Скрипты в Java-приложениях

что, как и зачем

Кто это?

Антон Архипов

Таллин, Эстония

ZeroTurnaround, JRebel

Java, Groovy

@antonarhipov

Devclub.eu

http://arhipov.blogspot.com


JRebel

Анти-Шипилёвский слайд, а-ля “дисклеймер”

Анти-Шипилёвский слайд, а-ля “дисклеймер”

Доклад про всякие няшные скрипто-сладости

Анти-Шипилёвский слайд, а-ля “дисклеймер”

Доклад про всякие няшные скрипто-сладости

Доклад простой, об очевидных вещах

Анти-Шипилёвский слайд, а-ля “дисклеймер”

Доклад про всякие няшные скрипто-сладости

Доклад простой, об очевидных вещах

Докладчик дружелюбен и легко отвлекается на разговоры

Анти-Шипилёвский слайд, а-ля “дисклеймер”

Доклад про всякие няшные скрипто-сладости

Доклад простой, об очевидных вещах

Докладчик дружелюбен и легко отвлекается на разговоры

В докладе **нет** боли, крови, кишок и расчленёнки JVM

Анти-Шипилёвский слайд, а-ля “дисклеймер”

Доклад про всякие няшные скрипто-сладости

Доклад простой, об очевидных вещах

Докладчик дружелюбен и легко отвлекается на разговоры

В докладе **нет** боли, крови, кишок и расчленёнки JVM

Да! Пушистые котики и розовые пони – это к нам!

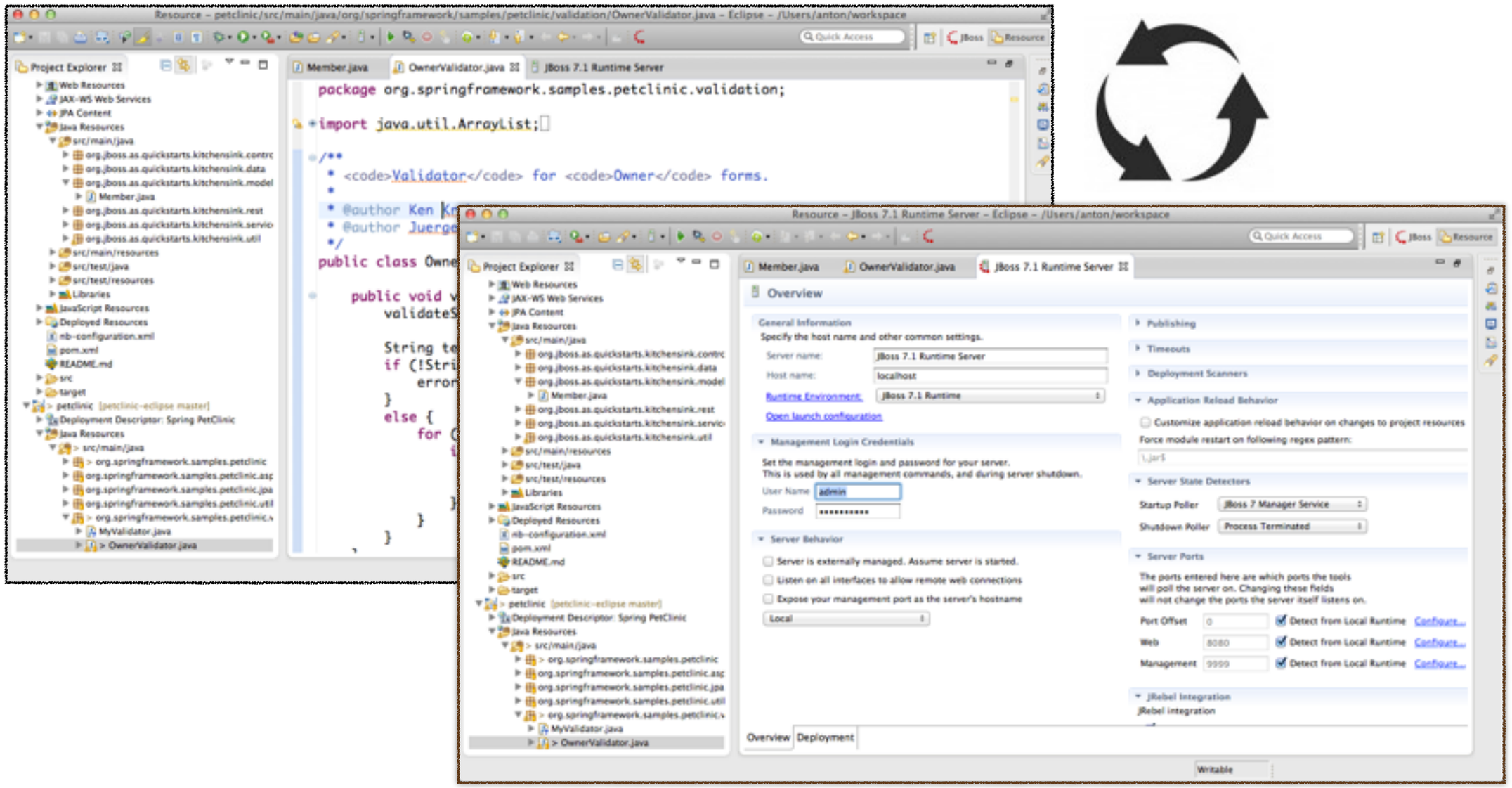
План

- Весёлые истории
- JSR 223
- Groovy
- Горячая подмена кода
- Groovy DSL и настройки
- JRuby
- и ещё

