Code Generation in Groovy

Hamlet D'Arcy
@HamletDRC

canoo

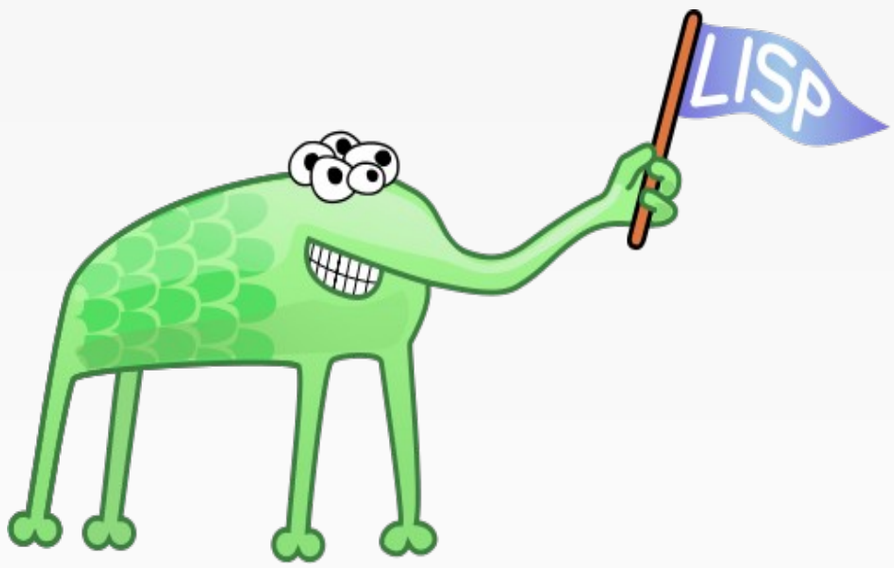
Bean Generation

Stub Generation

```
- <binding name="TestSoapBinding" type="tns:TestSoapPortType">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" />
  - <operation name="Multiply">
      <soap:operation style="rpc" soapAction="http://soapinterop.org/Multiply" />
    - <input>
        <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
          namespace="http://soapinterop.org" />
      </input>
    - <output>
        <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
          namespace="http://soapinterop.org" />
      </output>
    </operation>
  - <operation name="Add">
      <soap:operation style="document" soapAction="http://soapinterop.org/Add" />
    - <input>
        <soap:body use="literal" />
      </input>
    - <output>
        <soap:body use="literal" />
      </output>
    </operation>
  </binding>
- <service name="MSInterop1DocAndRPCService">
  - <port name="TestSoap" binding="tns:TestSoapBinding">
      <soap:address location="http://localhost/services/msInteropDocAndRpc" />
    </port>
  </service>
</definitions>
```

WSDL Generation

I hate code generation too...



… but it isn't all bad.

```
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
}
```

canoo

```
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
}
```

canoo

```
@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('...')
    }
}
```

canoo

```
@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, '...')
        }
    }
}
```

canoo

```
@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, '...')
        }
    }
}
```


