {
   public GrannyInternal(File buildScript, Project project) {



}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
                group: 'junit',
                version: '4.12'
}

task hello {
  println 'hello'
}
```

ОК-ОК Groovy, я сделаю как ты хочешь

```
public abstract class ProjectScript extends Script {




}
```

```
public class GrannyInternal {
   public GrannyInternal(File buildScript, Project project) {
...

config.setScriptBaseClass(ProjectScript.class.getName());
...
   }
}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
                group: 'junit',
                version: '4.12'
}

task hello {
  println 'hello'
}
```

ОК-ОК Groovy, я сделаю как ты хочешь

```
public abstract class ProjectScript extends Script {

   public void apply(Map<String,String> args) {



   }
}
```

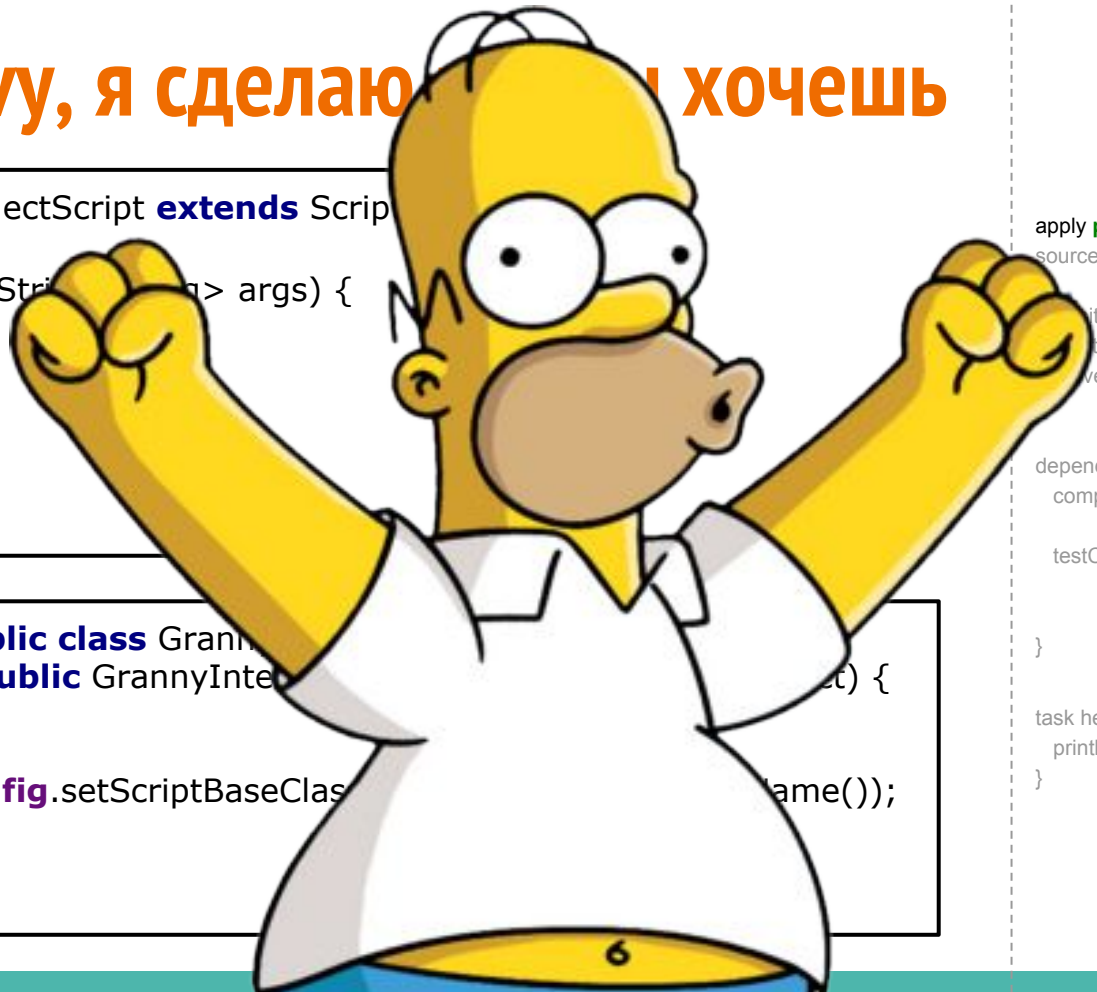
```
public class GrannyInternal {
   public GrannyInternal(File buildScript, Project project) {
...

config.setScriptBaseClass(ProjectScript.class.getName());
...
   }
}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
                  group: 'junit',
                  version: '4.12'
}

task hello {
  println 'hello'
}
```

ОК-ОК Groovy, я сделаю [illegible] хочешь

```
public abstract class ProjectScript extends Scrip

   public void apply(Map<Str[illegible]g> args) {



   }
}
```

```
public class Gran[illegible]
   public GrannyInte[illegible]t) {
...

config.setScriptBaseClas[illegible]ame());
...
   }
}
```



```
apply plugin: "java"
sourceCompatibility = '1.9'

[illegible]itory {
[illegible]ter()
[illegible]ven {  url 'http://my_repo.org' }


dependencies {
   compile 'one.util:streamex:0.6.5',
           'org.yaml:snakeyaml:1.17'
   testCompile name: 'junit',
                group: 'junit',
                version: '4.12'
}

task hello {
   println 'hello'
}
```

OK-OK Groovy, я сделаю как ты хочешь

```
public abstract class ProjectScript extends Script {
   @Override
   public Object invokeMethod(String name, Object args) {



   }
}
```

```
public class GrannyInternal {
   public GrannyInternal(File buildScript, Project project) {
...

config.setScriptBaseClass(ProjectScript.class.getName());
...
   }
}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
                  group: 'junit',
                  version: '4.12'
}

task hello {
  println 'hello'
}
```

ОК-ОК Groovy, я сделаю как ты хочешь

```
public abstract class ProjectScript extends Script {
  @Override
  public Object invokeMethod(String name, Object args) {
    if ("apply".equals(name)) {
      //А ТУТ У НАС КОД ЛОГИКИ!!!
    }
  }
}
```

```
public class GrannyInternal {
  public GrannyInternal(File buildScript, Project project) {
...

config.setScriptBaseClass(ProjectScript.class.getName());
...
  }
}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
              group: 'junit',
              version: '4.12'
}

task hello {
  println 'hello'
}
```

ОК-ОК Groovy, я сделаю как ты хочешь

```
public abstract class ProjectScript extends Script {
   @Override
   public Object invokeMethod(String name, Object args) {
      if ("apply".equals(name)) {
         //А ТУТ У НАС КОД ЛОГИКИ!!!
      }
   }
}
```

```
public class GrannyInternal {
   public GrannyInternal(File buildScript, Project project) {
...

config.setScriptBaseClass(ProjectScript.class.getName());
...
   }
}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:strea
        'org.yaml:snak
  testCompile name
                group
                versio
}

task hell
  p
}
```

Как-то этот код пахнет

```
if (name == "apply") {
...
} else if (name == "...." && args.length == ...) {
...
} else if
...
}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
                    group: 'junit',
                    version: '4.12'
}

task hello {
  println 'hello'
}
```

Как-то этот код пахнет

```
if (name == "apply") {
...
} else if (name == “....” &
...
} else if
...
}
```

Вынесем обработку

```
public class Project {
  public void apply(Map<String, String> options) { }
}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
              group: 'junit',
              version: '4.12'
}

task hello {
  println 'hello'
}
```

Вынесем обработку

```
public class Project {
  public void apply(Map<String, String> options) { }
}
```

```
public abstract class ProjectScript extends Script {
  private final Project project = new Project();
  @Override
  public Object invokeMethod(String name, Object args) {
    return project.invokeMethod(name, args);
  }
}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
              group: 'junit',
              version: '4.12'
}

task hello {
  println 'hello'
}
```

Вынесем обработку

```
public class Project {
  public void apply(Map<String, String> options) { }
}
```

```
public abstract class ProjectScript extends Script {
  private final Project project = new Project();
  @Override
  public Object invokeMethod(String name, Object args) {
    return project.invokeMethod(name, args);
  }
}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
              group: 'junit',
              version: '4.12'
}

task hello {
  println 'hello'
}
```

Вынесем обработку

```
public class Project extends GroovyObjectSupport {
  public void apply(Map<String, String> options) { }
}
```

```
public abstract class ProjectScript extends Script {
  private final Project project = new Project();
  @Override
  public Object invokeMethod(String name, Object args) {
    return project.invokeMethod(name, args);
  }
}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
              group: 'junit',
              version: '4.12'
}

task hello {
  println 'hello'
}
```

Но есть проблема

```
public class Project extends GroovyObjectSupport {
  public void apply(Map<String, String> options) { }
}
```

```
public abstract class ProjectScript extends Script {
  private final Project project = new Project();
  @Override
  public Object invokeMethod(String name, Object args) {
    return project.invokeMethod(name, args);
  }
}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
              group: 'junit',
              version: '4.12'
}

task hello {
  println 'hello'
}
```

Но есть проблема

```
public class Project extends GroovyObjectSupport {
  public void apply(Map<String, String> options) { }
}
```

```
public class GrannyInternal {
  public GrannyInternal(File buildScript, Project project) {
...
    config.setScriptBaseClass(ProjectScript.class.getName());
  }
}
```

```
  public Object invokeMethod(String name, Object args) {
    return project.invokeMethod(name, args);
  }
}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
              group: 'junit',
              version: '4.12'
}

task hello {
  println 'hello'
}
```

Binding

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
                  group: 'junit',
                  version: '4.12'
}

task hello {
  println 'hello'
}
```

Binding

```
public class GrannyInternal {
  public GrannyInternal(File buildScript, Project project) {
...
      binding = new Binding();
      binding.setProperty("project", project);
  }



}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
        'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
                group: 'junit',
                version: '4.12'
}

task hello {
  println 'hello'
}
```

Binding

```
public class GrannyInternal {
  public GrannyInternal(File buildScript, Project project) {
...
      binding = new Binding();
      binding.setProperty("project", project);
  }
  public void build() throws IOException {
      GroovyShell shell = new GroovyShell(binding, config);
      shell.evaluate(buildScript);
  }
}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
              group: 'junit',
              version: '4.12'
}

task hello {
  println 'hello'
}
```

Binding

```
public class GrannyInternal {
  public GrannyInternal(File buildScript, Project project) {
...
      binding = new Binding();
      binding.setProperty("project", project);
  }
  public void build() throws IOException {
      GroovyShell shell = new GroovyShell(binding, config);
      shell.evaluate(buildScript);
  }
}
```

```
public abstract class ProjectScript extends Script {
  private final Project project = new Project();
  @Override
  public Object invokeMethod(String name, Object args) {
    return project.invokeMethod(name, args);
  }
}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
              group: 'junit',
              version: '4.12'
}

task hello {
  println 'hello'
}
```

Binding

```
public class GrannyInternal {
  public GrannyInternal(File buildScript, Project project) {
...
      binding = new Binding();
      binding.setProperty("project", project);
  }
  public void build() throws IOException {
      GroovyShell shell = new GroovyShell(binding, config);
      shell.evaluate(buildScript);
  }
}
```

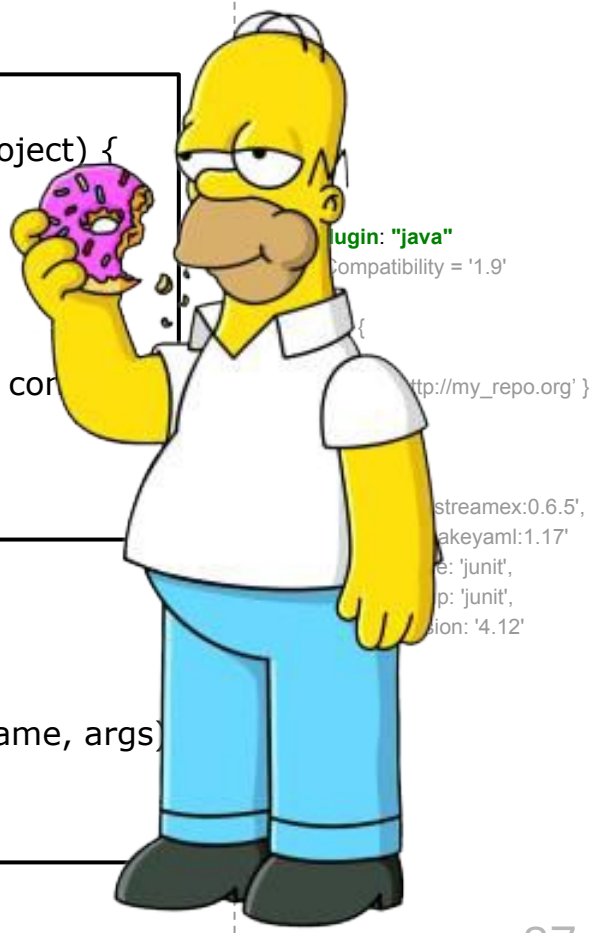
```
public abstract class ProjectScript extends Script {

  @Override
  public Object invokeMethod(String name, Object args) {
    return ((GroovyObjectSupport) getProperty("project")).invokeMethod(name, args);
  }
}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
              group: 'junit',
              version: '4.12'
}

task hello {
  println 'hello'
}
```

Binding

```
public class GrannyInternal {
  public GrannyInternal(File buildScript, Project project) {
...
      binding = new Binding();
      binding.setProperty("project", project);
  }
  public void build() throws IOException {
      GroovyShell shell = new GroovyShell(binding, cor
      shell.evaluate(buildScript);
  }
}
```

```
public abstract class ProjectScript extends Script {

  @Override
  public Object invokeMethod(String name, Object args) {
    return ((GroovyObjectSupport) getProperty("project")).invokeMethod(name, args)
  }
}
```



Как же это работает?

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
                  group: 'junit',
                  version: '4.12'
}

task hello {
  println 'hello'
}
```

Как же это работает?

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

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
                group: 'junit',
                version: '4.12'
}

task hello {
  println 'hello'
}
```

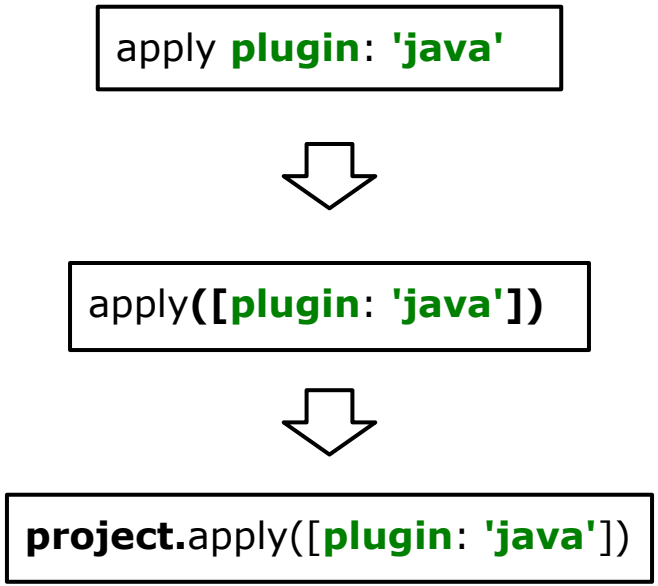
Как же это работает?

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

```
apply([plugin: 'java'])
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
        'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
                group: 'junit',
                version: '4.12'
}

task hello {
  println 'hello'
}
```

Как же это работает?

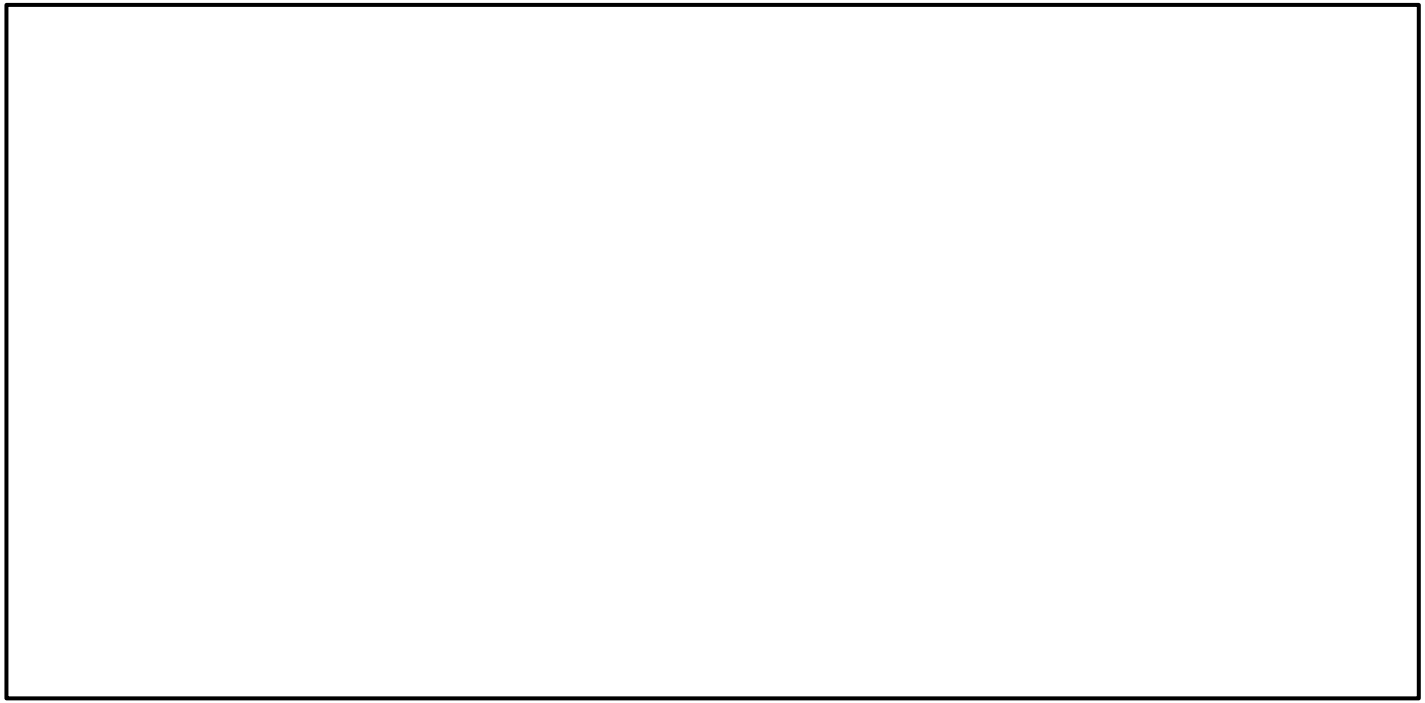



```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
                group: 'junit',
                version: '4.12'
}

task hello {
  println 'hello'
}
```

Как же не указать версию?

