

[推酷](#)

- [文章](#)
- [站点](#)
- [主题](#)
- [公开课](#)
- [活动](#)
- [客户端](#) [荐](#)
- [周刊](#)
 - [编程狂人](#)
 - [设计匠艺](#)
 - [创业周刊](#)
 - [科技周刊](#)
 - [Guru Weekly](#)
 - [一周拾遗](#)

30分钟groovy快速入门并掌握 (ubuntu 14.04+IntelliJ 13) - Hi_Amos • [登录](#)

时间 2014-09-14 15:02:00 [博客园-原创精华区](#)

原文 <http://www.cnblogs.com/amosli/p/3970810.html>

主题 [Groovy Ubuntu](#)

本文适合于不熟悉 Groovy，但想快速轻松地了解其基础知识的 Java开发人员。了解 Groovy 对 Java 语法的简化变形，学习 Groovy 的核心功能，例如本地集合、内置正则表达式和闭包。编写第一个 Groovy 类，然后学习如何使用 JUnit 轻松地进行测试。借助功能完善的 Groovy 开发环境和使用技能，您将轻松完成本教程的学习。最重要的是，您将学会如何在日常 Java 应用程序开发中联合使用 Groovy 和 Java 代码。

阅读本文的前提条件：为了从本教程得到最大收获，您应该熟悉 Java 语法和在 Java 平台上进行面向对象开发的基本概念。如果熟悉C#的话，基本上也能很快上手。

本文预计时长：30分钟

一、groovy简介和 环境搭建

本机环境：

ubuntu 14.04 64bit

JDK 1.7.67

IDE：intellij idea 13.1

1、groovy简介

其官方介绍为，Groovy...

- is an agile and dynamic language for the Java Virtual Machine
- builds upon the strengths of Java but has additional power features inspired by languages like Python, Ruby and Smalltalk
- makes modern programming features available to Java developers with almost-zero learning curve

- provides the ability to statically type check and statically compile your code for robustness and performance
- supports Domain-Specific Languages and other compact syntax so your code becomes easy to read and maintain
- makes writing shell and build scripts easy with its powerful processing primitives, OO abilities and an Ant DSL
- increases developer productivity by reducing scaffolding code when developing web, GUI, database or console applications
- simplifies testing by supporting unit testing and mocking out-of-the-box
- seamlessly integrates with all existing Java classes and libraries
- compiles straight to Java bytecode so you can use it anywhere you can use Java

简单说来: Groovy是一种运行在JVM上的动态语言, 它吸取了Python, Ruby和Smalltalk等语言的优点, 在Java语言的基础之上增加了许多特色功能; 对于Java开发人员来说掌握Groovy是没有什么太大障碍的; 相比Java而言, 语法更易懂, 更易上手, 更易调试; 无缝的集成了Java的类和库; 编译后的.groovy也是以class的形式出现的。

2、groovy 下载

网址: <http://groovy.codehaus.org/Download>

目前最新稳定版为Groovy 2.3 (2014-09-14)

这里下载:

Download zip: [Binary Release](#): groovy-binary-2.3.6.zip

Download documentation: [JavaDoc and zipped online documentation](#): groovy-docs-2.3.6.zip

3、groovy环境配置和Hello World!

1) 首先解压:

```
unzip groovy-binary-2.3.6.zip #解压groovy
unzip groovy-docs-2.3.6.zip #解压docs
```

2) 进入到Groovy Shell命令界面:

```
amosli@amosli-ThinkPad:~/developsoft/language/groovy/groovy-2.3.6/bin$ ./groovysh
Groovy Shell (2.3.6, JVM: 1.7.0_67)
Type ':help' or ':h' for help.

-----
groovy:000> println "hello world!"
hello world!
==> null
groovy:000> :h
```

在Groovy Shell里不必定义class可以直接写代码, 如下面进行一个for循环:

```
groovy:000> for(i=0;i<10;i++){
groovy:001> println("i:"+i);}
i:0
i:1
i:2
i:3
i:4
i:5
i:6
i:7
i:8
```

```
i:9  
==> null
```

注意这里，你可以发现i是没有指定int类型的，这里也是写法上也是比较随意的。

3)、将groovy加入到环境变量（可选）

将解压后的groovy拷到/usr/local/groovy 目录下：

```
root@amosli-ThinkPad:/usr/local/groovy# cp -r /home/amosli/developsoft/language/groovy/groovy-2.3.6 .
```

将groovy路径拷到/etc/profile里：

```
gedit /etc/profile #使用gedit打开profile, 也可以使用vi等工具
```

将下面内容拷到profile里最后位置：

```
export GROOVY_HOME=/usr/local/groovy/groovy-2.3.6  
export PATH=$GROOVY_HOME/bin:$PATH:  
export GROOVY_HOME  
export PATH
```