Cité
de
l'espace
arianespace
esa
esa
cnes
cnes

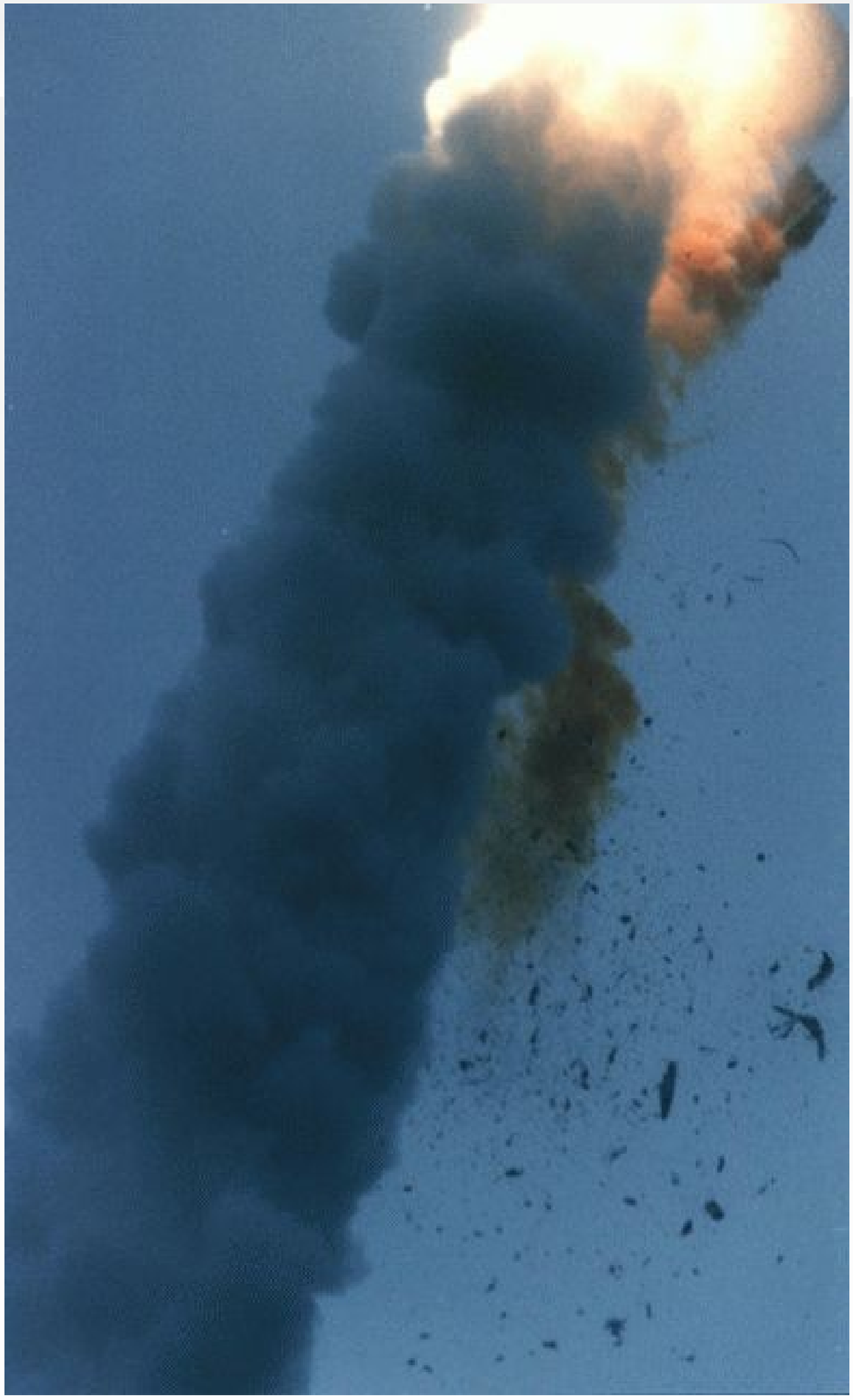

Cité
de
l'espace
arianespace
esa
esa
cnes
cnes

canoo

Ariane 5 Destroyed