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
   jcenter()
   maven { url 'http://custom-repository.com' }
}

dependencies {
   compile 'one.util:streamex:0.6.5',
           'org.yaml:snakeyaml:1.17'
   testCompile name: 'junit', group: 'junit', version: '4.12'
}

task hello {
   println 'hello'
}
```

Попробуем запустить



```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
                  group: 'junit',
                  version: '4.12'
}

task hello {
  println 'hello'
}
```

Попробуем запустить

OK

Process finished with exit code 0

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
        'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
                group: 'junit',
                version: '4.12'
}

task hello {
  println 'hello'
}
```

Попробуем запустить

OK

Process finished with exit code 0

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
              group: 'junit',
              version: '4.12'
}

task hello {
  println 'hello'
}
```

Как? Куда делась?

```
public abstract class Script extends GroovyObjectSupport {




}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
                  group: 'junit',
                  version: '4.12'
}

task hello {
  println 'hello'
}
```

Как? Куда делась?

```
public abstract class Script extends GroovyObjectSupport {
...
   public void setProperty(String property, Object newValue) {
      if ("binding".equals(property))
         setBinding((Binding) newValue);
      else if("metaClass".equals(property))
         setMetaClass((MetaClass)newValue);
      else
         binding.setVariable(property, newValue);
   }
...
}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
                  group: 'junit',
                  version: '4.12'
}

task hello {
  println 'hello'
}
```

Как? Куда делась?

```
public abstract class Script extends GroovyObjectSupport {
...
   public void setProperty(String property, Object newValue) {
      if ("binding".equals(property))
         setBinding((Binding) newValue);
      else if("metaClass".equals(property))
         setMetaClass((MetaClass)newValue);
      else
         binding.setVariable(property, newValue);
   }
...
}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
                group: 'junit',
                version: '4.12'
}

task hello {
  println 'hello'
}
```

Переменная должна быть в проекте

Переменная должна быть в проекте

```
sourceCompatibility = '1.9'
```

Переменная должна быть в проекте

```
sourceCompatibility = '1.9'
```

```
project.sourceCompatibility = '1.9'
```

Переменная должна быть в проекте

```
sourceCompatibility = '1.9'
```

```
project.sourceCompatibility = '1.9'
```

```
project.setSourceCompatibility("1.9")
```

Переменная должна быть в проекте

```
sourceCompatibility = '1.9'
```

```
project.sourceCompatibility = '1.9'
```

```
project.setSourceCompatibility("1.9")
```

```
public class Project extends GroovyObjectSupport {
  private String sourceCompatibility;

  public void setSourceCompatibility(String sourceCompatibility) {
    this.sourceCompatibility = sourceCompatibility;
  }
}
```

Осталось пробросить

```
public abstract class ProjectScript extends Script {




}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
                  group: 'junit',
                  version: '4.12'
}

task hello {
  println 'hello'
}
```

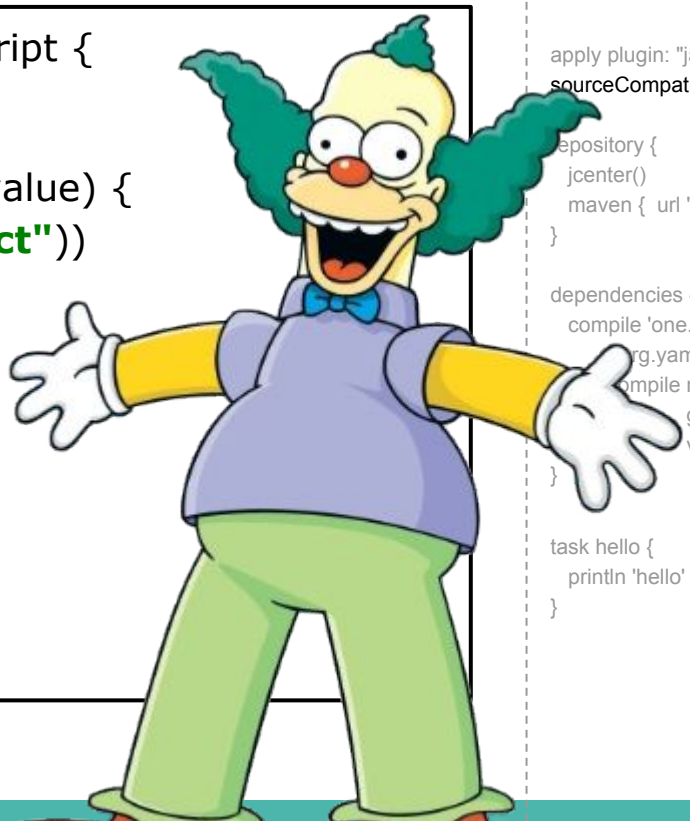
Осталось пробросить

```
public abstract class ProjectScript extends Script {
...
   @Override
   public void setProperty(String name, Object value) {
      ((GroovyObjectSupport) getProperty("project"))
                  .setProperty(name, value);
   }
...
}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
                  group: 'junit',
                  version: '4.12'
}

task hello {
  println 'hello'
}
```

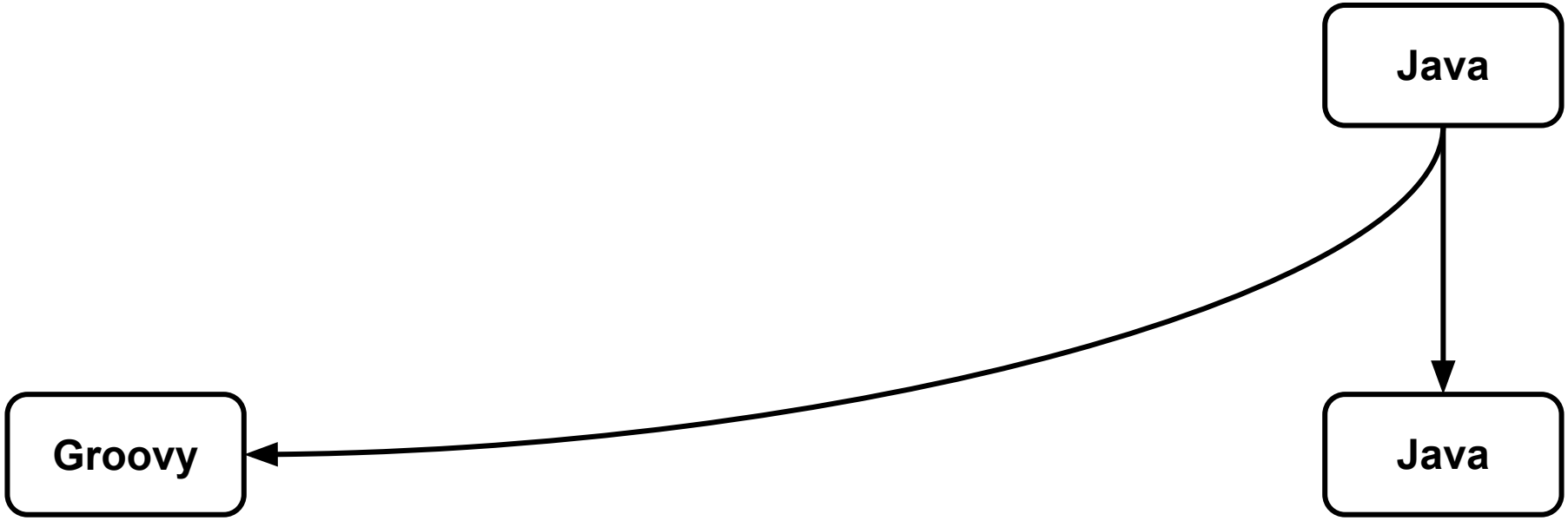
Осталось пробросить

```
public abstract class ProjectScript extends Script {
...
   @Override
   public void setProperty(String name, Object value) {
      ((GroovyObjectSupport) getProperty("project"))
                  .setProperty(name, value);
   }
...
}
```

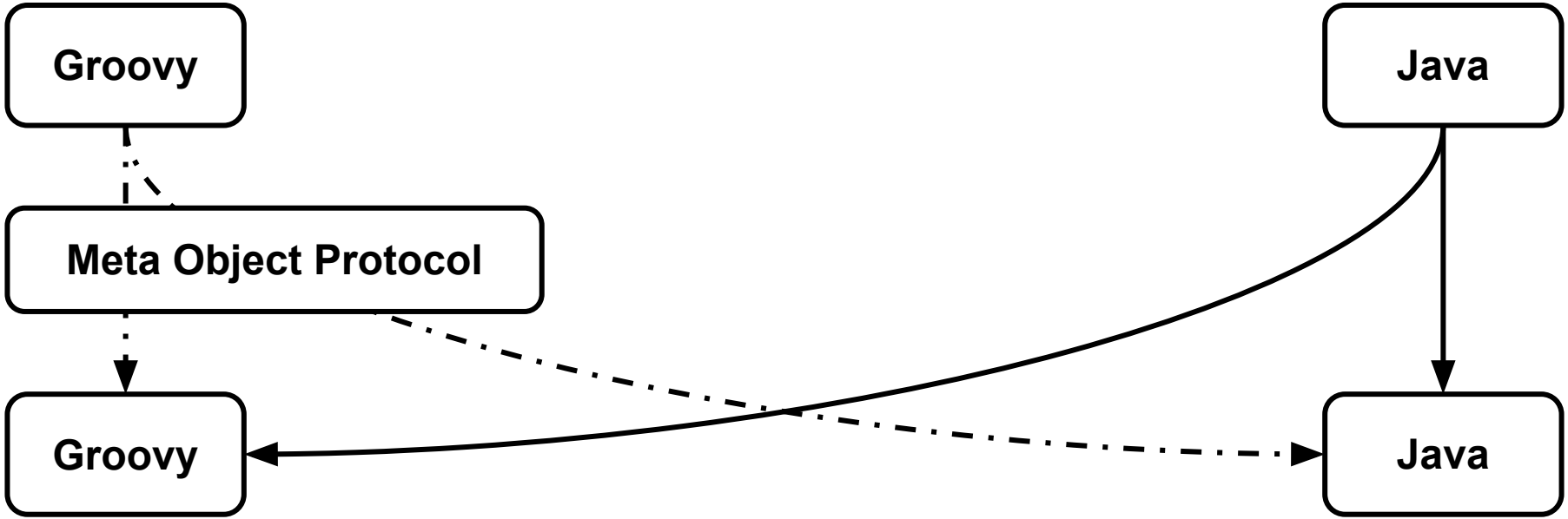


```
apply plugin: "java"
sourceCompatibility = '1.9'

epository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
      rg.yaml:snakeyaml:1.17'
      mpile name: 'junit',
          group: 'junit',
          version: '4.12'
}

