



**Chicago, October 19 - 22,
2010**

Code Generation on the JVM

Hamlet D'Arcy
Canoo Engineering AG
@HamletDRC

10/10/10



 **springsource¹**
A division of **VMWARE**

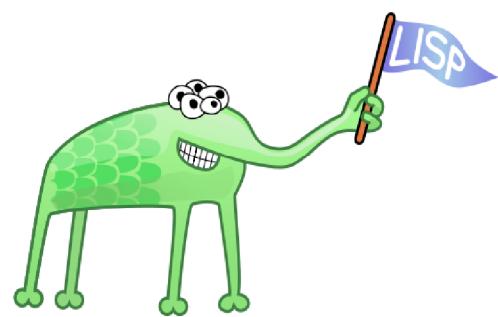

CORBA
Stub
Generation


Bean
Generation

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE wsdl:definitions PUBLIC "http://schemas.xmlsoap.org/wsdl/" "http://www.w3.org/2001/XMLSchema#wsdl">
<wsdl:definitions name="HelloWorld" targetNamespace="http://HelloWorldService" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://HelloWorldService" xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/">
    <wsdl:types>
        <xsd:schema targetNamespace="http://HelloWorldService">
            <xsd:import namespace="http://HelloWorldService" schemaLocation="http://HelloWorldService.wsdl" />
        </xsd:schema>
    </wsdl:types>
    <wsdl:message name="HelloWorldServicePortBindingMessage">
        <wsdl:part name="parameters" type="xsd:string" />
        <wsdl:part name="result" type="xsd:string" />
    </wsdl:message>
    <wsdl:portType name="HelloWorldServicePortType">
        <wsdl:operation name="HelloWorldServiceOperation" />
    </wsdl:portType>
    <wsdl:binding name="HelloWorldServicePortBinding" type="soap:rpc">
        <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />
        <wsdl:operation name="HelloWorldServiceOperation">
            <soap:operation message="HelloWorldServicePortBindingMessage" />
            <wsdl:input>
                <soap:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal" />
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="HelloWorldService">
        <wsdl:port name="HelloWorldServicePort" binding="HelloWorldServicePortBinding" />
    </wsdl:service>
</wsdl:definitions>
```

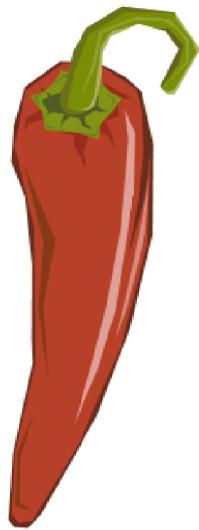
WSDL
Generation

I hate code generation too...



... but it isn't all bad.

Project Lombok



10/10/10

4

```
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;
    }

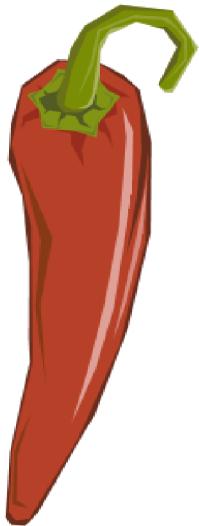
    public String getFirstName() {
        return firstName;
    }

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

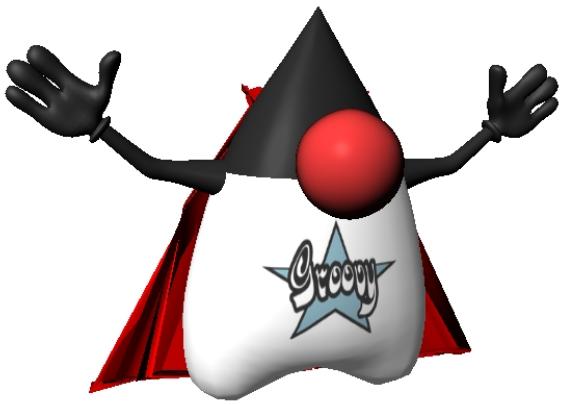
    public String getLastName() {
        return firstName;
    }
}
```