US$370 *million* in damages

… from a integer overflow error



canoo

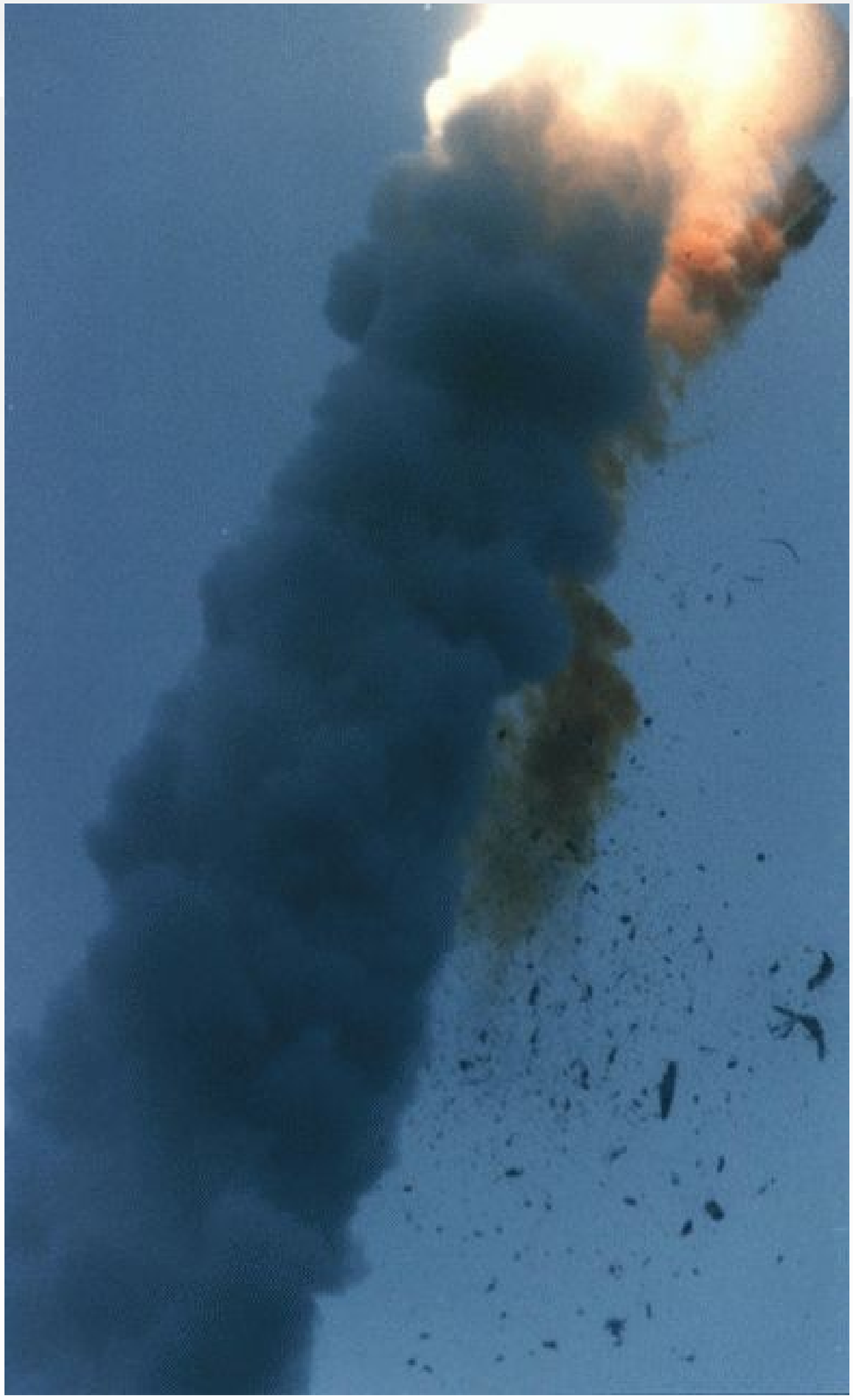
Ariane 5 Destroyed

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)

