TO THE NEW™ DIGITAL

Groovy For Java Developers

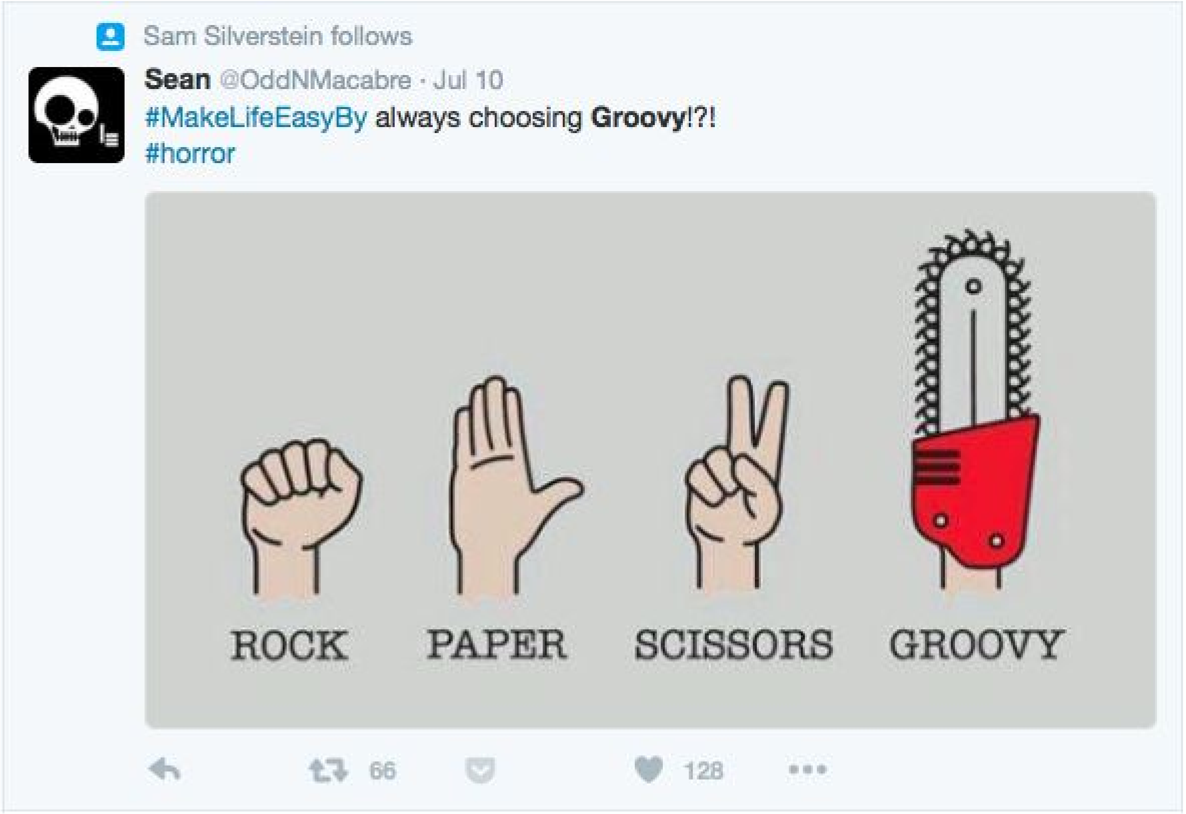
About Me

Puneet Behl
Associate Technical Lead
TO THE NEW Digital
puneet.behl@tothenew.com
GitHub: https://github.com/puneetbehl/
Twitter: @puneetbhl
LinkedIn: https://in.linkedin.com/in/puneetbhl

Agenda

- Background
- From Java to Groovy
- Groovy Closure, Traits
- Static Type Checking and Compilation
- How can I get started with Groovy?
- Groovy Ecosystem

What is Groovy?


Sam Silverstein follows

Sean @OddNMacabre · Jul 10

#MakeLifeEasyBy always choosing **Groovy**!?!
#horror

What is Groovy?

What is Groovy?