История №1

Плагины для Eclipse

История №1

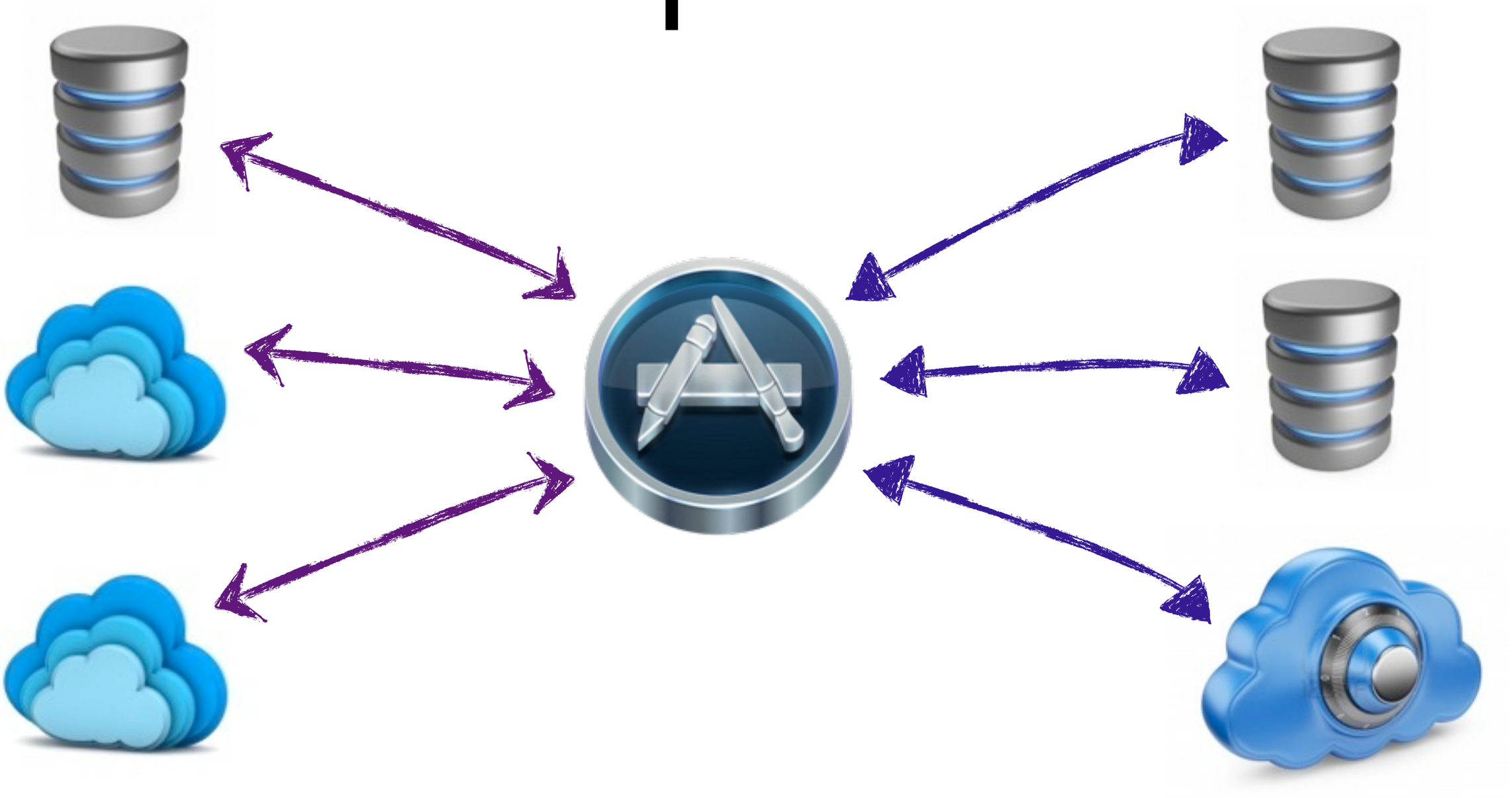


http://wiki.eclipse.org/
Add_the_ability_to_write_plugins_using_jruby_or_groovy.

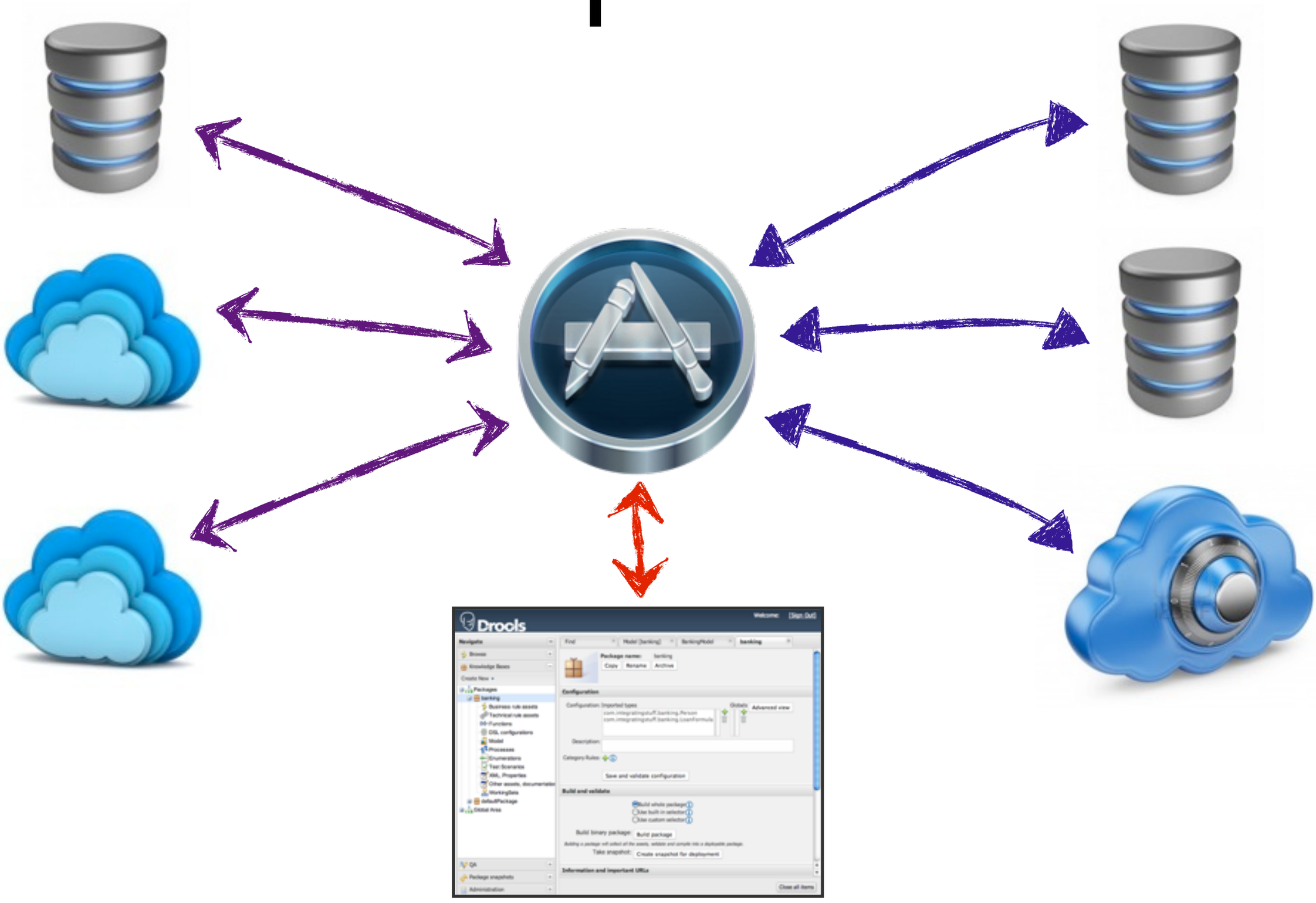
История №2

BRMS и все, все, все

История №2



История №2



Drools

Welcome: [Sign Out]

