

Pragmatic Groovy



Dion Almaer
dion@adigio.com
CTO, Founder, Adigio, Inc.
Editor-in-Chief, TheServerSide.com
www.adigio.com • info@adigio.com

Copyright © 2004

Presentation Agenda

In this presentation we will discuss:

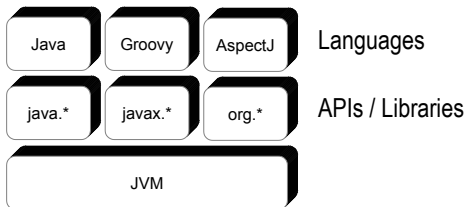
- What is groovy about Groovy?
 - Why another scripting language?
 - How can you use Groovy?
 - How does it fit in with Java?
 - A standard?
- The Groovy Language
 - How does the Groovy language work?
 - How is it different, and similar to Java?
 - Closures are your friend
- Learning how to be Groovy
 - Practical uses of Groovy
 - XML, Unit Testing, Markup Language Integration, IO
 - Databases, Build Systems, UI (Swing/SWT)
 - Groovlets, Groovy Templates, Groovy Beans, XML-RPC

Copyright © 2004 by Adigio, Inc.

Why do we need Groovy?

When we talk about Java what do we mean?

- "Java" isn't just a language, it is a platform



Copyright © 2004 by Adigio, Inc.

Reusability

- Java has a lot of great APIs and libraries
 - Core libraries (java[x].*)
 - Open source libraries
 - Third party commercial libraries
- What is it that we are *reusing* when we use these tools?
 - We are reusing the bytecode
 - We are reusing the fact that the JVM has a nice spec
- This means that we can innovate on top of this binary class file nonsense ☺

Copyright © 2004 by Adigio, Inc.

What is Groovy?

Groovy is...

- A dynamic, agile OO language for the JVM
- Based on the Java syntax
- A glue language
- We reuse far more code that we write
- A lot of what we are doing is gluing together the myriad of libraries that are out there for us
- It is written for the Java world...

Copyright © 2004 by Adigio, Inc.

The Roles of Groovy Use

Groovy can use in different roles

- Scripter Role
 - When you need to do some one scripts
 - Potentially one off
 - Perfect to have a simple script that wraps around the APIs that you have already been built
 - For example,
 - We have one off import user scripts for TSS that integrates with our business logic
- Real Business Logic
 - You can use Groovy or Java classes depending on the situation
 - For example,
 - Main business logic is in Java
 - Groovy is used for items such as web Actions
 - On a case by case basis you choose to safe as .java or .groovy
 - If you need to do a lot of data munging / regex you may choose to use Groovy

Copyright © 2004 by Adigio, Inc.

Roadmap



Where are we?

- What is groovy about Groovy?
- **The Groovy Language**
 - How does the Groovy language work?
 - How is it different, and similar to Java?
 - Closures are your friend
- Learning how to be Groovy
- Conclusion

Copyright © 2004 by Adigio, Inc.

Groovy Example



Is this Groovy, or Java?

```
public class Filter {
    public static void main( String[] args ) {
        List list = new java.util.ArrayList();
        list.add( "Rod" ); list.add( "Jane" ); list.add( "Freddy" );
        Filter filter = new Filter();
        List shorts = filter.filterLongerThan( list, 4 )
        for ( String item : shorts ) { System.out.println( item ); }
    }

    public List filterLongerThan( List list, int length ) {
        List result = new ArrayList();
        for ( String item : list ) {
            if ( item.length() <= length ) { result.add( item ); }
        }
        return result;
    }
}
```

- It is a lie to say that all Java is groked. There are still some small problems, such as handling inner classes

Copyright © 2004 by Adigio, Inc.

Groovy Example



Off Topic: New JDKs have helped here too!

```
List list = new java.util.ArrayList();

WAS:
list.add( "Rod" ); list.add( "Jane" ); list.add( "Freddy" );

NOW:
List<String> list = Arrays.asList<String>( "Rod", "Jane", "Freddy" );
```

- Credit where credit is due ☺

Copyright © 2004 by Adigio, Inc.

Groovy Example



This is much groovier though...

```
list = [ "Rod", "Jane", "Freddy" ]
shorts = list.findAll { it.size() <= 4 }
shorts.each { println it }

-> Rod
```

Copyright © 2004 by Adigio, Inc.