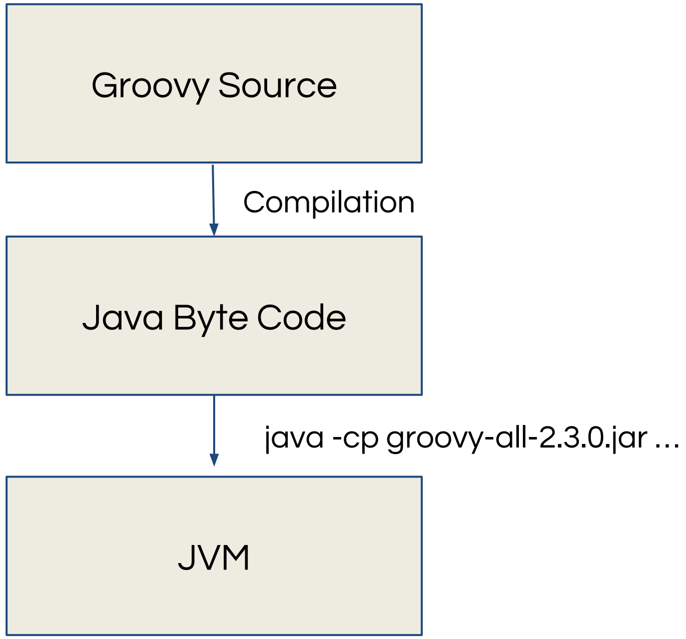
Groovy is:

- an alternate language which run on JVM
- dynamic language - add behavior to existing library at runtime

Groovy ...

But, also supports **static type checking** and **static compilation**

Groovy is compiled



How to setup Groovy?

http://www.groovy-lang.org

How to setup Groovy?

- Using Binary

  Download the binary from http://www.groovy-lang.org

  Install JDK > v1.6

  Set GROOVY_HOME to point to the installation.

  Add GROOVY_HOME/bin to the path variable.

How to setup Groovy?

- Using Binary

  Download the binary from http://www.groovy-lang.org

  Install JDK > v1.5

  Set GROOVY_HOME to point to the installation.

  Add GROOVY_HOME/bin to the path variable.

- Using SDKMAN

  [JDK should be present]

  curl -s get.sdkman.io | bash

  source "$HOME/.sdkman/bin/sdkman-init.sh"

  sdk install groovy

How to setup Groovy?

- Using Binary

  Download the binary from http://www.groovy-lang.org

  Install JDK > v1.6

  Set GROOVY_HOME to point to the installation.

  Add GROOVY_HOME/bin to the path variable.

- Using SDKMAN

  [JDK should be present]

  curl -s get.sdkman.io | bash

  source "$HOME/.sdkman/bin/sdkman-init.sh"

