Capgemini

CONSULTING.TECHNOLOGY.OUTSOURCING

GROOVY

2018

People matter, results count.

Agenda

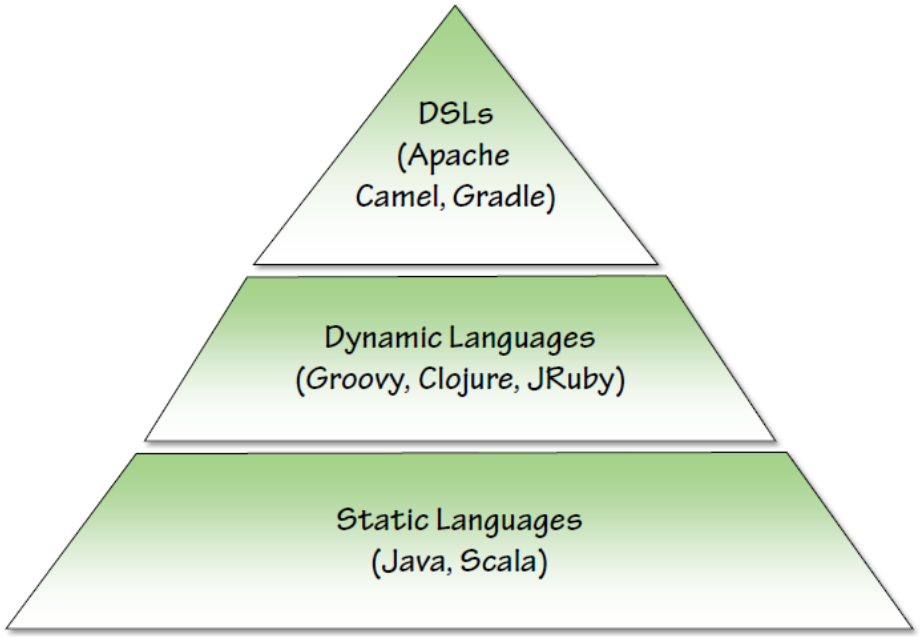
Groovy Introduction

Agenda

Groovy Overview

- **Java has become ubiquitous**
- **Slow evolution of the Java language**
- **Much innovation in alternative languages for the JVM**
- **The Ployglot Programmer**

- **Why Groovy?**
  - Most Java syntax is legal Groovy syntax
  - Groovy leverages existing Java libraries
  - Groovy objects extend the java.lang.Object



Installation

- To install groovy via windows installer use below link:
  - https://dl.bintray.com/groovy/Distributions/groovy-2.4.15-installer.exe

Groovy Console and Command prompt

- **To Check Version:**
  - groovy –v
- **To open Groovy Console**
  - groovyConsole
- **To compile groovy**
  - groovyc Sample.groovy
- **To see all the class details**
  - javap Sample.class
- **To run groovy**
  - groovy Sample.groovy

Command Prompt

```
D:\vidavid\groovy\courseware>javap Sample.class
Compiled from "Sample.groovy"
public class Sample extends groovy.lang.Script {
  public static transient boolean __$stMC;
  public Sample();
  public Sample(groovy.lang.Binding);
  public static void main(java.lang.String...);
  public java.lang.Object run();
  protected groovy.lang.MetaClass $getStaticMetaClass();
}

D:\vidavid\groovy\courseware>
```

Groovy Console

Command Prompt

```
Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\vidavid>groovy -v
Groovy Version: 2.4.15 JVM: 1.8.0_45 Vendor: Oracle Corporation OS: Windows 8

C:\Users\vidavid>groovyConsole

C:\Users\vidavid>
```

GroovyConsole

File Edit View History Script Help

```
1 println "Hello! welcome to Groovy"
```

Execute Groovy Script

```
groovy> println "Hello! welcome to Groovy"

Hello! welcome to Groovy
```

Execution complete. Result was null. 1:31

Groovy Script

Sample.groovy

```
println "Welcome to Groovy!"
println "Today's date is: " + new Date()
```

To Compile and Run the above Groovy Script:

groovyc Sample.groovy

groovy Sample.groovy

IDE Support

- To install groovy plugin in STS/Eclipse
- Help → Install new Software
- Enter the below URL
  - http://dist.springsource.org/snapshot/GRECLIPSE/e4.5/
- Choose only **Groovy Eclipse (required)** and finish.
- This will install Groovy in your IDE.