Navigate

- Browse
- Knowledge Bases
- Create New
- Packages
  - banking
    - Business rule assets
    - Technical rule assets
    - Functions
    - DSL configurations
    - Model
    - Processes
    - Enumerations
    - Test Scenarios
    - XML, Properties
    - Other assets, documentatior
    - WorkingSets
  - defaultPackage
- Global Area

QA

Package snapshots

Administration

Find | Model [banking] | BankingModel | **banking**

Package name: banking

Copy | Rename | Archive

Configuration

Configuration: Imported types

com.integratingstuff.banking.Person
com.integratingstuff.banking.LoanFormula

Globals

Advanced view

Description:

Category Rules:

Save and validate configuration

Build and validate

Build whole package
Use built-in selector
Use custom selector

Build binary package: Build package

Building a package will collect all the assets, validate and compile into a deployable package.

Take snapshot: Create snapshot for deployment

Information and important URLs

Close all items

Drools

Welcome: [Sign Out]

Navigate

- Browse
- Knowledge Bases
- Create New
- Packages
  - banking
    - Business rule assets
    - Technical rule assets
    - Functions
    - DSL configurations
    - Model
    - Processes
    - Enumerations
    - Test Scenarios
    - XML, Properties
    - Other assets, documentatior
    - WorkingSets
  - defaultPackage
- Global Area

- QA
- Package snapshots
- Administration

WHEN

There is a Person **[$p]** with:

1. birthDate less than 19-Dec-1982
   (x)= car.brand == "Ford" && salary > (2500 * 4.1)

There is an Address with:

2. street equal to Elm St.

From **$p.addresses.** Choose...

The following does not exist:

3. There is a Person with:
   salary equal to (x)= $p.salary * 2

There is a Number **[$totalAddresses]**

From Accumulate

4. There is an Address **[$a]** with:
   zipCode equal to 43240

   From **$p.addresses.** Choose...

   Custom Code | **Function**

   Function: count($a)

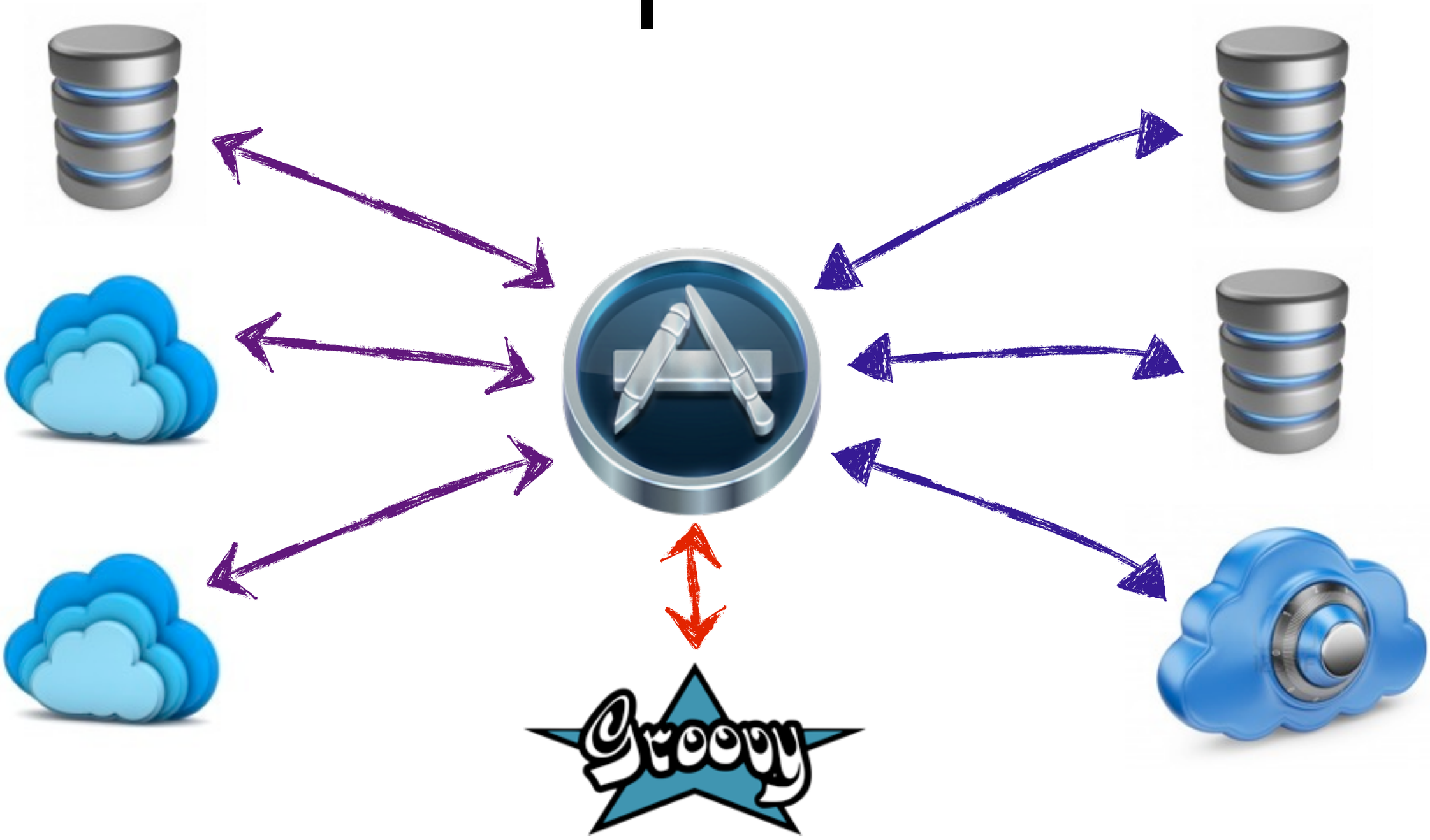
THEN

1. Insert **Person**:
   name $p.name

(show options...)

