springone 2GX

Chicago, October 19 - 22, 2010

Code Generation on the JVM

Hamlet D'Arcy
Canoo Engineering AG
@HamletDRC

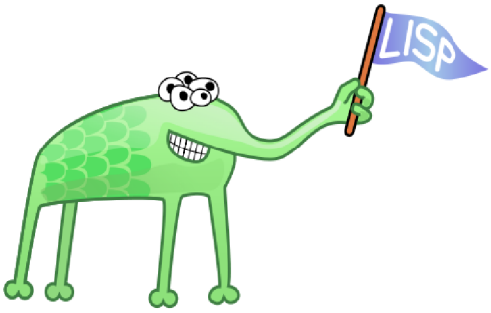
springsource
A division of vmware

Bean Generation

Stub Generation

WSDL Generation

I hate code generation too...



… but it isn't all bad.

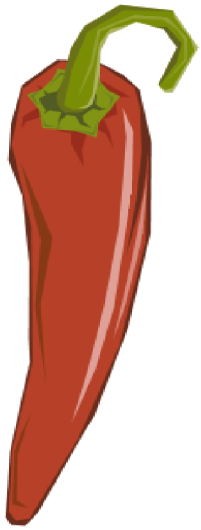
Project Lombok

```
import lombok.Getter;
import lombok.Setter;

public class Person {

  @Getter
  @Setter
  private String firstName;

  @Getter
  @Setter
  private String lastname;
}
```

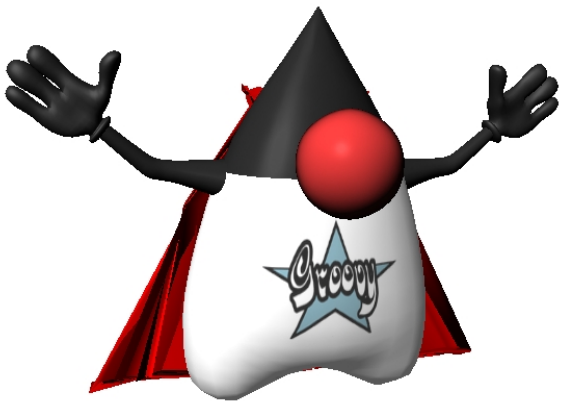
```
public class Person {
  private String firstName;
  private String lastName;

  void setFirstName(String fName) {
    this.firstName = fName;
  }

  public String getFirstName() {
    return firstName;
  }

  public void setLastName(String lName) {
    this.lastName = lName;
  }

  public String getLastName() {
    return firstName;
  }
}
```

```
import lombok.Synchronized;

public class SynchronizedExample
{

    @Synchronized
    public void doSomething() {
        return ...;
    }
}
```

```
public class SynchronizedExample {
   private final Object $lock = new
Object[0];

   public void doSomething() {
      synchronized($lock) {
         return ...;
      }
   }
}
```



- Generates Java Boilerplate
- Compile Time Only
  - For Eclipse and javac
- Removable with delombok
- Read the fine print
  - You should know what is generated


Groovy

```
class Event {
   String title
}
```

```
class Event {
   String title
}
```

```
class Event {
  String title

  public void getTitle() {
    title
  }
  public String setTitle(String t) {
    this.title = t
  }
}
```

```
class Event {
    @Delegate Date when
}
```

```
class Event {
    @Delegate Date when
}
```

```
class Event implements Comparable, Clonable {
    Date when
    boolean    after(Date when) {
        this.when.after(when)
    }
    boolean    before(Date when)  {
        this.when.before(when)
    }
    Object clone() {
        this.when.clone()
    }
    int        compareTo(Date anotherDate)  {
        this.when.compareTo(otherDate)
    }
    int        getDate()  {
        this.when.date
    }
    int        getDay()  {
        this.when.day
    }
    int        getHours()  {
        this.when.hours
    }
    int        getMinutes()  {
        this.when.minutes
    }
    int        getMonth()  {
        this.when.month
    }
    int        getSeconds()  {
        this.when.seconds
    }
    long getTime()  {
        this.when.time
    }
    int        getTimezoneOffset() {
        this.when.timezoneOffset
    }
    int        getYear()  {
        this.when.year
    }
    void setDate(int date) {
        this.when.date = date
    }
    void setHours(int hours)  {
        this.when.hours = hours
    }
    void setMinutes(int minutes)  {
        this.when.minutes = minutes
    }
    void setMonth(int month)  {
        this.when.month = month
    }
    void setSeconds(int seconds) {
        this.when.seconds = seconds
    }
    void setTime(long time) {
        this.when.time = time
    }
    void setYear(int year)  {
        this.when.year = year
    }
    String toGMTString()  {
        this.when.toGMTString()
    }
    String toLocaleString() {
        this.when.toLocaleString()
    }
}
```

```
class Event {
   @Lazy ArrayList speakers
}
```

```
class Event {
   @Lazy ArrayList speakers
}
```

```
class Event {
  ArrayList speakers

  def getSpeakers() {
    if (speakers != null) {
      return speakers
    } else {
      synchronized(this) {
        if (speakers == null) {
          speakers = []
        }
        return speakers
      }
    }
  }
}
```