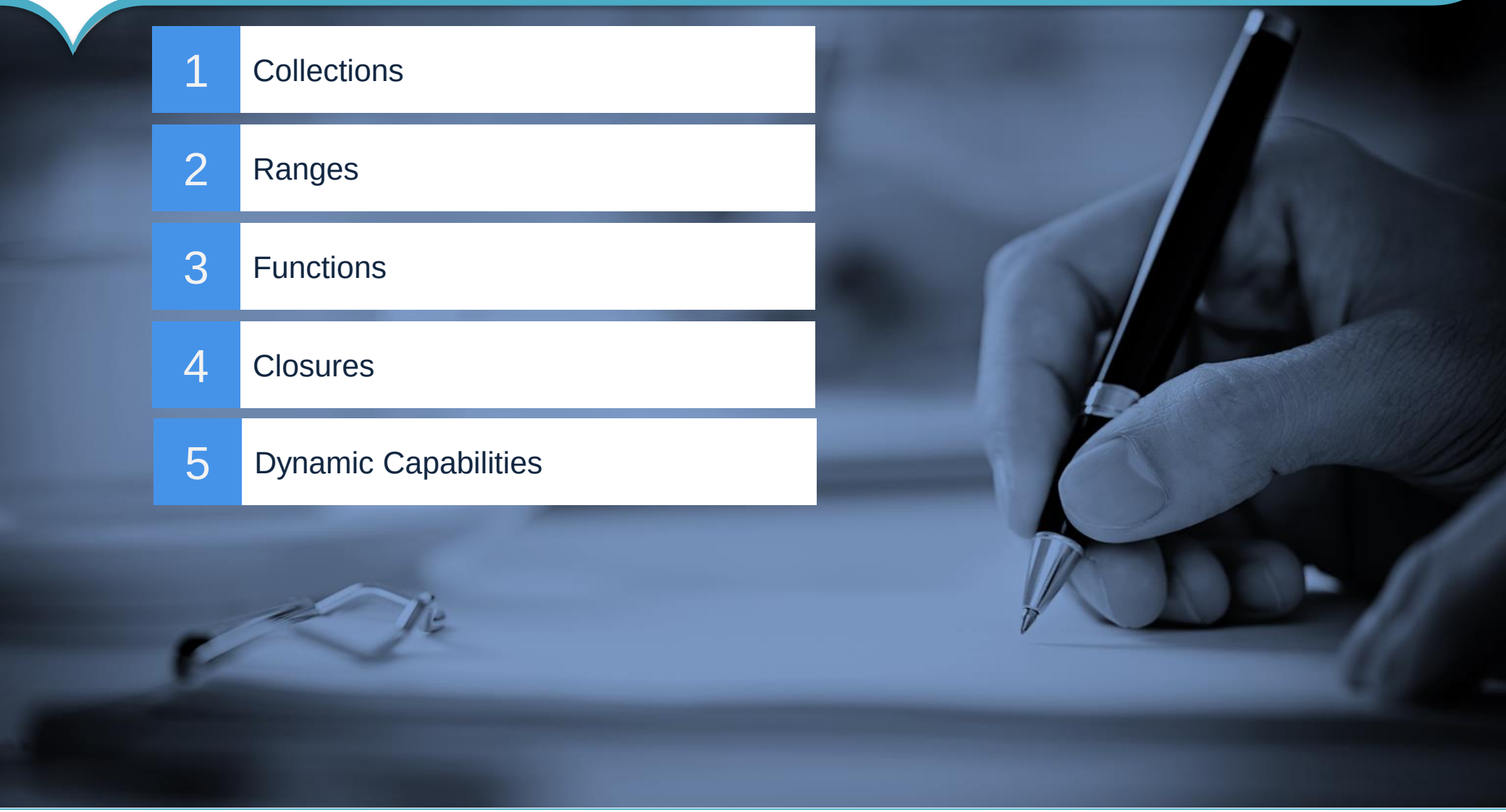
Summary

- Groovy Overview
- Installation
- Run Groovy via Console, Command prompt, Script
- Eclipse / STS IDE support
- Good understanding of groovy

Basic Syntax

Outline

1. Collections
2. Ranges
3. Functions
4. Closures
5. Dynamic Capabilities



Basic Syntax

- Class
- Object
- Variable declaration using def
- Dynamic Typing
- Variable binding using $
- Control Statements

```
println "Welcome to Groovy!"
println "Today's date is: " + new Date()
```