```
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



10/10/10

8

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

10/10/10

9

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

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

10/10/10

10

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

10/10/10

11

```
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(anotherDate)  
    }  
    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()  
    }  
}
```

10/10/10

12

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

10/10/10

13

```
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  
}
```

10/10/10

15

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

- Class is final
- Properties must be @Immutable or effectively immutable
- Properties are private
- Mutators 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('...')  
    }  
}
```

10/10/10

18

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

10/10/10

19

Coming in Groovy 1.8

@Log

@Synchronized

@IndexedProperty

@InheritConstructors

10/10/10

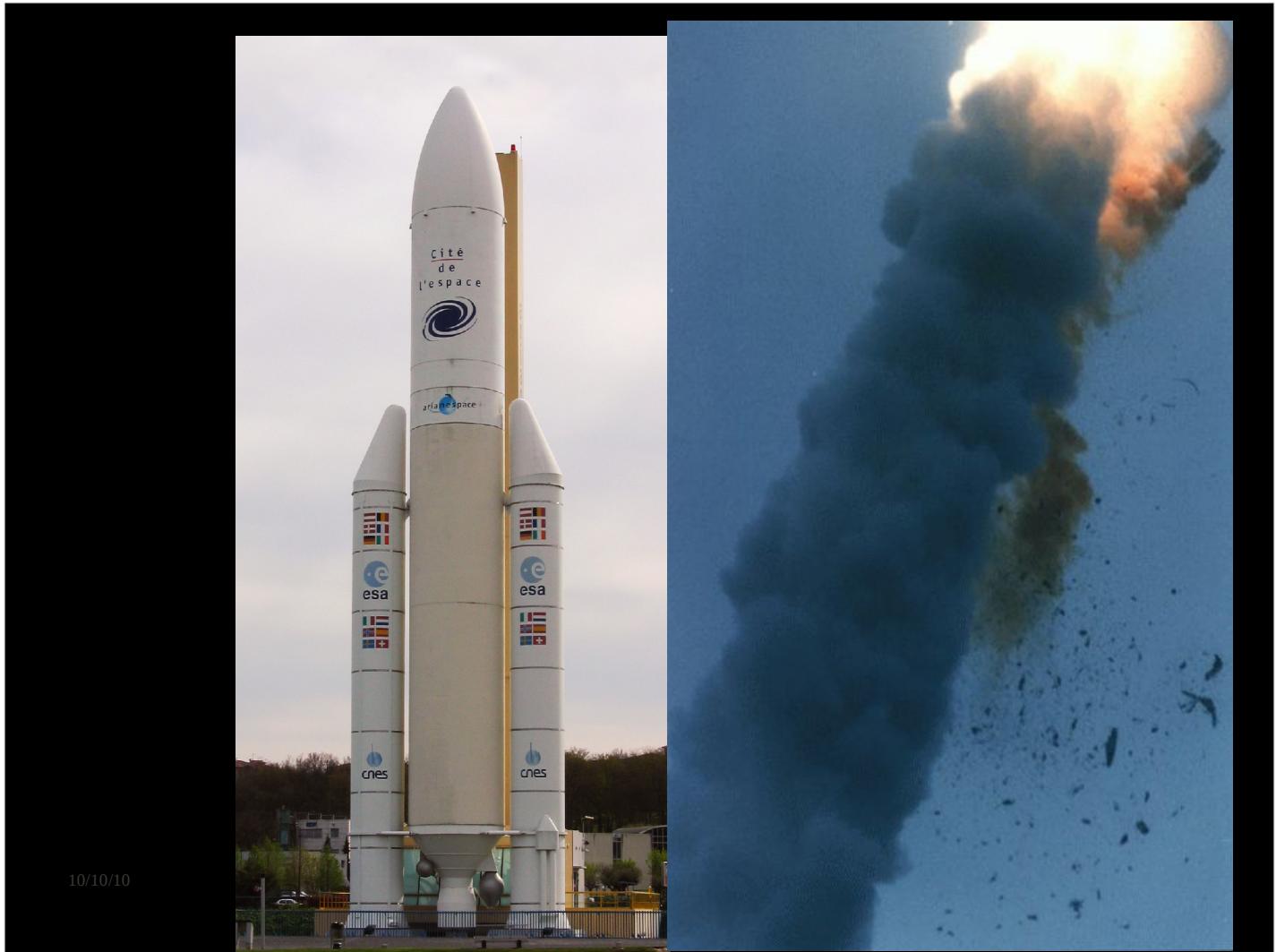
20



10/10/10

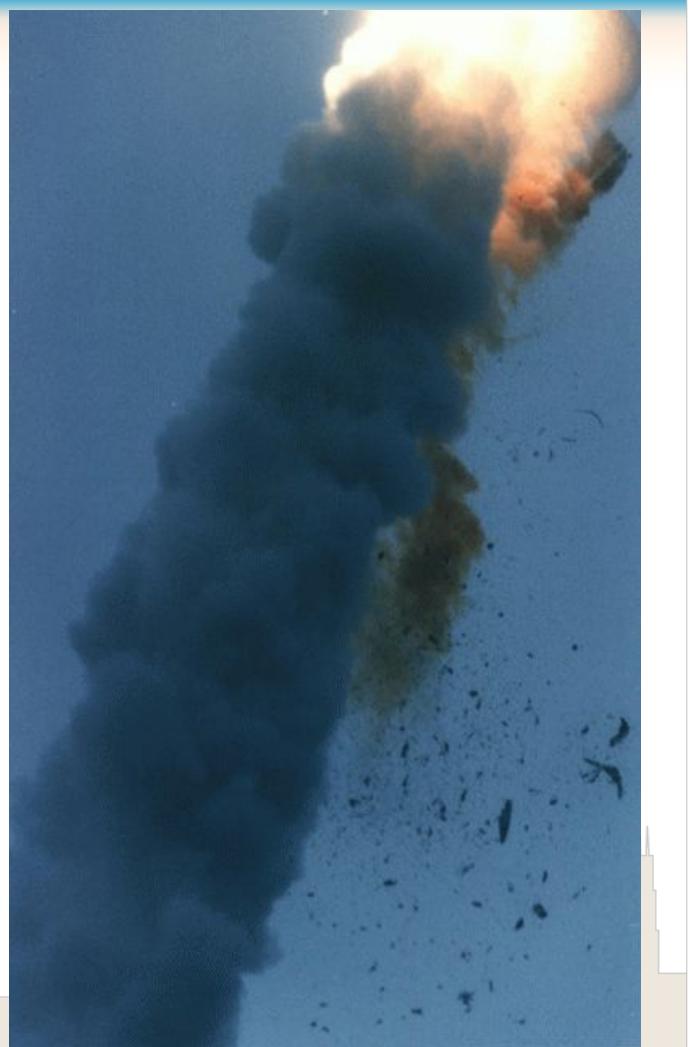
21

10/10/10



Ariane 5 Destroyed
US\$370 *million* in damages

... from a integer overflow error



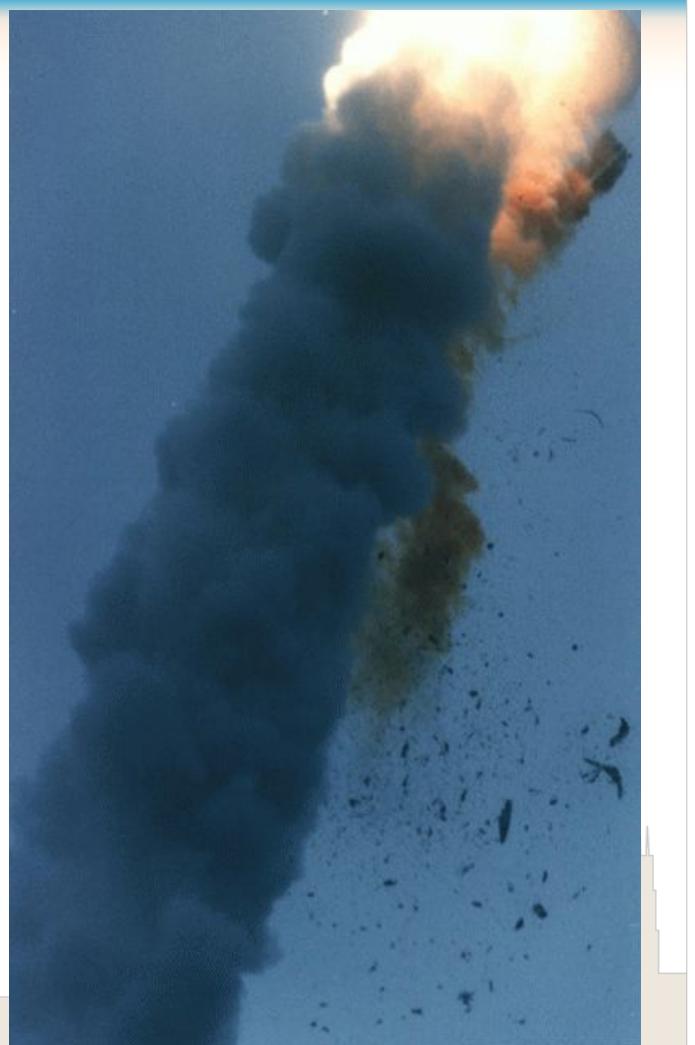
Ariane 5 Destroyed
US\$370 *million* in damages

... from a integer overflow error

Greece bailout is \$145 *billion*

... so we look quite good
compared to the bankers.

10/10/10



The lessons of Ariane

archive.eiffel.com/doc/n

Eiffel Software

Simple.

Products Downloads Services Developers Executives

Company Customers Partners News/Press Contact Us

This site contains older material on Eiffel. For the main Eiffel page, see <http://www.eiffel.com>.

Design by Contract: The Lessons of Ariane

by Jean-Marc Jézéquel, IRISA and Bertrand Meyer, ISE

This article appeared in a slightly different form in Computer (IEEE), as part of the Object-Oriented department, in January of 1997 (vol. 30, no. 2, pages 129-130).

Reader reactions to the article published in IEEE's Computer magazine appear at the end of the article.