- Also handles:
  - –Initial values
  - –Volatile fields

```
@Immutable
class Event {
   String title
}
```

```
@Immutable
class Event {
   String title
}
```

- Class is final
- Properties must be @Immutable or effectively immutable
- Properties are private
- Mutatators throw ReadOnlyPropertyException
- Map constructor created
- Tuple constructor created
- Equals(), hashCode() and toString() created
- Dates, Clonables, and arrays are defensively copied on way in and out (but not deeply cloned)
- Collections and Maps are wrapped in Immutable variants
- Non-immutable fields force an error
- Special handling for Date, Color, etc
- Many generated methods configurable

@Newify
@Category
@Package Scope
@Grab

… and many more as libraries

```
@Log
class Event {
   def breakForLunch() {
      log.debug('...')
   }
}
```

```
@Log
class Event {
   def breakForLunch() {
      log.debug('...')
   }
}
```

```
@Log
class Event {
   private static final transient
       Logger log = Logger.getLogger('Event')

   def breakForLunch() {
      if (log.isLoggable(Level.DEBUG) {
         log.log(Level.DEBUG, '...')
      }
   }
}
```

Coming in Groovy 1.8

@Log

@Synchronized

@IndexedProperty

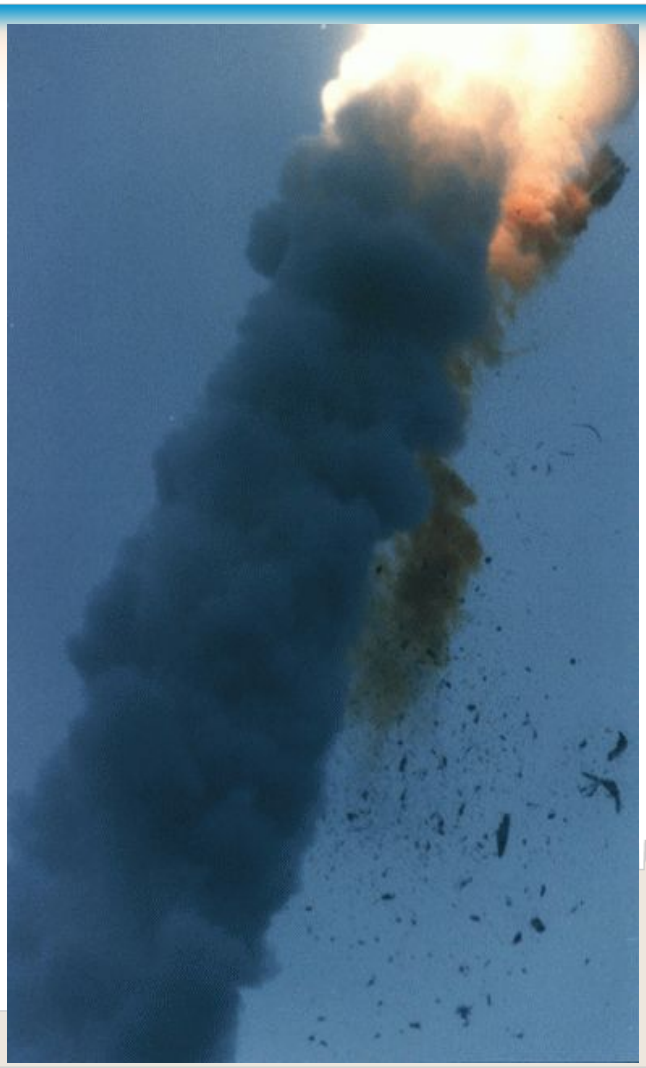
@InheritConstructors


Cité
de
l'espace
esa
esa
cnes
cnes


Cité
de
l'espace
esa
esa
cnes
cnes

Ariane 5 Destroyed
US$370 *million* in damages

… from a integer overflow error

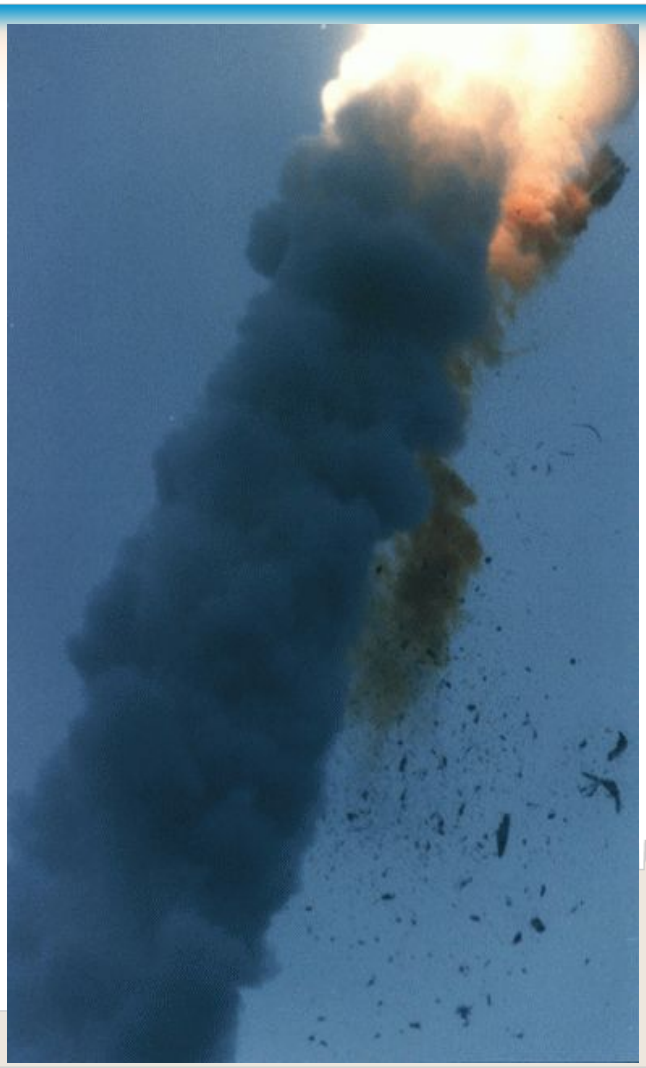


Ariane 5 Destroyed
US$370 *million* in damages

… from a integer overflow error

Greece bailout is $145 *billion*

… so we look quite good compared to the bankers.



"Design by Contract and Eiffel would have automatically avoided the crash..."
Sincerely,
The Eiffel Guys

