Groovy

JSPRING May 11, 2004

Speaker Laurent Weichberger – Trivera Technologies

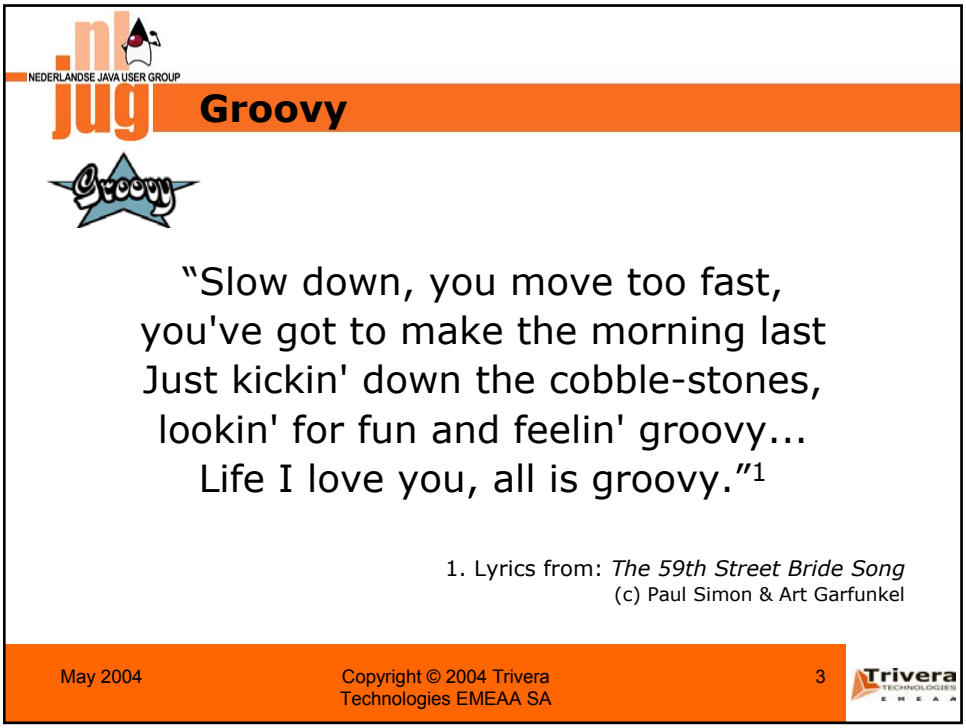
Disclaimer

1. Content
TRIVERA TECHNOLOGIES EMEAA reserves the right not to be responsible for the topicality, completeness or quality of the information provided. Liability claims regarding damage caused by the use of any information provided, including any kind of information which is incomplete or incorrect, will therefore be rejected. Parts of the pages or the complete publication including all information might be extended, changed or partly or completely deleted by the authors of TRIVERA TECHNOLOGIES EMEAA without separate announcements.

2. Referrals
TRIVERA TECHNOLOGIES EMEAA is not responsible for any contents referred to from their presentation -unless TRIVERA TECHNOLOGIES EMEAA has full knowledge of illegal contents and would be able to prevent the readers from viewing those pages. If any damage occurs by the use of information presented there, only the author of the respective pages might be liable, not TRIVERA TECHNOLOGIES EMEAA who has linked to these pages.

3. Copyright
TRIVERA TECHNOLOGIES EMEAA intended not to use any copyrighted material for the publication or, if not possible, to indicate the copyright of the respective object. The copyright for any material created by the authors of TRIVERA TECHNOLOGIES EMEAA is reserved. Any duplication or use of objects such as diagrams, texts in electronic form or printed publications is not permitted without TRIVERA TECHNOLOGIES EMEAA's agreement in writing.

4. Legal validity of this disclaimer
This disclaimer is to be regarded as part of this presentation. If sections or individual terms of this statement are not legal or correct, the content or validity of the other parts remain uninfluenced by this fact.



Introduction to Groovy

What is Groovy?

- "Groovy is an agile, dynamic programming language for the Java Virtual Machine. Groovy includes features found in Python, Ruby, and Smalltalk, but uses syntax similar to the Java programming language."[2]

2. From JSR-241 web page at http://www.jcp.org/en/jsr/detail?id=241

Where does Groovy come from?