The Groovy Language



Groovy is like Java, except when it isn't

- Some of the main differences are:
 - Groovy offers dynamic and optionally static typing
 - Magic Strings
 - **Consistency**
 - Native syntax for common data types
 - Collections (Lists, Maps), Arrays, JavaBeans
 - Closures (*more soon!*)
 - Built-in support for Regular Expressions
 - Operator overloading
 - Autoboxing across collections, arrays, maps, etc
 - int's are Integers!
 - Additional convenience methods for the JDK (Groovy Dev Kit)

Copyright © 2004 by Adigio, Inc.

It's Optional



- Developers either love, or hate some restrictions in Java
 - Pro: Restrictions == One Way To Do Things == **Easy to read code**
 - Con: Restrictions == One Way To Do Things == **Not as expressive**

- Semi Colons

```
name = "dion";
name = "dion"
```

- Parentheses

```
println "dion"
println("dion")
```

NOTE: This is a very contentious issue!

Copyright © 2004 by Adigio, Inc.

Dynamic / Static Typing



- Groovy allows you to ignore typing if you want

```
size = "Large"

println "Size: " + size
println size.class

size = 1

println "Size: " + size
println size.class
```

```
C:\Temp\Groovy>groovy dynamic.g
Size: Large
class java.lang.String
Size: 1
class java.lang.Integer
C:\Temp\Groovy>
```

NOTE: You can put types on if you want!

Copyright © 2004 by Adigio, Inc.

GStrings



Groovy has special Strings, GStrings

- Strings are a lot more flexible

- Various ticks work
 - s = "string's are cool"
 - s = 'Dion "Beckham" Almaer'
- Strings wrap, so you don't need to "" + ""
 - s = "strings now\n" + "allow you to wrap"
- More quoting
 - s = ""I don't need to "quote" things""
- HereDoc (a la Unix shell and Perl)
 - s <<EndOfText
 - I still don't need to "quote" things
 - EndOfText

```
s = "strings now\n" +
    "allow you to wrap";
```

Copyright © 2004 by Adigio, Inc.

GStrings 2



- You can also embed variables into your strings

```
name = "dion"
println "hello ${name}"
println 'hello ${name}'
println "You have ${getAccount(name)} in your account."

def getAccount(theName) {
    "1 million dollars!!!"
}
```

No need for "return"

Single quotes don't expand

```
C:\Temp\Groovy>groovy string.g
hello dion
hello dion
You have 1 million dollars!!! in your account.
C:\Temp\Groovy>
```

NOTE: Remember that the `$$` is only within Strings!

I have seen: `println $$foo` and `foo = "${bar}"` too many times!

Copyright © 2004 by Adigio, Inc.

First Class Support for Common Data Structures



Java uses classes to implement most data structures

- Groovy offers first class support for

- Collections
 - Lists
 - Maps
- Arrays
- JavaBeans
- Iterating through Arrays or Collections is the same!

Copyright © 2004 by Adigio, Inc.

Native Lists



Square brackets are for lists

```
list = ["dion", 30, 'monkey', new Integer(3)]
println list[0]
println list.size()

list += ["one more", "ok make it two"]
list -= [30]
println list

# How about a slice. This shows ranges!
println list[1..3]

# How do you make an empty list?
list = []
```

```
C:\Temp\Groovy>groovy lists.g
dion
size: 4
one more, ok make it two
monkey, 3, one more
size: 3
C:\Temp\Groovy>
```

Copyright © 2004 by Adigio, Inc.

Native Lists... and Arrays and Strings



Square brackets are for Strings and Arrays too!

- Shows how in Groovy they have tried to unify things

```
s = "in the waiting line"
println s[0]
println s.size()

s += ", again"
s -= "waiting "
println s

println s[1..3]
```

```
C:\Temp\Groovy>groovy strings@lists.g
i
n the line, again
n the
C:\Temp\Groovy>
```

```
s = new String[] { "dion", "is", "a", "monkey" }
println s[0]
println s.size() # not .length!

println s[1..3]
```

```
C:\Temp\Groovy>groovy arrays@lists.g
dion
size: 4
is, monkey
C:\Temp\Groovy>
```

Copyright © 2004 by Adigio, Inc.

Native Maps



Square brackets are also for Maps!

```
map = ["name": "dion", "age": 30, "pet": 'monkey']
println map.name
println map['pet']

println map

map.phone = '123 123 1234'

println map

# How do you make an empty list?
map = [:] # ugly I know!
```

Groovy Beans



Groovy Beans offer a simple way to work with JavaBeans

- You can simply define properties and the get/set methods will be handled for you

```
class Employee {
    String name
    int salary
    Date hireDate
}
emp = new Employee(name:"Bob", salary:50000,
    hireDate:new Date())
```

Mmm. Named parameters!