task hello {
  println 'hello'
}
```

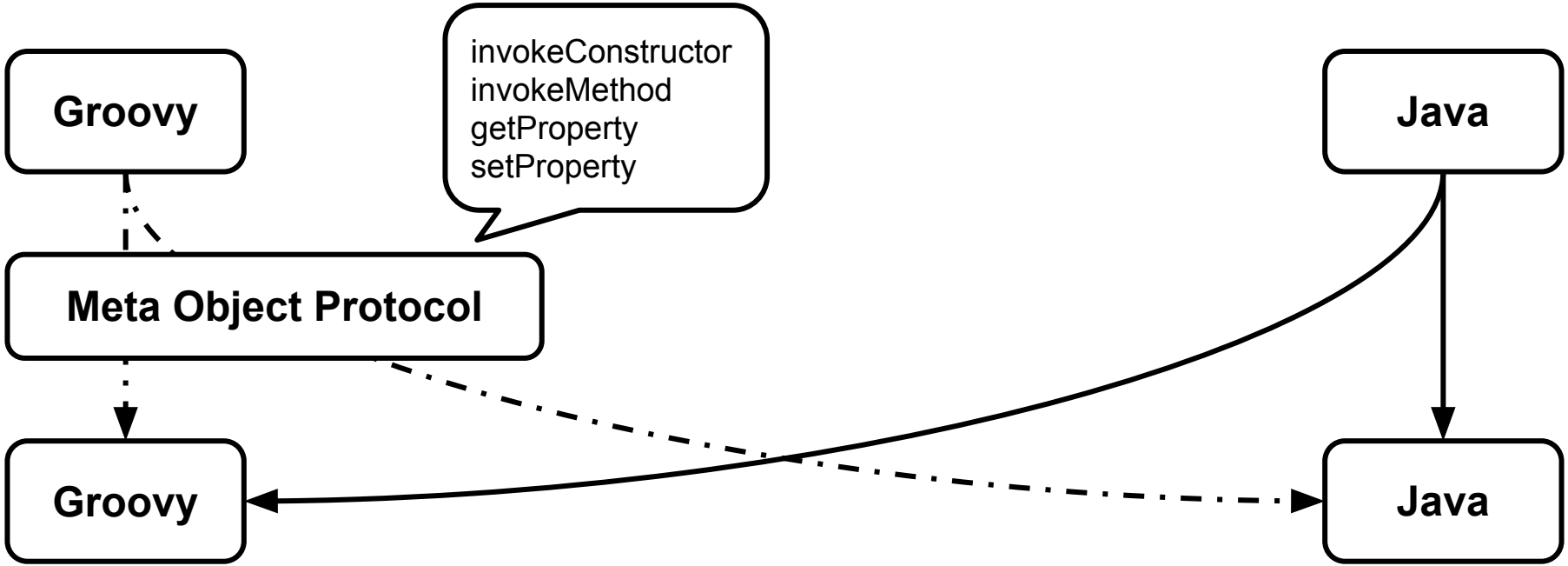
MetaClass



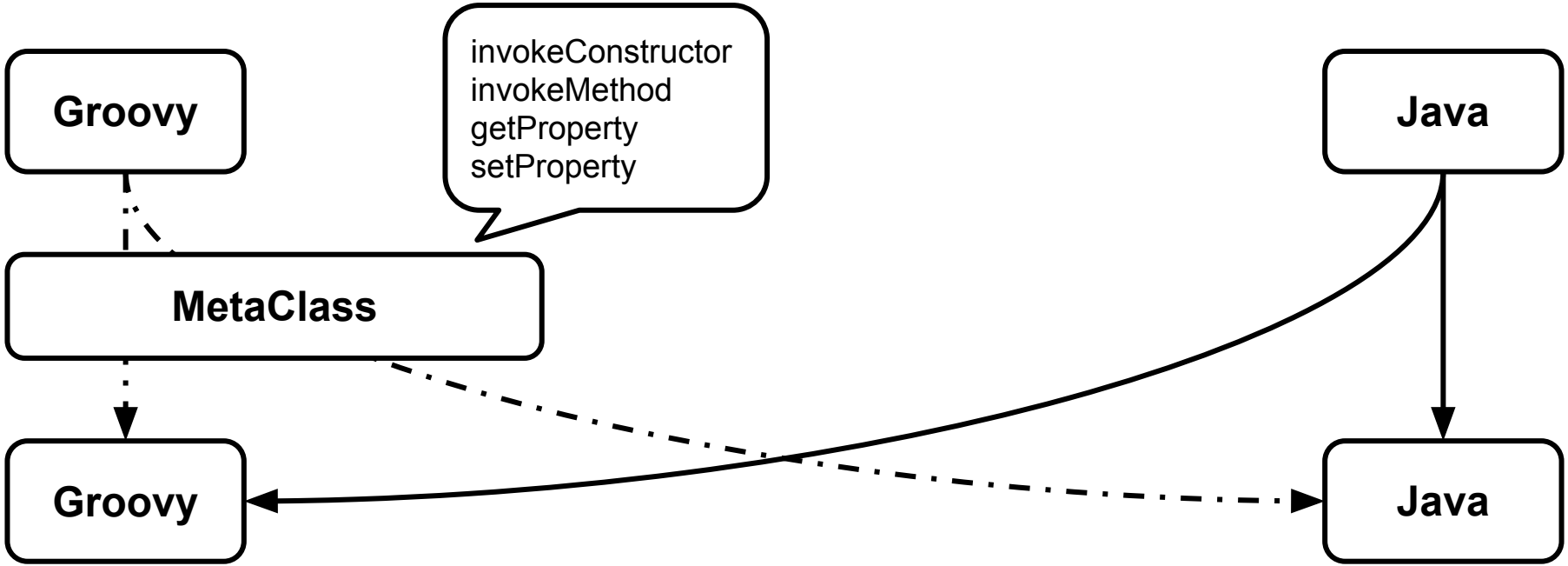
MetaClass



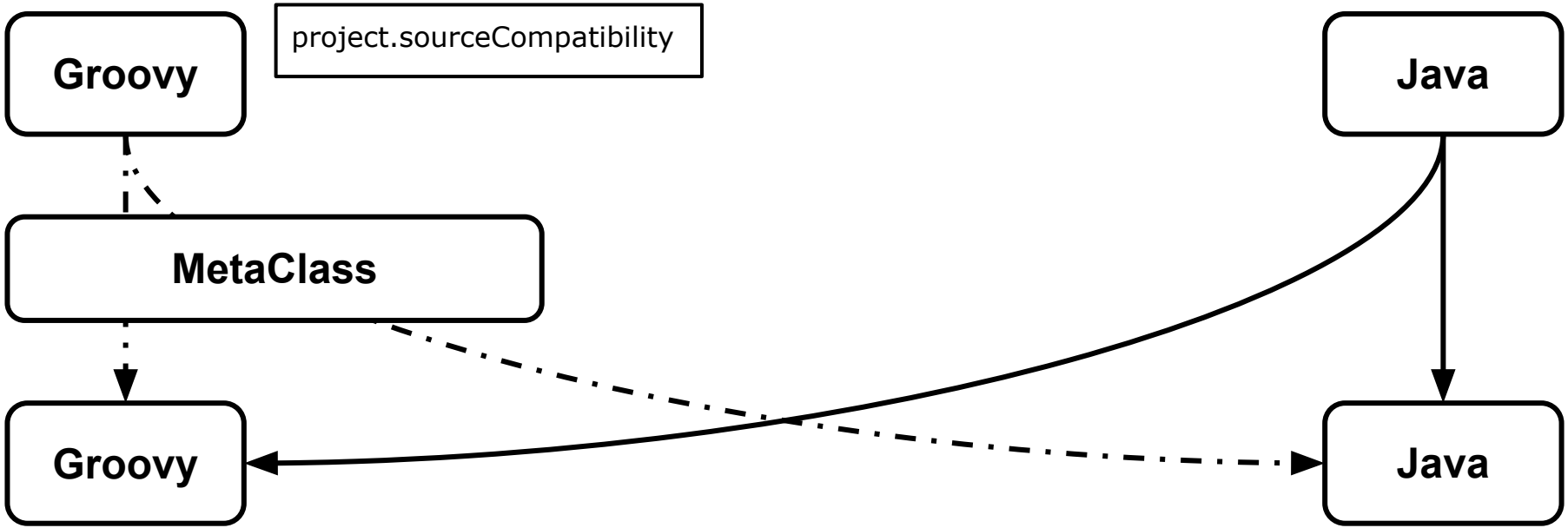
MetaClass


Groovy
invokeConstructor
invokeMethod
getProperty
setProperty
Java
Meta Object Protocol
Groovy
Java

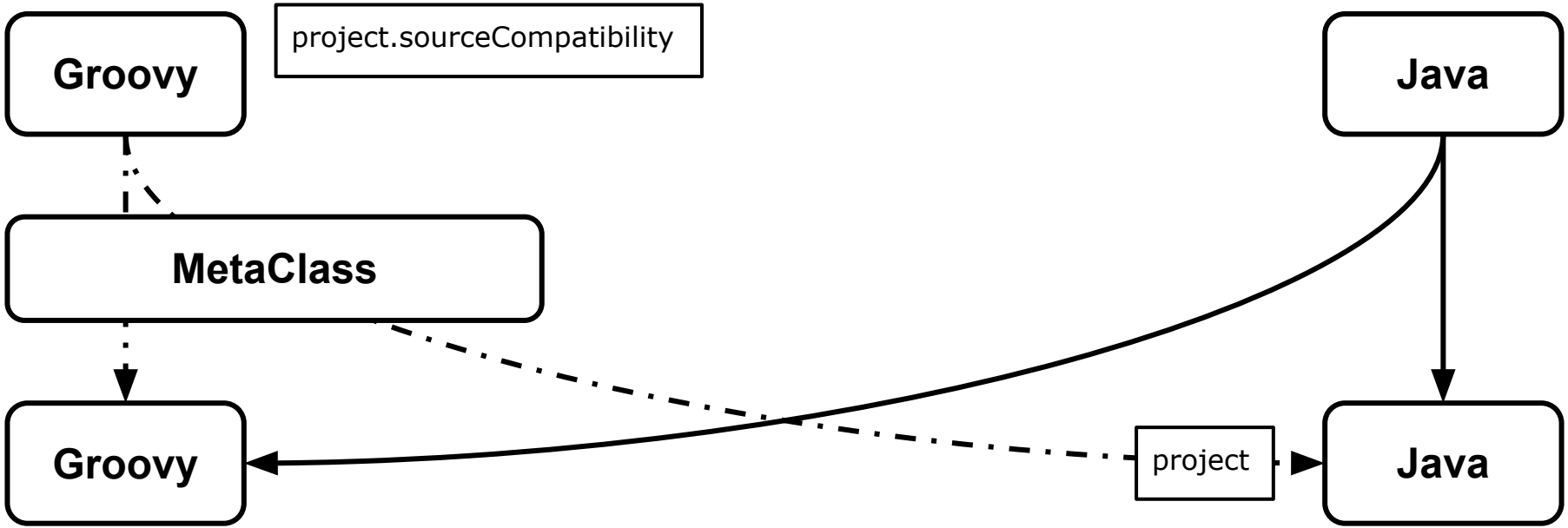
MetaClass


Groovy
invokeConstructor
invokeMethod
getProperty
setProperty
Java
MetaClass
Groovy
Java

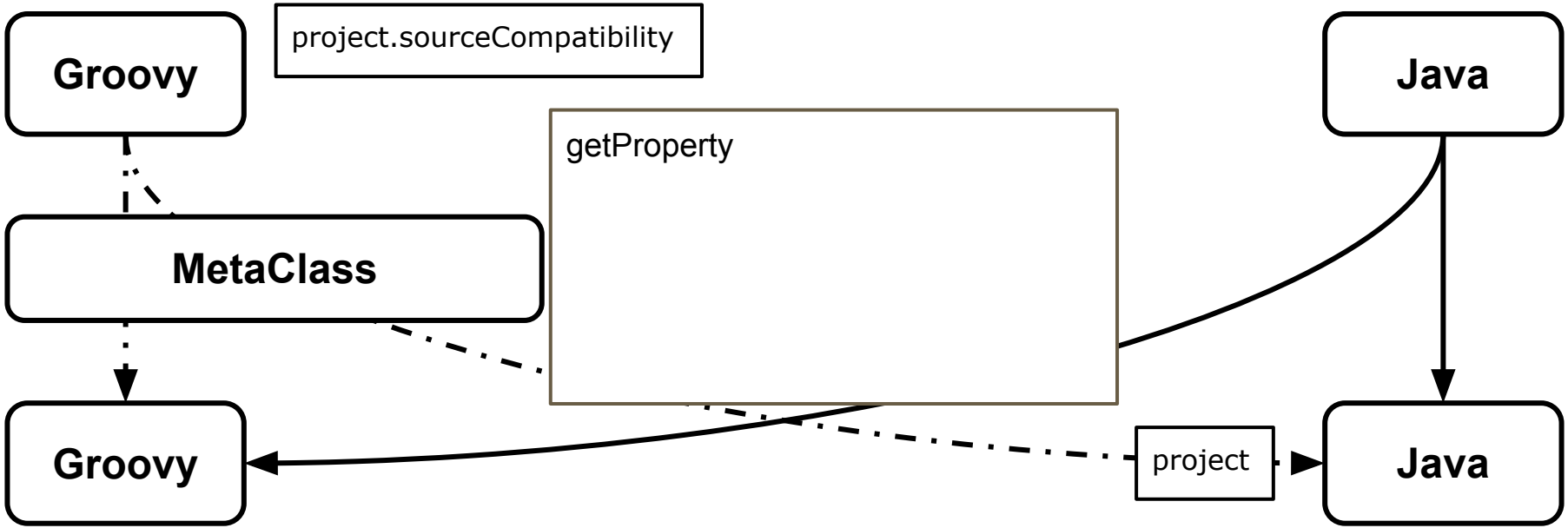
MetaClass


Groovy
project.sourceCompatibility
Java
MetaClass
Groovy
Java

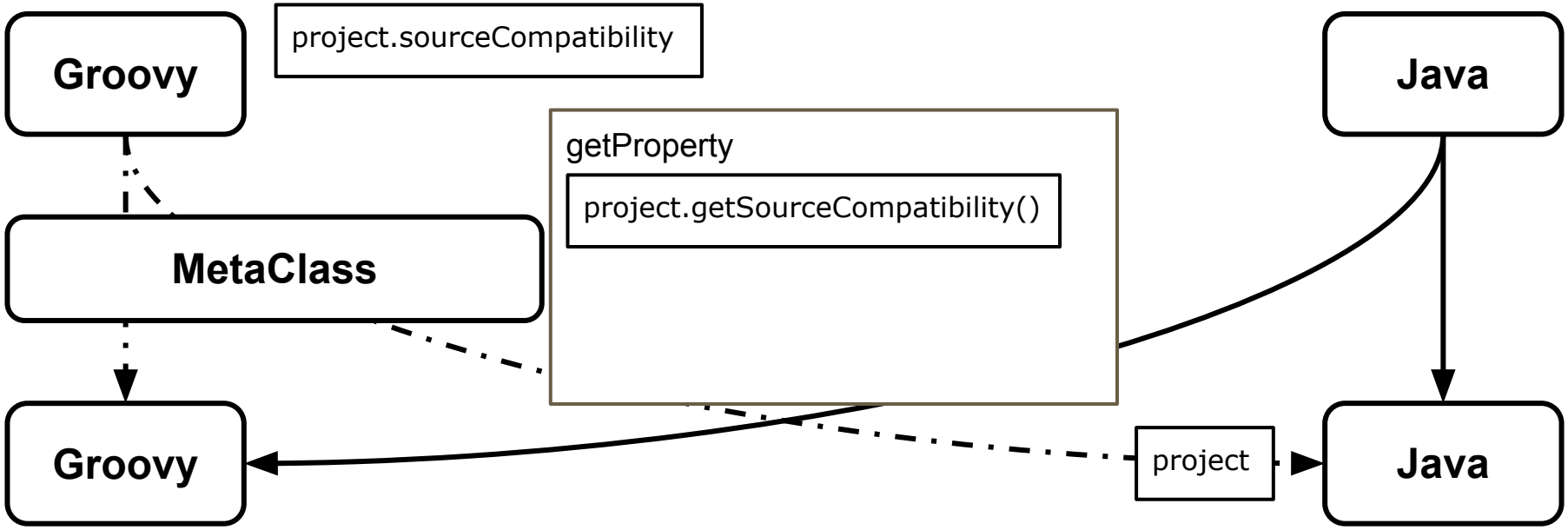
MetaClass


Groovy
project.sourceCompatibility
Java
MetaClass
Groovy
project
Java

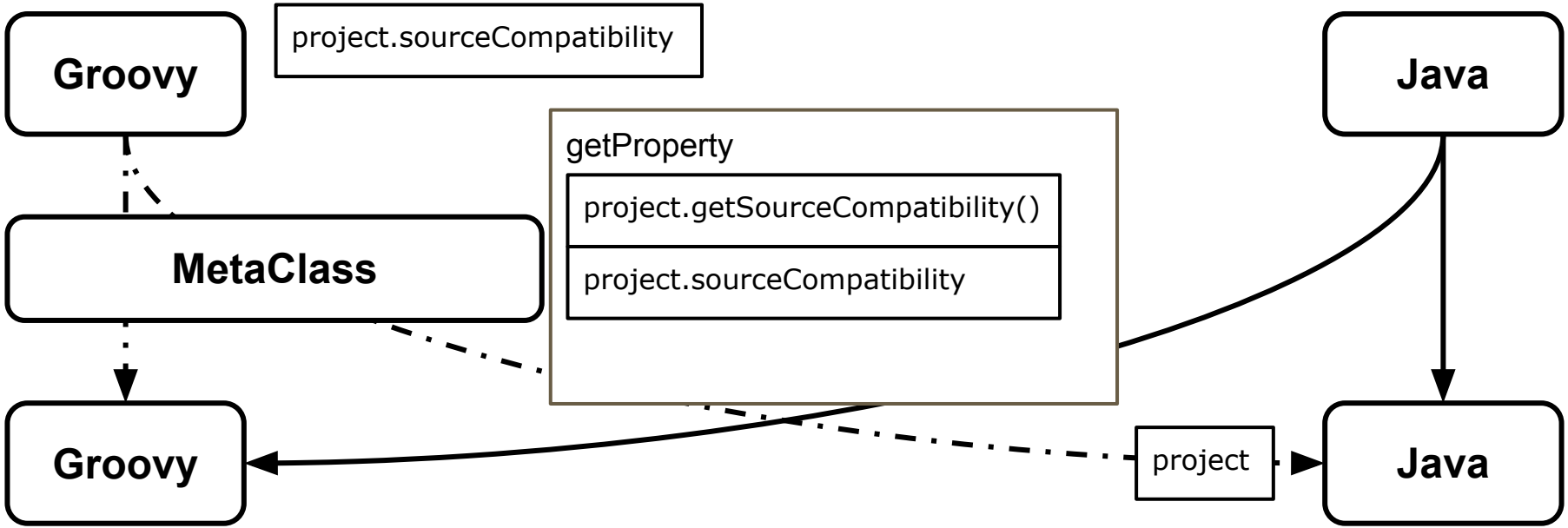
MetaClass


Groovy
project.sourceCompatibility
Java
getProperty
MetaClass
Groovy
project
Java

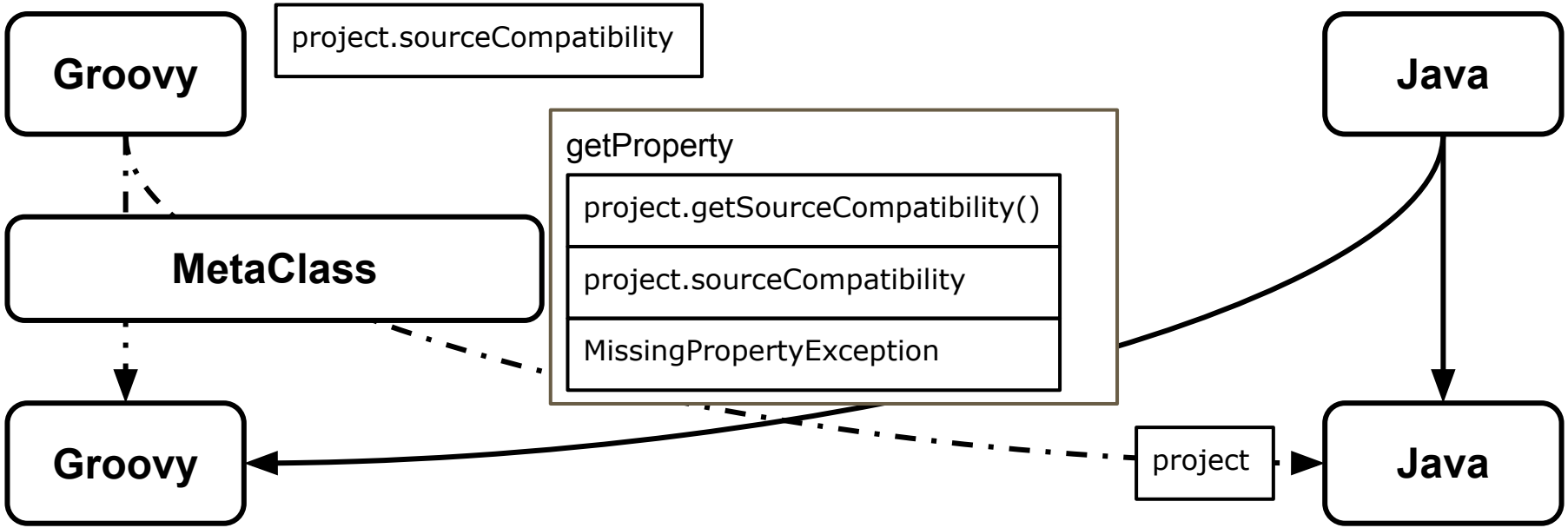
MetaClass


Groovy
project.sourceCompatibility
Java
getProperty
project.getSourceCompatibility()
MetaClass
Groovy
project
Java

MetaClass


Groovy
project.sourceCompatibility
Java
getProperty
project.getSourceCompatibility()
project.sourceCompatibility
MetaClass
Groovy
project
Java

MetaClass


Groovy
project.sourceCompatibility
Java
getProperty
project.getSourceCompatibility()
project.sourceCompatibility
MissingPropertyException
MetaClass
Groovy
project
Java

Идём Дальше

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
   jcenter()
   maven { url 'http://custom-repository.com' }
}

dependencies {
   compile 'one.util:streamex:0.6.5',
           'org.yaml:snakeyaml:1.17'
   testCompile name: 'junit', group: 'junit', version: '4.12'
}

task hello {
   println 'hello'
}
```

