

GEP 4 - AstBuilder AST Templates

Metadata

Number:	GEP-4
Title:	AstBuilder AST Templates
Version:	1
Type:	Feature
Status:	Draft
Leader:	Hamlet D'Arcy
Created:	2010-02-16
Last modification:	2010-02-16

Abstract

The AstBuilder was introduced in Groovy 1.7 as a way to ease the task of writing AST transformations. The "fromCode" approach allows you generate the AST from a piece of code. Currently, the AstBuilder fromCode API accepts a closure as a parameter, and the code within the closure is transformed to AST. A limitation of the AstBuilder is that the closure parameter does not allow variables in the surrounding context to be referenced: there is no way to pass a parameter into the code block. It was initially discussed but the idea was deemed too much effort to fit into 1.7.

This AST Template feature allows you to pass parameters into the AstBuilder. Similar to how a \$ works in GString interpolation, there needs to be a way to bind a variable from the enclosing scope into an AstBuilder closure. The syntax proposed is to use oxford brackets: `var`. An AstBuilder closure containing [GEP 4 - AstBuilder AST Templates](#) will have the variable within the brackets bound in.

Approach

The oxford brackets are used to bind, or quasi-quote, variables from the enclosing scope into an AstBuilder parameter.

Example 1:

String concatenation

```
def constant = new
ConstantExpression("World")
List<ASTNode> greeting =
AstBuilder.buildFromCode { "Hello" + [ |
constant | ] }
```

Produces the AST

```
BlockStatement
-> ExpressionStatement
  -> BinaryExpression +
    -> ConstantExpression - Hello
    -> ConstantExpression - World
```

Example 2:

Producing a println

```
new AstBuilder().buildFromCode { println([ |
message | ]) }[0]
```

is equivalent to invoking this method:

```
private Statement createPrintlnAst(String
message) {
    return new ExpressionStatement(
        new MethodCallExpression(
            new VariableExpression("this"),
            new
ConstantExpression("println"),
            new ArgumentListExpression(
                new
ConstantExpression(message)
            )
        )
    )
}
```

Example 3.

Memoization of a method. Presumably, you would have the existing method AST in a variable called "methodNode":

```

def methodNode = ...
def parameters = methodNode.parameters

def newMethod = new
AstBuilder().buildFromCode {
  if (cache.contains([ | parameters | ])) {
    return cache.get([ | parameters | ])
  }
  def result = [ | methodNode.code | ]
  cache.put([ | parameters | ])
  return result
}

methodNode.code = newMethod[0]

```

Alternatives

There are many syntax alternatives:

- Lisp uses ` and ,
- Template Haskell uses something similar to oxford brackets but with slightly different semantics
- Boo uses [GEP 4 - AstBuilder AST Templates](#) and \$ but with slightly different semantics
- A \$ was considered but cannot be used as it is already a meaningful identifier.

Boo Meta Method Comparison

The Boo Metamethod feature overlaps with the Groovy AST Transformation features. Instead of an @AstTransformation interface with a separately defined AstTransformation subclass, Boo offers [Meta](#) methods, where the AST Transformation is called and generated at compile time.

While Groovy offers AstBuilder to turn code into AST, Boo uses the oxford brackets. And while GEP-4 proposed oxford brackets to signify an AST variable templating, Boo uses the \$. Consider a simple println AST in Groovy

```
new AstBuilder().buildFromCode {
    println([ | message | ])
}[0]
```

Compared to a similar construct in Boo:

```
[Meta]
static def methodname(expr as Expression):
    [ | println($expr) | ]
```

GEP-4 proposes the opposite syntax from Boo: oxford brackets for AST Templating instead of the \$. The \$ is already a character in Java, and can be part of a variable name.

There are no real drawbacks to having an opposite syntax. The number of Boo AST Transformation writers migrating to Groovy is most likely quite low.

References

Mailing List Discussions

- <http://old.nabble.com/Groovy-AstBuilder-AST-Templates-td27607915.html>

JIRA Issues

- None yet

Useful Links

- <http://docs.codehaus.org/display/GROOVY/Building+AST+Guide>
- <http://blog.kartikshah.info/2010/01/revisiting-groovy-ast-transformation.html>
- [Boo Meta Methods](#)
- [Template Haskell in Wikipedia](#)
- [Template Haskell Home Page](#)

Reference Implementation

- None yet