- Note:
 - You can overwrite get/set methods if you want to put logic in there
 - If you do create just a get method, then the property will be read only
 - You can use the [] operator to access a property: emp['salary']

Groovy Beans: Property Rules



- When Groovy is compiled to bytecode, the following rules are used.

- If the property is private, then a Java field is used to represent the property.
- If a public or protected property is declared (properties are public by default), then a public or protected getter and setter are created along with a private Java field.
- If you don't declare getters or setters for public or protected properties, they are automatically created for you at the bytecode level.
- If you create a public or protected property, you can overload any auto-created methods.

```
class Foo {
    // read only property
    private String name
    public String getName() { return name }

    // read only property with protected setter
    Integer amount
    protected void setAmount(Integer amount) { this.amount = amount }

    // dynamically typed property
    cheese
}
```

Groovy Beans: Default Params



Groovy allows for default params

```
class Employee {
    String name
    int salary
    int age
    yearsToRetirement(retAge = 65) { return retAge - age }
}
emp = new Employee(name:"Bob", salary:50000, age: 30)

println emp.yearsToRetirement(40) # uses what is passed
println emp.yearsToRetirement() # will use the default!
```

Iteration with for



Iterating through collections, arrays, and Strings

- Iteration with 'for'
 - for (var in sequence) { code }
 - JDK 5.0: for (var : sequence) { code }

```
Simple loops: for ( index in 1..3 )
    { println index }

Strings: for ( ch in 'dog' )
    { println ch }

Lists and Arrays: for ( item in [1,2,'dog'] )
    { println item }

Maps: for ( entry in ['rod':33,'james':35] )
    { println "${entry.key} = ${entry.value}" }
```

- Support for Java while, do, if, else, etc.
- No standard Java for (int i=0; i < BLAH; i++) implemented yet

Closures



Closures are an important and integral part of Groovy

- You will find yourself using Closures all over the shop

- What is a closure?
 - A code block which is made into an object
 - Language implementations of the command / template pattern
 - Closures can take parameters

- KEY POINT: Unlike functions, blocks, or method objects:
 - state is visible before and after, as
 - blocks of code PLUS the bindings to the environment they came FROM

Operator Overloading



We can overload our own classes

```
class Point {
    int x
    int y

    Point(theX, theY) {
        x = theX
        y = theY
    }

    Point plus(Point newPoint) {
        x += newPoint.x
        y += newPoint.y
    }

    String toString() {
        "X Coord: $x, Y Coord: $y"
    }
}
```

```
a = new Point(1,1)
b = new Point(2,2)
c = a + b
println c
```

Copyright © 2004 by Adigio, Inc.

Operator Overloading



The following are a few of the operator => method mappings

Operator	Method
a+b	a.plus(b)
a-b	a.minus(b)
a++ / ++a	a.next()
a b	a.getAll(b)
a b = c	a.putAll(b, c)
a == b	a.equals(b)
a === b	a == b
a <> b	a.compareTo(b)
a << b	a.leftShift(b) (e.g. System.out << "Wheel" for C++ lovers!)

NOTE: The Groovy Dev Kit does some overloading for us on standard APIs (Please Turn Slide)

Copyright © 2004 by Adigio, Inc.

Groovy Dev Kit



Groovy mixes in a lot of new methods to the JDK

- String
 - contains(), count(), execute(), padLeft(), center(), padRight(), reverse(), tokenize(), each(), etc.
- Collection
 - count(), collect(), join(), each(), reverseEach(), findAll(), min(), max(), inject(), sort(), etc.
- File
 - eachFile(), eachLine(), withPrintWriter(), write(), getText(), etc.
- Thread.start { ... do stuff ... } (I love this one!)
- Lots there and growing all the time
 - <http://groovy.codehaus.org/groovy-jdk.html>
- You can add methods programmatically

```
thread = Thread.start {
    process = "some long running process".execute();
    process.waitFor();
}
```

Copyright © 2004 by Adigio, Inc.

Roadmap



Where are we?

- What is groovy about Groovy?
- The Groovy Language
- Learning how to be Groovy
 - Using Groovy to work with real problems
- Conclusion

Copyright © 2004 by Adigio, Inc.