Попробуем запустить

```
groovy.lang.MissingMethodException: No signature of
method: com.jokerconf2017.Project.repository() is
applicable for argument types: (Script1$_run_clos
values: [Script1$_run_closure1@73e22a3d]
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  aven {  url 'http://my_repo.org' }

  cies {
  'one.util:streamex:0.6.5',
  g.yaml:snakeyaml:1.17'
  mpile name: 'junit',
        group: 'junit',
        version: '4.12'

ello {
  llo'
```

Попробуем запустить

```
groovy.lang.MissingMethodException: No signature of
method: com.jokerconf2017.Project.repository() is
applicable for argument types: (Script1$_run_closure1)
values: [Script1$_run_closure1@73e22a3d]
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
                group: 'junit',
                version: '4.12'
}

task hello {
  println 'hello'
}
```

Попробуем запустить

```
groovy.lang.MissingMethodException: No signature of method: com.jokerconf2017.Project.repository() is applicable for argument types: (Script1$_run_closure1) values: [Script1$_run_closure1@73e22a3d]
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
              group: 'junit',
              version: '4.12'
}

task hello {
  println 'hello'
}
```

Попробуем запустить

```
groovy.lang.MissingMethodException: No signature of
method: com.jokerconf2017.Project.repository() is
applicable for argument types:
values: [Script1$_run_closure1@73e22a3d]
```

Script1$_run_closure1

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
              group: 'junit',
              version: '4.12'
}

task hello {
  println 'hello'
}
```

Closure

Closure

```
{ 5 + 7 }
```

Closure

```
def closure = {
    // code
}
```

Closure

```
def closure = { a, b ->
    // code
}
```

Closure

```
def closure = { int a, b ->
    // code
}
```

Closure

```
Closure<Boolean> closure = { int a, b ->
    // code
}
```

Closure

```
Closure<Boolean> closure = { int a, b ->
    // code
}
closure(1, 2)
```

Closure

```
Closure<Boolean> closure = { int a, b ->
    // code
}
closure.call(1, 2)
```

repository { jcenter() }

```
public class Project extends GroovyObjectSupport {



}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
                group: 'junit',
                version: '4.12'
}

task hello {
  println 'hello'
}
```

repository { jcenter() }

```
public class Project extends GroovyObjectSupport {
   public void repository(Closure closure) {

   }
}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
                  group: 'junit',
                  version: '4.12'
}

task hello {
  println 'hello'
}
```

repository { jcenter() }

```
public class Project extends GroovyObjectSupport {
   public void repository(Closure closure) {
      closure.call();
   }
}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
              group: 'junit',
              version: '4.12'
}

task hello {
  println 'hello'
}
```

repository { jcenter() }

```
public class Project extends GroovyObjectSupport {
   public void repository(Closure closure) {
      closure.call();
   }
}
```

```
groovy.lang.MissingMethodException: No signature of
method: com.jokerconf2017.Project.jcenter() is
applicable for argument types: () values: []
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
                group: 'junit',
                version: '4.12'
}

task hello {
  println 'hello'
}
```

repository { jcenter() }

```
public class Project extends GroovyObjectSupport {
   public void repository(Closure closure) {
      closure.call();
   }
}
```

```
groovy.lang.MissingMethodException: No signature of
method: com.jokerconf2017
applicable for argument types: () values: []
```

Project.jcenter()

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
                group: 'junit',
                version: '4.12'
}

task hello {
  println 'hello'
}
```

repository { jcenter() }

```
public class Project extends GroovyO
   public void repository(Closure closu
      closure.call();
   }
}
```


```
groovy.lang.MissingMethodExc
method: com.jokerconf2017.Pr
applicable for argument types: () values: []
```

```
va"
ility = '1.9'

ttp://my_repo.org' }

til:streamex:0.6.5',
:snakeyaml:1.17'
ame: 'junit',
roup: 'junit',
ersion: '4.12'

task hello {
   println 'hello'
}
```

```
public class RepositoryHandler {
   public final Set<String> repositories = new LinkedHashSet<>();
   public void jcenter() {
      repositories.add("http://jcenter.bintray.com/");
   }
}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
                  group: 'junit',
                  version: '4.12'
}

task hello {
  println 'hello'
}
```

```java
public class RepositoryHandler {
  public final Set<String> repositories = new LinkedHashSet<>();
  public void jcenter() {
    repositories.add("http://jcenter.bintray.com/");
  }
}
```

```java
public class Project extends GroovyObjectSupport {
  public void repository(Closure closure) {

    closure.call();
  }
}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
              group: 'junit',
              version: '4.12'
}

task hello {
  println 'hello'
}
```

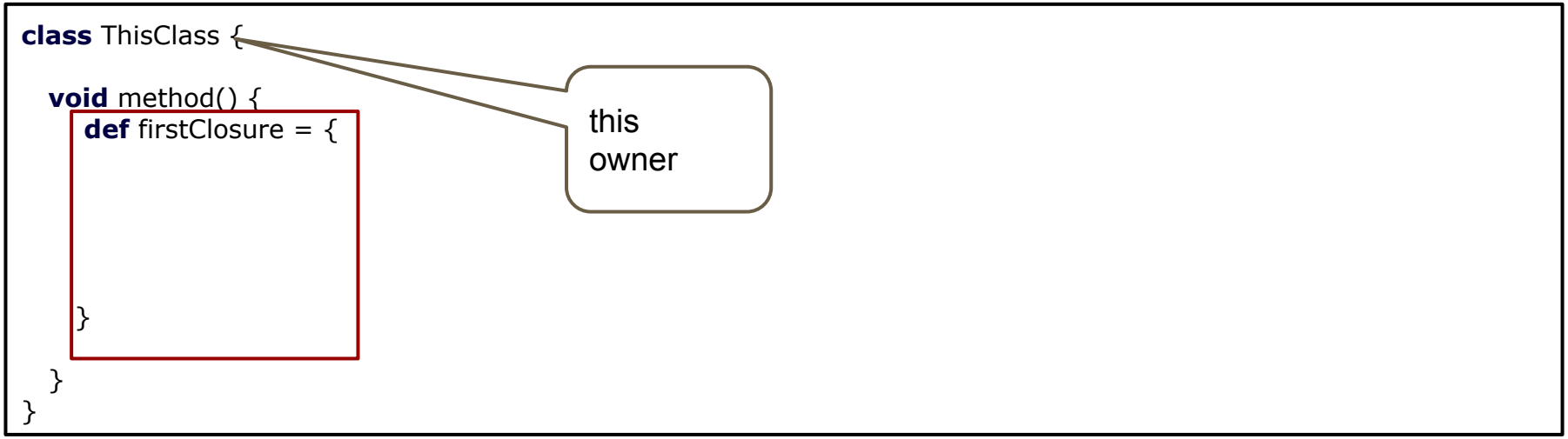
Closure context

this	Ссылка на породивший объект (до первого класса)
owner	Ссылка на породивший объект/closure (на 1 уровень вверх)
delegate	Ссылка на объект исполнитель

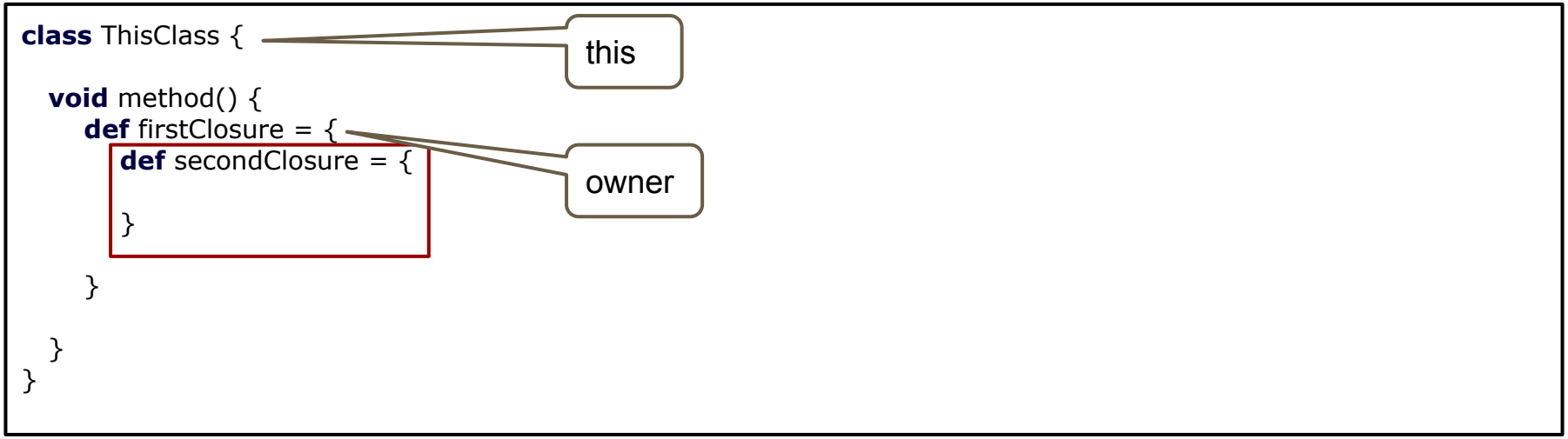
Closure context

```
class ThisClass {

   void method() {
      def firstClosure = {




      }

   }
}
```

Closure context



Closure context

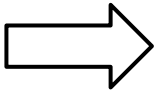


Closure - delegate context

```
def cl = { -> append "Hello" }
def sb = new StringBuilder()
cl.delegate = sb

cl()
println ("$sb closure!")
```

Closure - delegate context

```
def cl = { -> append "Hello" }
def sb = new StringBuilder()
cl.delegate = sb

cl()
println ("$sb closure!")
```



```
"Hello closure!"
```

Delegate

```
public class RepositoryHandler {
   public final Set<String> repositories = new LinkedHashSet<>();
   public void jcenter() {
      repositories.add("http://jcenter.bintray.com/");
   }
}
```

```
public class Project extends GroovyObjectSupport {
   public void repository(Closure cl) {
      cl.setDelegate(repositoryHandler);
      cl.call();
   }
}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
              group: 'junit',
              version: '4.12'
}

task hello {
  println 'hello'
}
```

Delegate

```
public class RepositoryHandler {
   public final Set<String> repositories = new LinkedHashSet<>();
   public void jcenter() {
      repositories.add("http://jcenter.bintray.com/");
   }
}
```

```
public class Project extends GroovyObjectSupport {
   public void repository(Closure cl) {
      cl.setDelegate(repositoryHandler);
      cl.call();
   }
}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

itory {
nter()
en {  url 'http://my_repo.org' }


encies {
mpile 'one.util:streamex:0.6.5',
      'org.yaml:snakeyaml:1.17'
estCompile name: 'junit',
                group: 'junit',
                version: '4.12'
}

task hello {
   println 'hello'
}
```

@DelegatesTo

```
public class RepositoryHandler {
   public final Set<String> repositories = new LinkedHashSet<>();
   public void jcenter() {
      repositories.add("http://jcenter.bintray.com/");
   }
}
```

```
public class Project extends GroovyObjectSupport {
   public void repository(@DelegatesTo(RepositoryHandler.class) Closure cl) {
      cl.setDelegate(repositoryHandler);
      cl.call();
   }
}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
              group: 'junit',
              version: '4.12'
}

task hello {
  println 'hello'
}
```

Идём Дальше

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
   jcenter()
   maven { url 'http://custom-repository.com' }
}

dependencies {
   compile 'one.util:streamex:0.6.5',
           'org.yaml:snakeyaml:1.17'
   testCompile name: 'junit', group: 'junit', version: '4.12'
}

task hello {
   println 'hello'
}
```

И ещё раз Delegate

```
public class RepositoryHandler {
  public final Set<String> repositories = new LinkedHashSet<>();
  public void maven(Closure closure) {


    closure.call();
    repositories.add(???);
  }
}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
              group: 'junit',
              version: '4.12'
}

task hello {
  println 'hello'
}
```

И ещё раз Delegate

```
public class RepositoryHandler {
  public final Set<String> repositories = new LinkedHashSet<>();
  public void maven(Closure closure) {


    closure.call();
    repositories.add(???);
  }
}
```

```
public class MavenRepository {

  public void url(String url) { this.url = url; }

}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
              group: 'junit',
              version: '4.12'
}

task hello {
  println 'hello'
}
```

И ещё раз Delegate

```
public class RepositoryHandler {
  public final Set<String> repositories = new LinkedHashSet<>();
  public void maven(@DelegatesTo(MavenRepository.class) Closure closure) {
    MavenRepository mavenRepository = new MavenRepository();
    closure.setDelegate(mavenRepository);
    closure.call();
    repositories.add(???);
  }
}
```

```
public class MavenRepository {

  public void url(String url) { this.url = url; }

}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
              group: 'junit',
              version: '4.12'
}

task hello {
  println 'hello'
}
```

И ещё раз Delegate

```
public class RepositoryHandler {
  public final Set<String> repositories = new LinkedHashSet<>();
  public void maven(@DelegatesTo(MavenRepository.class) Closure closure) {
    MavenRepository mavenRepository = new MavenRepository();
    closure.setDelegate(mavenRepository);
    closure.call();
    repositories.add(mavenRepository.getUrl());
  }
}
```

```
public class MavenRepository {
  private String url;
  public void url(String url) { this.url = url; }
  public String getUrl() { return url; }
}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
              group: 'junit',
              version: '4.12'
}

task hello {
  println 'hello'
}
```

Самое главное - зависимости

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
   jcenter()
   maven { url 'http://custom-repository.com' }
}

dependencies {
   compile 'one.util:streamex:0.6.5',
           'org.yaml:snakeyaml:1.17'
   testCompile name: 'junit', group: 'junit', version: '4.12'
}

task hello {
   println 'hello'
}
```

И ещё раз Delegate

```
public class DependencyHandler {
  public final Set<String> depends = new HashSet<>();


  public void compile(String... dependencies) {
    depends.addAll(Arrays.asList(dependencies));
  }




}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
              group: 'junit',
              version: '4.12'
}

task hello {
  println 'hello'
}
```

И ещё раз Delegate

```
public class DependencyHandler {
  public final Set<String> depends = new HashSet<>();
  public final Set<String> testDeps = new HashSet<>();

  public void compile(String... dependencies) {
    depends.addAll(Arrays.asList(dependencies));
  }

  public void testCompile(Map<String, String> d) {
    testDeps.add(d.get("group")+":"+d.get("name")+":"+d.get("version"));
  }
}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
              group: 'junit',
              version: '4.12'
}

task hello {
  println 'hello'
}
```

И ещё раз Delegate

```groovy
public class Project extends GroovyObjectSupport {
  public void dependencies(@DelegatesTo(DependencyHandler.class) Closure closure) {
    closure.setDelegate(dependencyHandler);
    closure.call();
  }
}
```

```groovy
public class DependencyHandler {
  public final Set<String> depends = new HashSet<>();
  public final Set<String> testDeps = new HashSet<>();

  public void compile(String... dependencies) {
    depends.addAll(Arrays.asList(dependencies));
  }

  public void testCompile(Map<String, String> d) {
    testDeps.add(d.get("group")+":"+d.get("name")+":"+d.get("version"));
  }
}
```

```groovy
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
              group: 'junit',
              version: '4.12'
}

task hello {
  println 'hello'
}
```

Дополнительные возможности Closure

Дополнительные возможности Closure

Curry

```
def sum = { a, b -> a + b }
```

Дополнительные возможности Closure

Curry

```
def sum = { a, b -> a + b }

def dec = sum.curry(a:-1)
assert dec(b: 5) == 4
```

Дополнительные возможности Closure

Curry

```
def sum = { a, b -> a + b }

def dec = sum.curry(a:-1)
assert dec(b: 5) == 4

def inc = sum.rcurry(b: 1)
assert inc(a: 5) == 6
```

Дополнительные возможности Closure

Curry

```
def sum = { a, b -> a + b }

def dec = sum.curry(a:-1)
assert dec(b: 5) == 4

def inc = sum.rcurry(b: 1)
assert inc(a: 5) == 6
```

Memoize

```
def fib
fib = { n ->
  n < 2 ? n : fib(n-1) + fib(n-2)
}

fib(10)
```

Вызовов: 177

Дополнительные возможности Closure

Curry

```
def sum = { a, b -> a + b }

def dec = sum.curry(a:-1)
assert dec(b: 5) == 4

def inc = sum.rcurry(b: 1)
assert inc(a: 5) == 6
```

Memoize

```
def fib
fib = { n ->
  n < 2 ? n : fib(n-1) + fib(n-2)
}.memoize()

fib(10)
```

Вызовов: 11

Дополнительные возможности Closure

Curry

```
def sum = { a, b -> a + b }

def dec = sum.curry(a:-1)
assert dec(b: 5) == 4

def inc = sum.rcurry(b: 1)
assert inc(a: 5) == 6
```

Memoize

```
def fib
fib = { n ->
  n < 2 ? n : fib(n-1) + fib(n-2)
}.memoize()

fib(10)
```

