[12pt]article [russian]babel

document Groovy is an object oriented programming language developed for the Java platform as a Supplement to the Java language with the capabilities of Python, Ruby and Smalltalk.

Groovy uses Java-like syntax, dynamically compiled to JVM bytecode and works directly with other Java code and libraries. Language can be used in any Java project or as a scripting language.

Groovy has completed the process of standardization in Java Community ProcessJSR 241. Features in Groovy (to distinguish it from Java):

Static and dynamic typing

Built-in syntax for lists, associative arrays, arrays and regular expressions

Circuit

Overload operations

History

The first mention of Groovy was a blog post James Strachey (eng. James Strachan (programmer))[3] from August 2003. It was later released several versions between 2004 and 2006. After the process of standardization of the JCP, version numbering was changed and a version called "1.0". The "1.0" version was released 2 Jan 2007. In December 2007, Groovy 1.1 was released, this version was soon renumbered as "1.5" as a result of significant changes in the language.

Strachan left the project a year before the Groovy 1.0 release in 2007, and in July 2009 Strachan wrote in his blog that perhaps would not have created Groovy if in 2003 he read a book by Martin Odersky et al programming in Scala (released in 2007)[4]. Project language development and the Committee of the JSR-241 in 2007 is headed by Guillaume La Forge (Guillaume Laforge).

IDE support

Programming Groovy is supported in the major integrated development environments software[5], in particular:

IntelliJ IDEA since version 7 or earlier versions using the JetGroovy Plugin;

Eclipse using the Groovy Eclipse;

Netbeans built NetBeans IDE.

Groovy

The latest version of the report generator iReport based on a Java library, JasperReports, allow to build reports expressions in Groovy and write on it with additional logic. The continuous integration system Jenkins allows you to use automation scripts created in Groovy.

Installing Groovy

Groovlets the ability to run Groovy scripts as servlets. GroovyBeans version Groovy JavaBeans. Unlike Java, a Groovy source code can be executed as a normal script, if it contains code outside of a class definition or a class with the main method or Runnable or Groovytestcases: !/usr/bin/env groovy println "I can execute this script now!" Strings in Groovy: Java Strings with single quotes and double quotes with GStrings. def javaStyleString = 'java-String style' def GStringsStyleString = "$javaStyleString" $def j = 'javaStyleString' def bigGroovyString = """ $javaStyleString GStringsStyleString """ println bigGroovyString Groovy implicitly generates methods to access the variables (setColor(String color) and getColor()): class AGroovyBean String color

def myGroovyBean = new AGroovyBean()

myGroovyBean.setColor('blue') assert myGroovyBean.getColor() == 'blue'

myGroovyBean.color = 'green' assert myGroovyBean.color == 'green' Groovy provides a simple and consistent access to lists, maps and arrays: def myList = ['One', 'Two', 'Three'] //looks like an array, but this is the list assert myList[2] == 'Three' myList[3] = 'Four' //add an item to the list assert myList.size() == 4

def monthMap = ['January' : 31, 'February' : 28, 'March' : 31] //define an associative array assert monthMap['March'] == 31 monthMap['April'] = 30 //add element to associative array assert monthMap.size() == 4 Closure (closure) is an anonymous function and the object in one view: def closureFunction = a, b -¿ println a println b

closureFunction(1, 2) return in a function is optional it will default the returned value of the latter variable. Immutable classes are marked with the annotation Immutable: @Immutable class ImmutableClass

String stringVariable Integer integerVariable def newVariable = new ImmutableClass(stringVariable : "some string", integerVariable : 23)