File IO



Groovy adds many convenience methods onto what you are used to with java.io.File, and the other parts of the IO APIs

```
import java.io.File

# Split up a name=value file
new File("input.txt").eachLine { line |
    s = line.split("=")

    println "Name: ${s[0]}"
    println "Value: ${s[1]}"
}

# Copy one file to another
new File("output.txt").withWriter { out |
    new File("input.txt").eachLine { line |
        out.writeLine(line)
    }
}

# cat all files that you pass into the command line and uppercase lines
args.collect { name | new File(name) }.each { file |
    file.eachLine {
        println it.toUpperCase()
    }
}
```

Copyright © 2004 by Adigio, Inc.

XML: Groovy Markup



Groovy Markup has node support for many types of markup, and even more (e.g. Swing, Ant, etc)

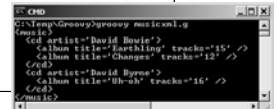
Firstly, let's see how we can produce XML

```
data = ['David Bowie': ['Changes':12, 'Earthling': 15],
        'David Byrne': ['Uh-oh':16] ]

xml = new groovy.xml.MarkupBuilder()

music = xml.music() {
    for ( entry in data ) {
        cd( artist: entry.key ) {
            for ( anAlbum in entry.value ) {
                album( title: anAlbum.key, tracks: anAlbum.value )
            }
        }
    }
}

println music
```



The power is how you can mix data, and real programming logic!

Copyright © 2004 by Adigio, Inc.

XML: Parsing and Navigating



- We can easily read in XML and poke and prod around

```
music = new XmlParser().parse("music.xml")

println music.cd.album['@title']
#-> [Earthling, Changes, Uh-oh]

println music.cd.album.findAll {
  Integer.parseInt(it["@tracks"]) > 14
}[["@title"]]
#-> [Earthling, Uh-oh]
```

```
1: CMD
C:\Temp\Groovy>groovy music.xml.g
<music>
  <cd artist="David Bowie">
    <album title="Earthling" tracks="15" />
    <album title="Changes" tracks="12" />
  </cd>
  <cd artist="David Byrne">
    <album title="Uh-oh" tracks="16" />
  </cd>
</music>
```

Copyright © 2004 by Adigo, Inc.

XML: Transformation



- Advanced XML Generation
 - StreamingMarkupBuilder in sandbox generates HTML and complex XML - used to generate GDK javadoc
- Groovy as Human-Readable XSLT
 - Use XML parsing and navigation (findAll, every, any, etc.) to locate data
 - Use MarkupBuilder, StreamingMarkupBuilder, etc. to render new format
 - Example in groovy.util.NavToWiki.groovy - transforms XDoc to Wiki

```
# Excerpt from groovy.util.NavToWiki.groovy
doc = new XmlParser().parse("navigation.xml")
items = doc.body.links.item
println items.collect {
  "[link:" + it["@name"] + "|" + it["@href"] + "]"
}.join(" | ")

# Original XML File
<body><links>
  <item name="Download" href="download.html"/>
  <item name="JavaDoc" href="apidocs/index.html"/> ...

# Wiki Output
[link:Download|download.html] | [link:JavaDoc|apidocs/index.html]
```

Copyright © 2004 by Adigo, Inc.

Gravy: Groovy + Ant



- Groovy Markup can be used all over.

```
class Build {
  ant = new groovy.util.AntBuilder()
  targets = ['clean', 'compile']

  static void main(args) {
    b = new Build()
    if (args.size() > 0) { b.targets = args }
    b.run()
  }

  run() {
    for (target in targets) {
      println "Target: ${target}"
      invokeMethod(target.toString(), null)
    }
  }

  clean() {
    ant.rmDir(dir:'target')
  }

  compile() {
    ant.mkDir(dir:'target/classes')
    ant.compile(srcDir:'src/main/java', destdir:'target/classes')
    fileset {
      includes(name:'**/*.java')
    }
  }
}
```

- One good example is the Ant support.
- Q. Why would you use this?

Copyright © 2004 by Adigo, Inc.

Groovy UI: Swing and SWT



- Groovy has support for Swing and SWT
- Here is a full Swing example

```
frame = swing.frame(title:'This is a Frame',
  location:[100,100], size:[800,400]) {
  menuBar {
    menu(text:'File') {
      menuItem() {
        action(name:'New',
          closure:{ println("clicked File->New") })
      }
    }
  }
  panel(layout:new BorderLayout()) {
    label(text:'Name', constraints:BorderLayout.WEST,
      tooltipText:'This is the name field')
    textField(text:'Rod', constraints:BorderLayout.CENTER,
      tooltipText:'Enter the name into this field')
    button(text:'Click me!', constraints:BorderLayout.SOUTH,
      actionPerformed: {println("Button clicked!") })
  }
}
frame.show()
```

Copyright © 2004 by Adigo, Inc.

