Introduction to Groovy

Part II – Groovy Grooviness

GROOVY… WHERE LESS REALLY IS MORE

Overview

Quick Review of Part I

Syntactical Sugar

Groovy truth

Groovy operators

Closures, A First Look

Intro to Collections

Look ahead to Part III

Part I Review

Getting Started

Groovy Scripts & Classes

POGO v POJO

Groovy Strings

Dynamic Typing

A Taste of Dynamic Behavior

Some Gotchas

Before I go on…

Let’s talk about two kinds of complexity

Essential complexity: The problems we try to solve have essential complexity. By their nature, they are complex

Accidental complexity: All the complexity we add in to implement a solution to the problem that is not essential complexity I accidental complexity

Using Groovy’s syntactical sugars and idioms, we can drive out a lot of the accidental complexity we added in because of Java

Syntactical Sugar

More default imported packages than Java

All classes assumed public

All fields assumed private; public getters / setters added automatically

Free map c'tor

Access / mutate properties like field level access (but it's not)

Parentheses optional in method calls (unless no arg)

Semi-colons optional unless multiple statements on line (or classic for loop)

Return statement optional at end of methods. Last evaluated expression is returned. Don't be obtuse!

Void methods return null

The Groovy compiler treats all exceptions as Runtime

Groovy Truth

Java is very rigid about truth

- if(someExpression) {…} // someExpression must be boolean result

This leads to a lot of noise in the code

```
if(arry != null && arry.length > 0)    // arrays have length property
if(name != null && name.length() > 0) // Strings have length()
if(list != null && list.size() > 0)    // collections have size()
if(map != null && map.size() > 0)      // maps have size()
if(iter != null & iter.hasNext())      // iterators
if(value != 0)                         // number not zero is true
if(anyObject != null)                  // object not null is true
```

Groovy Truth

Groovy has a relaxed definition of truth

- It's like JavaScript's definition

This leads to a lot of noise in the code

```
if(anyObject)      // true if not null
if(arry)           // true if array is not null and size() > 0
if(name)           // true if String is not null and size() > 0
if(list)           // true if collection not null and size() > 0
if(map)            // true if map is not null and size() > 0
if(iter)           // true if iteration (or enumeration) not null & has more
if(value)          // true if numeric not null and not zero

/* Groovy adds size() to arrays and Strings so we can finally have a
   consistent syntax about a thing's size */
```

Groovy Operators

Safe Navigation (?.)

Elvis (?:)

Spaceship (compareTo) (<=>)

Spread (*.)

Range operator (..)

getAt/putAt ([])

asType (as)

Regex find (=~) / Regex match (==~)

Method reference (.&)

Membership (in)

Identity (is)

Closures, A First Look

Groovy closures are first class functional citizens

Think of them as disembodied methods that can be passed around

There’s an elegant syntax for declaring and passing them inline

Intro to collections in Groovy

Groovy adds many enhancements to JDK collections

Groovy introduces a new type called a Range

Lists feel more like arrays

Maps feel more like beans

What's Next

At our next meeting we will have Part III of this Groovy introduction.

We will continue the Groovy Introduction series with the following:

- More on collections
- Traits & Java 8 Support
- Builders and Slurpers
- Some useful AST transformations
- Adding Groovy to your Maven Projects
- Using Groovy with Your Favorite IDEs
- What's coming in Groovy 3.0…

Resources & Links

Groovy

- http://groovy.codehaus.org
- http://groovy.codehaus.org/Differences+from+Java
- http://groovy.codehaus.org/Groovy+Truth
- http://groovy.codehaus.org/Operators