- James Strachan and Bob McWhirter, two open-source gurus, worked together to create Groovy
- It was submitted as a Java Specification Request (JSR-241) under the Java Community Process (JCP)
- Current status: In Progress (read: groovin')

Description of JSR-241

- Description: "This specification will standardize the Groovy programming language so that vendors can provide compliant implementations and developers will have a sanctioned scripting language they can use on the Java™ platform."[2]

Specification Support (1 of 2)

- Specification Lead(s): Richard Monson-Haefel and James Strachan
- There was a unanimous vote of "YES" from 16 individuals and organizations, as follows:
  - Apache Software Foundation
  - Apple Computer, Inc.
  - BEA Systems
  - Borland Software Corporation
  - Caldera Systems
  - Fujitsu Limited
  - Hewlett-Packard
  - IBM
  - IONA Technologies PLC
  - Mr. Doug Lea (Professor of Computer Science at SUNY and published Java author)

Specification Support (2 of 2)

- Macromedia, Inc.
- Mr. Richard Monson-Haefel (Specification Lead)
- Nokia Networks
- Oracle
- SAP AG
- Sun Microsystems, Inc. (with the following comment: ***"Sun is happy to see Groovy proposed as a JSR. Having additional interesting languages for the Java platform seems like a Good Thing!"***)

- On March 29, 2004 this JSR was officially approved(!)

Yet another language?

- Groovy is new, taking best of Ruby and Python and building from there
- Java-like syntax and produces byte-code
- Makes many people happy
- Many programmers feel Java is too verbose, strong typing (i.e. all variables must be declared with an unchangeable type)
- Groovy is being called a "native scripting language" for Java [Strachan]

Groovy and Java together

- Java classes are able to be called by Groovy classes (nice)
- Groovy classes are able to be called by Java classes (just be sure that you are ready with strongly typed Groovy code, so Java doesn't freak out)
- Uncompiled Groovy can be called by Java

Groovy: Nuts and Bolts

- Scripting
  - Groovy scripts can become Java byte-code ("`.class`" files) using the Groovy compiler `groovyc` (not `javac`)
  - Groovy scripts can also be run **un-compiled** by using the groovy engine `groovy`
  - `groovyc` can compile Java (but not 100%)
- Usage syntax:

```
prompt> groovyc <script-name>
```


Nuts and Bolts: Scripts

- Groovy scripts are source-code files that normally end with the "`.groovy`" extension
- Groovy scripts that want to work with other Groovy scripts have to "import" the other Groovy script
- What does Groovy code look like?

Groovy Code

```
["partly sunny" , "partly cloudy"].each{
weather | println "The weather today is
${weather}" }
```

```
The weather today is partly sunny

The weather today is partly cloudy
```


From Java to Groovy (1 of 2)

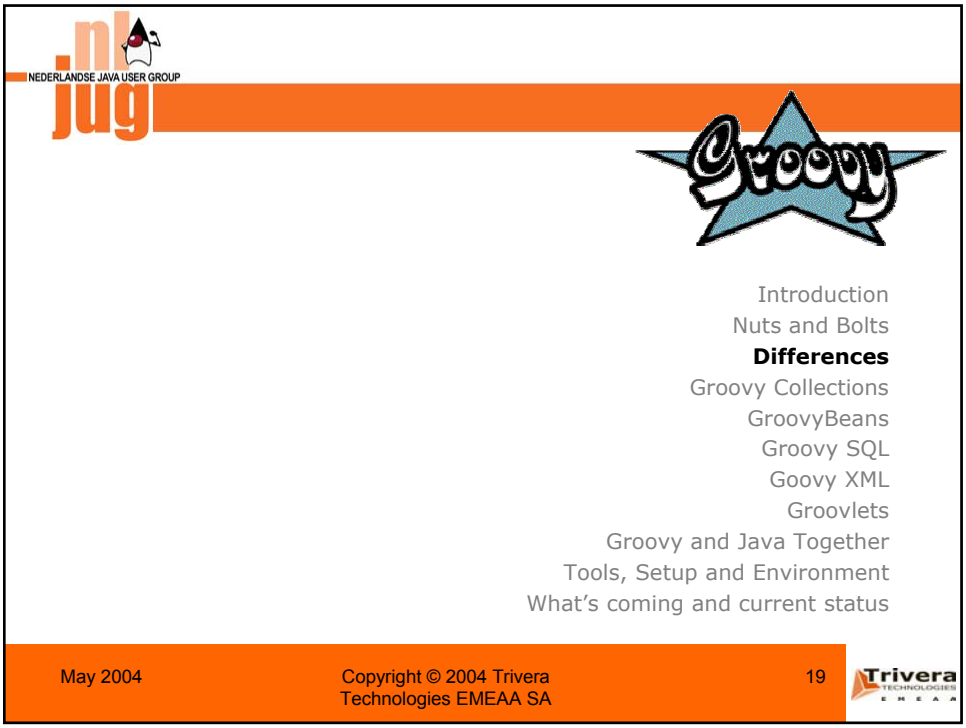
- To a Java developer, Groovy scripts are wild, untamed code
- Statements can appear outside of a method
- Methods can appear outside of a class
- Groovy can also be written more Java-like: declare a class containing variables and methods