canoo

DEMO

hooray

Design by Contract™

Component semantics part of interface

Semantics enforced by implementation

Contracts describe valid test results

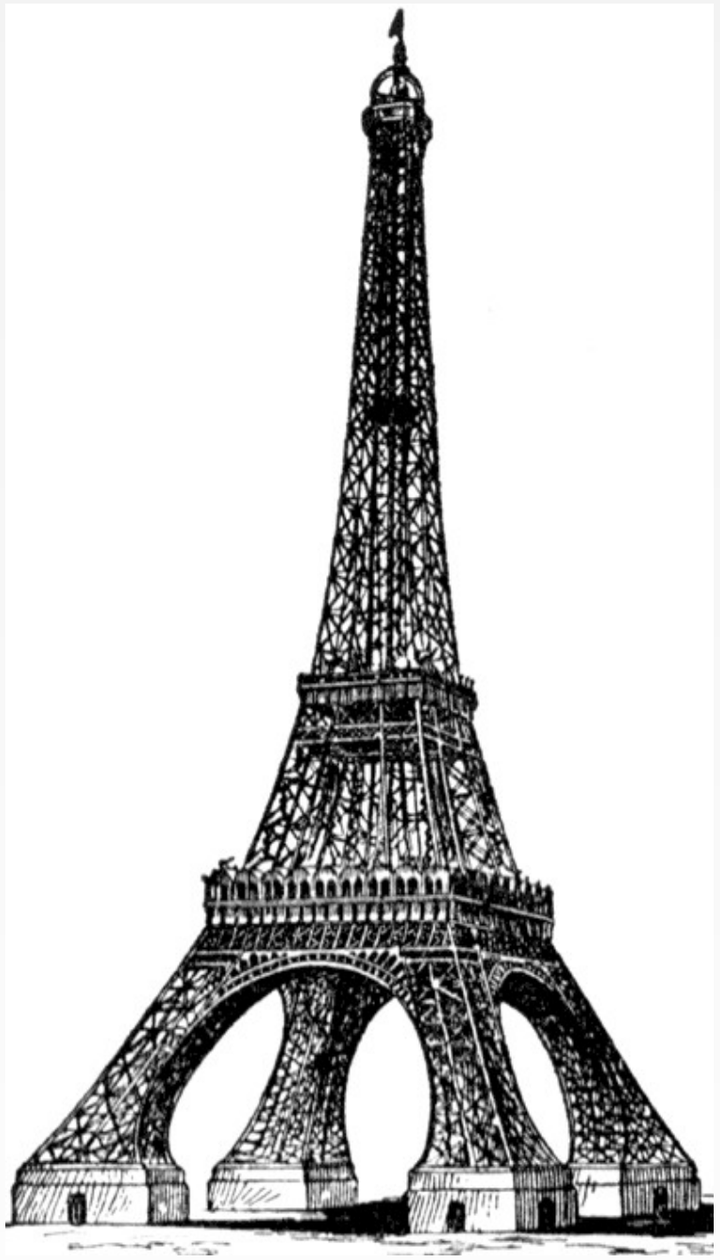
Self Documenting

Sound good?

canoo

Eiffel Envy

Symptoms include:



canoo

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)

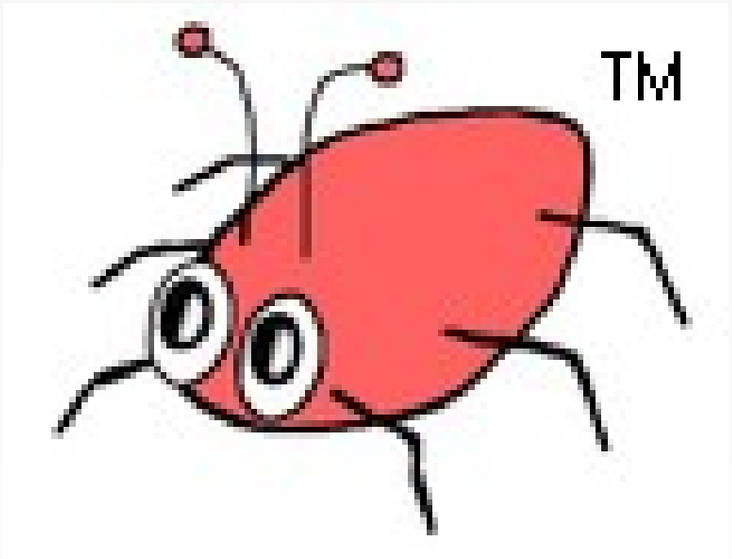
Best usage in Domain Model?

canoo

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

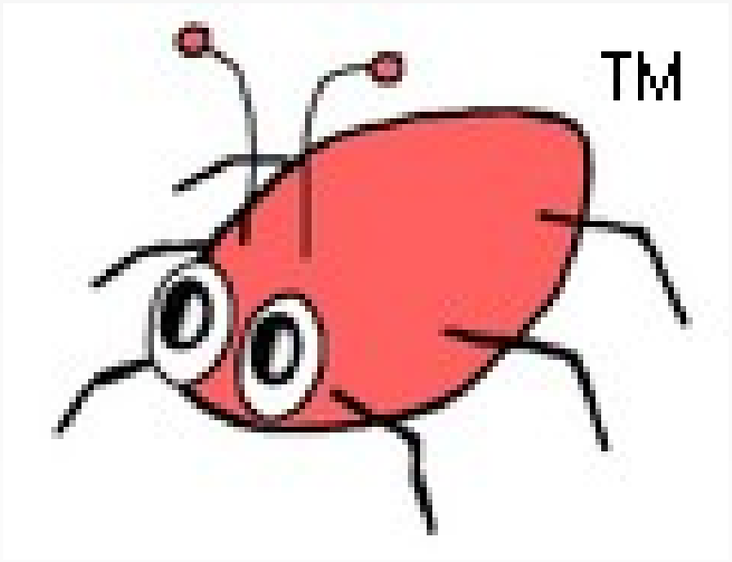
Q. Which OS Project won a 2008 Jolt Award for lamest logo?