Groovy SQL



- Groovy has very nice SQL support
- The power comes from closures again
 - Notice that there is no need to close connections, statements, and such!
 - Also, we are gluing into code to get connections

```
sql = new Sql(DBUtil.getDirectConnection())

sql.eachRow("select distinct m.userpk from ...") {
  userpk = it.userpk
  println "Building for ${userpk}"

  if (!CommunityAdmin.isAdmin(userpk)) {
    sharedSql = "SELECT DISTINCT uip.userpk FROM user_ip_map"

    sql.eachRow(sharedSql) { shared |
      if (!CommunityAdmin.isAdmin(shared.userpk)) {
        println "UserPK: ${userpk}, Shared: ${shared.userpk}"
        sql.execute("INSERT INTO s (userpk, shareduserpk) VALUES (?, ?)",
          [ userpk, shared.userpk ])
      }
    }
  }
}
```

Copyright © 2004 by Adigo, Inc.

Groovy SQL: Dataset



- Instead of coming up with INSERTs you can use Datasets
- Datasets allow you to have a record base view of your data
 - Create a new record, and it will be inserted
 - Change a record and it can be updated/deleted

```
sql = new groovy.sql.Sql( dataSource )

sql.execute( "create table person
  ( name varchar, age integer )" )

people = sql.dataSet( "person" )
people.add( name: "Bob", age: 30 )
people.add( name: "Harry", age: 32 )
goldenoldies = people.findAll { it.age > 50 }
```

Copyright © 2004 by Adigo, Inc.

Groovy Unit Testing



- Unit testing is important
- Writing Unit Tests can be laborious
- Sometimes you don't have to be as 'strict' with unit tests, and this is why Groovy is a great way of writing the tests

```
import groovy.util.GroovyTestCase

class MyTest extends GroovyTestCase {
    void testSomething() {
        assert 1 == 1
    }
}
```

- Then just run: `% groovy MyTest.groovy`
 - Or, have `<groovy>` compile the groovy to `.class` and let junit do its thing!
- NOTE: Using Groovy in small parts of your project first is a great way to play with it, without having it be part of your production system right away. We introduced Groovy into TSS code by first doing unit testing with it!*

Copyright © 2004 by Adigio, Inc.

Groovy Unit Testing: More Assertions



- The `GroovyTestCase` adds more assertions for you to use

- `assertArrayEquals(Object[] expected, Object[] value)`
- `assertLength(int length, char[] array)`
- `assertLength(int length, int[] array)`
- `assertLength(int length, Object[] array)`
- `assertContains(char expected, char[] array)`
- `assertContains(int expected, int[] array)`
- `assertToString(Object value, String expected)`
- `assertInspect(Object value, String expected)`
- `assertScript(final String script) // assert that a script runs without exceptions`
- `shouldFail(Closure code) // assert that an exception was thrown in that closure`
- `shouldFail(Class clazz, Closure code) // the same but for a class`

Copyright © 2004 by Adigio, Inc.

Groovy and Web Services



- Dynamic languages can be perfect for loosely coupled systems
- Groovy has built-in support for XML-RPC
 - Package: `groovy.net.xmlrpc`

```
import groovy.net.xmlrpc.*

# The Server Side
server = new XMLRPCServer( 2, 10, 8, 1000, 1000 )
server.testme = { | name | name + " is cool!" }
server.multiply = { | number | number * 10 }
serverSocket = new java.net.ServerSocket( 9047 )
server.startServer( serverSocket )

# Client
serverProxy = new XMLRPCServerProxy("http://127.0.0.1:9047")
println serverProxy.testme( "Groovy" )
-> "Groovy is cool!"
println serverProxy.multiply( 7 )
-> 70

# Kill the server
server.stopServer()
```

Copyright © 2004 by Adigio, Inc.