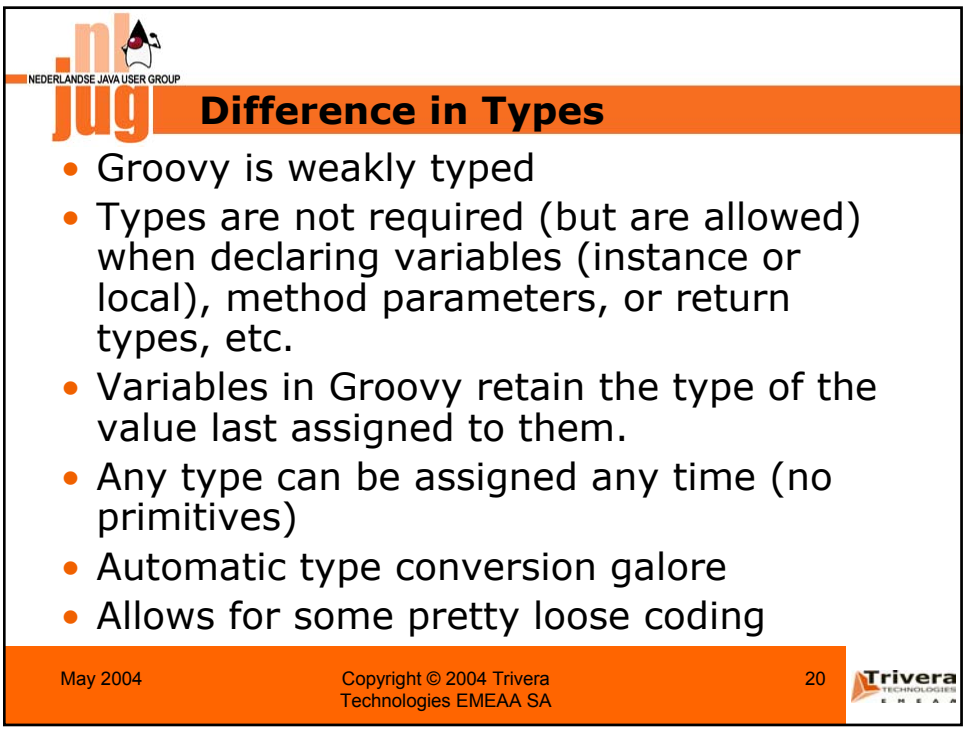
From Java to Groovy (2 of 2)

- Methods not associated with a class become static methods in a class related to Groovy source code.
- Statements outside of a method get put automatically in a `run()` method which is invoked automatically by the `main()` method.
- If there is no declared `main()` method, your Groovy class gets one automatically generated, which simply contains a call to the `run()` method.
- Groovy script therefore can be run as an application.

Groovy slower than Java

- Groovy is slower than Java. Why?
- Instead of explicit constructor calls and method calls, Groovy creates constructor calls, as well as `private` and `protected` method calls using reflection
- Less code for Groovy developers to write, more code is written for you by Groovy





New methods

- Groovy has new versions of some Java classes adding powerful methods
- New methods on the `java.lang.Object`
- New ways of working with strings based on a Groovy string `groovy.lang.GString`

Miscellaneous

- No support currently for Inner Classes
- Three free package imports for all classes: `java.lang.*` as well as `groovy.lang.*` and `groovy.util.*`
- Semicolon usage to terminate a statement is no longer required
- There are new Groovy keywords

Default Access modifiers

- The default access modifier is now `public` for methods (behavior) and attributes with a default access modifier are accessed by `get()` and `set()` methods.
- This encapsulation is done for you
- You can explicitly mark variables `private`

Parenthesis usage

- Under certain circumstances, parenthesis usage around arguments to a method, on the method signature (definition), is no longer required
- Under certain* circumstances, usage of parenthetical arguments must be continued on the method call itself (erratic, see samples on next slide)

*They are working to define this now

Parenthesis usage - Examples

```
// parenthesis here IS required
s = new JavaFromGroovy()
// parenthesis here NOT required
x = s.getMessage
// parenthesis here IS required
println x.getMessage()
// parenthesis here NOT required
s.setMessage "Happy?"
// parenthesis here NOT required
y = s.getMessage
// parenthesis here IS required
println y.getMessage()
```


Differences in Strings: The `GString`

- Groovy Strings can be created with double quotes or single quotes
- Strings can span multiple lines without the need for the plus "+" concatenation operator
- Groovy string `GString` has added new methods