  sdk install groovy

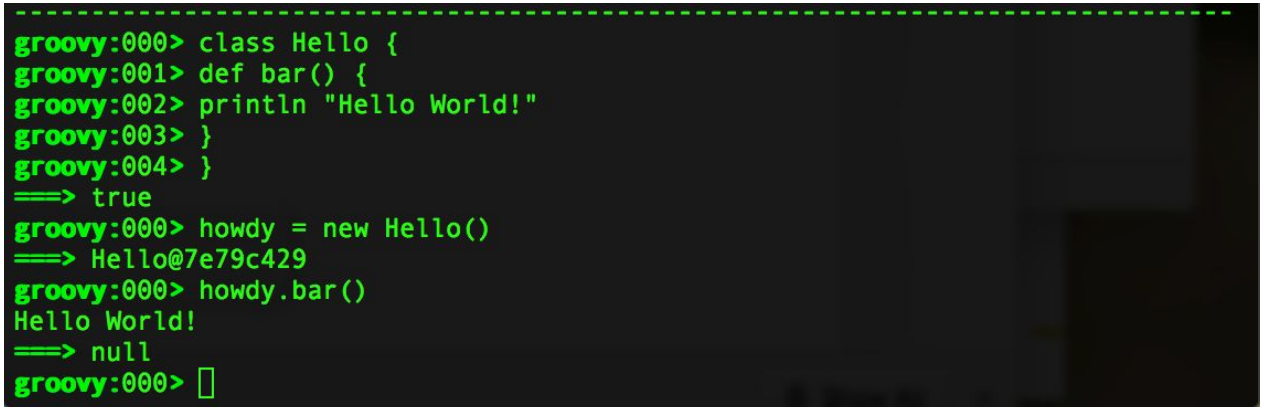
- Windows User

  Download and install Babun - a Windows shell you will love.

  Now, you can install SDKMAN & Groovy.

Groovy : Shell

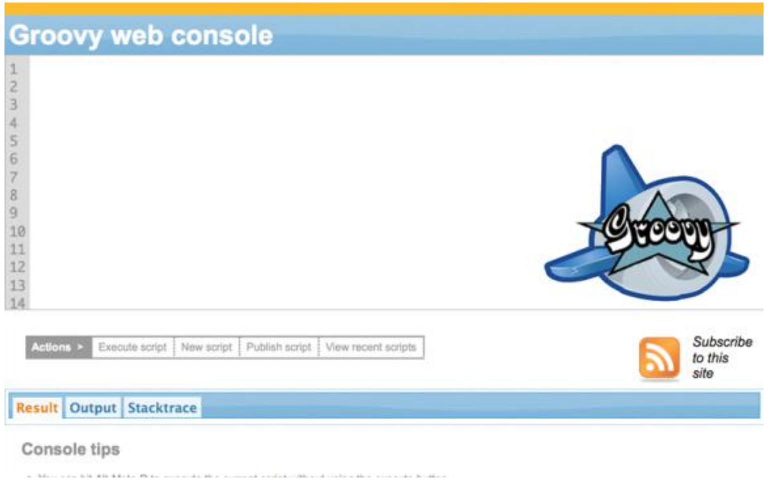
- Open a terminal window and type “groovysh”.
- Command line.
- It allows easy access to evaluate Groovy expressions, and run simple experiments.

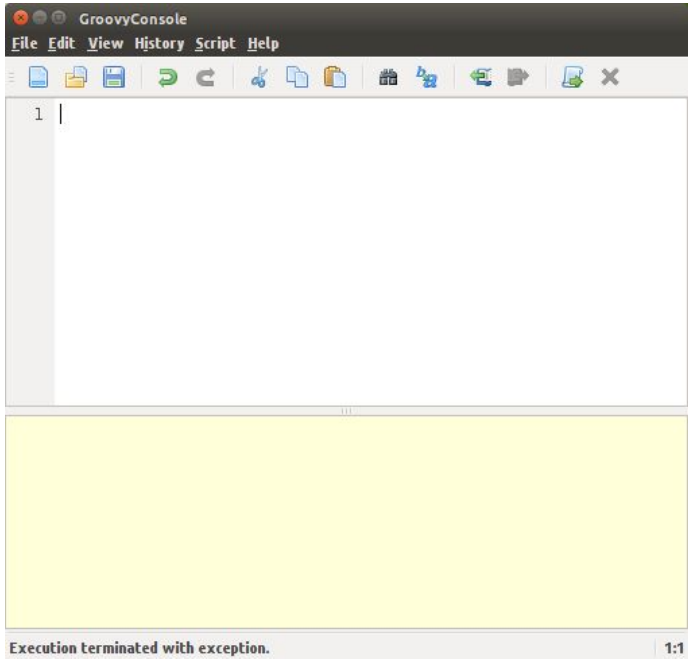


```
--------------------------------------------------------------------------
groovy:000> class Hello {
groovy:001> def bar() {
groovy:002> println "Hello World!"
groovy:003> }
groovy:004> }
===> true
groovy:000> howdy = new Hello()
===> Hello@7e79c429
groovy:000> howdy.bar()
Hello World!
===> null
groovy:000> 
```

