Groovy Language Overview

Rapid tour of Groovy
Differences from Java
Features used by Grails

@ Robert Rodini
Groovy Overview

• Every Groovy type is a subtype of java.lang.Object
• Every Groovy object is an instance of a type
  • no primitive types, everything is an object
  • all operators are method calls
Groovy Overview

• Comments
  • same as Java
  • also first line comment #!

• Groovy syntax is not a superset of Java but there's incredible overlap
  • general packaging mechanism
  • statements (except Java do...while)
  • class and method definitions (except nested classes)
  • operators, expressions, assignments
  • exception handling
Groovy Overview

- **Groovy automatic imports**
  - groovy.lang.*
  - groovy.util.*
  - java.lang.*
  - java.util.*
  - java.net.*
  - java.io.*
  - java.math.BigInteger
  - java.math.BigDecimal

- **Example**

```java
java.net.URLEncoder("a b")
```

@ Robert Rodini

- **Java:**

```java
java.net.URLEncoder("a b")
```
Groovy Overview

- `==` operator ('equals' method) on objects
- Groovy tests `value` equality
- Java tests `reference` identity

Demo: ShowEquality

- Assertions
  - first class statement in Groovy. Always active
  - Java requires run-time option to activate
### Groovy Truth

<table>
<thead>
<tr>
<th>Runtime type</th>
<th>Evaluation criteria for truth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>boolean value is 'true'</td>
</tr>
<tr>
<td>Matcher</td>
<td>matcher has a match</td>
</tr>
<tr>
<td>Collection</td>
<td>collection is non-empty</td>
</tr>
<tr>
<td>Map</td>
<td>map is non-empty</td>
</tr>
<tr>
<td>String, GString</td>
<td>string is non-empty</td>
</tr>
<tr>
<td>Number, Character</td>
<td>value is non-zero</td>
</tr>
<tr>
<td>None of the above</td>
<td>object reference is non-null</td>
</tr>
</tbody>
</table>

Note: Any object can be tested for truth by calling its `asBoolean()` method

@ Robert Rodini
Groovy Overview

• Strings
  • 'I am a string' // similar to Java string
  • "I am a string" // GString - supports
    // variable interpolation
  • /I am a string/ // good for regex patterns

• Extended strings (here documents)
  • '''
    for score and seven years ago
  ...
  ...'
  • """
    for score and seven years ago
  ..."""
Groovy Overview

- **GString**
  - string bounded by double quotes
  - permits substitution of variable values
  - **Example:** "The winning number is $n"

- **Regular expressions**
  - string bounded by forward slashes
  - interpreted as a regular expression in context
  - **Example:** `assert '12345' =~ /\d+/`
Groovy Overview

• Numbers are objects
  • Example:
    ```java
    int x = 1
    assert x instanceof Integer
    ```

• Groovy has its own coercion rules
  • Similar but different from Java
  • Attempts to be more "natural"
  • Example:
    ```java
    def x = 1/2
    assert x instanceof BigDecimal
    assert x == new BigDecimal(0.5)
    ```
Groovy Overview

• Lists
  • Similar to array literal, but bounded by square brackets
  • Instances of `java.util.List`
  • Indexing is zero-relative
  • Example:
    ```
    ['', 'I', 'II', 'III', 'IV', 'V', 'VI', 'VII', 'VIII', 'IX', 'X']
    ```

• Maps
  • Key/value binding
  • Similar to a list but : separates key and value
  • Example:
Groovy Overview

• Ranges
  • Language literal and language class
  • Example: `def x = 1..10`

• Groovy `for` loop
  • `for(i in x){println i} // block here`
  • `x` is anything that can be iterated

@ Robert Rodini
Groovy Overview

- Closures
- Code as Objects
- List of statements between curly brackets
- When used in an iterative method a closure functions like a "callback" function

- Example:

```groovy
def r = [1,2,3,4,5,6,7,8,9,10]
r.each { println 2 * it } // closure here
```