Вызовов: 11

Trampoline

```
def fact
fact = { int n, def acc = 1G ->
   if (n < 2) return acc
   fact(n - 1, n * acc)
}

fact(10_000)
```

java.lang.StackOverflowError

Дополнительные возможности Closure

Curry

```
def sum = { a, b -> a + b }

def dec = sum.curry(a:-1)
assert dec(b: 5) == 4

def inc = sum.rcurry(b: 1)
assert inc(a: 5) == 6
```

Memoize

```
def fib
fib = { n ->
  n < 2 ? n : fib(n-1) + fib(n-2)
}.memoize()

fib(10)
```

Вызовов: 11

Trampoline

```
def fact
fact = { int n, def acc = 1G ->
   if (n < 2) return acc
   fact.trampoline(n - 1, n * acc)
}.trampoline()

fact(10_000)
```

```
dependencies {
    compile group: 'junit', name: 'junit', version: '4.12'
}
```

```
public abstract class ProjectScript extends Script {

    @Override
    public Object invokeMethod(String name, Object args) {
        return ((GroovyObjectSupport) getProperty("project")).invokeMethod(name, args);
    }
}
```

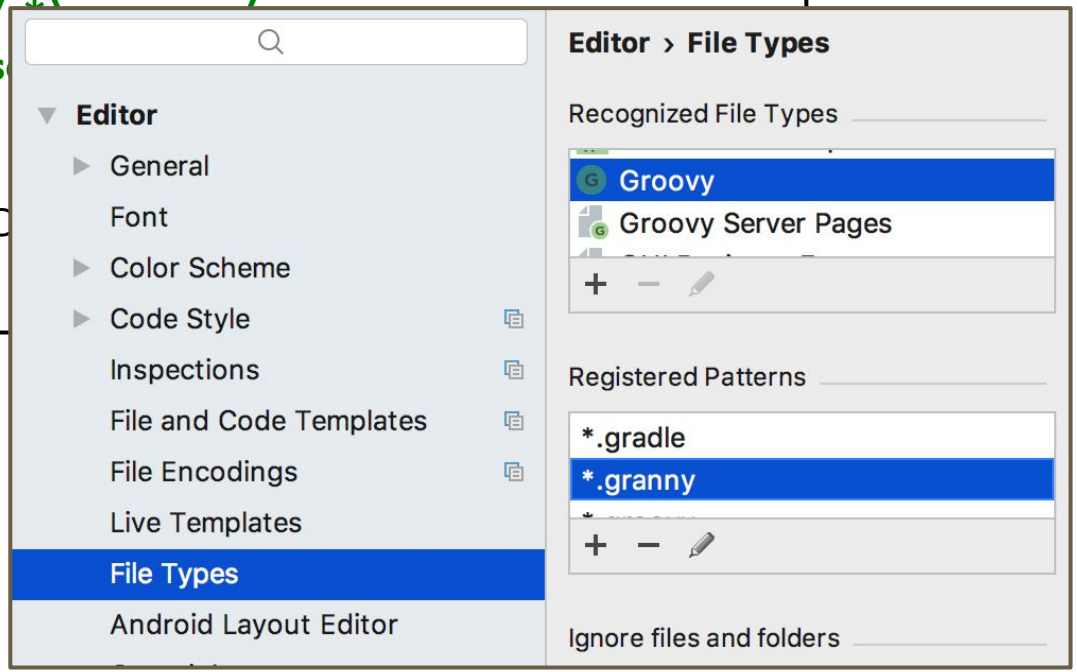
GDSL

GDSL

```
def scriptName = /.*\.granny/
def ctx = context(scope: scriptScope(name: scriptName))

contributor(ctx) {
   delegatesTo(findClass("com.jokerconf2017.Project"))
}
```

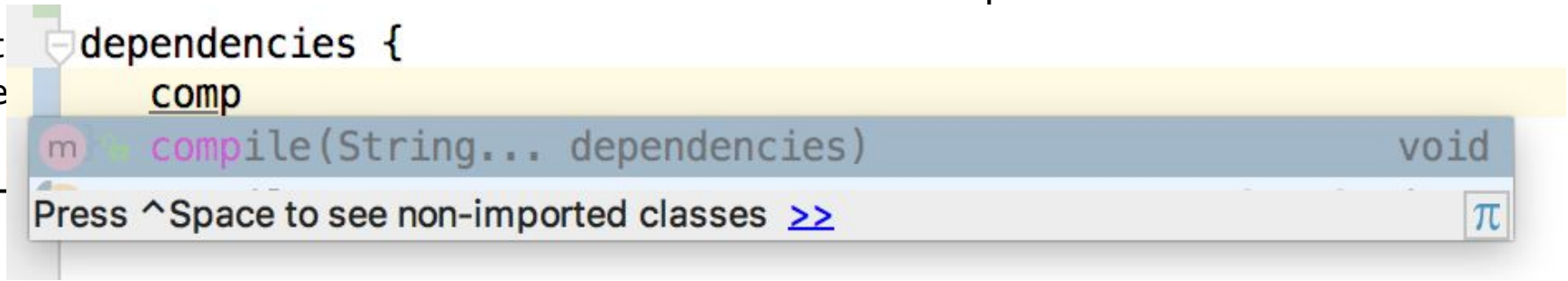
GDSL

```
def scriptName = /*
def ctx = context(s

contributor(ctx) {
   delegatesTo(findC
}
```



GDSL

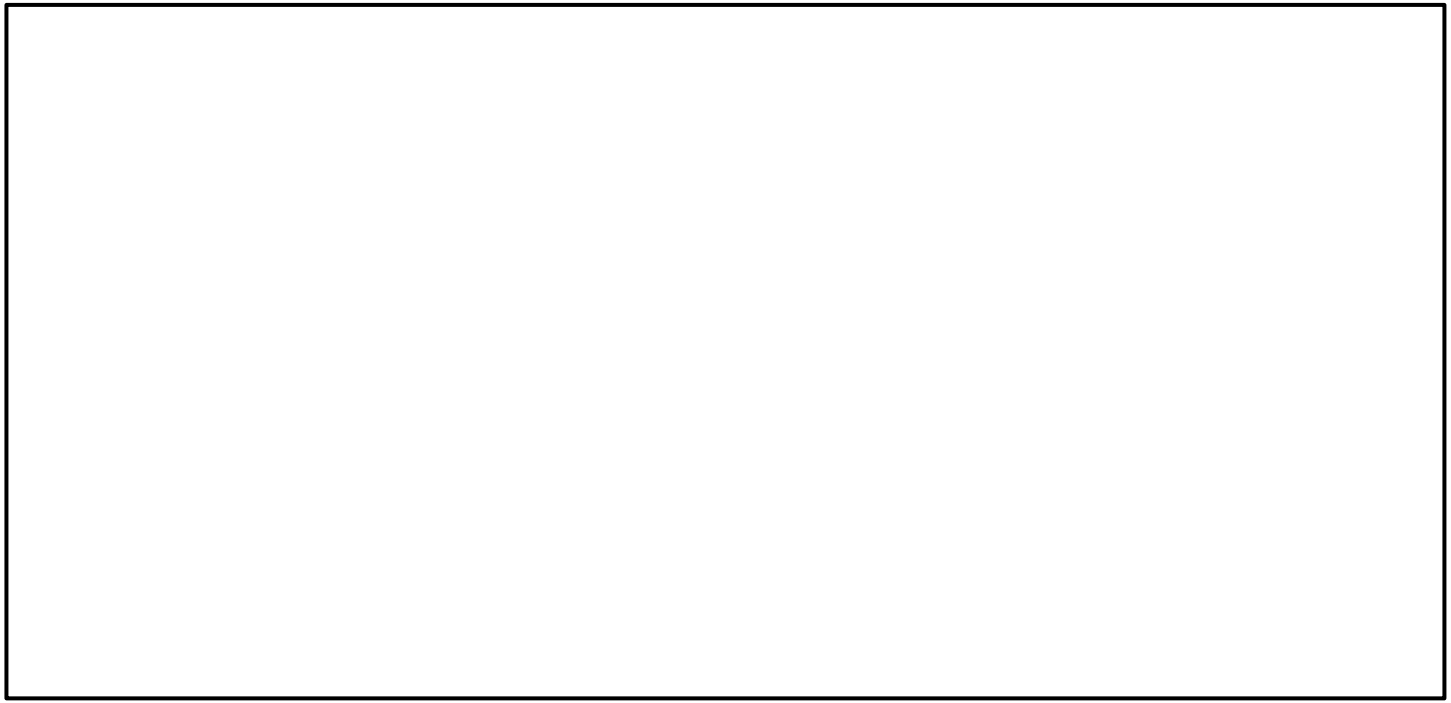
```
def scriptName = /.*\.granny/
def ctx = context(scope: scriptScope(name: scriptName))

cont
  de
}
```



И на вкусненькое

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
   jcenter()
   maven { url 'http://custom-repository.com' }
}

dependencies {
   compile 'one.util:streamex:0.6.5',
           'org.yaml:snakeyaml:1.17'
   testCompile name: 'junit', group: 'junit', version: '4.12'
}

task hello {
   println 'hello'
}
```

Попробуем запустить



```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
                group: 'junit',
                version: '4.12'
}

task hello {
  println 'hello'
}
```

Попробуем запустить

```
groovy.lang.MissingMethodException: No signature of
method: com.jokerconf2017.Project.hello() is applicable
for argument types: (Script1$_run_closure1) values:
[Script1$_run_closure1@73e22a3d]
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
                group: 'junit',
                version: '4.12'
}

task hello {
  println 'hello'
}
```

Попробуем запустить

```
groovy.lang.MissingMethodException: No signature of method: com.jokerconf2017.Project.hello() is applicable for argument types: (Script1$_run_closure1) values: [Script1$_run_closure1@73e22a3d]
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
                  group: 'junit',
                  version: '4.12'
}

task hello {
  println 'hello'
}
```

Попробуем запустить

groovy.lang.MissingMethodException: No signature of method: com.jokerconf2017.Project.hello() is applicable for argumen… …closure1) values: [Script1$_ru…

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
                group: 'junit',
                version: '4.12'
}

task hello {
  println 'hello'
}
```

Ну ОК! Попробуем

```
public class Project extends GroovyObjectSupport {
   public void hello(Closure closure) {}
}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
                  group: 'junit',
                  version: '4.12'
}

task hello {
  println 'hello'
}
```

Ну ОК! Попробуем

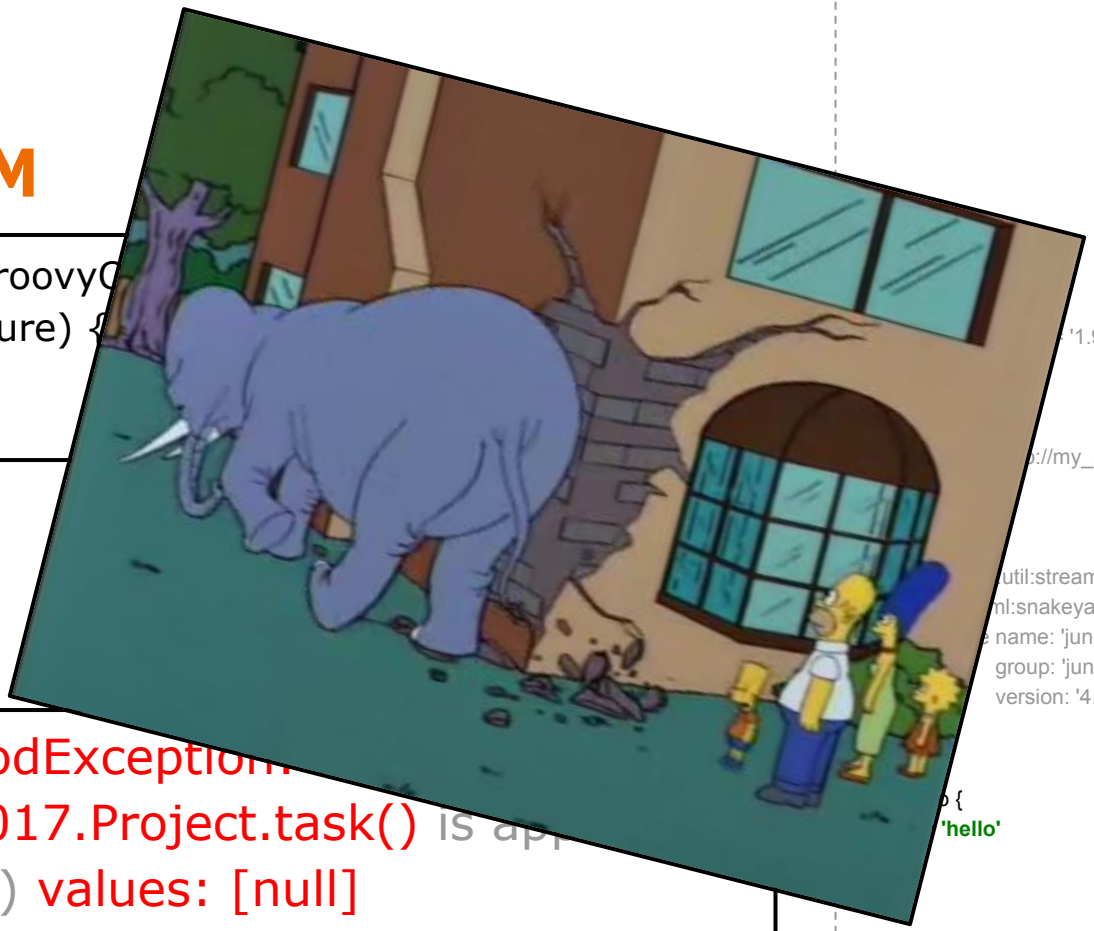
```
public class Project extends GroovyObjectSupport {
   public void hello(Closure closure) {}
}
```

```
groovy.lang.MissingMethodException: No signature of
method: com.jokerconf2017.Project.task() is applicable
for argument types: (null) values: [null]
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
                group: 'junit',
                version: '4.12'
}

task hello {
  println 'hello'
}
```

Ну ОК! Попробуем

```
public class Project extends GroovyC
   public void hello(Closure closure) {
}
```



```
groovy.lang.MissingMethodException:
method: com.jokerconf2017.Project.task() is app
for argument types: (null) values: [null]
```

TASK

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
                  group: 'junit',
                  version: '4.12'
}

task hello {
  println 'hello'
}
```

TASK

```
task hello {
   println "hello"
}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
                group: 'junit',
                version: '4.12'
}

task hello {
  println 'hello'
}
```

TASK

```
task hello {
   println "hello"
}
```

```
project.task(project.hello({
   println "hello"
}))
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
                  group: 'junit',
                  version: '4.12'
}

task hello {
  println 'hello'
}
```

TASK

```
task hello {
   println "hello"
}
```

```
project.task(project.hello({
   println "hello"
}))
```

```
project.task("hello", {
   println "hello"
})
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
              group: 'junit',
              version: '4.12'
}

task hello {
  println 'hello'
}
```

TASK

```
task hello {
   println "hello"
}
```

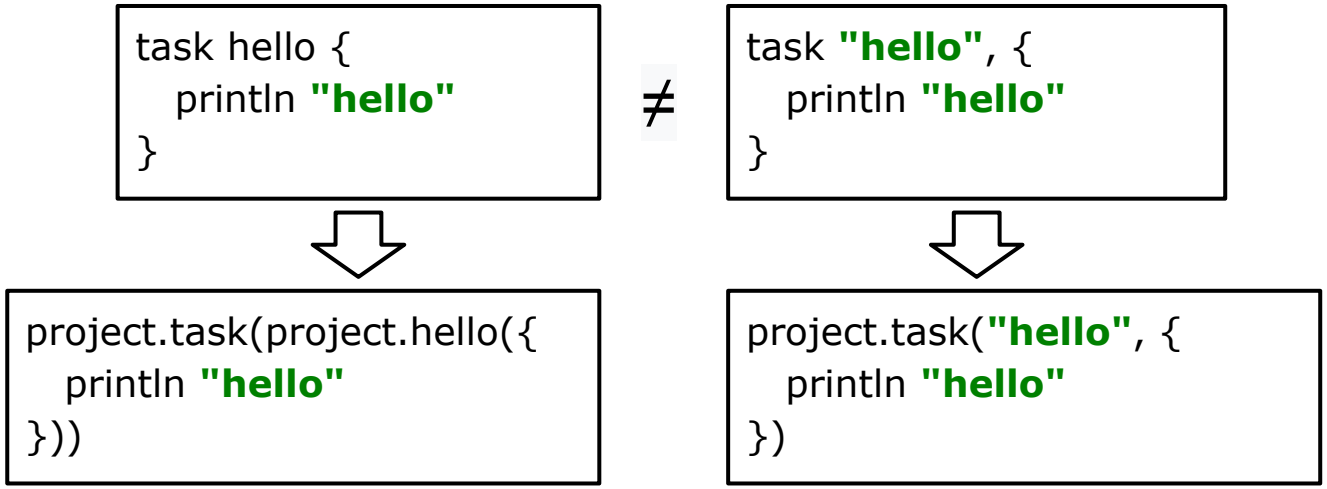
```
project.task(project.hello({
   println "hello"
}))
```