(not a direct quote)

Demo

hooray

Design by Contract™

Component semantics part of interface

Semantics enforced by implementation

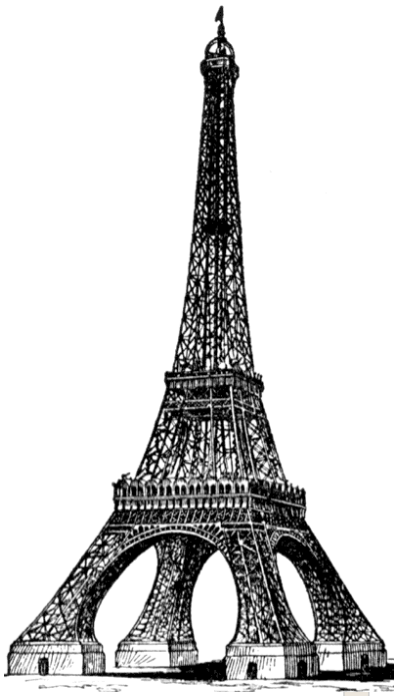
Contracts describe valid test results

Self Documenting

Sound good?

Eiffel Envy

Symptoms include:



Eiffel Envy

Symptoms include:

Unhealthy fixation on correctness
Pedantic use of unit tests
Domineering mothers
Having a 3 page interview questionnaire testing Java arcana
Dreams of being chased by goats

Eiffel Envy

Contracts must be:
- written correctly in the first place
- enforced by the compiler

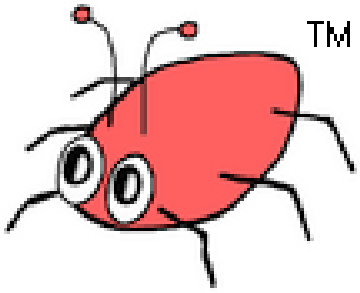
Subclasses can only:
- strengthen invariants (but not weaken)
- weaken preconditions (but not strengthen)
- and strengthen post conditions (but not weaken)

Best usage in Domain Model?

Q. Which OS Project won a 2008 Jolt Award for lamest logo?

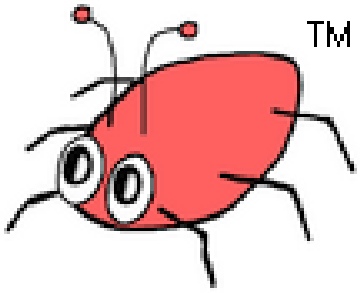
Q. Which OS Project won a 2008 Jolt Award for lamest logo?

A. FindBugs



Q. Which OS Project won a 2008 Jolt Award for lamest logo?

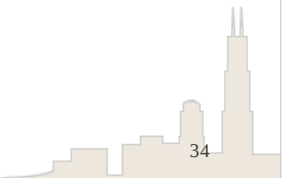
A. FindBugs



P.S. His name is “Buggy” and he is trademarked so no one steals him

FindBugs Finds Bugs

Empty synchronized block
Inconsistent synchronization
Synchronization on Boolean could lead to deadlock
Synchronization on boxed primitive could lead to deadlock
Synchronization on interned String could lead to deadlock
… and 364 more rules

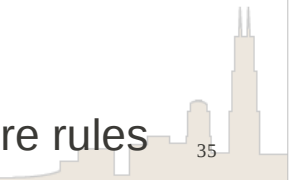


FindBugs Finds Bugs

Empty synchronized block
Inconsistent synchronization
Synchronization on Boolean could lead to deadlock
Synchronization on boxed primitive could lead to deadlock
Synchronization on interned String could lead to deadlock
… and 364 more rules