Operator Overloading

- There is limited support for operator overloading, and Groovy introduces another new operator:
  - ===
- Operators call predefined methods, which can be overloaded by the developer
- Some examples on next slide

Comparison Operator Overloading[3]

Operator	Method
a == b	a.equals(b)
a != b	! a.equals(b)
a===b	a==b (from Java)
a<=>b	a.compareTo(b)
a>b	a.compareTo(b)>0
a>=b	a.compareTo(b)>=0
a<b	a.compareTo(b)<0
a<=b	a.compareTo(b)<=0

3. From http://groovy.codehaus.org/operators.html

Difference in returns

- Keyword: `return` usage is, under certain circumstances, no longer required:

"In methods that return a value, if the last statement before the closing brace is reached, its value is returned. In the future, Groovy may be changed to return value of last statement evaluated." [Volkmann][4]

4. From: http://www.ociweb.com/jnb/jnbFeb2004.html

New Loops

Four new looping techniques:

- `each`
- `times`
- `upto`
- `step`

```
["partly sunny" , "partly cloudy"].each{
weather | println "The weather today is
${weather}" }
```

`switch()` Capabilities (1/2)

- A Groovy `switch()` block is not limited by type
  - Java can only `switch()` on these primitive types: `byte`, `short`, `int` and `char`
- There are no primitives in Groovy so you can switch on any type whatsoever
- This is cool and very powerful

`switch()` Capabilities (2/2)

- generates `if` statements in compiled code, using `compare` methods
- case value can be a regular expression (regex) and it will attempt to match it against switch value
- Should work with collections as well
- Example on next slide

Groovy `switch()` Example

```
x = 15
result = ""
   switch (x) {
              case "mom":
                     result = "hi mom"
                     break
       case 12..30:
                     result = "range"
                     break
       case Integer:
                     result = "integer"
                     break     } // this code was tested
   and worked
println result
assert result = = "range"
```


Groovy: Range

- Groovy has a `Range` type
- The interface `groovy.lang.Range` extends `java.util.AbstractList` to allow you to create a range of objects or values
- Powerful
- Two kinds of `Range` implementations
  - `groovy.lang.ObjectRange`
  - `groovy.lang.IntRange`
- You can make your own ranges

Groovy: `Closure` (1 of 3)

- Groovy has a `Closure` type
- Written as a bit of a code-snippet
- Closure code gets compiled into a class that extends `groovy.lang.Closure`
- Snippet code you write is used as the implementation of the method named `call()` which takes one parameter named `it` by default, but you can change it on the fly
- Sounds confusing? No need to worry.

Groovy: `Closure` (2 of 3)

- Since `Closure` is a class, you can write methods that take a `Closure` as a parameter
- Also calling a method on a Java object returns a subtype of Closure called `org.codehaus.groovy.runtime.MethodClosure`
- Figuring out how to work properly with `Closure` and `MethodClosure` is half the battle with Groovy

Groovy: Closure (3 of 3)

```
x = { println("Good to meet you ${it}") } // closure
   is here
/* Think of the above block as being cut and pasted
   into the call(Object it) method on a sub-class of
   type Closure
Why not the method call(String it) because any type
   is allowed to be passed over…
 */
x('Klaasjan Tukker')
// equivalent to x.call('Klaasjan Tukker')
```

```
Good to meet you Klassjan Tukker
```

May 2004 Copyright © 2004 Trivera Technologies EMEAA SA 37

Introduction
Nuts and Bolts
Differences
Groovy Collections
GroovyBeans
Groovy SQL
Goovy XML
Groovlets
Groovy and Java Together
Tools, Setup and Environment
What's coming and current status

May 2004 Copyright © 2004 Trivera Technologies EMEAA SA 38

Groovy Collections: `List` & `Map`

- There is a Groovy `List`, which is an instance of a `java.util.ArrayList` but with a more friendly syntax
- There is a Groovy `Map`, which is an instance of a `java.util.HashMap` but with a happy syntax
- Here are two samples of Groovy `List` and Groovy `Map`:

Groovy Collections: `List`

```
//in Java we write…
List myList = new ArrayList();
myList.add(new Integer(56));
myList.add(new Integer(122));
myList.add(new Integer(7000));

//in Groovy to do the same thing…
myList = [56, 122, 7000]
```