Close all items

История №2



Зачем встраивать скрипты?

Области применения

- Плагины
- Интеграция
- Автоматизация
- Горячая подмена кода

Скрипты везде!

- Jython в Oracle WebLogic и IBM WebSphere
- Groovy плагины в Jenkins
- Tcl в инструментах для EDA
- Lisp в EMACS
- Scheme и Python в GIMP
- JavaScript в 

JSR-223

- Java 6: javax.script
- ScriptEngineManager
- ScriptEngine
- Bindings
- Invocable
- Compilable

JSR-223

```
import javax.script.*;

ScriptEngineManager manager = new ScriptEngineManager();
ScriptEngine engine = manager.getEngineByName("javascript");
engine.eval("print('Hello, World!')");
```

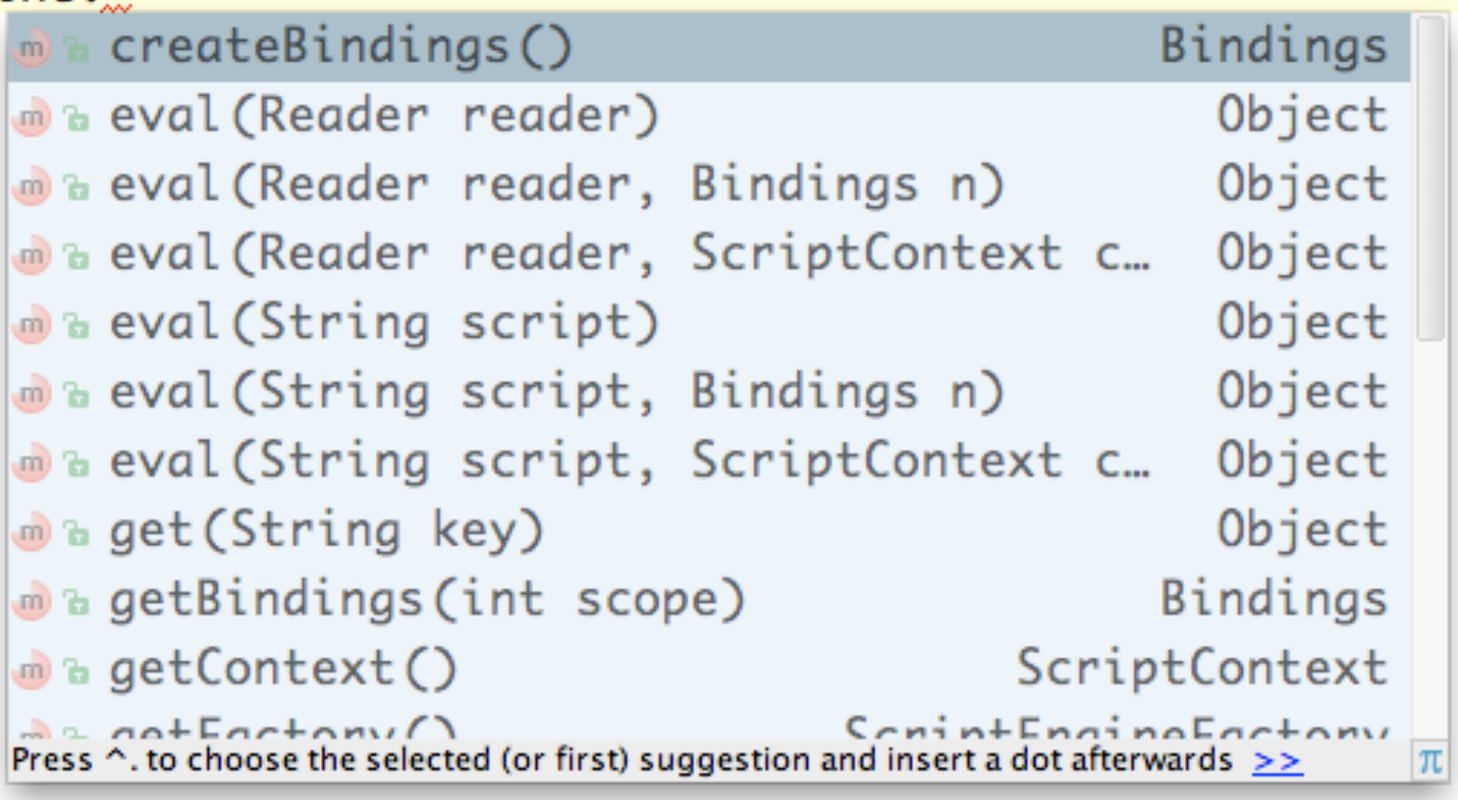
```
>> Hello, World!
```

JSR-223

```
import javax.script.*;

ScriptEngineManager manager = new ScriptEngineManager();
ScriptEngine engine = manager.getEngineByName("javascript");
engine.eval("print('Hello, World!')");
```

```
engine.
```



JSR-223

```
import javax.script.*;

ScriptEngineManager manager = new ScriptEngineManager();
ScriptEngine engine = manager.getEngineByName("javascript");
engine.eval((new java.io.FileReader("script.js"));
```

```
engine.
```

createBindings()	Bindings
eval(Reader reader)	Object
eval(Reader reader, Bindings n)	Object
eval(Reader reader, ScriptContext c…	Object
eval(String script)	Object
eval(String script, Bindings n)	Object
eval(String script, ScriptContext c…	Object
get(String key)	Object
getBindings(int scope)	Bindings
getContext()	ScriptContext