CodeNarc Finds Bugs

ThreadLocal not static final field
Volatile long or double field
Nested synchronization
Synchronized method, synchronized on this
Call to System.runFinalizersOnExit()
… and ~75 more rules



FindBugs Finds Bugs

Empty synchronized block
Inconsistent synchronization
Synchronization on Boolean could lead to deadlock
Synchronization on boxed primitive could lead to deadlock
Synchronization on interned String could lead to deadlock
… and 364 more rules

CodeNarc Finds Bugs

ThreadLocal not static final field
Volatile long or double field
Nested synchronization
Synchronized method, synchronized on this
Call to System.runFinalizersOnExit()
… and ~75 more rules

Embedded Languages

```
def s = new ArithmeticShell()

assert 2 == s.evaluate(' 1+1 ')
assert 1.0 == s.evaluate('cos(2*PI)')
```


… source in groovy/src/examples/groovyShell

Embedded Languages

```
def s = new ArithmeticShell()

assert 2 == s.evaluate(' 1+1 ')
assert 1.0 == s.evaluate('cos(2*PI)')

shouldFail(SecurityException) {
    s.evaluate('new File()')
}
```


… source in groovy/src/examples/groovyShell

Embedded Languages

Tired of hearing about DSLs?
Say “Embedded Language” instead!

ArithmeticShell ~= 300 lines of code

Alternative is
- javacc?
- custom interpreter?

Embedded Languages

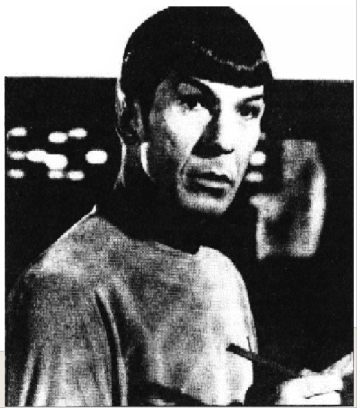
Tired of hearing about DSLs?
Say “Embedded Language” instead!

ArithmeticShell ~= 300 lines of code

Alternative is
- javacc?
- custom interpreter?
- seppuku?

Java Perversions

```
def "Does simple math work?"() {
    expect:
    def s = new ArithmeticShell()
    s.evaluate(input) == output

    where:
    input           | output
    '1 + 1'         | 2
    'cos(2*PI)'     | 1.0
}
```



Demo

again?

Groovy is a compiled language

...oh yes it is

Compiled changes visible in .class file

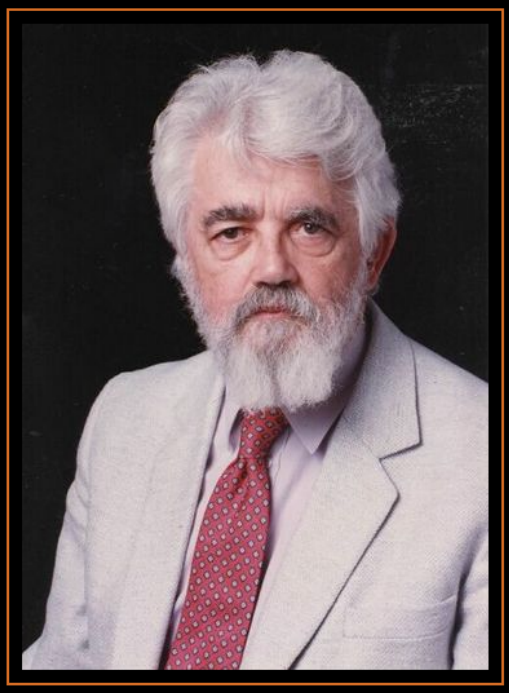
...visible to all JVM users

Language semantics are a library feature

...not hardcoded into the language

"A language should have access to its own abstract syntax"