Collections

- Collection of objects can be stored in a container, can be iterated and manipulated as per the requirement.

```
def names=["John","Paul","Rose","Ringo","Jack"]
for(name in names){
        def greetings = "Hello, "
        println "$greetings" + "$name"
}
```

Ranges

```
def range = 'a'..'g'
for(var in range){
        println var
}
```

```
def enum DAYS{
        SUN,
        MON,
        TUE,
        WED,
        THR,
        FRI,
        SAT
}

def weekdays= DAYS.MON..DAYS.FRI
for(var in weekdays){
        println var
}
println "Extents:"
println weekdays.from
println weekdays.to
```

Functions

```
def numbers = 0..9
for(num in numbers){
        if(isEven(num)){
        println num
        }
}

def isEven(def num){
        return num%2==0
}
```

Function without def and return:

```
def isEven(num){
        num%2==0
}
```

Closures

```
def myClosure = {
    println "I am in Closure"
    println new Date()
}
```

Dynamic Capabilities

- Declarative Variable
- Dynamically declare the variable
- Variable can be any type
- Based on the assignment variable nature will change
- Flexible to handle variable

```
def cat="meow"
//def one=1
//Integer one=1
def one =1
one="one"
println one.getClass()
println one.toUpperCase()
```

Working with XML

Outline

1. XML problem Statement
2. XML read
3. XML write
4. Grape – Dependency Management

XML Problem Statement

- **GPS and Weather Combiner**
  - Work with XML files
  - Consume JSON data from a REST service
  - Store data in a relational database
  - Interact with native Java libraries
  - Work with Groovy's dependency management scheme
  - Explore unit testing in Groovy

An XML format for exchange GPS data
Open format
Widely supported in the industry
More information available at www.topografix.com/gpx.asp

XML Read

- **DOMCategory**
  - Low-level access to Java's DOM API
- XmlParser
  - Eager evaluation
- XmlSlurper
  - Lazy evaluation

XML Writer

- **MarkupBuilder**
  - Simple XML creation
- **StreamingMarkupBuilder**
  - Advanced XML creation
  - Better support for large documents

XML Handling

- All of Java's native XML processing options are available
- Groovy brings new options for both low and high level processing of XML
- Reading and writing of XML is made easier due to Groovy's dynamicity

Calling Java from Groovy

Outline

1. Groovy with Java Library
2. Grape
3. IVY
4. Joda – Time Example

Groovy with Java Library

- Groovy works with existing Java libraries
- Including popular libraries such as **Google Guava** or **Apache Commons**
- Or your own **proprietary** libraries

Grape

- **Dependency management** for Groovy
- **Natively supported** in the Groovy language
- Built on **Ivy**

IVY

- **Maven Repository-compatible** dependency management
- Uses Maven coordinates for **dependency resolution**
- Automatically **installs** these dependencies at **runtime**

Joda Time

- An alternative to the standard Java date and time libraries
- Supports multiple calendar systems
- Includes functionality for easily working with date and times

Note:
Groovy allows you to continue to use existing Java libraries
Grape simplifies dependency management in Groovy

Unit Testing in Groovy

Outline

Why Should we Unit Test?

Finds problems **early** in development

Preserves **key** behavior of the system

Documents the code

What makes good Unit Test?

Fast

Isolated

Repeatable

Self-verifying

Timely

Test Driven Development (TDD)

Writing your unit tests before before your write production code

Red-Green-Refactor

Watch the test
Fail

Write just enough code
for the test to **Pass**

Refactor the code

Benefits of TDD

- Results in clean APIS
- Testable code is flexible code
- Results in much less production code
- No need to schedule 'unit-testing' time

Disadvantages of TDD

- Invcreases initial development time
- Has a significant learning curve
- Difficult without full team buy-in

Unit Testing Support in Groovy

JUnit is built **directly** into the Groovy runtime

Mocking APIs are **included** in the Groovy platform

Advanced testing frameworks are also available

Groovy is Dynamic

- The compiler doesn't enforce type-checking
- Unit Tests become your safety net
- Especially on a large team

Stub and Mock

A simulated object that mimics the behavior of a real object in a controller way.