Keywords: Contracts, Ariane, Eiffel, reliable software, correctness, specification, reuse, reusability, Java, CORBA, IDL.

How not to test your software

Several earlier columns in *IEEE Computer* have emphasized the importance of Design by Contract™ for constructing reliable software. A \$500-million software error provides a sobering reminder that this principle is not just a pleasant academic ideal.

On June 4, 1996, the maiden flight of the European Ariane 5 launcher crashed about 40

10/10/10

“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

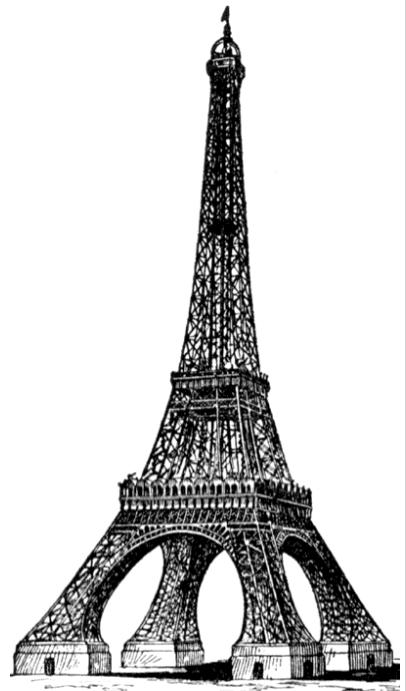
Sound good?

Eiffel Envy

Symptoms include:



10/10/10



28

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



10/10/10



29

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?

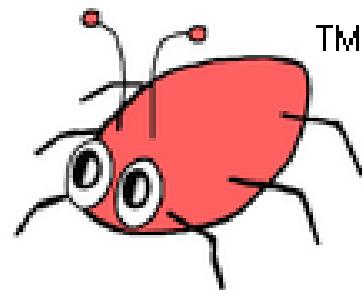
Q. Which OS Project won a 2008
Jolt Award for lamest logo?

10/10/10

31

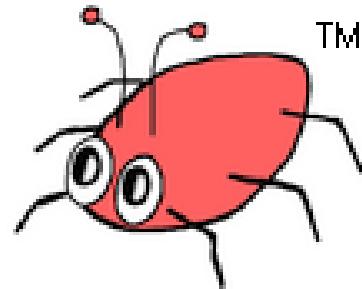
Q. Which OS Project won a 2008
Jolt Award for lamest logo?

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)')
```