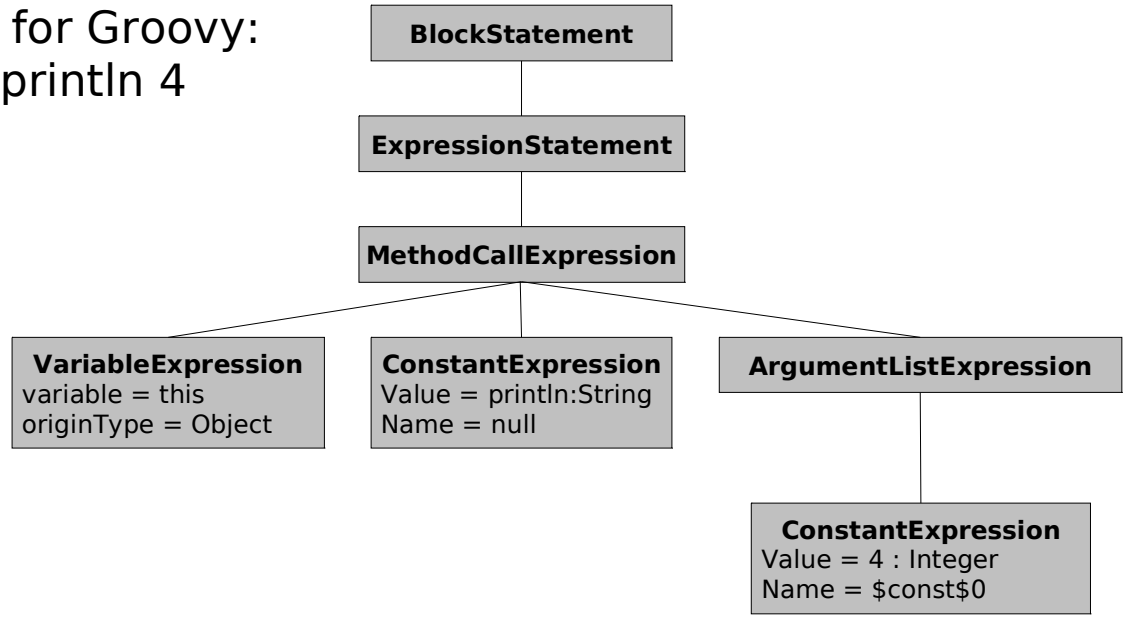
John McCarthy



PROGRAMMING

You're doing it completely wrong.

AST for Groovy: println 4



… have you seen Groovy's AST Browser?

Demo

OH YEAH

How It Works

Local AST Transformations
Global AST Transformations
AST Builder
AST Templates
ANTLR Plugins

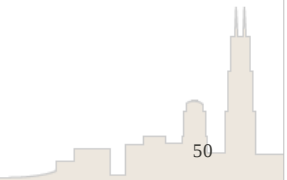

@Requires(...)
...
source.groovy

```
@Requires(...)
...
```

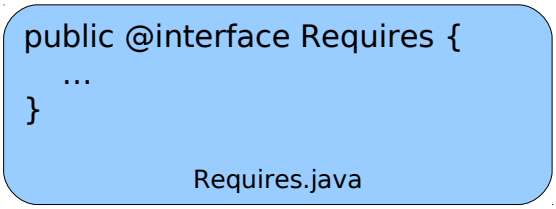
source.groovy

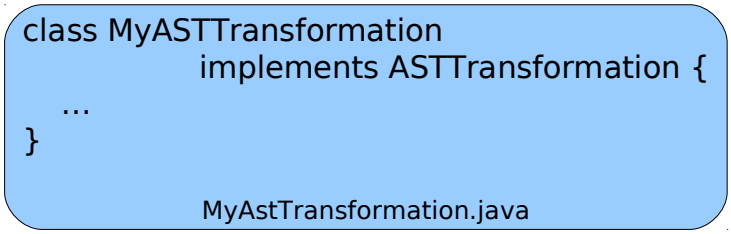
```
public @interface Requires {
   ...
}
```

