Automating Jenkins Using Groovy

GROOVY FUNDAMENTALS, BRIEFLY

Chris B. Behrens
SOFTWARE ARCHITECT
@chrisbbehrens

Our Demo Scenario

A DevOps Pro

Globlomantics Corporation

Groovy – the Jenkins automation language of choice

The Groovy Console

Something to validate and verify our code

Jenkins is the ultimate target

A place to experiment

http://groovy-lang.org/download.html

Demo

Navigate to the download link

Extract and launch our console

Write a super simple Groovy script with it

Execute it

Data Types

x is clearly a number

An optionally typed language

No type definition at all

A Partial List of Data Types

Data Type	Groovy Keyword	Sample Data
Strings	String	“Chris B. Behrens”
Integers	int	0,1,2,3
Floats	float	0.5, 3.8, 98345.12345
Boolean	Boolean	true, false

Data Types and Syntax

Optional “def” or explicit data type

Loose typing?

Meh

Okay when the return value is crystal clear

Control Structures

```
if (isProgrammer) {
    println "He's a programmer, alright"
}
else{
    println "Not a programmer, tho"
}
```

Control Structures

```
for (int i = 0; i < courseCount; i++){
    println "Chris made course " + (i+1) + "!!!"
}
```

Control Structures

```
int i = 0


while (i < courseCount){
    println "Chris made course " + (i+1) + "!!!"
    i++
}
```

Control Structures

```
String[] singers = ["Bob", "George", "Jeff", "Roy", "Tom"]


for(String singer: singers){
    println singer
}
```

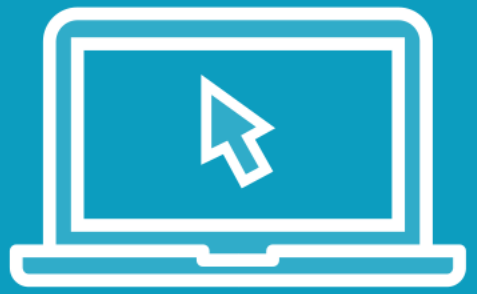
Control Structures

```
singers.each{x -> println(x)}
```

Control Structures

```
singers.each{println(it)}
```

Demo



Review our control structures

- If / Else
- For loop
- For-in loop
- Each loops

Subroutines

Functions

Methods

Groovy, Java and C#: *void* return type

A function to create credentials from a name

Demo

Create our username function

Properly assert that its return value is correct

Create a simple output function

Talk about parameters and types

Working with Classes and Objects

Base level of encapsulation

Wrap your code in classes

Parallel lists => object with two properties

Very similar to Java or C#

A Class in Groovy

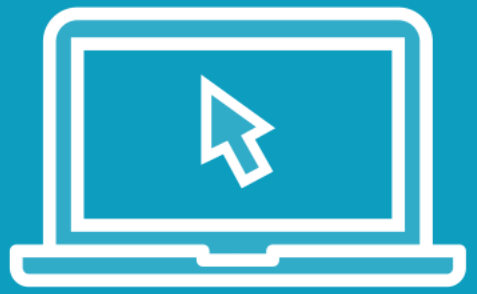
```
class User{

    // code and do stuff

}


User user = new User();
```

Demo



- Declare a new class
- Encapsulate our current work into a User class
- Wrap up our users with our objects
- Output their usernames

Inheritance

You might not need it

Groovy supports interfaces

- But the scripting case for interfaces is weaker
- Not so for regular inheritance and abstract classes

Abstract classes have *implementation*

Nobody has a *car,* exactly

They have car *instances*

Copy and paste is evil

Globlomantics Entertainment

Represent the people involved

- Artists
- Producers

They all have names

And they all get credentials

Artists will have a song collection

Producer will *void Produce()*

Demo

Change our User class to be an abstract class

Create two implementer classes

- Artist
- Producer

Add our concrete methods

Execute our script

Review the results

Working with External Packages

External packages

Leveraging the work and creativity of others

Default Imports in Groovy

java.io.*	java.lang.*	java.math.BigDecimal	java.math.BigInteger
java.net	java.util	groovy.lang.*	groovy.util.*

Default Groovy Imports (Code)

```
import java.lang.*

import java.util.*

import java.io.*

import java.net.*

import groovy.lang.*

import groovy.util.*

import java.math.BigInteger

import java.math.BigDecimal
```

Demo

- Json data file representing music data
- Opening the file from our hard drive
- Parsing it with a new library we need to import
- Outputting part of the contents

Summary

- The basics of the Groovy language
- The Groovy Console
- Data Types
- Control Structures
- Subroutines
- Defining Classes
- Inheritance
- Working with External Packages