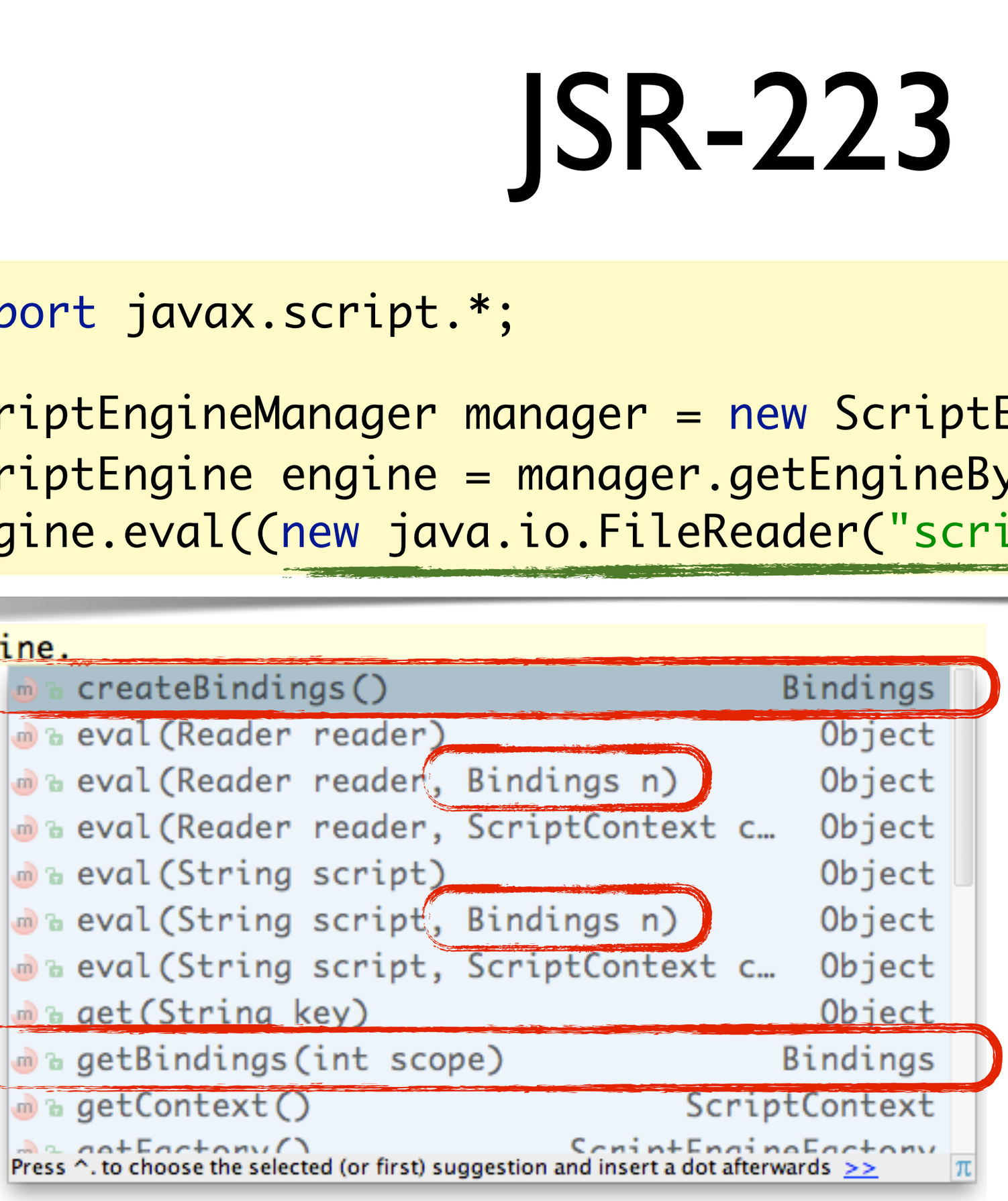
Press ^. to choose the selected (or first) suggestion and insert a dot afterwards >>

JSR-223

```
import javax.script.*;

ScriptEngineManager manager = new ScriptEngineManager();
ScriptEngine engine = manager.getEngineByName("javascript");
engine.eval((new java.io.FileReader("script.js"));
```

```
engine.
```



JSR-223

```
import javax.script.*;

ScriptEngineManager manager = new ScriptEngineManager();
ScriptEngine engine = manager.getEngineByName("javascript");

Bindings bindings = engine.createBindings();
bindings.put("a", 1L);
bindings.put("b", 10.0d);
bindings.put("c", "This is my string");
bindings.put("obj", new MyObject());
engine.setBindings(bindings, ScriptContext.ENGINE_SCOPE);

engine.eval((new java.io.FileReader("script.js")));
```

JSR-223

```
import javax.script.*;

ScriptEngineManager manager = new ScriptEngineManager();
ScriptEngine engine = manager.getEngineByName("javascript");


engine.put("a", 1L);
engine.put("b", 10.0d);
engine.put("c", "This is my string");
engine.put("obj", new MyObject());


engine.eval((new java.io.FileReader("script.js"));
```

script.js

```
println("a=" + a);
println("b=" + b);
println("c=" + c);
println("obj=" + obj);
```

```
>> a=1
>> b=10
>> c=This is my string
>> obj=java.lang.Object@12345
```