Stubs vs Mocks

Stubs

- Can be used as stand-in objects
- Will return canned values when asked

Mocks

- Supports all the functionality of stubs
- Can verify a member was called a certain number of times
- Can assert a specific interaction occurred

Digging Deeper

- Extension methods are available on native objects
- Groovy's truth model allows for more succinct null checks
- Catagories allow you to easily add custom logic to existing classes

REST services

Outline

1. REST Services and Groovy
2. REST service
3. JSON parser

REST Services and Groovy

Groovy has **support available** for working with REST services

RESTClient encapsulates HTTP **requests** and **responses**

Dynamic languages are **well-suited** to working with flexible data

REST Service

- Groovy has excellent support available for working with REST services
- Dynamic languages are well-suited to working with flexible data formats

JSON parser

Similar to the XmlSlurper → JsonSlurper for parsing JSON.

```
import groovy.json.JsonOutput
import groovy.json.JsonSlurper
def a = new JsonSlurper().parse(new File("./input/tasks.json"))
JsonOutput.prettyPrint(a.toString())
```

Working with databases

Outline

1. Working with Database
2. Getting SQL instance
3. Dataset and insert query
4. Dataset and update query

Working with Database

- Groovy provides built in support for working with database
- Dataset allows you to easily insert data
- eachRow and firstRow allow you to work with query results
- executeUpdate method allows for more complicated statements

Getting SQL instance

```
@GrabConfig(systemClassLoader=true)
@Grapes([
@Grab(group='mysql', module='mysql-connector-java', version='5.1.26'),     ])
import groovy.sql.Sql

def sql = Sql.newInstance("jdbc:mysql://localhost:3306/gps", "root", "root",
            "com.mysql.jdbc.Driver")
```

Dataset and insert query

```
def routepoints = sql.dataSet("routepoints")
        routepoints.add(latitude: it.@lat.toDouble(), longitude: it.@lon.toDouble(),
                timestep: new DateTime(it.time.toString()).toDate(),
                temperature: response.data.currently.temperature)
```

Dataset and update query

```
@GrabConfig(systemClassLoader=true)
@Grapes([
        @Grab(group='mysql', module='mysql-connector-java', version='5.1.26'),
        ])
import groovy.sql.Sql
 def sql = Sql.newInstance("jdbc:mysql://localhost:3306/gps", "root", "root",
        "com.mysql.jdbc.Driver")
 def newTemperature = 100
sql.executeUpdate("update routepoints set temperature = ${newTemperature}")


sql.close()
```

Other bits and pieces

Using Groovy in a Maven build

```
<dependencies>
    ... other dependencies
    <dependency>
        <groupId>org.codehaus.groovy</groupId>
        <artifactId>groovy-all</artifactId>
        <version>2.4.5</version>
    </dependency>
</dependencies>
```

Using Groovy in a Gradle build

```
apply plugin: 'groovy'

repositories {
    mavenCentral()
}

dependencies {
    compile 'org.codehaus.groovy:groovy-all:2.4.5'
}
```

THANK YOU

Capgemini

CONSULTING.TECHNOLOGY.OUTSOURCING

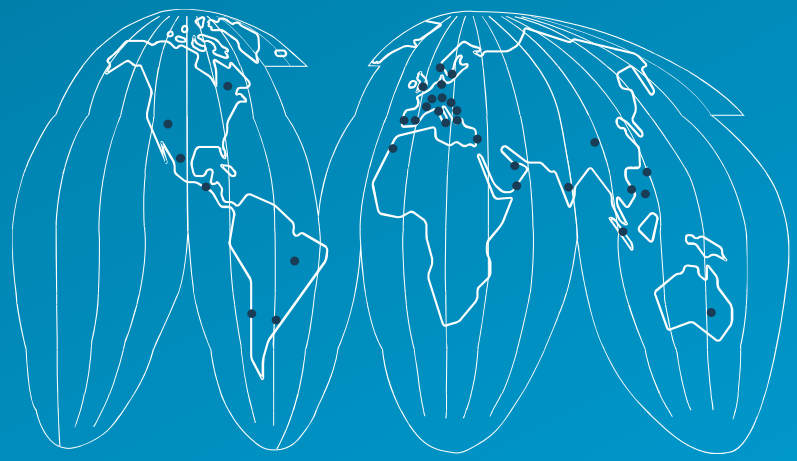
People matter, results count.

About Capgemini

With more than 145,000 people in 40 countries, Capgemini is one of the world's foremost providers of consulting, technology and outsourcing services. The Group reported 2014 global revenues of EUR 10.5 billion.

Together with its clients, Capgemini creates and delivers business and technology solutions that fit their needs and drive the results they want. A deeply multicultural organization, Capgemini has developed its own way of working, the Collaborative Business Experience™, and draws on Rightshore®, its worldwide delivery model.

Rightshore® is a trademark belonging to Capgemini



www.capgemini.com