Groovy Console

- Allows a user to enter and run Groovy scripts
- You can save your scripts, open existing scripts etc
- Groovy Web Console
  http://groovyconsole.appspot.com

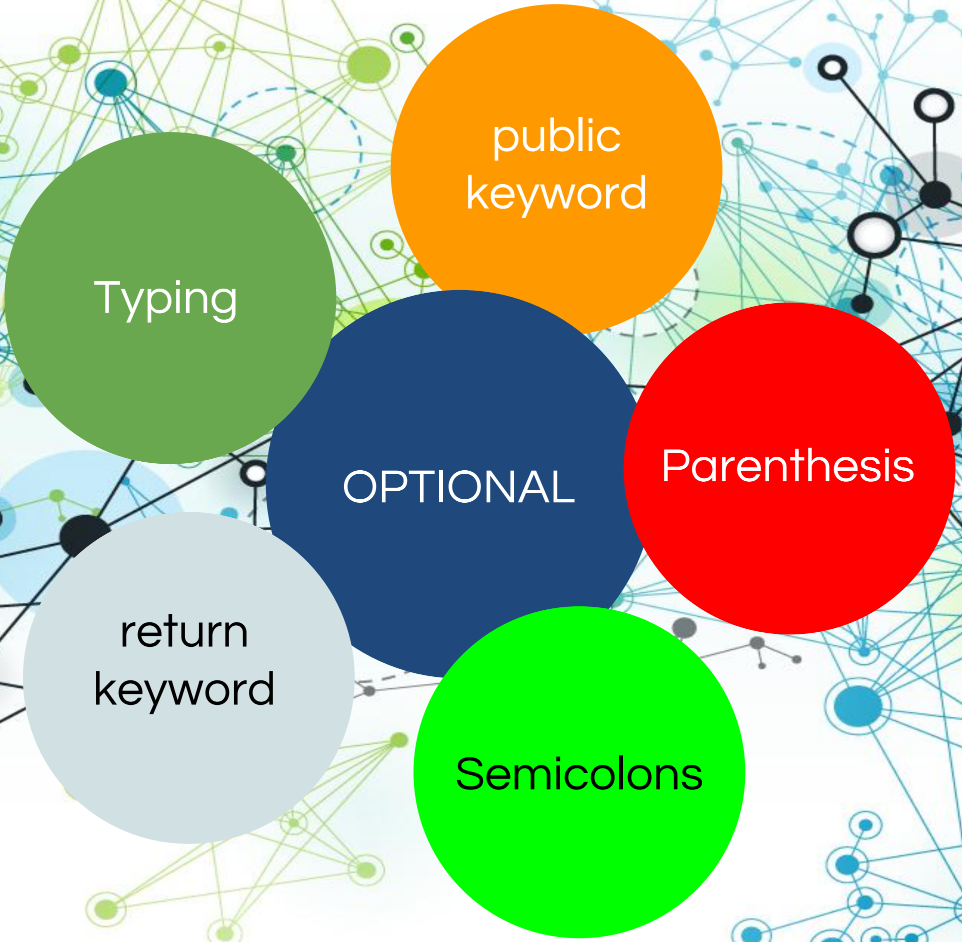




Execution terminated with exception.

Why Groovy?

- Powerful - *closure, traits, metaprogramming, DSL etc.*
- Optional - *no more ceremonies*
- Dynamic - *decides at runtime*
- Easy to learn - *syntax like Java*
- Simple and Expressive


public keyword
Typing
OPTIONAL
Parenthesis
return keyword
Semicolons

Optional

```
public class Greeter {

   private String place;

   public String getPlace() {
        return place;
   }

   public void setPlace(String place){
        this.place = place;
   }

   public String greet(String name) {
      return "Hello "+ name + "! Welcome to "+
place;
    }
```

```
public static void main(String[] args) {
    Greeter greeter = new Greeter();
    greeter.setPlace("GR8Conf US");
    System.out.println(
        greeter.greet("Everyone")
     );
   }
}
```

Optional ...