Groovy Collections: Map

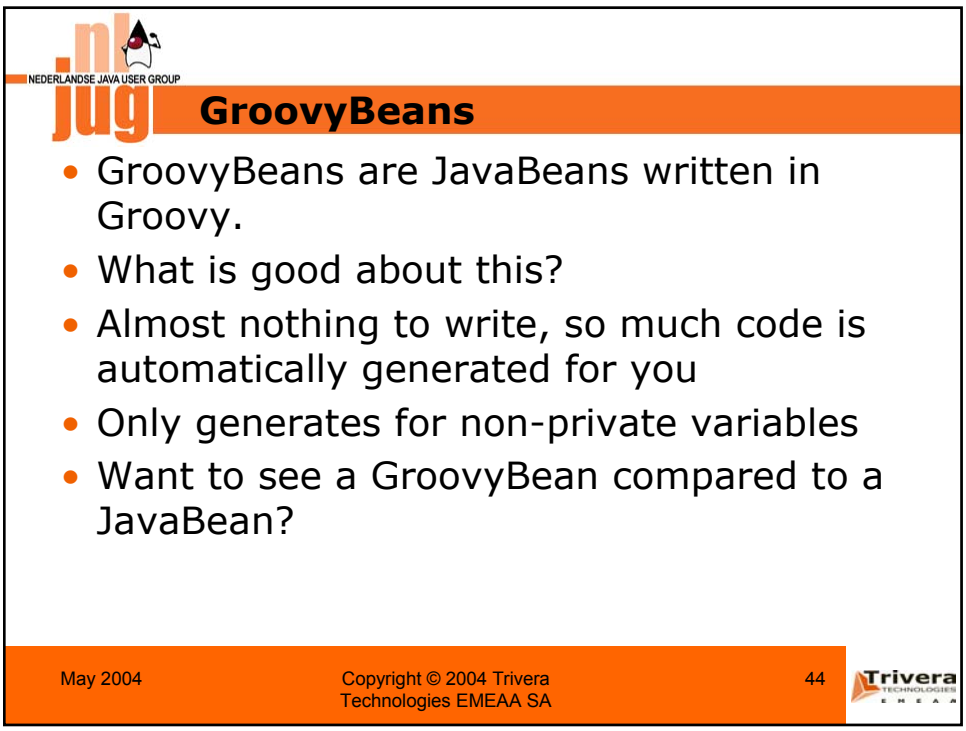
```
//in Java we write…
Map myMap = new HashMap();
myMap.put("realized" , new Integer(56));
myMap.put("cirlce" , new Integer(122));
System.out.println(myMap.get("circle"));

//in Groovy to do the same thing…
myMap = ['realized' :56, 'cirlce' :122]
print myMap['cirlce']
```


Groovy Collections: List

Also Groovy has added new methods to the `java.util.List` interface:

- `count()`
- `immutable()`
- `intersect()`
- `join()`
- `sort()`
- `min()`
- `max()`
- `reverse()`
- `plus()`
- `minus()`



GroovyBeans

```
//Java
class Writer{
   private String book;
   private String publisher;
   public void setBook(String
     book = b;
   }
   public String getBook(){
    return book;
   }
   public void setPublisher(String p){
     publisher = p;
   }
   public String getPublisher(){
    return publisher;
   }
}
```

```
//Groovy
class Writer {
   String book
   String publisher
}
```