@ Robert Rodini
Groovy Overview

- Checked exceptions don't have to be caught
  - can still use `try/catch/finally`
  - otherwise checked exception becomes runtime exception
Groovy Overview

• Syntactic sugar
  • `?.` null-safe operator
    • `obj?.member ≡ (obj==null) ? null : obj.member`
  • `?:` Elvis operator
    • `complex_exp ?: 1 ≡ complex_exp ? complex_exp : 1`
  • `<=>` spaceship operator
    • `a <=> b ≡ a.compareTo(b)`

• Less syntax
  • semicolons optional
  • `return` statement optional
Groovy Overview

• **GroovyBean**
  - is a JavaBean defined in Groovy
  - Groovy generates `public` getters/setters
  - Groovy translates property references to getter/setter calls

• Bean instantiation has special syntax
  - use a list of named arguments matching the property names (map here)

• Example:

```java
format = new SimpleDateFormat(
    lenient: false,
    numberFormat: NumberFormat.getIntegerInstance(),
    timeZone: TimeZone.getTimeZone('EST'))
```

@ Robert Rodini
Groovy Overview

- Groovy classes / Java classes
  - Use `groovyc` to compile Groovy code to Java byte code (.class files here)
  - A Groovy class can be called from Java provided it is compiled
- Groovy script
  - Uses groovy class loader to generate class objects (instances of `java.lang.Class`) that load into the JVM on the fly (no .class files here)
Libraries and JavaDocs

• Java SE library

• Groovy Development Kit library (GDK)
  • Extends the JDK classes by funnelling all method calls through a device called MetaClass
  • Homogenizes the JDK libraries
  • Example: size method in GDK

• Groovy Libraries

• Pure Groovy classes

@ Robert Rodini
Where to get Groovy?

- Groovy download
  - groovy shell, groovy console, groovy compiler, libraries
  - http://groovy.codehaus.org/
- Spring Tool Suite (STS) has Groovy and Grails plugins bundled
  - STS is Eclipse with enhancements
- Eclipse plugin
  - http://dist.springsource.org/release/GRECLIPSE/e3.5/

Other IDEs
How to run Groovy

- groovysh
  - Command line shell
- groovyConsole
  - Graphical interface (written in Groovy using a builder)
- groovy
  - Script interpreter (really custom class loader)

Demo: Fibonacci via groovyConsole, Fibonacci via groovy class loader, Fibonacci compiled class via java

@ Robert Rodini
Real World Example

- Q: What do you do when you have > 2,000 Hudson jobs to modify?
- A: Write a Groovy script.
- Modify Hudson job's config.xml
  - Parse the xml
  - Find the node to be updated
    - Delete the old node
    - Insert new child nodes
  - Output the xml (pretty print)
Modify config.xml

• Before (excerpt shown)

```xml
<builders>
  <hudson.tasks.Shell>
    <command>$CI_HOME/build/release.sh
      -Dbuild=milestone
      -Dworkdir=$WORKSPACE/trunk
      -Dprofile=quality-checks-1.5.1
    </command>
  </hudson.tasks.Shell>
</builders>
```
Modify config.xml

• After (excerpt shown)

```xml
<builders>
  <hudson.tasks.Maven>
    <targets>-B vgi-release:prepare
    -DpreparationGoals="clean install"
    </targets>

    <mavenName>Maven 2.1.0</mavenName>

    <properties>maven.repo.local=$M2_REPO$EXECUTOR_NUMBER</properties>
  </hudson.tasks.Maven>

  <hudson.tasks.Maven>
    <targets>-B vgi-release:perform
    -Dgoals=deploy
    -Darguments="-s $HUDSON_DATA/settings-releases.xml"
    </targets>

    <mavenName>Maven 2.1.0</mavenName>

    <properties>maven.repo.local=$M2_REPO$EXECUTOR_NUMBER</properties>
  </hudson.tasks.Maven>
</builders>
```
Modify config.xml

Demo: ConfigEditXml.groovy