script.js

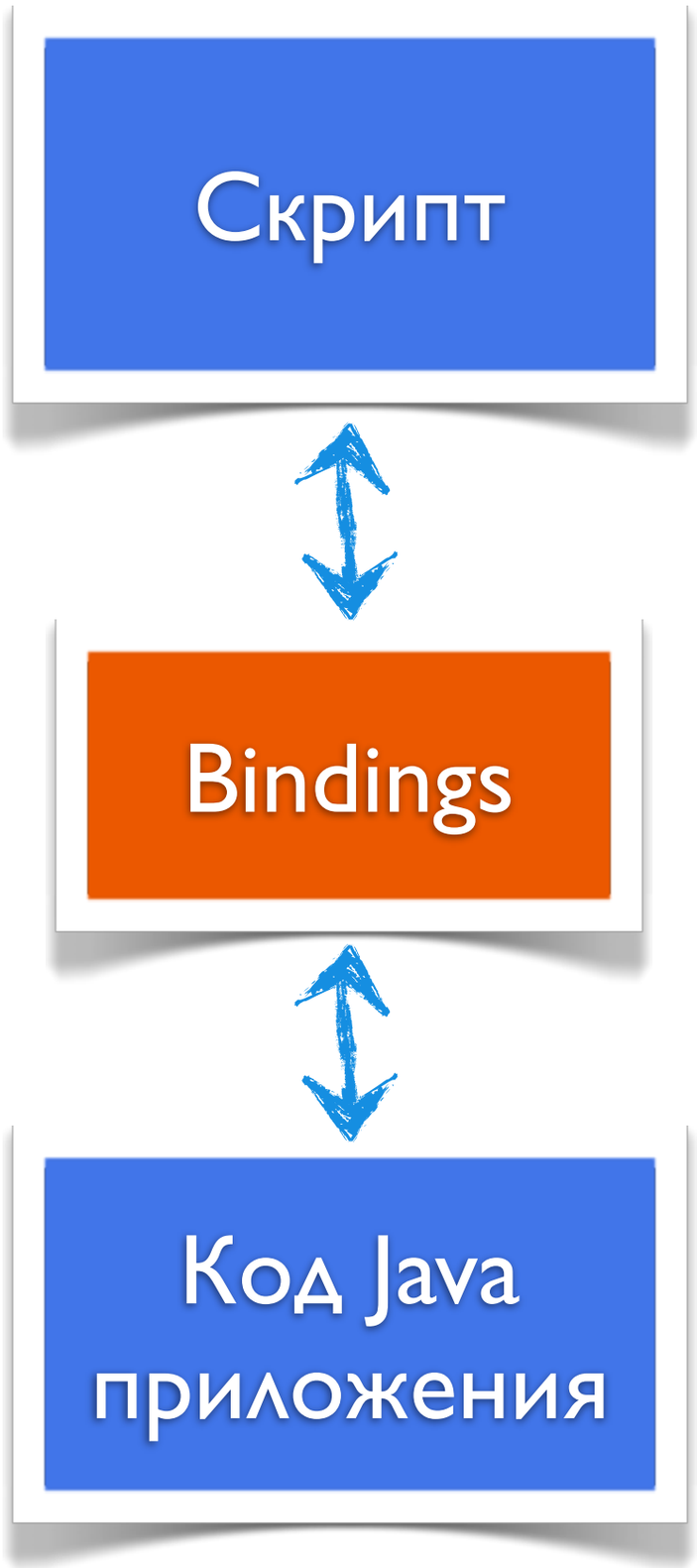
```
var x = "This variable is set by the script";
```

```
String x = (String) engine
   .getBindings(ScriptContext.ENGINE_SCOPE).get("x");
```

Скрипт

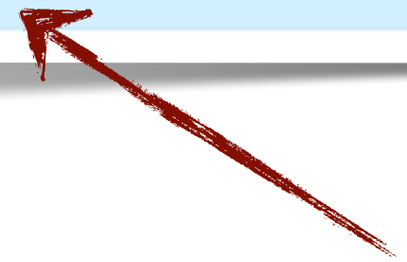
Bindings

Код Java приложения


Скрипт
Bindings
Код Java приложения

script2.js

```
//script2.js
importPackage(Packages.eu.devclub);
var o = new MySuperObject();
println(o);
o;
```



внезапно! **return**

script2.js

```
//script2.js
importPackage(Packages.eu.devclub);
var o = new MySuperObject();
println(o);
o;
```

```
//*.java
MySuperObject o = (MySuperObject)engine
                            .eval(new FileReader("script2.js"));
System.out.println("o = " + o);

>> o = eu.devclub.a.MySuperObject@38c2a17a
```

script3.js

```
//script3.js
var obj = {};

obj.a = function(){
    println("a()");
}

obj.b = function(x){
    println(x);
}
```

script3.js

```
//script3.js
var obj = {};

obj.a = function(){
    println("a()");
}

obj.b = functio
    println(x);
}
```

```
//*.java
engine.eval(new FileReader("script3.js"));
Invocable invocable = (Invocable) engine;

Object script = engine.get("obj");

invocable.invokeMethod(script, "a");
invocable.invokeMethod(script, "b", 1);
```

Наложение на интерфейс

```
String script = "function run() { " +
                    " println('I am running!'); }";

engine.eval(script);
Invocable inv = (Invocable) engine;
Runnable r = inv.getInterface(Runnable.class);
```

Компилируемые скрипты

```
ScriptEngineManager manager = new ScriptEngineManager();
ScriptEngine engine = manager.getEngineByName("javascript");

CompiledScript cs = ((Compilable) engine)
                    .compile(new FileReader("script.js"));
while(true){
    cs.eval();
}
```

А что если ...

... System.exit() !?

ClassShutter

```
new ClassShutter() {
  public boolean visibleToScripts(String className) {
    return className.startsWith("adapter"));
  }
}
```

* - специфика имплементации Rhino

```
ScriptEngineManager manager =
      new ScriptEngineManager(getClassLoader());
```

JSR-223

- Java 6: javax.script
- ScriptEngineManager
- ScriptEngine
- Bindings
- Invocable
- Compilable

JSR-223
javax.script

API
конкретного
языка

Общий API для скриптов

Имплементация
предоставляется самим
“языком”

Похоже на JSR-223,
но предоставляет более
детальные настройки

Популярные языки все
предоставляют API для
встраивания


Groovy


Groovy

Groovy: JSR-223

```
ScriptEngineManager factory = new ScriptEngineManager();
ScriptEngine engine = factory.getEngineByName("groovy");

System.out.println(engine.eval("(1..10).sum()"));
```

```
>> 55
```

http://groovy.codehaus.org/JSR+223+Scripting+with+Groovy

GroovyShell

```
Binding binding = new Binding();
binding.setVariable("foo", 2);
GroovyShell shell = new GroovyShell(binding);

Object value = shell.evaluate("println 'Hello World!'; x =
123; return foo * 10");
assert value.equals(20);
assert binding.getVariable("x").equals(123);
```

http://groovy.codehaus.org/Embedding+Groovy

GroovyClassLoader

```
ClassLoader parent = getClass().getClassLoader();
GroovyClassLoader loader = new GroovyClassLoader(parent);
Class groovyClass = loader.parseClass(new
                                File("MyClass.groovy"));

// let's call some method on an instance
GroovyObject groovyObject = (GroovyObject)
                                  groovyClass.newInstance();
Object[] args = {};
groovyObject.invokeMethod("run", args);
```

http://groovy.codehaus.org/Embedding+Groovy

GroovyShell

GroovyScriptEngine

ScriptClassLoader

GroovyClassLoader

История о маленьком “движке”

```java
public interface Engine {
    void start();
    void stop();
}
```

```groovy
def start(){
    println """Start the
                engines!"""
}
def stop(){
    println "Stop at once!"
}
this
```

http://zeroturnaround.com/rebellabs/scripting-your-java-application-with-groovy/

“Script Proxy”

```
final Object e = shell.evaluate(new File("engine.groovy"));

Engine engine =
(Engine) Proxy.newProxyInstance(getClassLoader(),
  new Class[]{ Engine.class },
  new InvocationHandler() {
  @Override
  public Object invoke(Object proxy,
                       Method method,
                       Object[] args) throws Throwable {
   Method m = e.getClass().getMethod(method.getName());
   return m.invoke(e, args);
  }
});
```