```
task "hello", {
   println "hello"
}
```

```
project.task("hello", {
   println "hello"
})
```

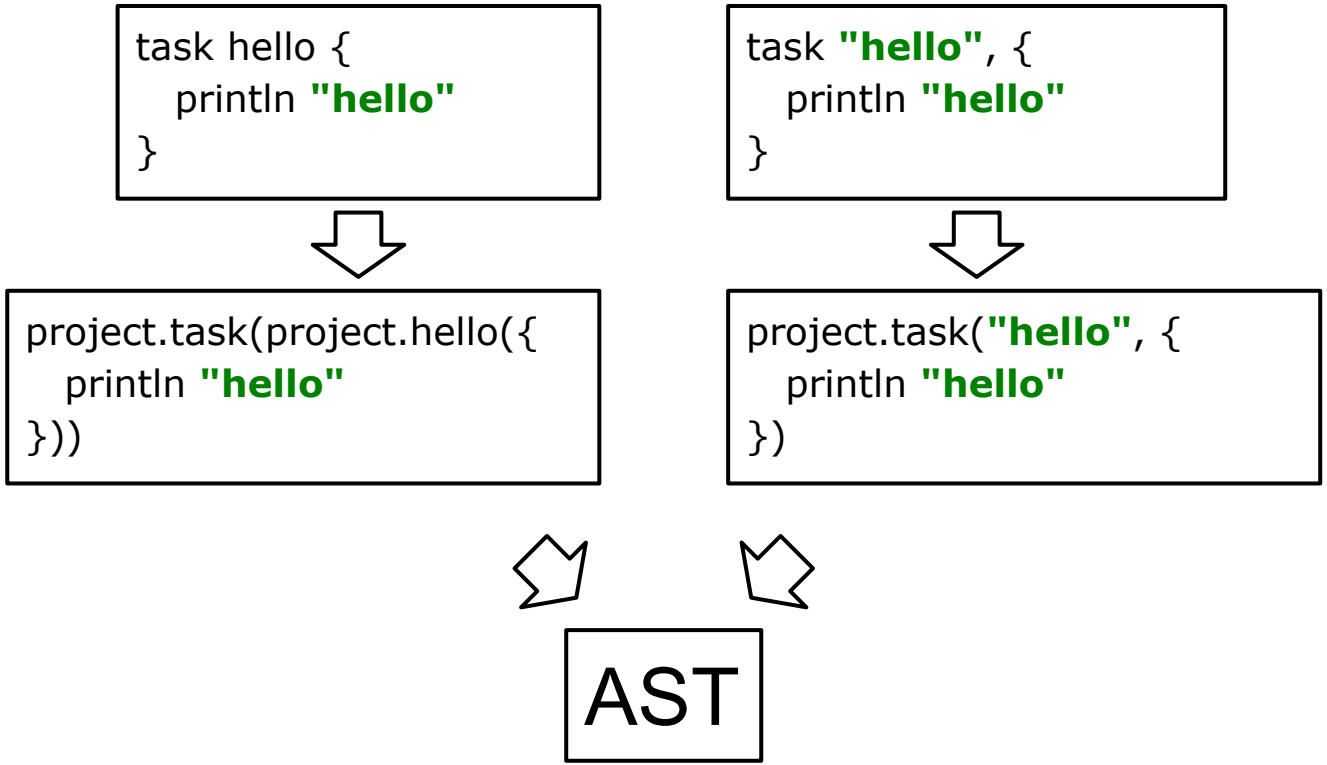
```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
              group: 'junit',
              version: '4.12'
}

task hello {
  println 'hello'
}
```

TASK




```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
                group: 'junit',
                version: '4.12'
}

task hello {
  println 'hello'
}
```

TASK




```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
                group: 'junit',
                version: '4.12'
}

task hello {
  println 'hello'
}
```

AST - Абстрактное синтаксическое дерево

AST - Абстрактное синтаксическое дерево

```
2 + 2 * 2
```

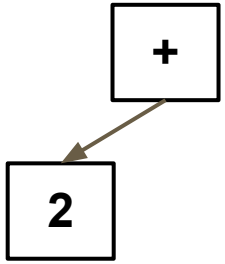
AST - Абстрактное синтаксическое дерево

2 + 2 * 2

2

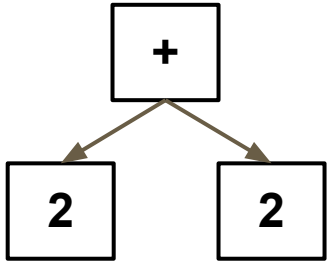
AST - Абстрактное синтаксическое дерево

2 + 2 * 2



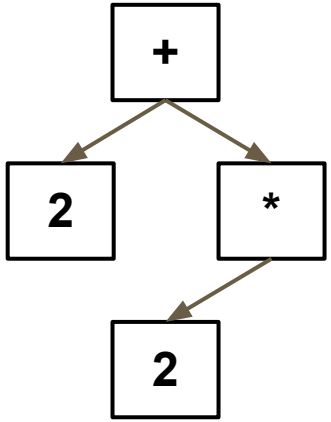
AST - Абстрактное синтаксическое дерево

2 + **2** * 2



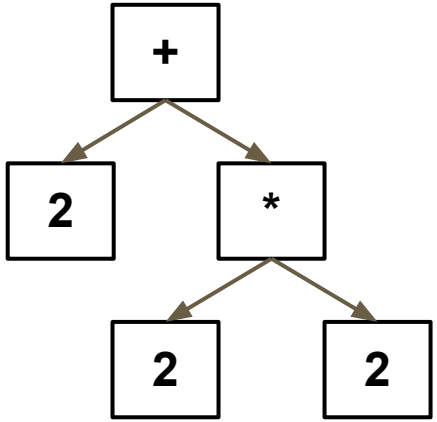
AST - Абстрактное синтаксическое дерево

2 + 2 * 2



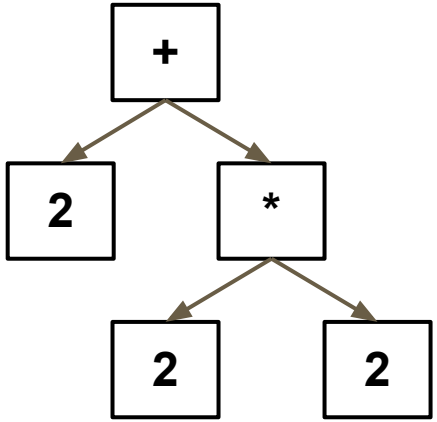
AST - Абстрактное синтаксическое дерево

2 + 2 * **2**



AST - Абстрактное синтаксическое дерево

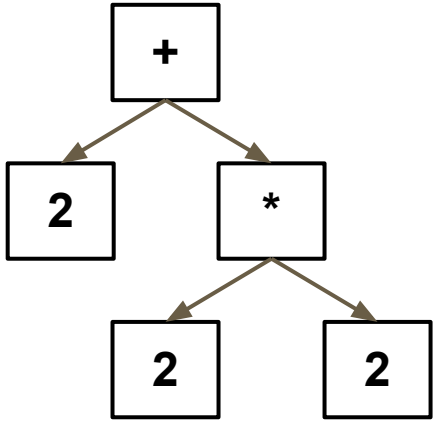
```
2 + 2 * 2
```



AST - Абстрактное синтаксическое дерево

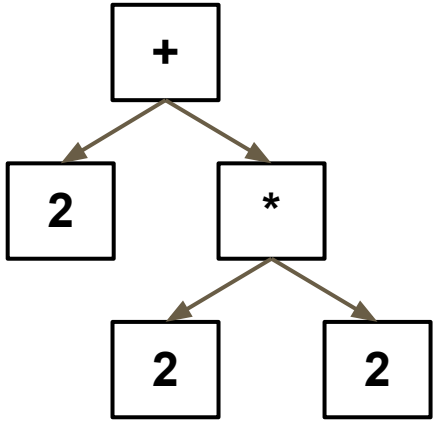
```
sum(a, 5)
```

```
2 + 2 * 2
```



AST - Абстрактное синтаксическое дерево

```
2 + 2 * 2
```

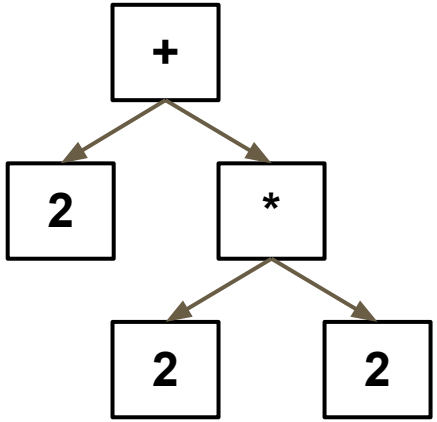



```
sum(a, 5)
```

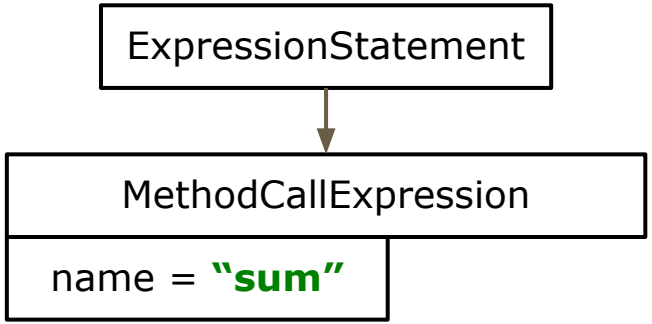
ExpressionStatement

AST - Абстрактное синтаксическое дерево

```
2 + 2 * 2
```

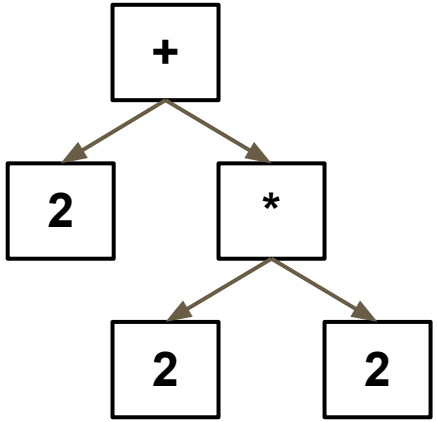



```
sum(a, 5)
```

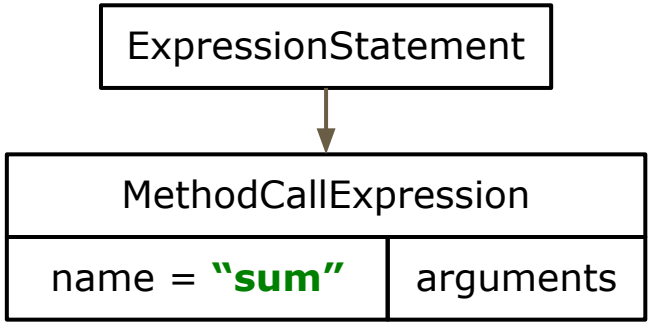


AST - Абстрактное синтаксическое дерево

```
2 + 2 * 2
```

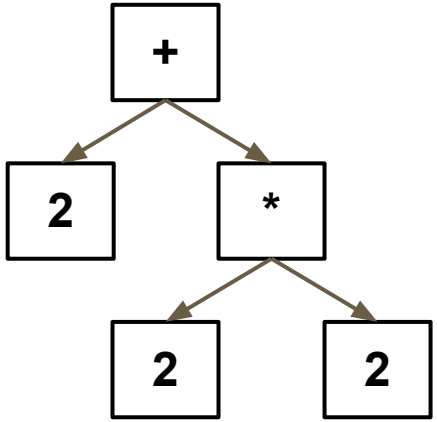



```
sum(a, 5)
```

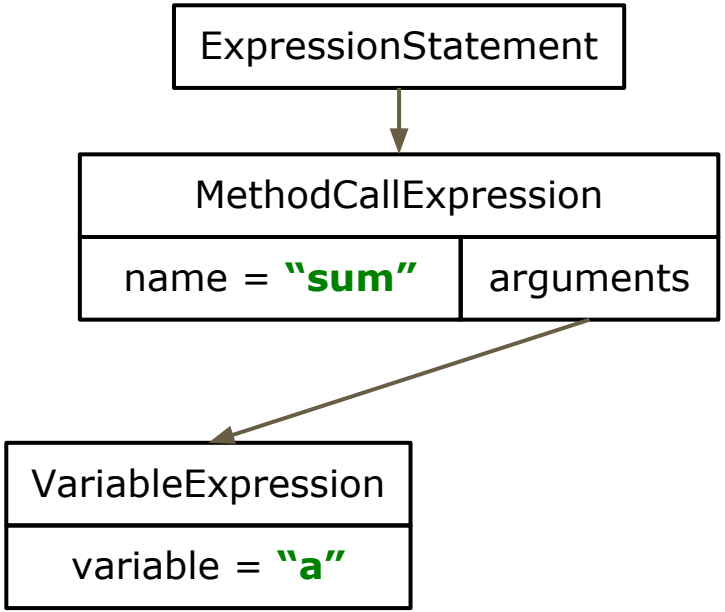


AST - Абстрактное синтаксическое дерево

```
2 + 2 * 2
```

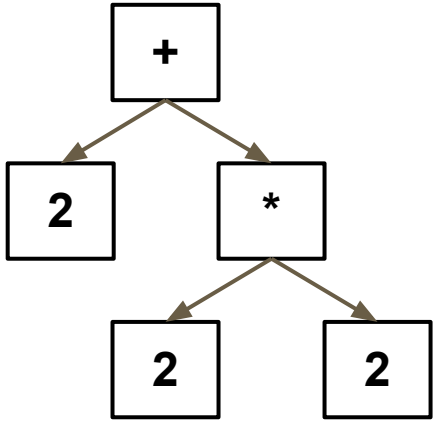



```
sum(a, 5)
```

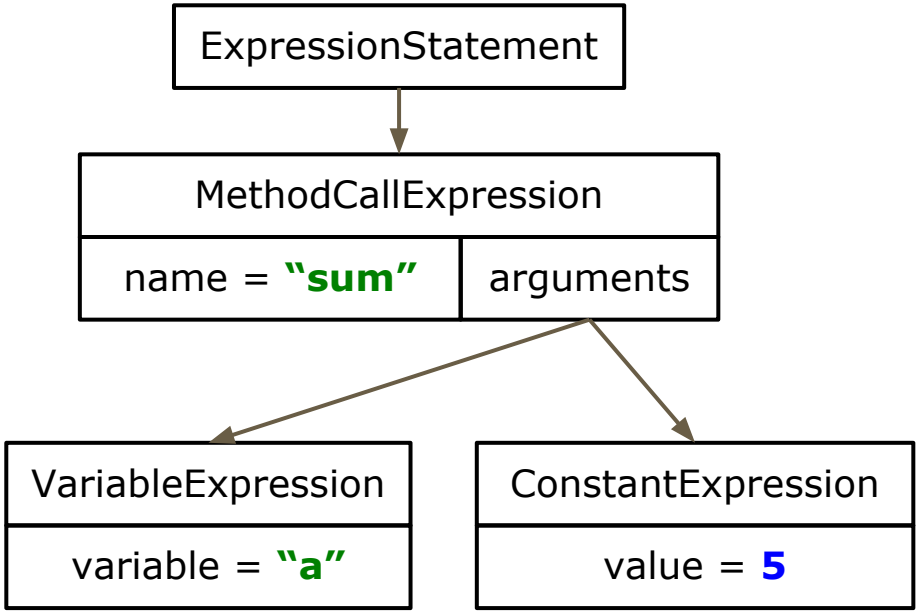


AST - Абстрактное синтаксическое дерево

```
2 + 2 * 2
```




```
sum(a, 5)
```



```
task( hello( { println "hello" } ) )
```

```
task( "hello", { println "hello" } )
```

```
task( hello( { println "hello" } ) )
```

```
task( "hello", { println "hello" } )
```

```
task( hello( { println "hello" } ) )

expression MethodCallExpression:
```

```
task( "hello", { println "hello" } )

expression MethodCallExpression:
```

```
task( hello( { println "hello" } ) )


expression MethodCallExpression:
  method: ConstantExpression("task")
```

```
task( "hello", { println "hello" } )


expression MethodCallExpression:
   method: ConstantExpression("task")
```

```
task( hello( { println "hello" } ) )


expression MethodCallExpression:
   method: ConstantExpression("task")
   arguments:
```

```
task( "hello", { println "hello" } )


expression MethodCallExpression:
   method: ConstantExpression("task")
   arguments:
```

```
task( hello( { println "hello" } ) )


expression MethodCallExpression:
   method: ConstantExpression("task")
   arguments:
      expressions:
```

```
task( "hello", { println "hello" } )


expression MethodCallExpression:
   method: ConstantExpression("task")
   arguments:
      expressions:
```

```
task( hello( { println "hello" } ) )
```

```
expression MethodCallExpression:
   method: ConstantExpression("task")
   arguments:
      expressions:
         MethodCallExpression
```

```
task( "hello", { println "hello" } )
```

```
expression MethodCallExpression:
   method: ConstantExpression("task")
   arguments:
      expressions:
```

```
task( hello( { println "hello" } ) )


expression MethodCallExpression:
  method: ConstantExpression("task")
  arguments:
    expressions:
      MethodCallExpression
        method: ConstantExpression("hello")
```

```
task( "hello", { println "hello" } )


expression MethodCallExpression:
  method: ConstantExpression("task")
  arguments:
    expressions:
```

```
task( hello( { println "hello" } ) )


expression MethodCallExpression:
  method: ConstantExpression("task")
  arguments:
    expressions:
      MethodCallExpression
        method: ConstantExpression("hello")
          arguments:
            expressions:
```