Groovy and the Web tier



There is a lot going on with Groovy on the web tier

- Groovlets
 - You can write Servlets in Groovy
 - A `GroovyServlet` will automatically compile your `.groovy` source files, turn them into bytecode, load the Class and cache it until you change the source file.
- GSPs
 - Forget JSP, try GSP ☺
 - GSP works with the Template support in Groovy
 - Groovy Templates look similar to Velocity
- Frameworks that work with Groovy:
 - `NanoWar`: Built on the `Pico/NanoContainer` IoC framework
 - `WebWork2`: We integrated Groovy into `WebWork2`. Now we can write actions in Groovy for a faster dev cycle (*no ant build deploy restart arggh!*)

Copyright © 2004 by Adigio, Inc.

Groovlets



Let's look at an example of a Groovlet

```
import java.util.Date

if (session.counter == null) {
    session.counter = 1
}

println ""

<html>
<head>
<title>Groovy Servlet</title>
</head>
<body>
Hello, ${request.remoteHost}: ${session.counter}! ${new Date()}
<br>src
</body>
</html>
***
session.counter = session.counter + 1
```

Copyright © 2004 by Adigio, Inc.

Groovy and Java



Since Groovy is meant to work so well with Java, what can you do?

- Using Java in Groovy
 - "Just Do It"
 - You can use any Java class, inherit from a Java class, or anything else that tickles your fancy
- Using Groovy from Java
 - You can use and run Groovy scripts
 - You can compile Groovy classes to Java classes and use them
 - Let's see!

Copyright © 2004 by Adigio, Inc.

Using Groovy from Java



- It is normally smart to hide items behind interfaces:

```
ClassLoader cl = Thread.currentThread().getContextClassLoader();
GroovyClassLoader groovyCl = new GroovyClassLoader(cl);
Class groovyClass = groovyCl.parseClass(
    cl.getResourceAsStream("FooImplementation.groovy"));
FooInterface foo = (FooInterface) groovyClass.newInstance();
foo.callSomeMethod();
```

- BSF Compliant Scripting

```
import groovy.lang.GroovyShell;

public class Foo {
    public static void main(String[] args) throws _ {
        Binding b = new Binding();
        b.setVariable("name", "Bob");
        GroovyShell sh = new GroovyShell(b);
        sh.evaluate("println \"Hello ${name}\"");
    }
}
```

Copyright © 2004 by Adigio, Inc.

Fun Stuff



- There is so much going on in the Groovy community that we can't cover in detail

- Some of the fun things out there are:

- Mixins:** The ability to mixin functionality is very powerful (mixin Foo { ... })
- Thicky:** Groovy-based fat client delivery (Rich Internet Apps here we come)
- Expando:** Auto properties

```
clark = new groovy.util.Expando(name: 'Clark Kent', codename: 'Superman')
clark.status = 'hero'
clark.shout = { println 'Help is on the way' }

println clark.status
clark.shout()

clark.shoutName = { println clark.name }
clark.shoutName();
```

Copyright © 2004 by Adigio, Inc.

Roadmap



Where are we?

- What is groovy about Groovy?
- The Groovy Language
- Learning how to be Groovy
- Conclusion
 - Groovy JSR Status
 - The real issues with Groovy

Copyright © 2004 by Adigio, Inc.

Conclusion: JSR 241 Status



- JSR-241: The Groovy Programming Language

- Standardized Language Specification
- Open Source Reference Implementation

- Ongoing Debates

- Optional Semicolons
- Optional Parentheses
- Properties/JavaBean Spec Compliance
- Closure Variable Scoping
- Closure Syntax Sugar/Whitespace

- Work starting up again

- Need to iron out sugar
- Need to solidify the implementation

Copyright © 2004 by Adigio, Inc.

Conclusion: The Honest Truth



- Groovy is great, and has potential
- However, there are serious issues to consider

- DEBUG HELL**

- This is the worst part of working with Groovy
- If a problem occurs, it can be hard to work out what is going wrong
- You end up staring at long, cryptic, stack traces ☹
- This will be fixed in the future. Please.

- Tool Support**

- IntelliJ and Eclipse can be contagious
 - There are plugins that are in a serious alpha state
 - You could argue that you can then dev faster in Java due to the tool
 - However you have to remember how much time you spend *READING* versus *WRITING* code!

- Language Wars**

- Java camp vs. non-Java camp
- How Java-like should it be?
- How much syntactic sugar should it have?

- Performance can be a little, or a lot slower

- It depends on how you use Groovy, how much reflection is going on, and if you turn on optimizations

Copyright © 2004 by Adigio, Inc.

Module Summary



In this module we discussed:

- What is groovy about Groovy?
- The Groovy Language
- Learning how to be Groovy
- Conclusion



Copyright © 2004 by Adigio, Inc.