Hamlet D'Arcy | Canoo Engineering AG

Code Generation on the JVM

- Professional Software Developer

canoo
› your provider for business web solutions ›

- Open Source Committer

_ JConch (Java Concurrency)
_ CrushGraphics (Logo for .NET)

- And Contributor
  - _ Griffon Plugins
  - _ CodeNarc
  - _ Gradle
  - _ Gpars
  - _ ...

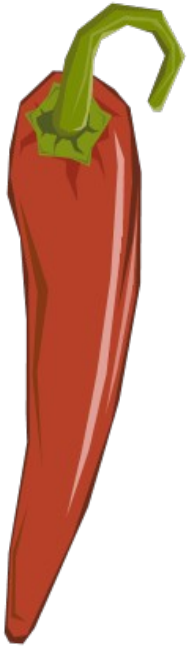
Bean Generation

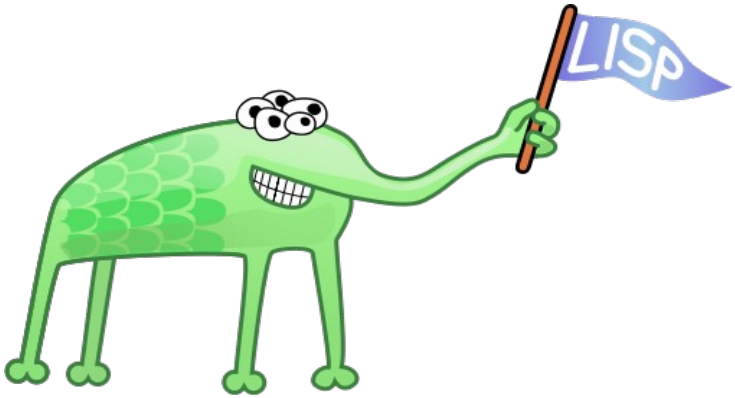
Stub Generation

```
- <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

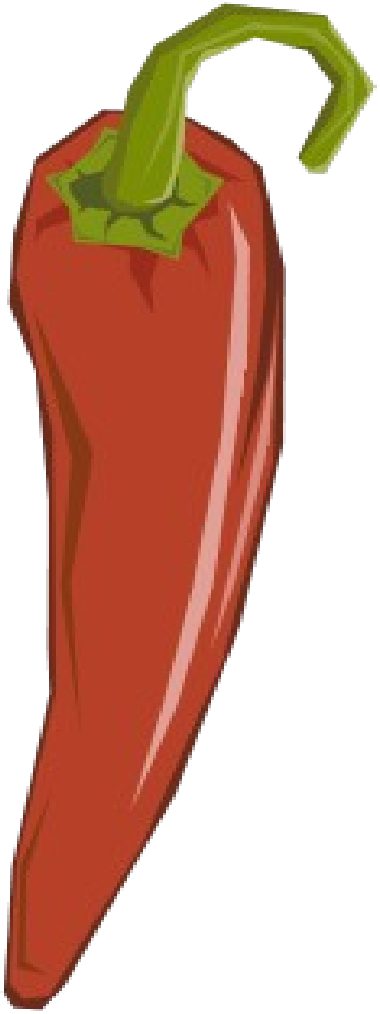
I hate code generation too!





... but it isn't all bad.

Project Lombok



```
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.Getter;
import lombok.Setter;

public class Person {

  @Getter @Setter private String firstName;
  @Setter @Setter private String lastname;

}
```

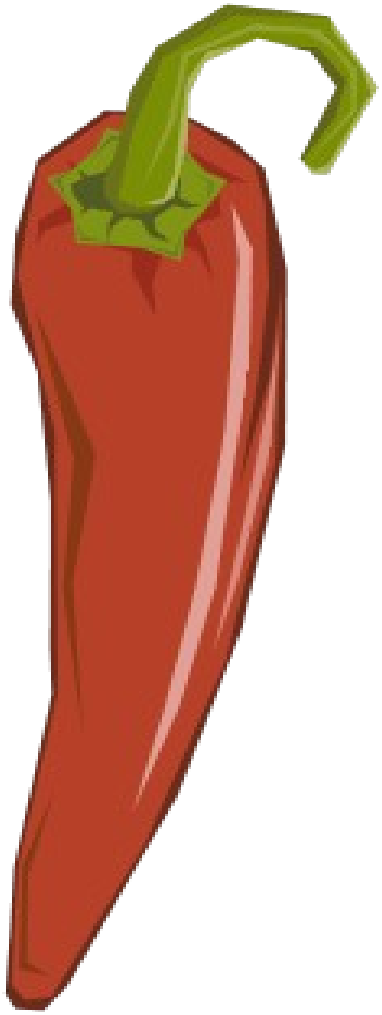
```
public class SynchronizedExample {
   private final Object $lock = new Object[0];