```
task( "hello", { println "hello" } )


expression MethodCallExpression:
   method: ConstantExpression("task")
   arguments:
      expressions:
```

```
task( hello( { println "hello" } ) )
```

```
expression MethodCallExpression:
  method: ConstantExpression("task")
  arguments:
    expressions:
      MethodCallExpression
        method: ConstantExpression("hello")
          arguments:
            expressions:
              ClosureExpression: {...}
```

```
task( "hello", { println "hello" } )
```

```
expression MethodCallExpression:
   method: ConstantExpression("task")
   arguments:
      expressions:
```

```
task( hello( { println "hello" } ) )
```

```
expression MethodCallExpression:
  method: ConstantExpression("task")
  arguments:
    expressions:
      MethodCallExpression
        method: ConstantExpression("hello")
          arguments:
            expressions:
              ClosureExpression: {...}
```

```
task( "hello", { println "hello" } )
```

```
expression MethodCallExpression:
  method: ConstantExpression("task")
  arguments:
    expressions:
      ConstantExpression("hello")
```

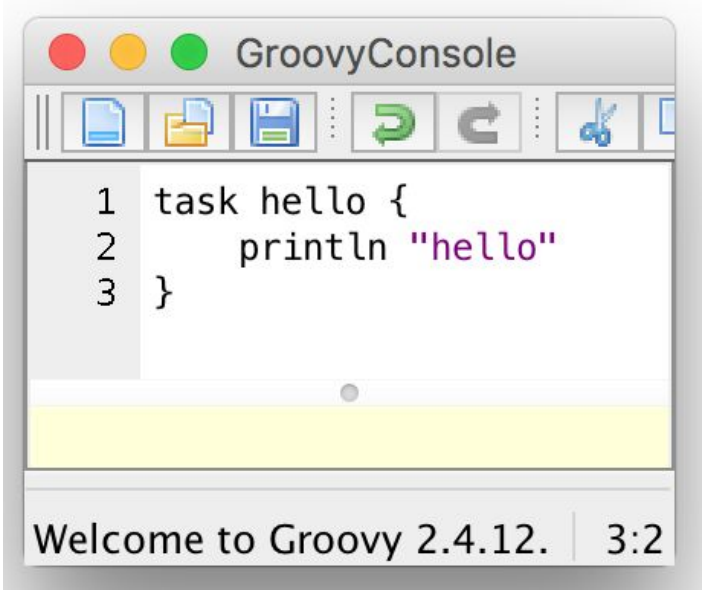
```
task( hello( { println "hello" } ) )
```

```
expression MethodCallExpression:
  method: ConstantExpression("task")
  arguments:
    expressions:
      MethodCallExpression
        method: ConstantExpression("hello")
          arguments:
            expressions:
              ClosureExpression: {...}
```

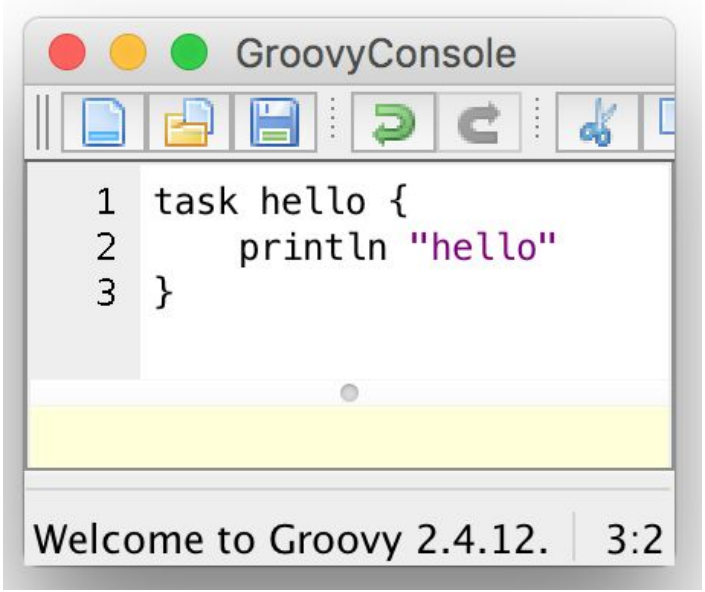
```
task( "hello", { println "hello" } )
```

```
expression MethodCallExpression:
   method: ConstantExpression("task")
   arguments:
      expressions:
         ConstantExpression("hello")
         ClosureExpression: {...}
```

groovyConsole

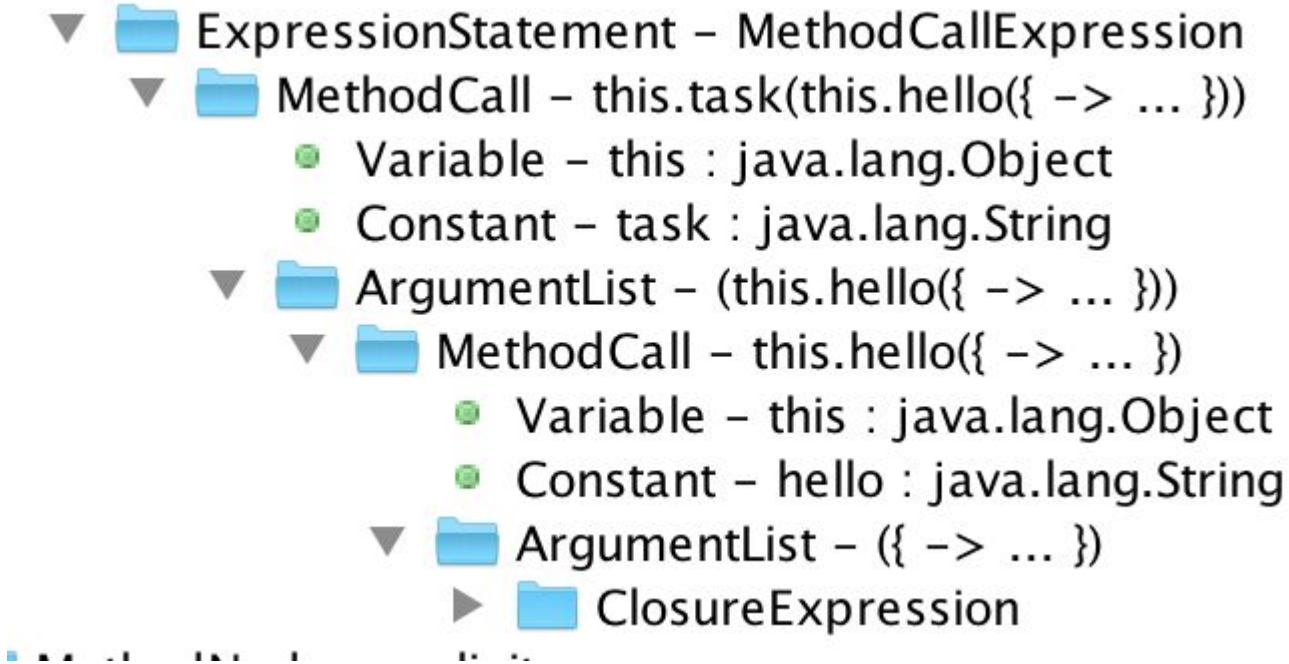


groovyConsole -> Script -> Inspect AST




```
public java.lang.Object run() {
    this.task(this.hello({
        this.println('hello')
    }))
}
```

groovyConsole -> Script -> Inspect AST



```
task( hello( { println "hello" } ) )


expression MethodCallExpression:
  method: ConstantExpression("task")
  arguments:
    expressions:
      MethodCallExpression
        method: ConstantExpression("hello")
          arguments:
            expressions:
              ClosureExpression: {...}
```

```
task( "hello", { println "hello" } )


expression MethodCallExpression:
  method: ConstantExpression("task")
  arguments:
    expressions:
      ConstantExpression("hello")
      ClosureExpression: {...}
```

```
task( hello( { println "hello" } ) )


expression MethodCallExpression:
  method: ConstantExpression("task")
  arguments:
    expressions:
      MethodCallExpression
        method: ConstantExpression("hello")
          arguments:
            expressions:
              ClosureExpression: {...}
```

```
task( "hello", { println "hello" } )


expression MethodCallExpression:
  method: ConstantExpression("task")
  arguments:
    expressions:
      ConstantExpression("hello")


      ClosureExpression: {...}
```

```
task( hello( { println "hello" } ) )


expression MethodCallExpression:
  method: ConstantExpression("task")
  arguments:
    expressions:
      MethodCallExpression
        method: ConstantExpression("hello")
          arguments:
            expressions:
              ClosureExpression: {...}
```

```
task( "hello", { println "hello" } )


expression MethodCallExpression:
  method: ConstantExpression("task")
  arguments:
    expressions:

      ConstantExpression("hello")


      ClosureExpression: {...}
```

Сделаем сначала очевидное

```
public class Project extends GroovyObjectSupport {
  private final Map<String, Closure> tasks = new HashMap<>();

  public void task(String name, Closure closure) {
     tasks.put(name, closure);
  }
}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
        'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
              group: 'junit',
              version: '4.12'
}

task hello {
  println 'hello'
}
```

Сделаем сначала очевидное

```
public class Project extends GroovyObjectSupport {
   private final Map<String, Closure> tasks = new HashMap<>();

   public void task(String name, Closure closure) {
      tasks.put(name, closure);
   }
```

```
public class GrannyInternal {
  public GrannyInternal(File buildScript, Project project) {
...
    config = new CompilerConfiguration();
    config.setScriptBaseClass(ProjectScript.class.getName());
    config.addCompilationCustomizers(new TaskDefinitionCustomizer());
    config.setDefaultScriptExtension(".granny");
...
  }
}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
              group: 'junit',
              version: '4.12'
}

task hello {
  println 'hello'
}
```

Сделаем сначала очевидное

```
public class Project extends GroovyObjectSupport {
  private final Map<String, Closure> tasks = new HashMap<>();

  public void task(String name, Closure closure) {
     tasks.put(name, closure);
  }
```

```
public class GrannyInternal {
  public GrannyInternal(File buildScript, Project project) {
...
    config = new CompilerConfiguration();
    config.setScriptBaseClass(ProjectScript.class.getName());
    config.addCompilationCustomizers(new TaskDefinitionCustomizer());
    config.setDefaultScriptExtension(".granny");
...
  }
}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
              group: 'junit',
              version: '4.12'
}

task hello {
  println 'hello'
}
```

Перехватчик компиляции

```
public class TaskDefinitionCustomizer {




}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
                group: 'junit',
                version: '4.12'
}

task hello {
  println 'hello'
}
```

Перехватчик компиляции

```
public class TaskDefinitionCustomizer extends CompilationCustomizer {




}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
                  group: 'junit',
                  version: '4.12'
}

task hello {
  println 'hello'
}
```

Перехватчик компиляции

```
public class TaskDefinitionCustomizer extends CompilationCustomizer {



   @Override
   public void call(SourceUnit source,
                    GeneratorContext context,
                    ClassNode node) {
      // ТУТ и начинаем работу
   }
}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
                  group: 'junit',
                  version: '4.12'
}

task hello {
  println 'hello'
}
```

Перехватчик компиляции

```
public class TaskDefinitionCustomizer extends CompilationCustomizer {

   public TaskDefinitionCustomizer() {
      super(CompilePhase.CONVERSION);
   }

   @Override
   public void call(SourceUnit source,
                    GeneratorContext context,
                    ClassNode node) {
      // ТУТ и начинаем работу
   }
}
```

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
   jcenter()
   maven {  url 'http://my_repo.org' }
}

dependencies {
   compile 'one.util:streamex:0.6.5',
           'org.yaml:snakeyaml:1.17'
   testCompile name: 'junit',
               group: 'junit',
               version: '4.12'
}

task hello {
   println 'hello'
}
```

Перехватчик компиляции

```
public class TaskDefinitionCustomizer extends CompilationCustomizer {

    public TaskDefinitionCustomizer() {
        super(CompilePhase.
    }

    @Override
    public void call(SourceU
                     Generat
                     ClassNo
        // ТУТ и начинаем
    }
}
```

INITIALIZATION
PARSING
CONVERSION
SEMANTIC_ANALYSIS
CANONICALIZATION
INSTRUCTION_SELECTION
CLASS_GENERATION
OUTPUT
FINALIZATION

```
apply plugin: "java"
sourceCompatibility = '1.9'

repository {
  jcenter()
  maven {  url 'http://my_repo.org' }
}

dependencies {
  compile 'one.util:streamex:0.6.5',
          'org.yaml:snakeyaml:1.17'
  testCompile name: 'junit',
                group: 'junit',
                version: '4.12'
}

task hello {
  println 'hello'
}
```

Фазы компиляции

INITIALIZATION	всё настроено, исходник открыт, но ничего не сделано
PARSING	разбор на токены
CONVERSION	AST созданное из токенов предыдущего шага
SEMANTIC_ANALYSIS	проверка согласованности и разрешение классов
CANONICALIZATION	завершение построения дерева
INSTRUCTION_SELECTION	выбор набора команд для байт-кода
CLASS_GENERATION	создание байт-кода в памяти
OUTPUT	запись байт-кода в файловую систему
FINALIZATION	освобождение ресурсов

Поиск узла AST

```
task( hello( { println "hello" } ) )
```

```
source.getAST().getUnit().getModules().stream()
   .map(ModuleNode::getStatementBlock)
   .map(BlockStatement::getStatements)
   .flatMap(Collection::stream)
   .filter(it -> it instanceof ExpressionStatement)
   .map(it -> (ExpressionStatement) it)
   .map(ExpressionStatement::getExpression)
   .filter(it -> it instanceof MethodCallExpression)
   .map(it -> (MethodCallExpression) it)
   .filter(method -> Stream.of(method)
       .map(MethodCallExpression::getMethod)
       .filter(it -> it instanceof ConstantExpression)
       .map(it -> (ConstantExpression) it)
       .map(ConstantExpression::getValue)
       .anyMatch("task"::equals))
   .forEach(this::transform);
```

```
expression MethodCallExpression:
    method: ConstantExpression("task")
    arguments:
        expressions:
            MethodCallExpression
                method: ConstantExpression("hello")
                arguments:
                    expressions:
                        ClosureExpression: {...}
```

Поиск узла AST

```
task( hello( { println "hello" } ) )
```

```
source.getAST().getUnit().getModules().stream()
   .map(ModuleNode::getStatementBlock)
   .map(BlockStatement::getStatements)
   .flatMap(Collection::stream)
   .filter(it -> it instanceof ExpressionStatement)
   .map(it -> (ExpressionStatement) it)
   .map(ExpressionStatement::getExpression)
   .filter(it -> it instanceof MethodCallExpression)
   .map(it -> (MethodCallExpression) it)
   .filter(method -> Stream.of(method)
       .map(MethodCallExpression::getMethod)
       .filter(it -> it instanceof ConstantExpression)
       .map(it -> (ConstantExpression) it)
       .map(ConstantExpression::getValue)
       .anyMatch("task"::equals))
   .forEach(this::transform);
```

```
expression MethodCallExpression:
   method: ConstantExpression("task")
   arguments:
      expressions:
         MethodCallExpression
            method: ConstantExpression("hello")
            arguments:
               expressions:
                  ClosureExpression: {...}
```

Поиск узла AST

```
task( hello( { println "hello" } ) )
```

```
source.getAST().getUnit().getModules().stream()
   .map(ModuleNode::getStatementBlock)
   .map(BlockStatement::getStatements)
   .flatMap(Collection::stream)
   .filter(it -> it instanceof ExpressionStatement)
   .map(it -> (ExpressionStatement) it)
   .map(ExpressionStatement::getExpression)
   .filter(it->it instanceof MethodCallExpression)
   .map(it->(MethodCallExpression) it)
   .filter(method -> Stream.of(method)
       .map(MethodCallExpression::getMethod)
       .filter(it -> it instanceof ConstantExpression)
       .map(it -> (ConstantExpression) it)
       .map(ConstantExpression::getValue)
       .anyMatch("task"::equals))
   .forEach(this::transform);
```

```
expression MethodCallExpression:
    method: ConstantExpression("task")
    arguments:
        expressions:
            MethodCallExpression
                method: ConstantExpression("hello")
                arguments:
                    expressions:
                        ClosureExpression: {...}
```

Поиск узла AST

```
task( hello( { println "hello" } ) )
```

```
source.getAST().getUnit().getModules().stream()
   .map(ModuleNode::getStatementBlock)
   .map(BlockStatement::getStatements)
   .flatMap(Collection::stream)
   .filter(it -> it instanceof ExpressionStatement)
   .map(it -> (ExpressionStatement) it)
   .map(ExpressionStatement::getExpression)
   .filter(it -> it instanceof MethodCallExpression)
   .map(it -> (MethodCallExpression) it)
   .filter(method -> Stream.of(method)
      .map(MethodCallExpression::getMethod)
      .filter(it -> it instanceof ConstantExpression)
      .map(it -> (ConstantExpression) it)
      .map(ConstantExpression::getValue)
      .anyMatch("task"::equals))
   .forEach(this::transform);
```

```
expression MethodCallExpression:
   method: ConstantExpression("task")
   arguments:
      expressions:
         MethodCallExpression
            method: ConstantExpression("hello")
            arguments:
               expressions:
                  ClosureExpression: {...}
```