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()')  
}
```

Embedded Languages

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

ArithmeticShell \approx 300 lines of code

Alternative is
javacc?
custom interpreter?

Embedded Languages

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

ArithmeticShell \approx 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  
}
```

10/10/10



Demo

again?

Groovy is a compiled language

...oh yes it is

Compiled changes visible in .class file

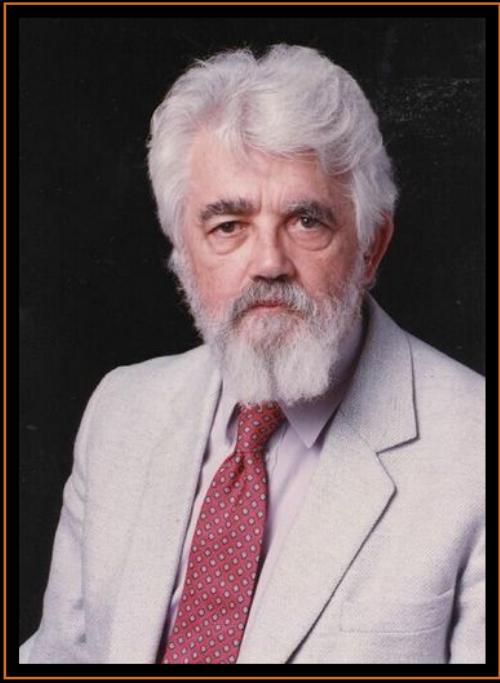
...visible to all JVM users

Language semantics are a library feature

...not hardcoded into the language

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

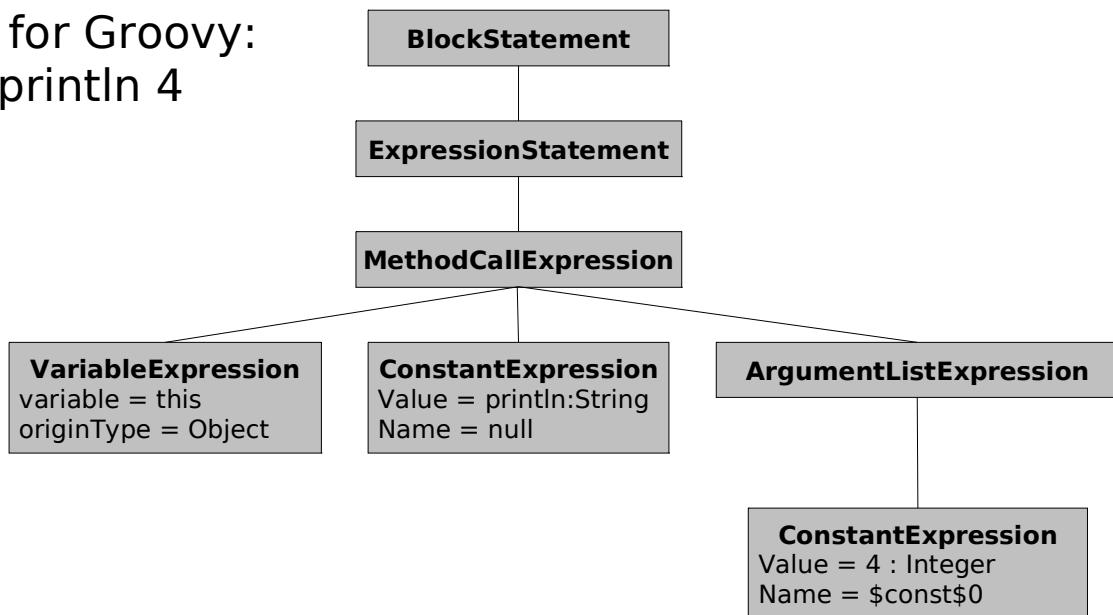
John McCarthy



PROGRAMMING

YOU'RE DOING IT COMPLETELY WRONG.