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

```
import lombok.Synchronized;

public class SynchronizedExample {

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



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

Spring Roo

- Moves Java Boilerplate to .aj files
  - Uses AspectJ's inter-type declaration (ITD) features
  - “Roo doesn't believe in magic!”

- Compile Time Only
  - Not a runtime
  - Not an Annotation Processing library
  - No dynamic proxy objets
- Removable with rollout mechanism
- There is a *lot* of code on disk
  - Is this Roo's fault?


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

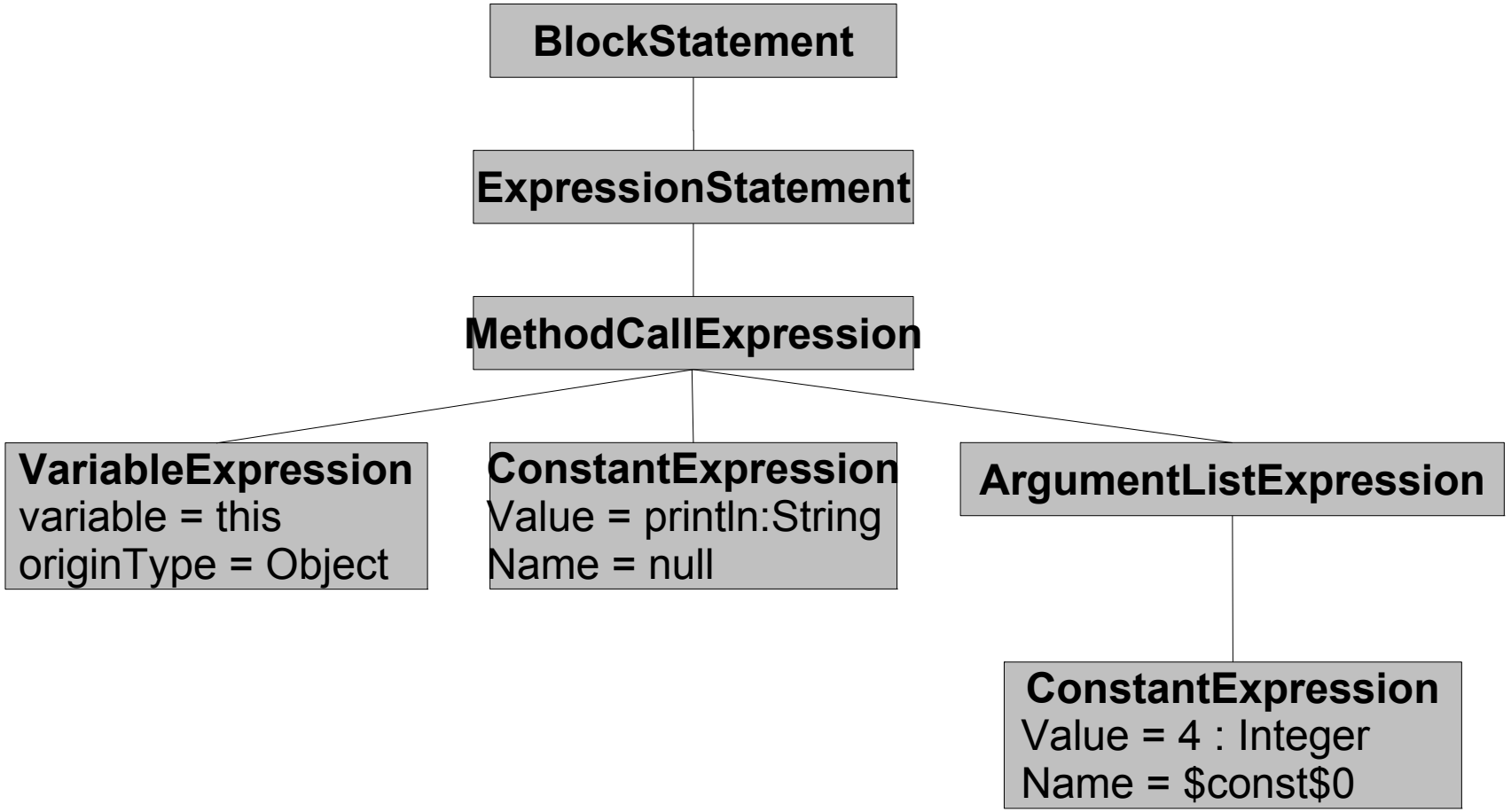
@Grab

... and many more as libraries

GContracts - @Insures, @Requires, @Invariant

Spock - @RunIf({jdkVersion >= 1.6 })

AST for “println 4”



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

GroovyCodeVisitor

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

GroovyCodeVisitor

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

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

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

- CodeNarc – static analysis for Groovy

```
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);
    }
}
```

- Spock – Testing Framework for Groovy

```
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
}
```

... from “Hijacking Goto Labels”

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

How it Works

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

Local AST Transformations

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

```
@GroovyASTTransformationClass("org.pkg.DelegateTransform")
public @interface Delegate {
    ...
}
```

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

Global AST Transformations

```
spock-core-0.3.jar!
/META-INF/services/org.codehaus.groovy.transform.ASTTransformation
    org.spockframework.compiler.SpockTransform
```

```
@GroovyASTTransformation(phase = CompilePhase.SEMANTIC_ANALYSIS)
public class SpockTransform implements ASTTransformation {
    public void visit(ASTNode[] nodes, SourceUnit sourceUnit) {
        ...
    }
}
```

AST Builder

```
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"
}
```

AST Templates

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

... from “GEP 4 - AstBuilder AST Templates”


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

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

Thanks!

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

- I am speaking at:
  - GR8 Conference
  - CZ Jug
  - Your JUG? Please?