Поиск узла AST

```
task( hello( { println "hello" } ) )
```

```
source.getAST().getUnit().getModules().stream()
   .map(ModuleNode::getStatementBlock)
   .map(BlockStatement::getStatements)
   .flatMap(Collection::stream)
   .filter(it -> it instanceof ExpressionStatement)
   .map(it -> (ExpressionStatement) it)
   .map(ExpressionStatement::getExpression)
   .filter(it -> it instanceof MethodCallExpression)
   .map(it -> (MethodCallExpression) it)
   .filter(method -> Stream.of(method)
       .map(MethodCallExpression::getMethod)
       .filter(it->it instanceof ConstantExpression)
       .map(it->(ConstantExpression) it)
       .map(ConstantExpression::getValue)
       .anyMatch("task"::equals))
   .forEach(this::transform);
```

```
expression MethodCallExpression:
    method: ConstantExpression("task")
    arguments:
        expressions:
            MethodCallExpression
                method: ConstantExpression("hello")
                arguments:
                    expressions:
                        ClosureExpression: {...}
```

Поиск узла AST

```
task( hello( { println "hello" } ) )
```

```
source.getAST().getUnit().getModules().stream()
   .map(ModuleNode::getStatementBlock)
   .map(BlockStatement::getStatements)
   .flatMap(Collection::stream)
   .filter(it -> it instanceof ExpressionStatement)
   .map(it -> (ExpressionStatement) it)
   .map(ExpressionStatement::getExpression)
   .filter(it -> it instanceof MethodCallExpression)
   .map(it -> (MethodCallExpression) it)
   .filter(method -> Stream.of(method)
       .map(MethodCallExpression::getMethod)
       .filter(it -> it instanceof ConstantExpression)
       .map(it -> (ConstantExpression) it)
       .map(ConstantExpression::getValue)
       .anyMatch("task"::equals))
   .forEach(this::transform);
```

```
expression MethodCallExpression:
    method: ConstantExpression("task")
    arguments:
        expressions:
            MethodCallExpression
                method: ConstantExpression("hello")
                arguments:
                    expressions:
                        ClosureExpression: {...}
```

Поиск узла AST

```
task( hello( { println "hello" } ) )
```

```
source.getAST().getUnit().getModules().stream()
   .map(ModuleNode::getStatementBlock)
   .map(BlockStatement::getStatements)
   .flatMap(Collection::stream)
   .filter(it -> it instanceof ExpressionStatement)
   .map(it -> (ExpressionStatement) it)
   .map(ExpressionStatement::getExpression)
   .filter(it -> it instanceof MethodCallExpression)
   .map(it -> (MethodCallExpression) it)
   .filter(method -> Stream.of(method)
       .map(MethodCallExpression::getMethod)
       .filter(it -> it instanceof ConstantExpression)
       .map(it -> (ConstantExpression) it)
       .map(ConstantExpression::getValue)
       .anyMatch("task"::equals))
   .forEach(this::transform);
```

```
expression MethodCallExpression:
   method: ConstantExpression("task")
   arguments:
      expressions:
         MethodCallExpression
            method: ConstantExpression("hello")
            arguments:
               expressions:
                  ClosureExpression: {...}
```

Изменение узла AST

```
task( hello( { println "hello" } ) )
```

```
MethodCallExpression nameExp = expression
        .getArguments()
        .getExpressions().get(0);

ConstantExpression name = nameExp.getMethod();

ClosureExpression closureExpression = nameExp
        .getArguments()
        .getExpressions().get(0);

expression.setArguments(
        new TupleExpression(name,
                            closureExpression)
);
```

```
expression MethodCallExpression:
    method: ConstantExpression("task")
    arguments:
        expressions:
            MethodCallExpression
                method: ConstantExpression("hello")
                arguments:
                    expressions:
                        ClosureExpression: {...}
```

Изменение узла AST

```
task( hello( { println "hello" } ) )
```

```
MethodCallExpression nameExp = expression
        .getArguments()
        .getExpressions().get(0);

ConstantExpression name = nameExp.getMethod();

ClosureExpression closureExpression = nameExp
        .getArguments()
        .getExpressions().get(0);

expression.setArguments(
        new TupleExpression(name,
                            closureExpression)
);
```

```
expression MethodCallExpression:
    method: ConstantExpression("task")
    arguments:
        expressions:
            MethodCallExpression
                method: ConstantExpression("hello")
                arguments:
                    expressions:
                        ClosureExpression: {...}
```

Изменение узла AST

```
task( hello( { println "hello" } ) )
```

```
MethodCallExpression nameExp = expression
        .getArguments()
        .getExpressions().get(0);

ConstantExpression name = nameExp.getMethod();

ClosureExpression closureExpression = nameExp
        .getArguments()
        .getExpressions().get(0);

expression.setArguments(
        new TupleExpression(name,
                            closureExpression)
);
```

```
expression MethodCallExpression:
    method: ConstantExpression("task")
    arguments:
        expressions:
            MethodCallExpression
                method: ConstantExpression("hello")
                arguments:
                    expressions:
                        ClosureExpression: {...}
```

Изменение узла AST

```
task( hello( { println "hello" } ) )
```

```
MethodCallExpression nameExp = expression
        .getArguments()
        .getExpressions().get(0);

ConstantExpression name = nameExp.getMethod();

ClosureExpression closureExpression = nameExp
        .getArguments()
        .getExpressions().get(0);

expression.setArguments(
        new TupleExpression(name,
                            closureExpression)
);
```

```
expression MethodCallExpression:
    method: ConstantExpression("task")
    arguments:
        expressions:
            MethodCallExpression
                method: ConstantExpression("hello")
                arguments:
                    expressions:
                        ClosureExpression: {...}
```

Изменение узла AST

```
task( hello( { println "hello" } ) )
```

```
MethodCallExpression nameExp = expression
        .getArguments()
        .getExpressions().get(0);

ConstantExpression name = nameExp.getMethod();

ClosureExpression closureExpression = nameExp
        .getArguments()
        .getExpressions().get(0);

expression.setArguments(
        new TupleExpression(name,
                            closureExpression)
);
```

```
expression MethodCallExpression:
    method: ConstantExpression("task")
    arguments:
        expressions:
            MethodCallExpression
                method: ConstantExpression("hello")
                arguments:
                    expressions:
                        ClosureExpression: {...}
```

Изменение узла AST

```
task( hello( { println "hello" } ) )
```

```
MethodCallExpression nameExp = expression
        .getArguments()
        .getExpressions().get(0);

ConstantExpression name = nameExp.getMethod();

ClosureExpression closureExpression = nameExp
        .getArguments()
        .getExpressions().get(0);

expression.setArguments(
        new TupleExpression(name,
                            closureExpression)
);
```

```
expression MethodCallExpression:
    method: ConstantExpression("task")
    arguments:
        expressions:
            MethodCallExpression
                method: ConstantExpression("hello")
                arguments:
                    expressions:
                        ClosureExpression: {...}
```

Изменение узла AST

```
task( hello( { println "hello" } ) )
```

```
MethodCallExpression nameExp = expression
        .getArguments()
        .getExpressions().get(0);

ConstantExpression name = nameExp.getMethod();

ClosureExpression closureExpression = nameExp
        .getArguments()
        .getExpressions().get(0);

expression.setArguments(
        new TupleExpression(name,
                                closureExpression)
);
```

```
expression MethodCallExpression:
    method: ConstantExpression("task")
    arguments:
        expressions:
            MethodCallExpression
                method: ConstantExpression("hello")
                arguments:
                    expressions:
                        ClosureExpression: {...}

expressions
```

Изменение узла AST

```
task( hello( { println "hello" } ) )
```

```
MethodCallExpression nameExp = expression
        .getArguments()
        .getExpressions().get(0);

ConstantExpression name = nameExp.getMethod();

ClosureExpression closureExpression = nameExp
        .getArguments()
        .getExpressions().get(0);

expression.setArguments(
        new TupleExpression(name,
                            closureExpression)
);
```

```
expression MethodCallExpression:
    method: ConstantExpression("task")
    arguments:
        expressions:
            MethodCallExpression
                method: ConstantExpression("hello")
                arguments:
                    expressions:
                        ClosureExpression: {...}

expressions
```

Изменение узла AST

```
task( hello( { println "hello" } ) )
```

```
MethodCallExpression nameExp = expression
        .getArguments()
        .getExpressions().get(0);

ConstantExpression name = nameExp.getMethod();

ClosureExpression closureExpression = nameExp
        .getArguments()
        .getExpressions().get(0);

expression.setArguments(
        new TupleExpression(name,
                            closureExpression)
);
```

```
expression MethodCallExpression:
    method: ConstantExpression("task")
    arguments:
        expressions:
            MethodCallExpression
                method:
                arguments:
                    expressions:
                        ClosureExpression: {...}

expressions
    ConstantExpression("hello")
```

Изменение узла AST

```
task( hello( { println "hello" } ) )
```

```
MethodCallExpression nameExp = expression
        .getArguments()
        .getExpressions().get(0);

ConstantExpression name = nameExp.getMethod();

ClosureExpression closureExpression = nameExp
        .getArguments()
        .getExpressions().get(0);

expression.setArguments(
        new TupleExpression(name,
                            closureExpression)
);
```

```
expression MethodCallExpression:
    method: ConstantExpression("task")
    arguments:
        expressions:
            MethodCallExpression
                method:
                arguments:
                    expressions:
                        ClosureExpression: {...}

expressions
    ConstantExpression("hello")
```

Изменение узла AST

```
task( hello( { println "hello" } ) )
```

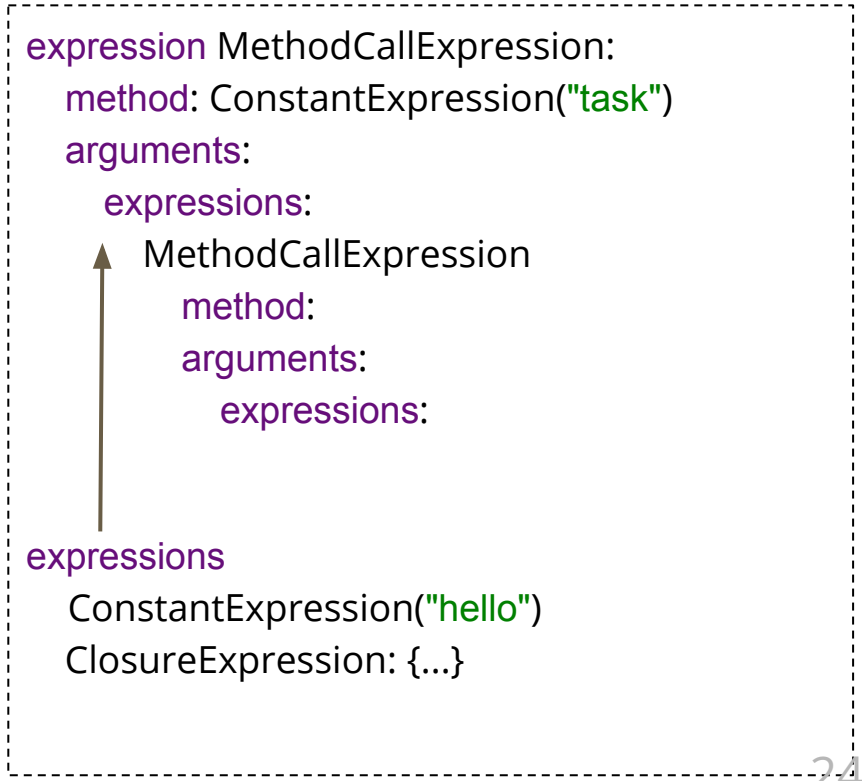
```
MethodCallExpression nameExp = expression
        .getArguments()
        .getExpressions().get(0);

ConstantExpression name = nameExp.getMethod();

ClosureExpression closureExpression = nameExp
        .getArguments()
        .getExpressions().get(0);

expression.setArguments(
        new TupleExpression(name,
                            closureExpression)
);
```

```
expression MethodCallExpression:
    method: ConstantExpression("task")
    arguments:
        expressions:
            MethodCallExpression
                method:
                arguments:
                    expressions:


expressions
    ConstantExpression("hello")
    ClosureExpression: {...}
```

Изменение узла AST

```
task( hello( { println "hello" } ) )
```

```
MethodCallExpression nameExp = expression
        .getArguments()
        .getExpressions().get(0);

ConstantExpression name = nameExp.getMethod();

ClosureExpression closureExpression = nameExp
        .getArguments()
        .getExpressions().get(0);

expression.setArguments(
        new TupleExpression(name,
                            closureExpression)
);
```



Изменение узла AST

```
task( "hello", { println "hello" } )
```

```
MethodCallExpression nameExp = expression
        .getArguments()
        .getExpressions().get(0);

ConstantExpression name = nameExp.getMethod();

ClosureExpression closureExpression = nameExp
        .getArguments()
        .getExpressions().get(0);

expression.setArguments(
        new TupleExpression(name,
                            closureExpression)
);
```

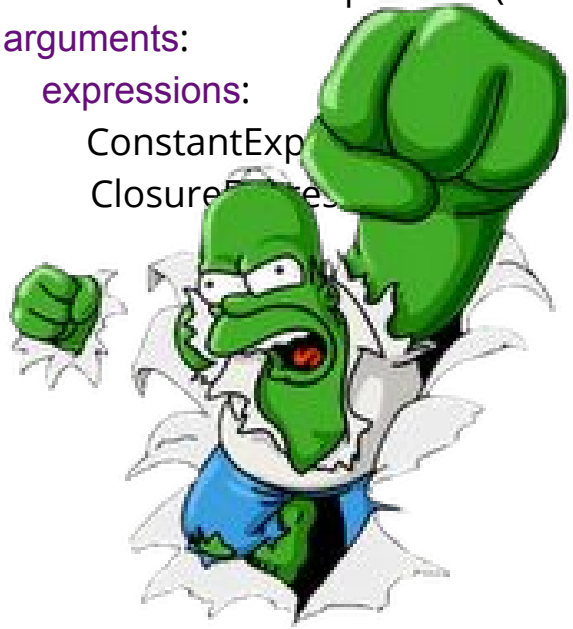
```
expression MethodCallExpression:
    method: ConstantExpression("task")
    arguments:
        expressions:
            ConstantExpression("hello")
            ClosureExpression: {...}
```

Изменение узла AST

```
task( "hello", { println "hello" } )
```

```
MethodCallExpression nameExp = expression
        .getArguments()
        .getExpressions().get(0);

ConstantExpression name = nameExp.getMethod();

ClosureExpression closureExpression = nameExp
        .getArguments()
        .getExpressions().get(0);

expression.setArguments(
        new TupleExpression(name,
                            closureExpression)
);
```

```
expression MethodCallExpression:
    method: ConstantExpression("task")
    arguments:
        expressions:
            ConstantExp
            Closure
```



Сила DSL

Сила DSL - MacroGroovy

```groovy
return new SomeCoolClass("someValue");
```

```groovy
def someVariable = new ConstantExpression("someValue");
def returnStatement = new ReturnStatement(
    new ConstructorCallExpression(
        ClassHelper.make(SomeCoolClass),
        new ArgumentListExpression(someVariable)
    )
);
```

Сила DSL - MacroGroovy

```
return new SomeCoolClass("someValue");
```

```
def someVariable = macro { "someValue" };
def returnStatement = macro { return new SomeCoolClass($v{ someVariable }) }
```

```
def someVariable = new ConstantExpression("someValue");
def returnStatement = new ReturnStatement(
    new ConstructorCallExpression(
        ClassHelper.make(SomeCoolClass),
        new ArgumentListExpression(someVariable)
    )
);
```

https://github.com/bsideup/MacroGroovy/

Сила DSL - MacroGroovy

```
return new SomeCoolClass("someValue");
```

```
def someVariable = macro { "someValue" };
def returnStatement = macro { return new SomeCoolClass($v{ someVariable }) }
```

```
def someVariable = new ConstantExpression("someValue");
def returnStatement = new ReturnStatement(
    new ConstructorCallExpression(
        ClassHelper.make(SomeCoolClass),
        new ArgumentListExpression(someVariable)
    )
);
```

https://github.com/bsideup/groovy-macro-methods

Сила DSL - Bytecode AST

```
@groovyx.ast.bytecode.Bytecode
int fib(int n) {
    l0:
    iload 1
    iconst_2
    if_icmpge l1
    iload 1
    _goto l2
    l1:
    aload 0
    iload 1
    iconst_2
    isub
    invokevirtual '.fib','(I)I'
    aload 0
    iload 1
    iconst_1
    isub
    invokevirtual '.fib', '(I)I'
    iadd
    l2:
    ireturn
}
```

https://github.com/melix/groovy-bytecode-ast

Сила DSL - Bytecode AST

```
@groovyx.ast.bytecode.Bytecode
int fib(int n) {
    l0:
    iload 1
    iconst_2
    if_icmpge l1
    iload 1
    _goto l2
    l1:
    aload 0
    iload 1
    iconst_2
    isub
    invokevirtual '.fib','(I)I'
    aload 0
    iload 1
    iconst_1
    isub
    invokevirtual '.fib', '(I)I'
    iadd
    l2:
    ireturn
}
```

b.com/melix/groovy-bytecode-ast

Сила DSL

Заключение

Заключение

Заключение

Q&A

Алексей Добрынин
lexa@trifle.one
@mad_lexa

Руслан Михалёв
mikhalev.ruslan@gmail.com
@CryonixMe

https://github.com/DNAlchemist/joker2017-gradle