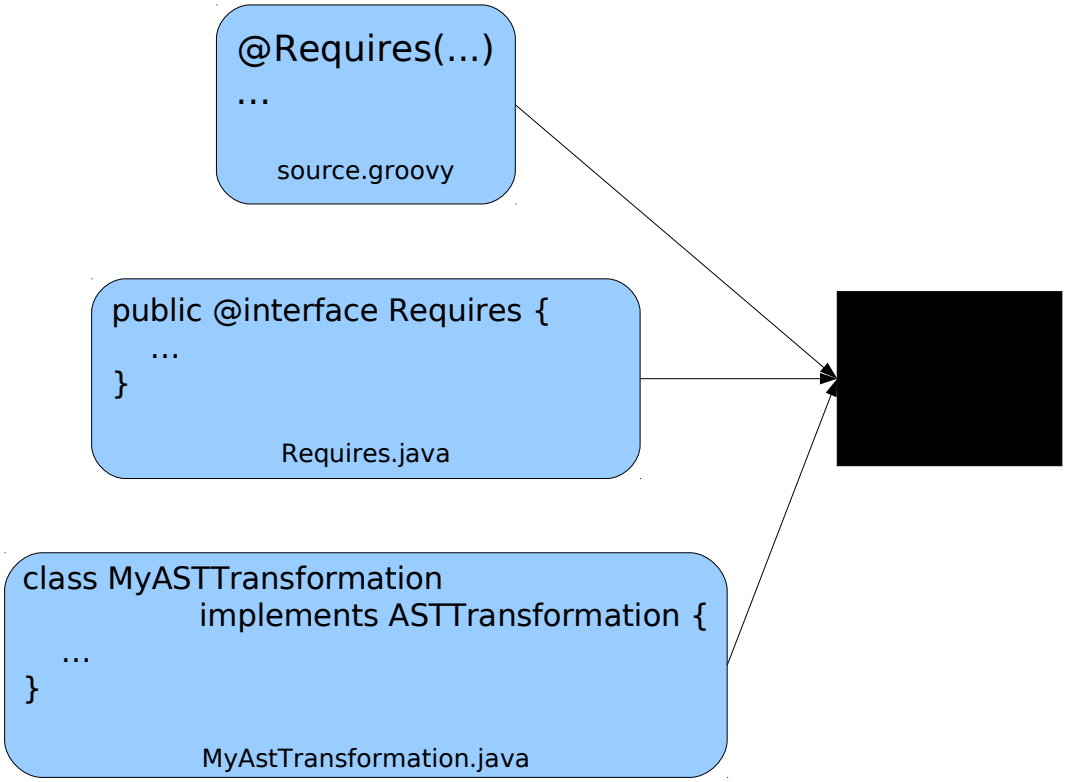
Requires.java

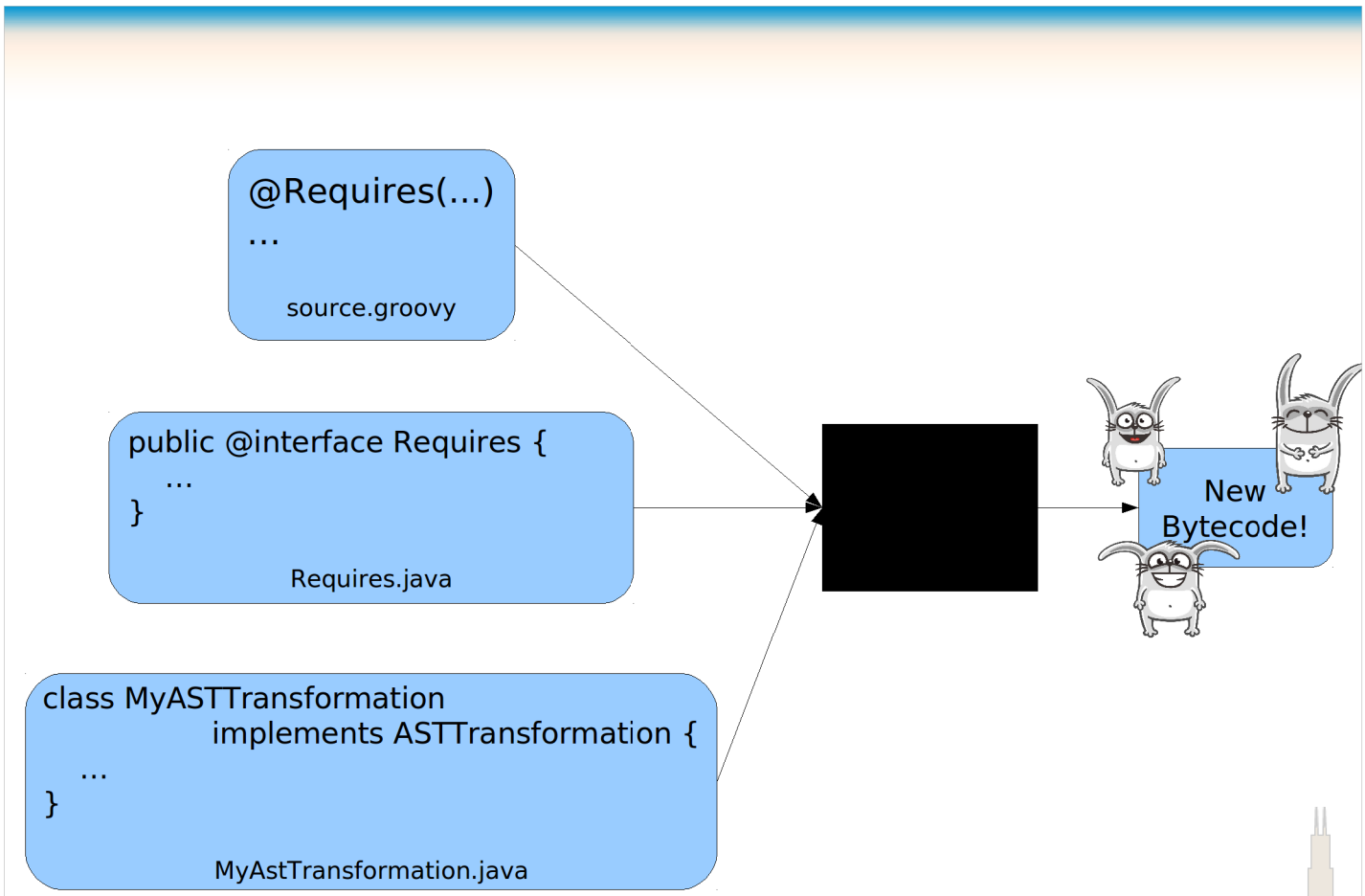



@Requires(...)
...
source.groovy


public @interface Requires {
...
}
Requires.java


class MyASTTransformation
implements ASTTransformation {
...
}
MyAstTransformation.java


@Requires(...)
...
source.groovy
public @interface Requires {
...
}
Requires.java
class MyASTTransformation
implements ASTTransformation {
...
}
MyAstTransformation.java


@Requires(...)
...
source.groovy
public @interface Requires {
...
}
Requires.java
class MyASTTransformation
implements ASTTransformation {
...
}
MyAstTransformation.java
New
Bytecode!

```
@Requires(...)
void startEngine() { ... }
```