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 67 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 67 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

canoo

Embedded Languages

Tired of hearing about DSLs?

Say “Embedded Language” instead!

ArithmeticShell ~= 300 lines of code

Alternative is

javacc?

custom interpreter?

canoo

Embedded Languages

Tired of hearing about DSLs?

Say “Embedded Language” instead!

ArithmeticShell ~= 300 lines of code

Alternative is

javacc?

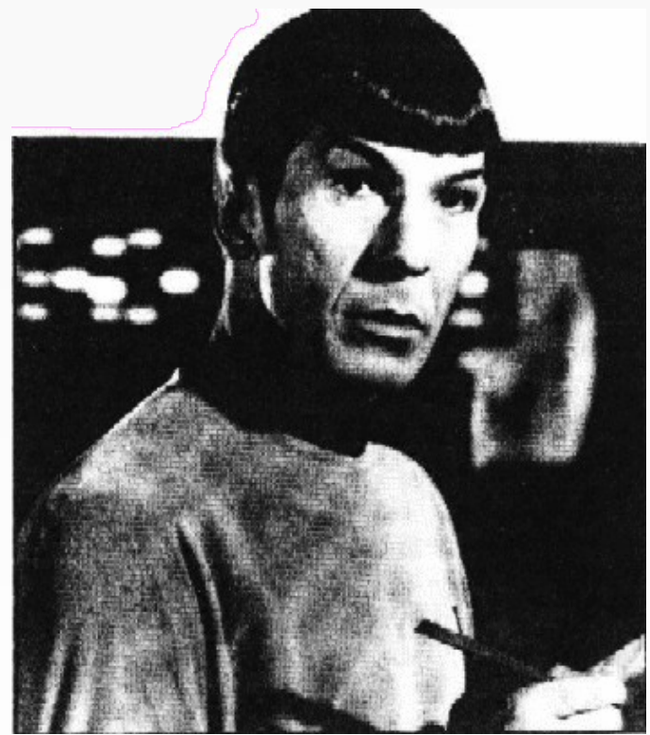
custom interpreter?

seppuku?