Получить объект скрита

```
final Object e = shell.evaluate(new File("engine.groovy"));

Engine engine =
(Engine) Proxy.newProxyInstance(getClassLoader(),
  new Class[]{ Engine.class },
  new InvocationHandler() {
  @Override
  public Object invoke(Object proxy,
                       Method method,
                       Object[] args) throws Throwable {
   Method m = e.getClass().getMethod(method.getName());
   return m.invoke(e, args);
  }
});
```

Привести к интерфейсу

```
final Object e = shell.evaluate(new File("engine.groovy"));

Engine engine =
(Engine) Proxy.newProxyInstance(getClassLoader(),
  new Class[]{ Engine.class },
  new InvocationHandler() {
  @Override
  public Object invoke(Object proxy,
                       Method method,
                       Object[] args) throws Throwable {
   Method m = e.getClass().getMethod(method.getName());
   return m.invoke(e, args);
  }
});
```

Делегирование вызова объекту скрипта

```
final Object e = shell.evaluate(new File("engine.groovy"));

Engine engine =
(Engine) Proxy.newProxyInstance(getClassLoader(),
  new Class[]{ Engine.class },
  new InvocationHandler() {
   @Override
   public Object invoke(Object proxy,
                        Method method,
                        Object[] args) throws Throwable {
    Method m = e.getClass().getMethod(method.getName());
    return m.invoke(e, args);
   }
});
```

com.sun.proxy.$Proxy4

```
final Object e = shell.evaluate(new File("engine.groovy"));

Engine engine =
(Engine) Proxy.newProxyInstance(getClassLoader(),
  new Class[]{ Engine.class },
  new InvocationHandler() {
  @Override
  public Object invoke(Object proxy,
                       Method method,
                       Object[] args) throws Throwable {
   Method m = e.getClass().getMethod(method.getName());
   return m.invoke(e, args);
  }
});
```

```
// Start the engines!
engine.start();
engine.stop();
```

А можно и без Proxy...

```
this as Engine
```

```
Engine engine = (Engine) shell
                .evaluate(new File("engine.groovy"));
```

```
>> class engine1_groovyProxy
```

Горячая подмена кода

```
final File file = new File("engine.groovy");

GroovyShell shell = new GroovyShell();
Object app = shell.evaluate(file);
Engine e = toEngine(app);
```

Горячая подмена кода

```
final File file = new File("engine.groovy");

GroovyShell shell = new GroovyShell();
Object app = shell.evaluate(file);
Engine e = toEngine(app);


if (timestamp < file.lastModified()) {
  timestamp = file.lastModified();
  app = shell.evaluate(file);
  e = toEngine(app);
}
```

Горячая подмена кода

```
this as Engine
```


```
java.lang.IllegalArgumentException: argument type mismatch
        ...
        at org.codehaus.groovy.runtime.DefaultGroovyMethods.asType(DefaultGroovyMethods.java:11816)
        at engine.run(engine.groovy:13)
        at groovy.lang.GroovyShell.evaluate(GroovyShell.java:518)
        at com.zt.groovy.LiveReload$1.execute(LiveReload.java:33)
```

Демо

Groovy DSL

Замыкания aka closures

“делегаты”

methodMissing()

Мета-классы

invokeMethod()

“Категории”

Настройки

CompilerConfiguration

CompilerConfiguration

```
import groovy.lang.Script;
public abstract class EngineScript
                    extends Script
                    implements Engine {}
```

CompilerConfiguration

```
import groovy.lang.Script;
public abstract class EngineScript
                    extends Script
                    implements Engine {}
```

```
CompilerConfiguration config = new CompilerConfiguration();
config.setScriptBaseClass("EngineScript");

GroovyShell shell = new GroovyShell(config);

(Engine) shell.evaluate(new File("engine.groovy"));
```

ImportCustomizer

ImportCustomizer

```
ImportCustomizer customizer = new ImportCustomizer();

customizer.addImports(
    "java.util.concurrent.atomic.AtomicInteger",
    "java.util.concurrent.atomic.AtomicBoolean");

customizer.addStarImports("java.util.concurrent");

customizer.addStaticStars("java.util.Math");
```

ImportCustomizer

```
ImportCustomizer customizer = new ImportCustomizer();

customizer.addImports(
    "java.util.concurrent.atomic.AtomicInteger",
    "java.util.concurrent.atomic.AtomicBoolean");

customizer.addStarImports("java.util.concurrent");

customizer.addStaticStars("java.util.Math");
```

```
CompilerConfiguration configuration =
                        new CompilerConfiguration();
configuration.addCompilationCustomizers(customizer);
```

Restricting The Language

```
final ImportCustomizer imports =
    new ImportCustomizer().addStaticStars("java.lang.Math");
final SecureASTCustomizer secure = new SecureASTCustomizer();

secure.setClosuresAllowed(true);
secure.setMethodDefinitionAllowed(true);

secure.setTokensWhitelist(
   Arrays.asList(Types.ASSIGN, Types.PLUS));

CompilerConfiguration configuration =
   new CompilerConfiguration();
configuration.addCompilationCustomizers(imports, secure);
```

Preventing System.exit

```
class MethodCallExpressionChecker
  implements SecureASTCustomizer.ExpressionChecker {

  public boolean isAuthorized(Expression expression) {
    if (expression instanceof MethodCallExpression) {
      if (expression.objectExpression instanceof ClassExpression) {
        if (expression.objectExpression.type.name==System.name) {
          if (expression.meth =='exit') return false
        }
      }
    }

    return true;
  }
}
```

Preventing System.exit

```
class MethodCallExpressionChecker
  implements SecureASTCustomizer.ExpressionChecker {

  public boolean isAuthorized(Expression expression) {
    if (expression instanceof MethodCallExpression) {
      if (expression.objectExpression instanceof ClassExpression) {
        if (expression.objectExpression.type.name==System.name) {
          if (expression.meth =='exit') return false
        }
      }
    }

    return true;
  }
}
```