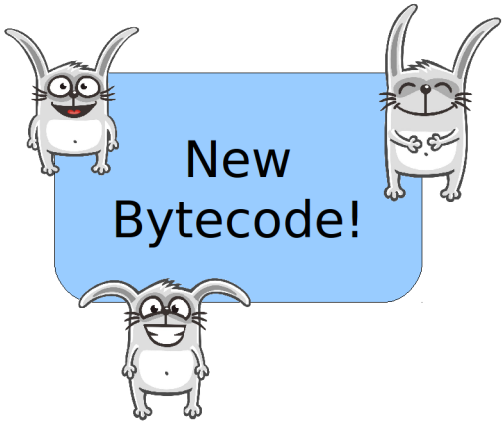
```
@Requires(...)
void startEngine() { ... }
```

```
@GroovyASTTransformationClass("org.pkg.MyTransformation")
public @interface Requires {
    ...
}
```

```
@Requires(...)
void startEngine() { ... }
```

```
@GroovyASTTransformationClass("org.pkg.MyTransformation")
public @interface Requires {
    ...
}
```

```
@GroovyASTTransformation(phase = CompilePhase.CANONICALIZATION)
public class MyTransformation implements ASTTransformation {
    public void visit(ASTNode[] nodes, SourceUnit source) {
        ...
    }
}
```


New
Bytecode!

Groovy Code Visitors

```
def s = new ArithmeticShell()
assert 2 == s.evaluate(' 1+1 ')
assert 1.0 == s.evaluate('cos(2*PI)')
```

```
public interface GroovyCodeVisitor {
    void visitBlockStatement(BlockStatement statement);
    void visitForLoop(ForStatement forLoop);
    void visitWhileLoop(WhileStatement loop);
    void visitDoWhileLoop(DoWhileStatement loop);
    ...
}
```


… source in groovy/src/examples/groovyShell

Groovy Code Visitors

```
CodeNarc Rule:
    Ban System.runFinalizersOnExit()
```

Groovy Code Visitors

CodeNarc Rule:
Ban System.runFinalizersOnExit()

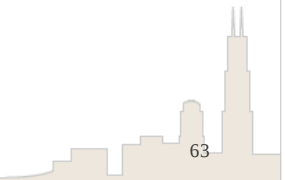
```
class SystemRunFinalizersOnExitAstVisitor extends AbstractAstVisitor  {
    def void visitMethodCallExpression(MethodCallExpression call) {
        if (call.objectExpression in VariableExpression) {
            def target = call.objectExpression.variable
            if (target == "System" && call.method in ConstantExpression) {
                if (call.method.value == "runFinalizersOnExit") {
                    addViolation(call)
                }
            }
        }
        super.visitMethodCallExpression(call);
    }
}
```


... source from codenarc

Pitfalls!

Pitfalls!

Testing AST Transformations
Writing AST
Rigid Groovy/Java syntax
Splicing source into AST
Finding insertion points
Splicing AST into source
Variable capture



TranformTestHelper and IDE Support

```
def file = new File('./MyExample.groovy')

def transform = new MainTransformation()
def phase = CompilePhase.CANONICALIZATION

def invoker = new
TranformTestHelper(transform, phase)

def clazz = invoker.parse(file)
def instance = clazz.newInstance()
```

… source in groovy/src/examples/astbuilder

Pitfalls!

~~Testing AST Transformations~~
Writing AST
Rigid Groovy/Java syntax
Splicing source into AST
Finding insertion points
Splicing AST into source
Variable capture

Writing AST

```
def ast = new ExpressionStatement(
        new MethodCallExpression(
            new VariableExpression("this"),
            new ConstantExpression("println"),
            new ArgumentListExpression(
                new ConstantExpression("Hello World")
            )
        )
    )
```

Writing AST

```
def ast = new ExpressionStatement(
        new MethodCallExpression(
            new VariableExpression("this"),
            new ConstantExpression("println"),
            new ArgumentListExpression(
                new ConstantExpression("Hello World")
            )
        )
    )
```

```
def ast = new AstBuilder().buildFromCode {
    println "Hello World"
}
```

Writing AST

```
def ast = new AstBuilder().buildFromString(
    ' println "Hello World" '
)
```

```
def ast = new AstBuilder().buildFromSpec {
    methodCall {
        variable('this')
        constant('println')
        argumentList {
            constant 'Hello World'
        }
    }
}
```


… from “Building AST Guide” on Groovy Wiki

Pitfalls!

~~Testing AST Transformations~~
~~Writing AST~~
Rigid Groovy/Java syntax
Splicing source into AST
Finding insertion points
Splicing AST into source
Variable capture

Rigid Syntax

```
given "some data", {
    ...
}
when "a method is called", {
    ...
}
then "some condition should exist", {
    ...
}
```

Rigid Syntax

```
given "some data" {
    ...
}
when "a method is called" {
    ...
}
then "some condition should exist" {
    ...
}
```


```
String addCommas(text) {
    def pattern = ~/(.*)(given|when|then) "([^"\\]*(\\.[^"\\]*)*)" \{(.*)/
    def replacement = /$1$2 "$3", {$4/
    (text =~ pattern).replaceAll(replacement)
}
```


… from “Groovy ANTLR Plugins for Better DSLs”

Pitfalls!