canoo

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
}
```



canoo

DEMO

again?

canoo

Groovy is a compiled language

...oh yes it is

Compiled changes visible in .class file

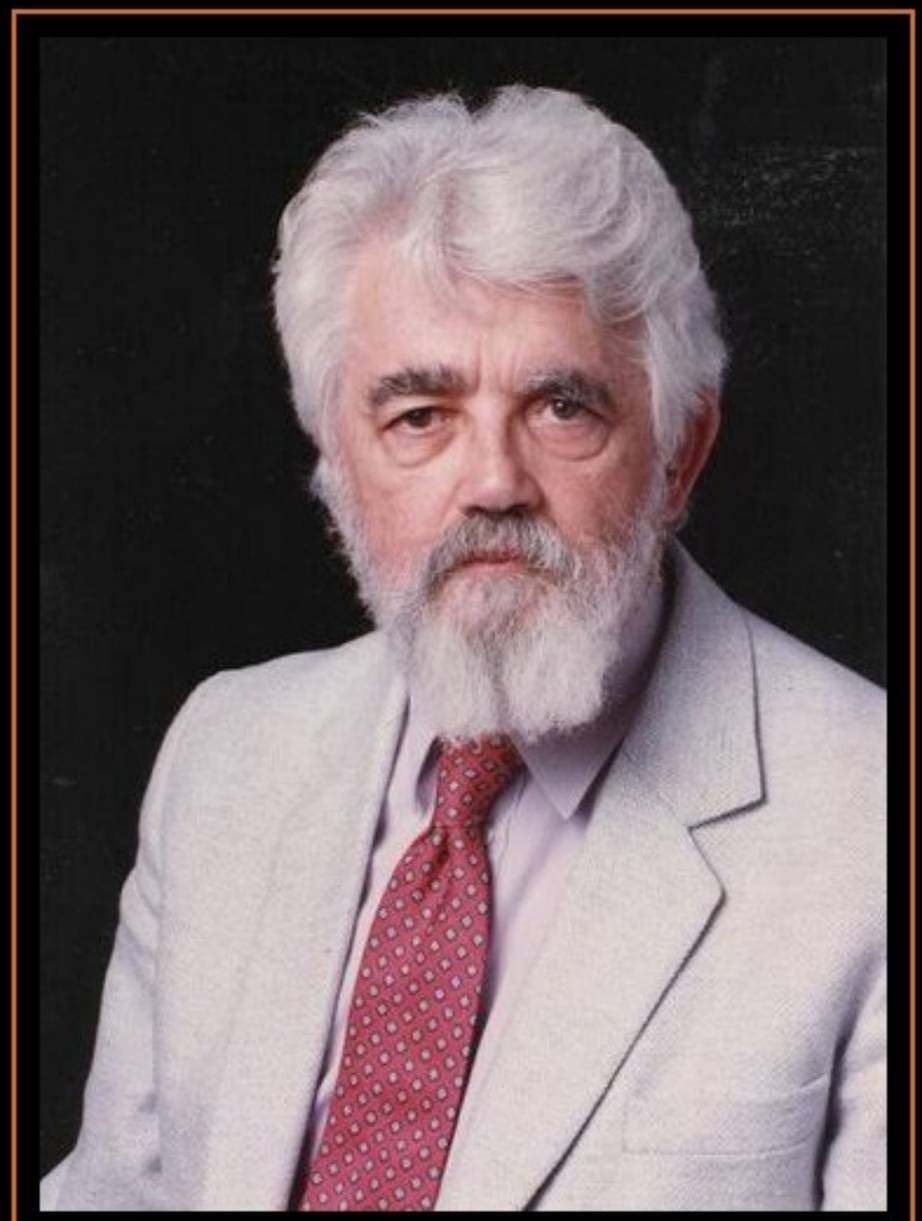
...visible to all JVM users

Language semantics are a library feature

...not hardcoded into the language

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