AST for Groovy:
println 4



10/10/10

... have you seen Groovy's AST Browser? 46

Demo

OH YEAH

How It Works

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

@Requires(...)

...

source.groovy

10/10/10

49

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

```
...
```

```
source.groovy
```

```
public @interface Requires {
```

```
...
```

```
Requires.java
```

10/10/10

50

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

```
...
```

```
source.groovy
```

```
public @interface Requires {
```

```
...
```

```
Requires.java
```

```
class MyASTTransformation  
    implements ASTTransformation {
```

```
}
```

```
MyAstTransformation.java
```

10/10/10

51

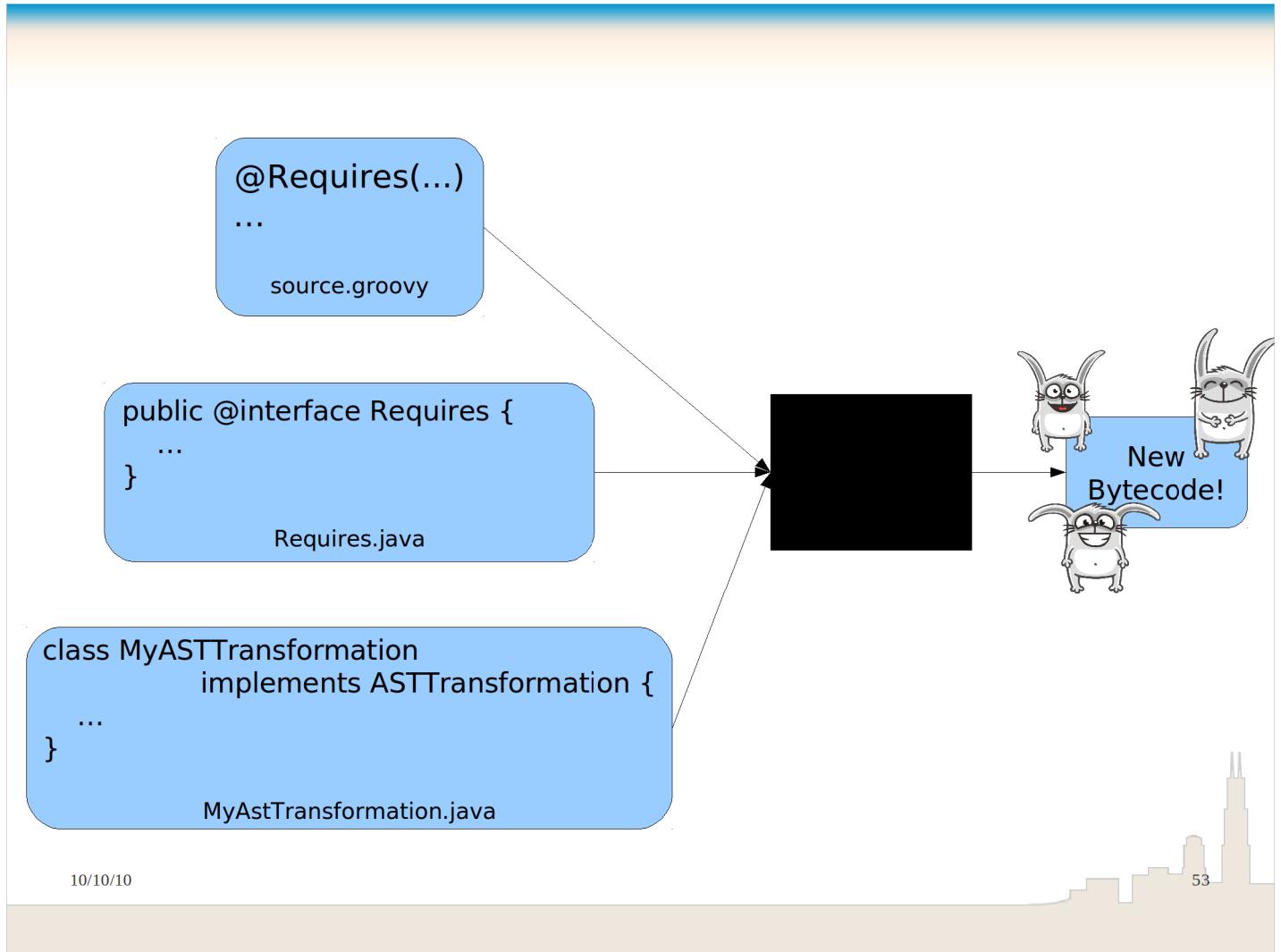
```
@Requires(...)  
...  
source.groovy
```