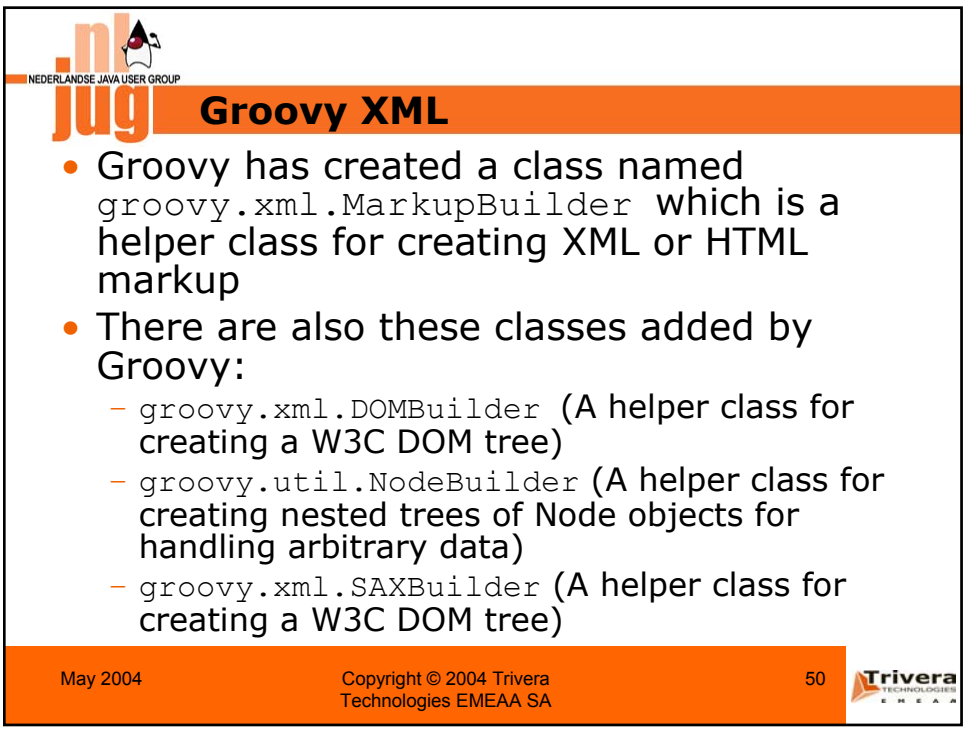

Groovy SQL

- Groovy provides a class named `groovy.sql.Sql` which is supposed to make JDBC work easier
- Does it? It has about forty methods to work with
- `Connection` is established (if you provide a `DataSource`) and is put back in pool after use
- Existing `Connection` can be sent to `Sql` object
- Creates a `Statement` & retrieves `ResultSet`
- All you do is provide raw SQL code

Groovy SQL: source

```
datasource=//get your DataSource (e.g. JNDI)
doesItAll = new Sql(datasource)
doesItAll.eachRow("SELECT fname, lname,
  email FROM contacts") { currentRow |
  println "Name: ${currentRow.fname}
  ${currentRow.lname} email:
  ${currentRow.email}
} // this outer block is the Closure code

/* eachRow() method calls Closure with each
  row of ResultSet.  This code has not been
  tested */
```



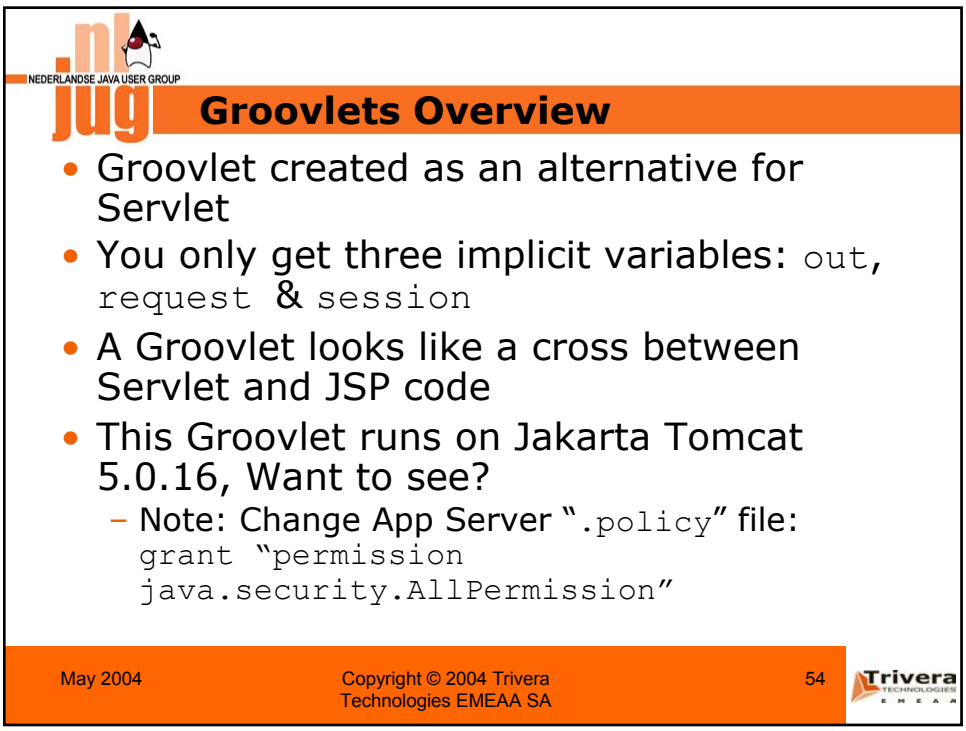
Groovy XML Example (1/2)

```
import groovy.xml.MarkupBuilder;
import java.io.*;
pw = new PrintWriter (…)
mb = new MarkupBuilder(pw)
mb.root() {
  garden(id:'Wakehurst Place') {
    bed(area:'Rose Garden') {
      flower(name:'Persian Yellow' , species:'Rosa
  foetida persiana')
      flower(name:'Mountain Rose' , species:'Rosa
  woodsii')
      flower(name:'Venusta Pendula' , species:'Rosa
  arvensis hybrid')
    }
  }
}
```

Groovy XML: Example (2/2)