John McCarthy

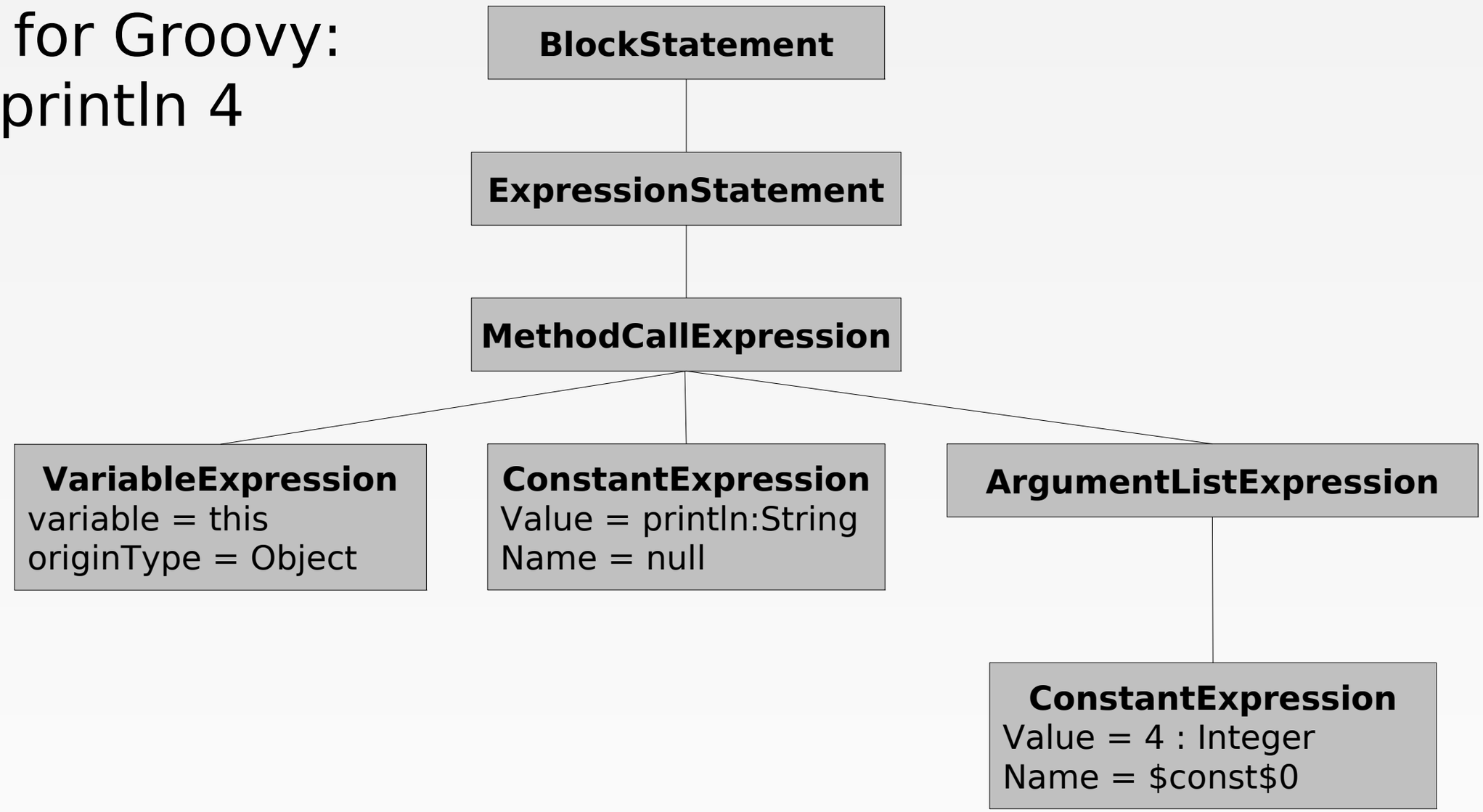


PROGRAMMING

YOU'RE DOING IT COMPLETELY WRONG.

canoo

AST for Groovy: println 4



… have you seen Groovy's AST Browser?

canoo

DEMO

OH YEAH

How It Works

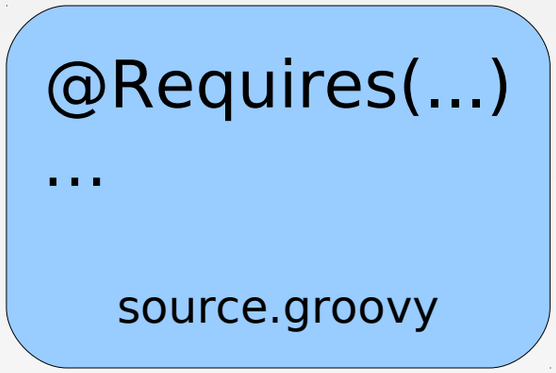
Local AST Transformations

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(...)
...
```