Semicolons

```
public class Greeter {

   private String place;

   public String getPlace() {
        return place;
   }

   public void setPlace(String place){
        this.place = place;
   }

   public String greet(String name) {
      return "Hello "+ name + "! Welcome to "+
place;
    }
```

```
public static void main(String[] args) {
    Greeter greeter = new Greeter();
    greeter.setPlace("GR8Conf US");
    System.out.println(
        greeter.greet("Everyone")
    );
   }
}
```

Optional ...

```
public class Greeter {

   private String place

   public String getPlace() {
        return place
   }

   public void setPlace(String place){
        this.place = place
   }

   public String greet(String name) {
      return "Hello "+ name + "! Welcome to "+
place
    }
```

```
public static void main(String[] args) {
    Greeter greeter = new Greeter()
    greeter.setPlace("GR8Conf US")
    System.out.println(
        greeter.greet("Everyone")
     )
   }
}
```

verbose Java properties

Optional ...

```
public class Greeter {

   String place
```

```
Greeter greeter = new Greeter()
greeter.setPlace("GR8Conf US")
System.out.println(
    greeter.greet("Everyone")
)
```

public keyword

```
   public String greet(String name) {
      return "Hello "+ name + "! Welcome to "+
place
    }
```

Optional ...

```
class Greeter {

   String place
```

```
Greeter greeter = new Greeter()
greeter.setPlace("GR8Conf US")
System.out.println(
    greeter.greet("Everyone")
 )
}
```

```
   String greet(String name) {
      return "Hello "+ name + "! Welcome to "+
place
    }
```

optional typing

property notation

Parenthesis

println shortcuts

return keyword

Optional ...

```
class Greeter {

   String place
```

```
def greeter = new Greeter()
greeter.place = "GR8Conf US"
println greeter.greet("Everyone")
```

```
   String greet(String name) {
      "Hello "+ name + "! Welcome to "+ place
    }
```

Optional ...

```
class Greeter {

   String place




   String greet(String name) {
       "Hello $name! Welcome to $place"
    }
```

GString

```
def greeter = new Greeter()
greeter.place = "GR8Conf US"
println greeter.greet("Everyone")
```

Optional ...

Let’s refactor the white space...

Optional ...

```
class Greeter {
    String place
    String greet(String name) {
        "Hello $name! Welcome to $place"
    }
}

def greeter = new Greeter()
greeter.place = "GR8Conf US"
println greeter.greet("Everyone")
```

Demo

Let’s say you want parse an XML...

Let’s say you want parse an XML...

```
<langs type="current">
  <language>Java</language>
  <language>Groovy</language>
  <language>Scala</language>

<language>JavaScript</language>
</langs>
```

From Java to Groovy - Parsing an XML in Java

From Java to Groovy - Parsing an XML in Java

```java
import org.xml.sax.SAXException;
import org.w3c.dom.*;
import javax.xml.parsers.*;
import java.io.IOException;

public class ParseXml {
  public static void main(String[] args) {
    DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
    try {
      DocumentBuilder db = dbf.newDocumentBuilder();
      Document doc = db.parse("src/languages.xml");

      //print the "type" attribute
      Element langs = doc.getDocumentElement();
      System.out.println("type = " + langs.getAttribute("type"));

      //print the "language" elements
      NodeList list = langs.getElementsByTagName("language");
      for(int i = 0 ; i < list.getLength();i++) {
        Element language = (Element) list.item(i);
        System.out.println(language.getTextContent());
      }
    }catch(ParserConfigurationException pce) {
      pce.printStackTrace();
    }catch(SAXException se) {
      se.printStackTrace();
    }catch(IOException ioe) {
      ioe.printStackTrace();
    }
  }
}
```

From Java to Groovy - Parsing an XML

```
def langs = new XmlParser().parse("/Users/puneet/groovy/gr8us/src/languages.xml")
println "Type: ${langs.attribute('type')}"
langs.each { println it.text() }
```

From Java to Groovy - Parsing an XML

```
def langs = new XmlParser().parse("/Users/puneet/groovy/gr8us/src/languages.xml")
println "Type: ${langs.attribute('type')}"
langs.each { println it.text() }
```

Now, I want to generate an XML...

