Groovy for Java Developers

Course Summary

Description
This course teaches experienced Java developers how to write programs in Groovy, both by itself and in conjunction with existing Java code. An optional additional one day session on Gradle is available as an add-on to this class. This is a four day course with an optional fifth day on Gradle.

Objectives
After taking this course, students will be able to:

- Understand how dynamic languages differ from languages like Java or C#
- Work with Groovy closures
- Use Groovy control structures
- Build object-oriented programs in Groovy
- Use Groovy builders to reduce complexity
- Access databases using Groovy
- Write unit tests in Groovy
- Access Java classes from Groovy, and vice versa
- Build a simple web application using Grails
- Work with Groovy strings, closures, and collections
- Develop model-view-controller web applications using Grails
- Access relational databases from Grails
- Build unit and functional tests for web applications
- Groovy support in the Spring framework
- Build and test RESTful web services
- Use the Gradle build tool

Topics
- Groovy Fundamentals
- Object-Oriented Programming In Groovy
- Groovy Collections
- Closures in Groovy
- Groovy Control Structures
- Miscellaneous Groovy operators
- Unit Testing in Groovy
- Abstract Syntax Tree (AST) Transformations
- Groovy Builders
- Runtime Metaprogramming
- Database Access with Groovy
- The Spring Framework
- RESTful web services
- The Gradle Build Tool

Audience
This course is for experienced Java developers wanting to learn how to write programs in Groovy, both by itself and in conjunction with existing Java code.

Prerequisites
Before taking this course, all attendees should have a working knowledge of Java.

Duration
Four to five days
Groovy for Java Developers

Course Outline

I. Groovy Fundamentals
   A. Differences between Groovy and Java
   B. Compiling and executing Groovy programs
   C. The basic Groovy data types and optional typing
   D. Writing Groovy scripts
   E. Declaring classes
   F. Overriding operators and type coercion
   G. The Groovy JDK: Groovy extensions to the Java standard library
   H. Groovy strings
   I. Regular expressions in Groovy

II. Object-Oriented Programming In Groovy
    A. Plain Old Groovy Objects (POGOs)
    B. Generated getters/setters
    C. Public vs private defaults
    D. Map-based constructors
    E. @ToString, @EqualsAndHashCode, @TupleConstructor, and @Canonical

III. Groovy Collections
    A. Ranges
    B. Lists
    C. Maps
    D. Iterators and polymorphic algorithms

IV. Closures in Groovy
    A. Declaring closures
    B. Available options for calling closures

V. Groovy Control Structures
   A. The "Groovy truth"
   B. Conditional execution
   C. Looping constructs

VI. Miscellaneous Groovy operators
    A. Safe navigation with the ?. operator
    B. The Elvis operator
    C. The spaceship operator

VII. Unit Testing in Groovy
     A. GroovyTestCase
     B. JUnit tests in Groovy
     C. The Spock testing framework
     D. Spock mocks and stubs

VIII. Abstract Syntax Tree (AST) Transformations
    A. @Delegate
    B. @Immutable
    C. @Singleton
    D. @TypeChecked and @CompileStatic

IX. Groovy Builders
    A. MarkupBuilder
    B. JsonBuilder
    C. AntBuilder and others

X. Runtime Metaprogramming
    A. Working with Expandos
    B. The MetaClass class
    C. Adding fields and methods at runtime
    D. The Intercept-Cache-Invoke design pattern

XI. Database Access with Groovy
    A. Basic database operations
    B. DataSet operations
    C. Groovy and ORM solutions

XII. The Spring Framework
     A. Using Groovy beans in Spring applications
     B. "Refreshable" beans
     C. Inline scripted beans
     D. AOP using Groovy
     E. The Groovy BeanBuilder in Spring

XIII. RESTful web services
      A. JAX-RS annotations on Groovy classes
      B. JsonSlurper and JsonBuilder
      C. REST clients with the HtptBuilder project
      D. The ratpack framework
      E. REST additions to Grails 2.3+

XIV. The Gradle Build Tool
     A. The build challenge
     B. The Gradle plugin system
     C. Defining tasks and dependencies
     D. Using repositories
     E. Multi-project builds
Groovy for Java Developers

Course Outline (cont’d)

Additional one-day Gradle session:

XV. Quick Tour of Gradle
A. Creating build scripts
B. Declaring dependencies
C. Accessing repositories
D. Using plugins
E. Configuring the directed acyclic graph

XVI. Building Java and Groovy projects
A. Standard project structure
B. The Java plugin
C. The Groovy plugin
D. Running tests

XVII. Defining Custom Tasks
A. Declaring tasks
B. Defining project properties
C. Setting dependencies
D. Using doFirst and doLast
E. Using the built-in task types
F. Inserting your tasks into the DAG

XVIII. The Gradle Daemon
A. Usage and troubleshooting
B. Configuring the daemon

XIX. Web projects
A. Standard web layout
B. The war and jetty plugins
C. Customizing web projects

XX. IDE Integration
A. Eclipse projects
B. IntelliJ IDEA

XXI. The Gradle wrapper
A. Specifying versions
B. Generating the scripts

XXII. Multi-project builds
A. Using settings.xml
B. Consolidating configuration properties
C. Making one subproject depend on another