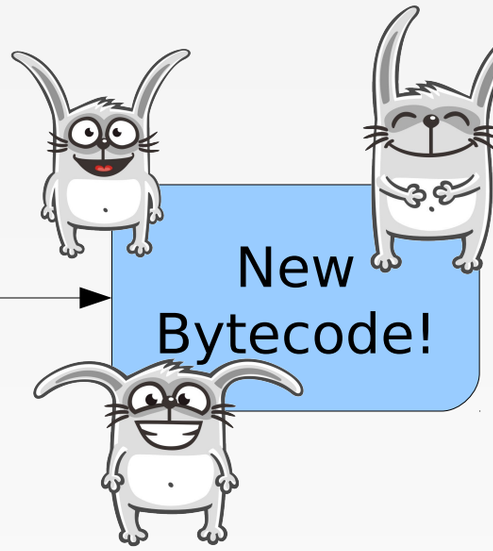
source.groovy

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

Requires.java

```
class MyASTTransformation
            implements ASTTransformation {
    ...
}
```

MyAstTransformation.java



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

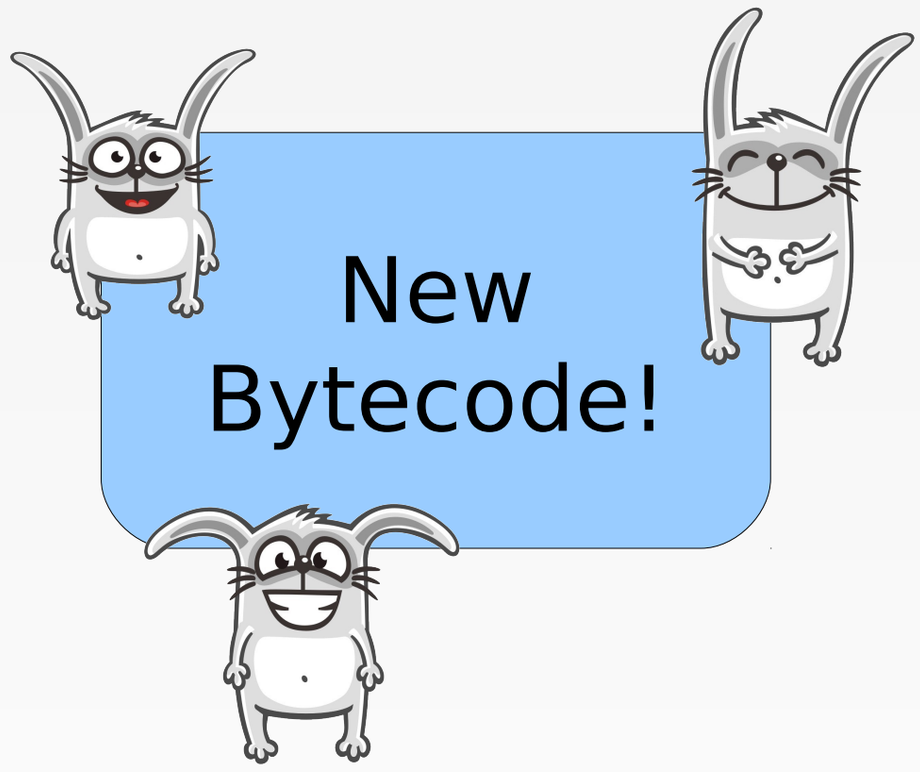
```
@Requires(...)
void startEngine() { ... }
```

```
@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!

canoo

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

canoo

Groovy Code Visitors

CodeNarc Rule:

```
    Ban System.runFinalizersOnExit()
```

canoo

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!

ACTIVISION

Pitfalls!

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

TransformTestHelper and IDE Support

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

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

def invoker = new TransformTestHelper(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

canoo

Writing AST

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

canoo

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

canoo

Rigid Syntax

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

canoo

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

canoo

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.

canoo

Combining AST with AST Builder

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

canoo

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

canoo

Combining AST with AST Builder

```
def memoizeMethod(MethodNode original) {
  def parameters = methodNode.parameters

  return new AstBuilder().buildFromCode {
    if (cache.contains( $parameters )) {
      return cache.get( $parameters )
    }
    def result = $methodNode.code
    cache.put($parameters, result)
    return result
  }[0]
}
```

… 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

canoo

Finding Insertion Points
&
Splicing AST into Source

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

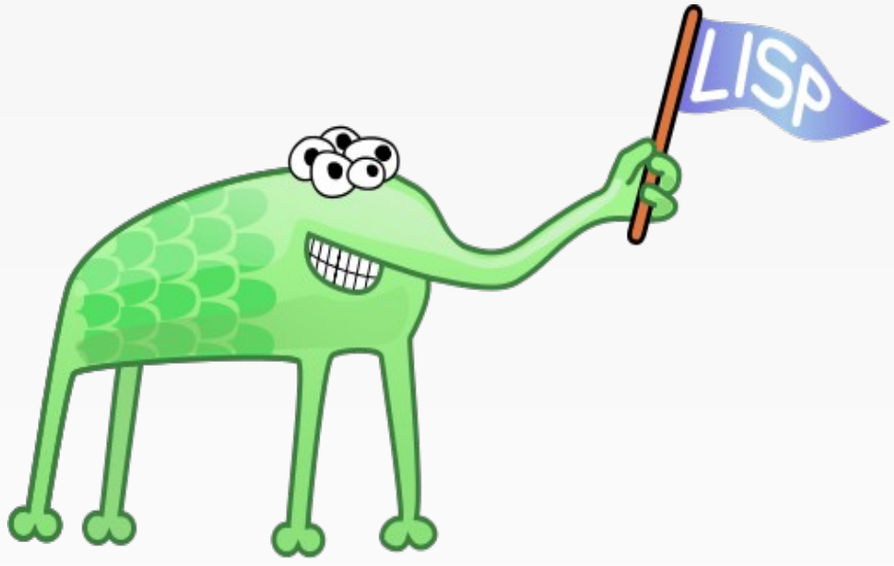
canoo

Finding Insertion Points
&
Splicing AST into Source

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

canoo

Warning... Lisp Ahead



canoo

Finding Insertion Points
&
Splicing AST into Source

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

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

canoo

Variable Capture

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

canoo

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

canoo

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

canoo

Thanks!

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

- I am speaking at:
  - CZ Jug
  - SpringOne/2GX
  - JavaOne
  - Your JUG? Please?