~~Testing AST Transformations~~
~~Writing AST~~
~~Rigid Groovy/Java syntax~~
Splicing source into AST
Finding insertion points
Splicing AST into source
Variable capture

Combining AST with AST Builder

```
def wrapWithLogging(MethodNode original) {
    new AstBuilder().buildFromCode {
        println "starting $original.name"
        $original.code
        println "ending $original.name"
    }
}
```

Combining AST with AST Builder

```
def wrapWithLogging(MethodNode original) {
    new AstBuilder().buildFromCode {
        println "starting $original.name"
        $original.code
        println "ending $original.name"
    }
}
```

Too bad this is not valid code.

Combining AST with AST Builder

```
[meta]
def wrapWithLogging(original as Expression):
    return [|
        println "starting " + $original.name
        $(original.ToCodeString())
        println "ending " + $original.name
    |]
```

Combining AST with AST Builder

```
def wrapWithLogging(MethodNode original) {
    new AstBuilder().buildFromCode {
        println "starting $original.name"
        $original.code
        println "ending $original.name"
    }
}
```

… from GEP-4 AST Templates

Pitfalls!

~~Testing AST Transformations~~
~~Writing AST~~
~~Rigid Groovy/Java syntax~~
~~Splicing source into AST~~ GEP-4 in 1.8?
Finding insertion points
Splicing AST into source
Variable capture

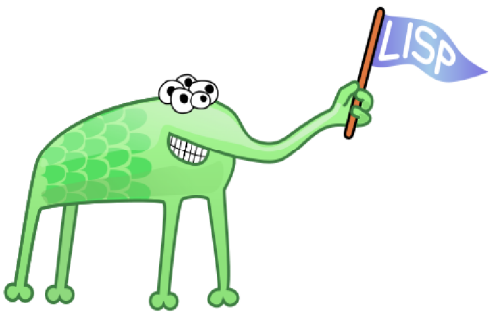
Finding Insertion Points
&
Splicing AST into Source

```
public interface ASTTransformation {
    public void visit(ASTNode[] nodes,
                      SourceUnit source);
}
```

Finding Insertion Points
&
Splicing AST into Source

```
def x = "some value"
setNull x
assert x == null
```

Warning... Lisp Ahead



Finding Insertion Points
&
Splicing AST into Source

```
def x = "some value"
setNull x
assert x == null
```

```
(defmacro setNull (var)
    (list 'setq var nil))
```

Variable Capture

```
def ast = new AstBuilder().buildFromCode {
    String syntheticField = ...
}
```

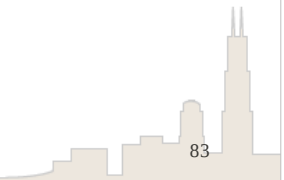
Groovy Code Generation

Good Things

- When nothing else will do
- To call functions without evaluating arguments
- To modify variables in calling scope

Bad Things

- Difficult to write
- Difficult to write *correctly*
- Source code clarity
- Runtime clarity
- Version compatibility



Groovy Code Generation

Good Things

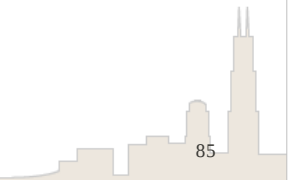
- When nothing else will do
- To call functions without evaluating arguments
- To modify variables in calling scope

Bad Things

- Difficult to write
- Difficult to write *correctly*
- Source code clarity
- Runtime clarity
- Version compatibility

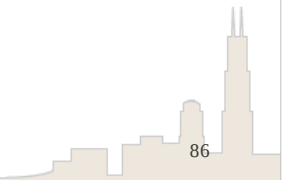
… from “On Lisp” by Paul Graham

Q. You are marooned on a deserted island with only one programming language. Which do you want?



Q. You are marooned on a deserted island with only one programming language. Which do you want?

A. One with AST Transformations

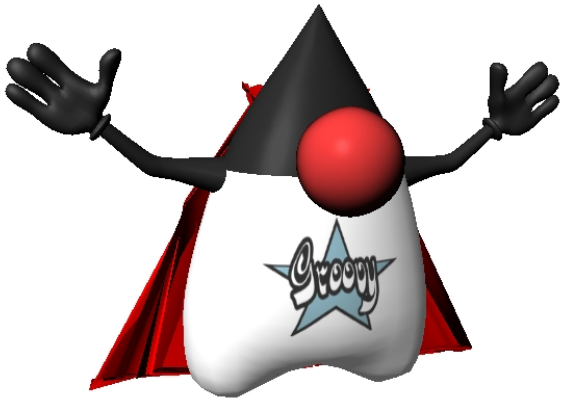


...and a freakin' sweet logo

Thanks!

- What to do next:
  - Groovy Wiki and Mailing List is amazingly helpful
  - Use your creativity and *patience*
  - http://hamletdarcy.blogspot.com & @HamletDRC

canoo
› your provider for business web solutions ›



Griffon, Grails, Groovy, and Agile Consulting
info@canoo.com or hamlet.darcy@canoo.com