From Java to Groovy - Generate an XML

```
def xml = new groovy.xml.MarkupBuilder()
xml.langs(type: "current") {
    language("Java")
    language("Groovy")
    language("Javascript")
    language("Haskell")
    language("Scala")
}
```

http://groovy-lang.org/processing-xml.html

Operator Overloading

Operator Overloading

- Groovy supports operator overloading which makes working with Numbers, Collections, Maps and various other data structures easier to use.

```
def date = new Date()
date++
println date
```

- All operators in Groovy are method calls.
- Various operators in Groovy are mapped onto regular Java method calls on objects.

Operator Corresponding Method Call

The following few of the operators supported in Groovy and the methods they map to

`a + b`	`a.plus(b)`
`a - b`	`a.minus(b)`
`a * b`	`a.multiply(b)`
`a ** b`	`a.power(b)`
`a / b`	`a.div(b)`
`a % b`	`a.mod(b)`
`a++`	`a.next()`
`a--`	`a.previous()`

http://www.groovy-lang.org/operators.html#Operator-Overloading

Groovy Closures

Groovy Closures

- A Closure is a block of code given a name.
- Groovy feature that will be used the most.
- Methods can accept closure as parameters.

Groovy Closures...

```
def adder = {a, b->  a + b }
```

Groovy Closures...

```
def adder = {a, b->  a + b }
```

Assign a function into a variable

Groovy Closures...

```
def adder = {a, b->  a + b }
```

Closure parameters

Groovy Closures...

```
def adder = {a, b->  a + b }


assert adder(1, 2) == 3
```

Groovy Closures...

```
def adder = {a, b->  a + b }


assert adder(1, 2) == 3

assert adder('a', 'b') == 'ab'
```

Groovy Closures...

```
def adder = {a, b->  a + b }
```

```
assert adder(1, 2) == 3

assert adder('a', 'b') == 'ab'
```

Genericity with duck typing & operator overloading

Closures - explicit type

```
def intAdder = {int a, int b->  a + b }
```

Closures - implicit parameter

```
doubleIt = { it * 2 }

assert doubleIt(3) == 6

assert doubleIt(‘a’) == ‘aa’
```

Closures - variable arguments

```
def sum = {... elements ->
                 elements.sum() }


assert sum(1,2) == 3

assert sum(‘a’, ‘b’, ‘c’) == ‘abc’
```

Closures - default values

```
def mult = { int a, int b = 10 -> a * b }

assert mult(2, 3) == 6

assert mult(5) == 50
```

Closures - methods as functions

```
def logBase10 = Math.&log10
def printer = System.out.&println

assert logBase10(10) == 1
printer ‘abc’
```

Groovy Collections

Groovy Collections

Let's say we want to double the salary of all the employees who are having more than 4-years of experience.

Groovy Collections...

```
class Employee {

     String name
     Integer experience
     Double salary
}

List<Employee> employees = []
employees << new Employee(name: "John Doe", experience: 2, salary: 1000)
employees << new Employee(name: "Hanna Mackey", experience: 10, salary: 3000)
employees << new Employee(name: "Himanshu Seth", experience: 8, salary: 2000)
employees << new Employee(name: "Roni Thomas", experience: 6, salary: 1500)
employees << new Employee(name: "Bob", experience: 5, salary: 2000)
```

Groovy Collections...

```
List<Employee> premiumEmployees = employees.findAll { it.experience > 4}
                                           .collect {
                                               it.salary = it.salary * 2
                                               return it
                                           }
                                           .sort {it.name}

premiumEmployees.each { println "Name: $it.name, Salary: $it.salary"}
```

Groovy Traits

Traits are functional construct of language, which allow:

- Composition of behaviours
- Runtime implementation of interfaces
- Behaviour overriding
- Compatibility with static type checking/compilation

Traits ...

```
trait Greetable {
    void greeting() { “Welcome to GR8Conf US” }
}
```

Traits ...

```
trait Greetable {
    void greeting() { “Welcome to GR8Conf US” }
}

class Person implements Greetable {}

Person p = new Person()
assert p.greeting() == “Welcome to GR8Conf US”
```

Traits - abstract methods

```
trait Greetable {
    String getName()
    void greeting() { “Hello ${getName()}! Welcome to GR8Conf US” }
}

class Person implements Greetable {
    String getName() { “Bob” }
}

Person p = new Person()
assert p.greeting() == “Hello Bob! Welcome to GR8Conf US”
```

Traits - composition of behaviors

```
trait Greetable {
    void greeting() { “Welcome to GR8Conf US” }
}

trait Presentable {
    void present() { “I am presenting at GR8Conf US” }
}
```

Traits - composition of behaviors

```
trait Greetable {
    void greeting() { “Welcome to GR8Conf US” }
}

trait Presentable {
    void present() { “I am presenting at GR8Conf US” }
}

class Speaker implements Greetable, Presentable { }
```

Traits - composition of behaviors

```
trait Greetable {
    void greeting() { “Welcome to GR8Conf US” }
}

trait Presentable {
    void present() { “I am presenting at GR8Conf US” }
}

class Speaker implements Greetable, Presentable { }

def speaker = new Speaker()
assert speaker.greeting() == “Welcome to GR8Conf US”
assert speaker.present() == “I am presenting at GR8Conf US”
```

Traits - runtime implementation

```
trait Greetable {
    void greeting() { “Welcome to GR8Conf US” }
}

class Attendee { }
```

Traits - runtime implementation

```
trait Greetable {
    void greeting() { “Welcome to GR8Conf US” }
}

class Attendee { }

def attendee = new Attendee()
attendee.greeting()
```

Traits - runtime implementation

```
trait Greetable {
    void greeting() { “Welcome to GR8Conf US” }
}

class Attendee { }

def attendee = new Attendee() as Greetable
attendee.greeting()
```

Traits - overriding default behavior

```
trait Greetable {
    void greeting() { “Welcome to GR8Conf US” }
}

class Attendee implements Greetable {
    void greeting() { “I am very excited about GR8Conf US” }
}
```

Traits - overriding default behavior

```
trait Greetable {
    void greeting() { “Welcome to GR8Conf US” }
}

class Attendee implements Greetable {
    void greeting() { “I am very excited about GR8Conf US” }
}

def attendee = new Attendee()
assert attendee.greeting() == “I am very excited about GR8Conf US”
```

What Next?

Groovy for Shell scripting

Groovy for Shell scripting

You know Java better than Shell scripting

Groovy For Shell Scripting

Let’s write a Groovy script to count characters & words in a file...

Groovy For Shell Scripting

```
#!/usr/bin/env groovy

def content = new File(args[0]).text
def charCount = content.size()
def wordCount = content.split(/\s/).size()

def stringCount
if( args.size() == 2 ) {
     stringCount = content.count(args[1])
}
println "Characters Count:".padRight(25) + charCount
println "Word Count:".padRight(25) + wordCount
```

Using Spock For Testing Java Classes

Using Spock For Testing Java Classes

```
class LibraryTest extends Specification {
    def "someLibraryMethod returns true"() {
        setup:
        Library lib = new Library()
        when:
        def result = lib.someLibraryMethod()
        then:
        result == true
    }

}
```

Groovy For DSL

```
List<User> users = FollowersGraphBuilder.build {
    users 'John', 'Billy', 'Kate', 'Bob'

    user('Kate').follows 'Bob'
    user('Billy').follows 'Bob'
    user('Bob').follows 'John'
}
```

Groovy Ecosystem

GRAILS

GRIFFON

SDKMAN!

Groovy

spock

Ratpack

GPars

gradle

Questions ?

Thank you

References