Preventing System.exit

```
class MethodCallExpressionChecker
  implements SecureASTCustomizer.ExpressionChecker {

  public boolean isAuthorized(Expression expression) {
    if (expression instanceof MethodCallExpression) {
      if (expression.objectExpression instanceof ClassExpression) {
        if (expression.objectExpression.type.name==System.name) {
          if (expression.meth =='exit') return false
        }
      }
    }

    return true;
  }
}
```

Preventing System.exit

```
class MethodCallExpressionChecker
  implements SecureASTCustomizer.ExpressionChecker {

  public boolean isAuthorized(Expression expression) {
    if (expression instanceof MethodCallExpression) {
      if (expression.objectExpression instanceof ClassExpression) {
        if (expression.objectExpression.type.name==System.name) {
          if (expression.meth =='exit') return false
        }
      }
    }

    return true;
  }
}
```

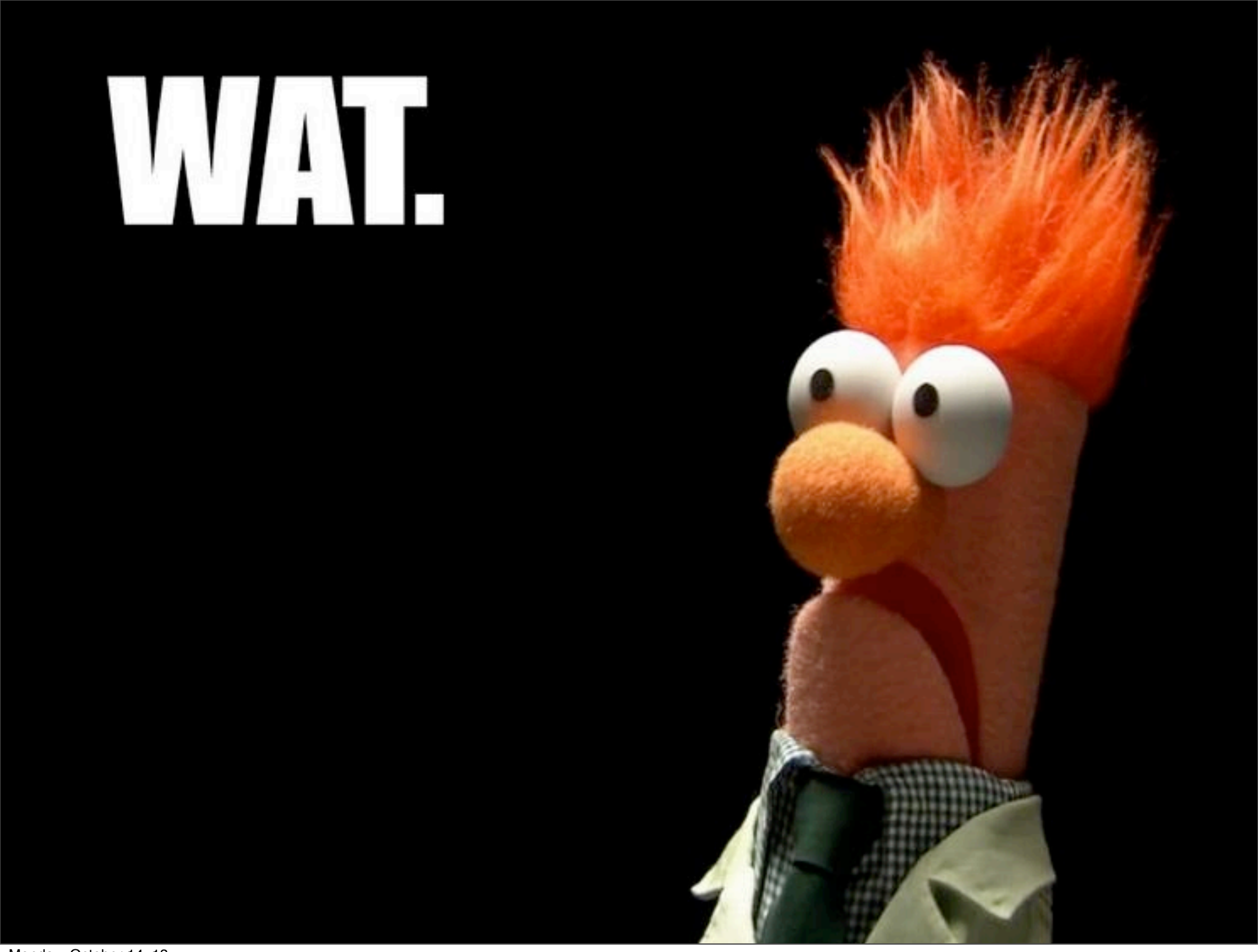
Preventing System.exit

```
class MethodCallExpressionChecker
  implements SecureASTCustomizer.ExpressionChecker {

  public boolean isAuthorized(Expression expression) {
    if (expression instanceof MethodCallExpression) {
      if (expression.objectExpression instanceof ClassExpression) {
        if (expression.objectExpression.type.name==System.name) {
          if (expression.meth =='exit') return false
        }
      }
    }

    return true;
  }
}
```

WAT.



Здесь будет ЫИнтерпрайзное

демо

JRuby

Здесь должна быть болтовня про JRuby
как его встраивать и настраивать

JavaScript,
JRuby, Groovy,
Jython ...

... и чё, как теперь
выбирать?

Всё дело в совместимости с JAVA

Всё дело в

DSL

Всё дело в

SECURITY

(да?)

Всё дело в

$$0.1+0.1+0.1$$

Clojure

```
user=> (+ 0.1 0.1 0.1)
0.30000000000000004
```

JRuby

```
1.9.3p327 :001 > 0.1 + 0.1 + 0.1
 => 0.30000000000000004
```

Scala

```
scala> 0.1 + 0.1 + 0.1
res0: Double = 0.30000000000000004
```

Groovy

```
groovy:000> 0.1 + 0.1 + 0.1
===> 0.3
```

```
groovy:000> 0.1.class
===> class java.math.BigDecimal
```

ИТОГО

- Иногда разумно использовать скрипты
- Горячая подмена кода - это здорово
- Есть всяческие способы для встраивания скриптов в свои приложения
- API JSR-223 не всегда достаточно
- Groovy рулит! :)

Вопросы,
пожелания,
комментарии

Anton Arhipov

@antonarhipov

anton@zeroturnaround.com

JRebel