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

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

10/10/10

52



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

10/10/10

54

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

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

10/10/10

55

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



---


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

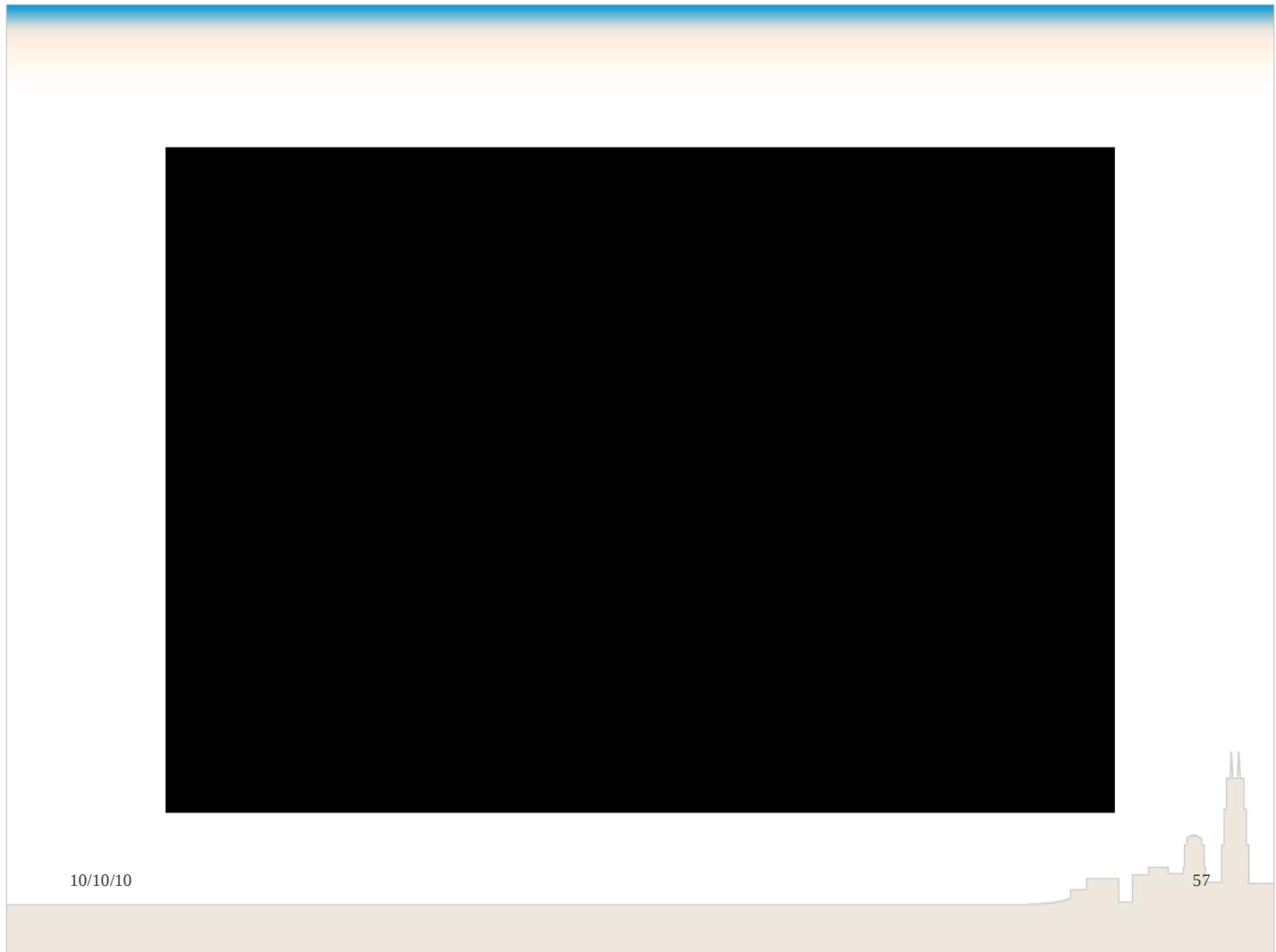


---


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

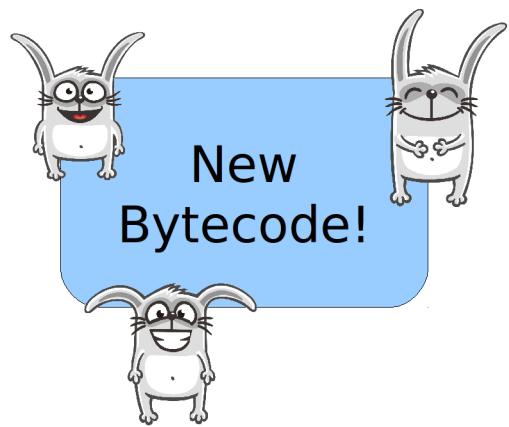
10/10/10

56



10/10/10

57



10/10/10

58

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);  
    ...  
}
```

10/10/10

... source in groovy/src/examples/groovyShell

Groovy Code Visitors

CodeNarc Rule:
Ban System.runFinalizersOnExit()

10/10/10

60

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);
    }
}
```

10/10/10

... source from codenarc