- result:

```
<root>
  <garden id='Wakehurst Place'>
    <bed area='Rose Garden'>
      <flower name='Persian Yellow' species='Rosa
foetida persiana' />
      <flower name='Mountain Rose' species='Rosa
woodsii' />
      <flower name='Venusta Pendula' species='Rosa
arvensis hybrid' />
    </bed>
  </garden>
</root>
```



Groovlets Example

```
/* Thanks go to Mark Volkmann for helping me get
   Groovlets working! */
out.println <<<EOS
<html>
  <head>
     <title> Groovlets for JSPRING </title>
  </head>
  <body>
     <p> JSPRING is on May 11, today is: ${new
  java.util.Date()} </p>
     <p> "Life I love you, all is groovy." </p>
  </body>
</html>
EOS //the "here-doc" terminator EOS has to be all
   the way left (column 1)
```


Groovlets Deployment (1/2)

- If deploying with a WAR file:
  - Put your Groovlet source code in the top level directory of your web application
  - Must register `groovy.serlvet.GroovyServlet` in the web.xml deployment descriptor
  - `GroovyServlet` is responsible for compiling Groovlets, a job which is usually done by the Application Server

Groovlets Deployment (2/2)

- Set the `<servlet-mapping>` elements in `web.xml` appropriately
- Make sure that the `WEB-INF/lib` directory has your "`groovy`*`XXX`*`.jar`" and "`asm`*`XXX`*`.jar`" files in it

- Deploy your web application!

Groovy and Java work together

- Groovy scripts can be written (and not compiled) to invoke Java code
- It works
- It looks funny to a Java developer
- Here is an example: `InvokeJava.groovy` source code

Groovy -> Java: Example (1/3)

- The Groovy source

```
s = new JavaFromGroovy()
x = s.getMessage
println x.getMessage()
s.setMessage "Happy?"
y = s.getMessage
println y.getMessage()
```

Groovy -> Java: Example (2/3)

- The Java Code

```
public class JavaFromGroovy {
  private String m_message =
    "This came from a Java object at runtime";
  public String toString() {
    return "JavaFromGroovy: message= " +
                getMessage();
  }
  public String getMessage() {
    return m_message;
  }
  public void setMessage(String mes) {
    m_message = mes;
  }
}
```


Groovy -> Java: Example (3/3)

- Running

```
prompt> groovy InvokeJava.groovy

This came from a Java object at run time

Happy?
```

Java -> Groovy Class (1/2)

- You can also have Java invoke your Groovy just add a `main()` method to Java code like this:

```
public static void main(String[] args) {
  GroovyFromJava groovyObject =
         new GroovyFromJava();
  groovyObject.grooveBaby("I did it!");
}
```


Java -> Groovy Class (2/2)

- The Groovy source:

```
class GroovyFromJava {
  grooveBaby (x) {
    println x
  }
}
```

Java -> Groovy Script

- Dynamically loading Groovy:

```
ClassLoader parent;
GroovyClassLoader loader;
parent = getClass().getClassLoader();
loader = new GroovyClassLoader(parent);
File file = new File("SomeScript.groovy");
Class gClass = loader.parseClass(file);
GroovyObject gObject;
gObject = (GroovyObject) gClass.newInstance();
Object[] args = {};
gObject.invokeMethod("run", args);
// run is the only non-static method
```


Compilation

- You can compile a Groovy source code file with `groovyc` from the command line
- It creates a `.class` byte-code file with the same name as the source-code file
- Most Groovy source ends with a `.groovy` extension
- If it works, it returns this: `"[.]"` on the command line.

Groovy Shell

- There is a Groovy Shell, which is invoked by typing `groovysh` at the command line
- It responds with:

```
[.]
Let's get Groovy!
=================
Version 1.0-beta-4 JVM: 1.4.1_01-b01
Type 'exit' to terminate the shell
Type 'help' for command help
```

Groovy Shell

```
1> println 'Then you can type Groovy code here line by line'
2> println 'And it will only be compiled and run when you type:'
3> execute
...
```

or more advanced code like this:

```
1> ['partly sunny' , 'partly cloudy'].each { weather | println
"The weather today is ${weather}" }
2> execute
The weather today is partly sunny
The weather today is partly cloudy
```


Groovy Set Up Issues (1 of 2)

How to download Groovy (free):

- Go to http://groovy.codehaus.org/download.html and click the link "this site" which takes you to http://dist.codehaus.org/groovy/distributions/
- Pick a distribution, and download it.
  - this presentation uses: `groovy-1.0-beta-4-tar.gz`

Groovy Set Up Issues (2 of 2)

- Unzip the download
- Make sure you set the environment:
  - GROOVY_HOME = *<Groovy Install Dir>*
  - PATH = *<GROOVY_HOME>*\bin
- The Java CLASSPATH requires references to the groovy*XXX*.jar and asm*XXX*.jar files which reside in the *<GROOVY_HOME>*\lib

Introduction
Nuts and Bolts
Differences
Groovy Collections
GroovyBeans
Groovy SQL
Goovy XML
Groovlets
Groovy and Java Together
Tools, Setup and Environment
What's coming and current status

Groovy Features Coming Soon?

I wrote to Strachan, and he wrote back on (April 27, 2004):

LW: Since Groovy now has world wide attention, what are the top three major features that you are going to try to add, issues that need to be changed about Groovy, for the near future, like immediately?

JS: Getting the compile / bytecode generation solid & free from bugs is top on my list. Also having a cast iron specification is high up there.
In terms of features I want that are not yet implemented in order of priority these are my personal favourites

- **JDK 1.5 metadata** *// LW note: (part of the JSR-175)*
- **generators** *//LW note: automatic code generation*
- **mixins / AOP features** *//LW note: Aspect Oriented Programming*

See also: http://wiki.codehaus.org/groovy/MixinSupport and http://wiki.codehaus.org/groovy/AspectOrientedProgramming

Groovy Features Coming Soon?

Already requested (not yet present) but coming soon, we hope:

- Well commented JavaDocs ☺
- See the Groovy (long) page of requests: http://wiki.codehaus.org/groovy/WishList

Perspective: Problems

As of this writing, **"Groovy 1.0-beta-4"** only had four problems that are marked "Unresolved" as follows:

1. [Groovy-200] (Since February 17, 2004): "at runtime, groovy incorrectly fails when choosing between signature-specialized versions of a method."

- Which means: Method overloading in Groovy has trouble under certain circumstances

Perspective: Problems

2. [Groovy-105] (Since January 6, 2004): "Parent class print method is wrongly picked up."

- Which means: When combining method overloading and inheritance, sometimes Groovy gets confused

Perspective: Problems

3. [Groovy-305] (Since March 20, 2004): single character strings should be coerced to characters if it helps.

- Which means: When using strings in Groovy, and the `replace()` method `MissingMethodException` can be thrown under certain circumstances

Perspective: Problems

4. [Groovy-134] (Since January 16, 2004): "New 'def' has short term memory in shell."

- Which means: Certain Groovy code throws a `MissingMethodException` but it is not clear why

Perspective: Weakness

- Still early, but if the Groovy language is solid enough for sixteen organizations (and rare individuals) to vote "**yes**" that is good enough
- Groovy JavaDocs not great (after years of Java APIs it is too hard to understand what is going on in Groovy) Are many of the details of how Groovy works still in Strachan's head?
- Descriptions of classes, methods, etc. are either missing entirely, or not very helpful. But, at least there *are* docs! It is a wild horse, but elegant
- No books yet either, hard to find code examples, have to rely on web articles for now. Look for books coming soon.

Groovy Summary

- Groovy works and it is a versatile language (do we have to say "agile" again?)
- There are many things that you can do fast and it is still an extremely young language
- Groovy is maturing rapidly and has tremendous industry support
- Expect to be hearing a lot more

Questions

- Does anyone have any Groovy questions?

Groovy Resources (1 of 2)

- Official Groovy web site (Strachan) at http://groovy.codehaus.org or email: GroovyTeam@Codehaus.org
- James Strachan PowerPoint Groovy presentation "Groovy, Making Java more funky" (August 2003) at http://codehaus.org/~jstrachan/Groovy/html/siframes.html

Groovy Resources (2 of 2)

- Object Computing (excellent) article by Mark Volkmann at http://www.ociweb.com/jnb/jnbFeb2004.html
- Groovy, An Object Oriented Dynamic Language for the JVM (Groovy SeaJUG) at http://www.sauria.com/presentations
- Groooooovy Babe: Jazzing Up Plain Old Java Scripting Power for Java - Do More With Less (Lines of Code) by Gerald Bauer (March 2004) at: http://viva.sourceforge.net/talk/jug-mar2004/slides.html

Thank You

Laurent Weichberger
Senior Technical Trainer
laurent@triveratech.com

Raphaël Parrée
Director Courseware & Consulting Services
Raphael@triveratech.com

www.triveratech.com

Enjoy Education

Trivera Technologies EMEAA
Europe, Middle East, Africa & Asia Pacific
P +33 442 163594
F +33 153 013205
Espace Cézanne
14 Parc Golf du Club
13856 Aix-en-Provence (France)

Trivera Technologies
USA & Canada
P +1 609 953 1515
F +1 609 953 6886
135 Meeshaway Trail
Medford Lakes, NJ08055 (USA)