61

Pitfalls!

2000



ACTIVISION

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()
```

10/10/10

... 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'  
        }  
    }  
}
```

10/10/10

... from “Building AST Guide” on Groovy Wiki

68

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)  
}
```

10/10/10

... from “Groovy ANTLR Plugins for Better DSLs”

71

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.ToString()}
        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

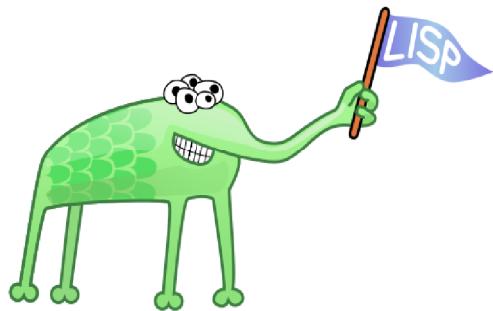
Finding Insertion Points & Splicing AST into Source

```
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



10/10/10

80

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 = ...  
}
```

10/10/10

82

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



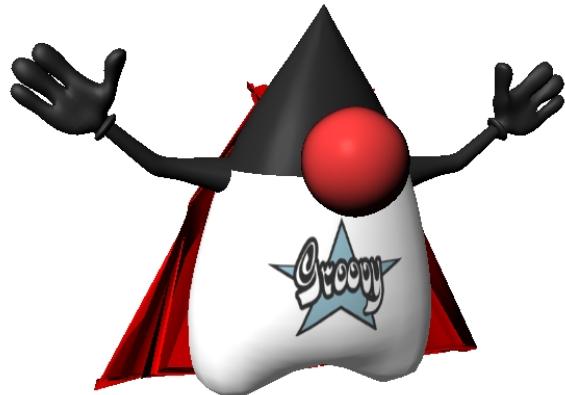
10/10/10

87

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