全部的profile内容：



```
# /etc/profile: system-wide .profile file for the Bourne shell (sh(1))  
# and Bourne compatible shells (bash(1), ksh(1), ash(1), ...).  
  
if [ "$PS1" ]; then  
  if [ "$BASH" ] && [ "$BASH" != "/bin/sh" ]; then  
    # The file bash.bashrc already sets the default PS1.  
    # PS1='\h:\w\$ '  
    if [ -f /etc/bash.bashrc ]; then  
      . /etc/bash.bashrc  
    fi  
  else  
    if [ "`id -u`" -eq 0 ]; then  
      PS1='# '  
    else  
      PS1='$ '  
    fi  
  fi  
fi  
  
# The default umask is now handled by pam_umask.  
# See pam_umask(8) and /etc/login.defs.  
  
if [ -d /etc/profile.d ]; then  
  for i in /etc/profile.d/*.sh; do  
    if [ -r $i ]; then  
      . $i  
    fi  
  done  
  unset i  
fi  
  
JAVA_HOME=/usr/local/java/jdk1.7.0_67  
PATH=$PATH:$HOME/bin:$JAVA_HOME/bin  
export JAVA_HOME  
export PATH  
  
export GROOVY_HOME=/usr/local/groovy/groovy-2.3.6  
export PATH=$GROOVY_HOME/bin:$PATH:  
export GROOVY_HOME  
export PATH
```

[View Code](#)

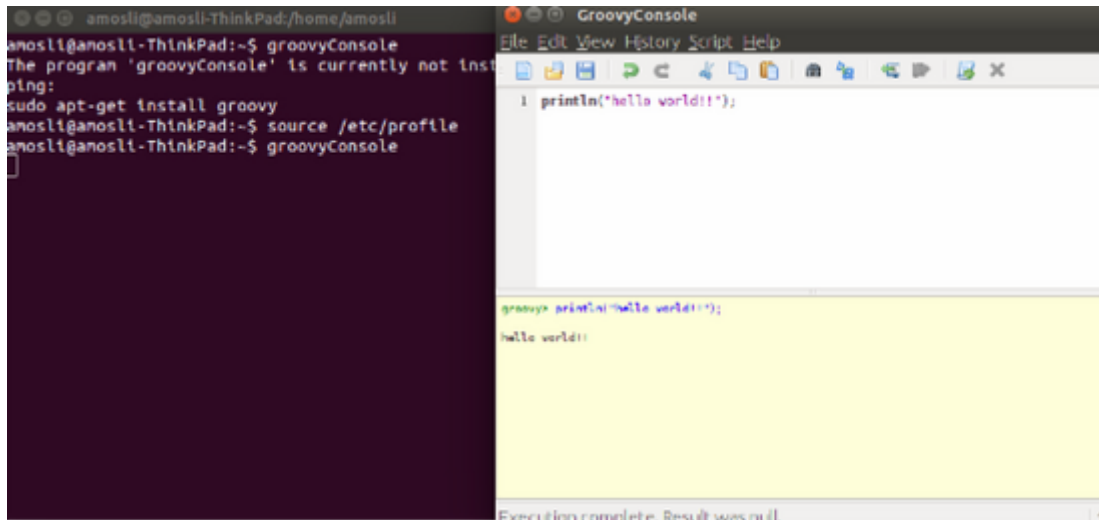
更新环境变量:

```
source /etc/profile
```

验证是否成功:

```
# groovy -version
Groovy Version: 2.3.6 JVM: 1.7.0_67 Vendor: Oracle Corporation OS: Linux
```

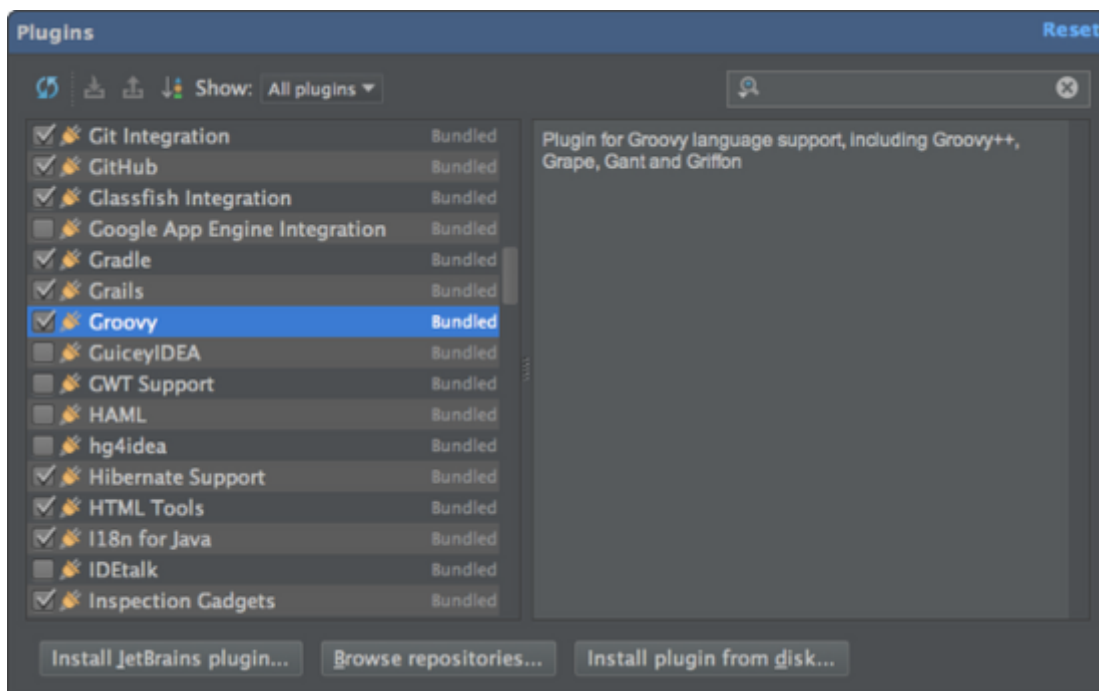
打开groovyConsole:



4)、使用IDE进行开发(这里以IntelliJ idea 13.1为例)

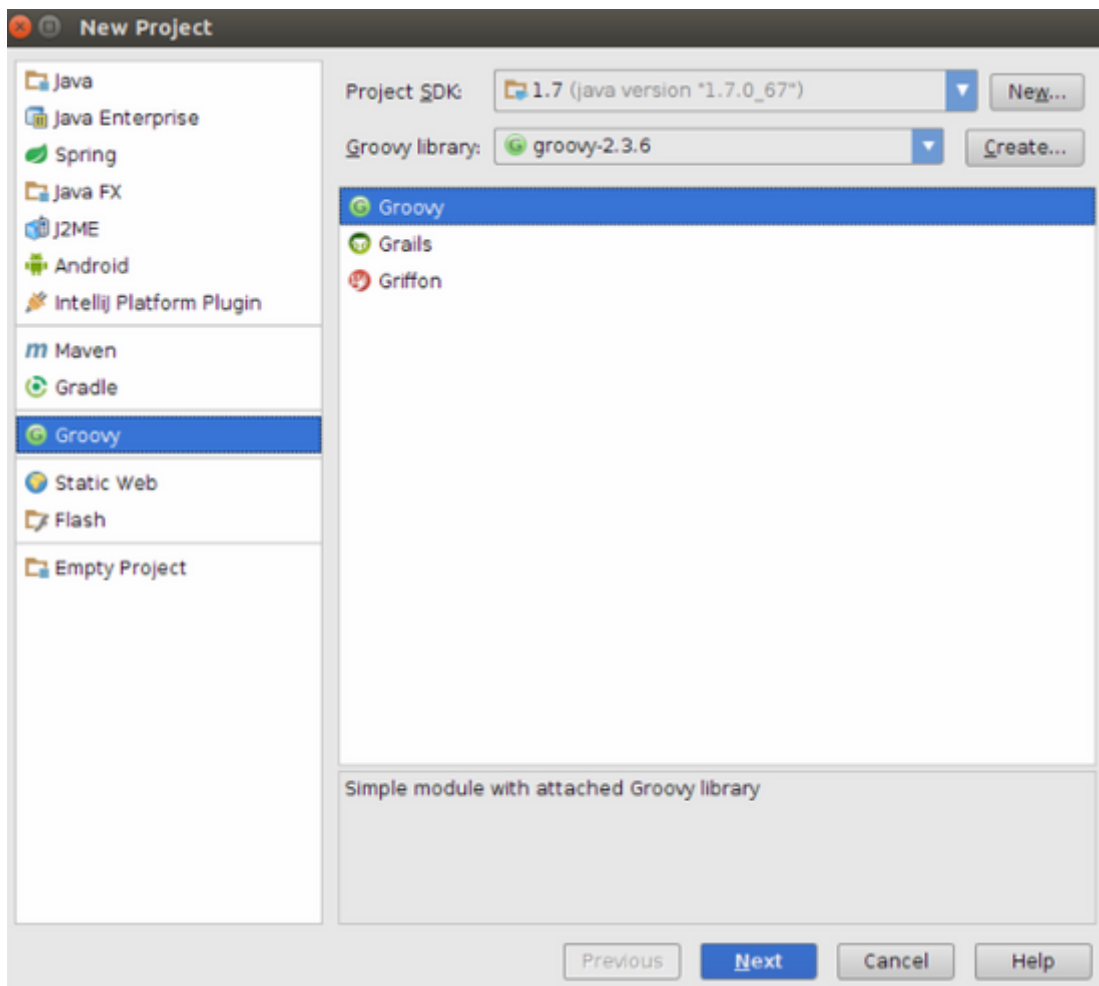
1. Enable the plugin

Before you create your first Groovy application make sure Groovy plugin is enabled in Settings → Plugins.

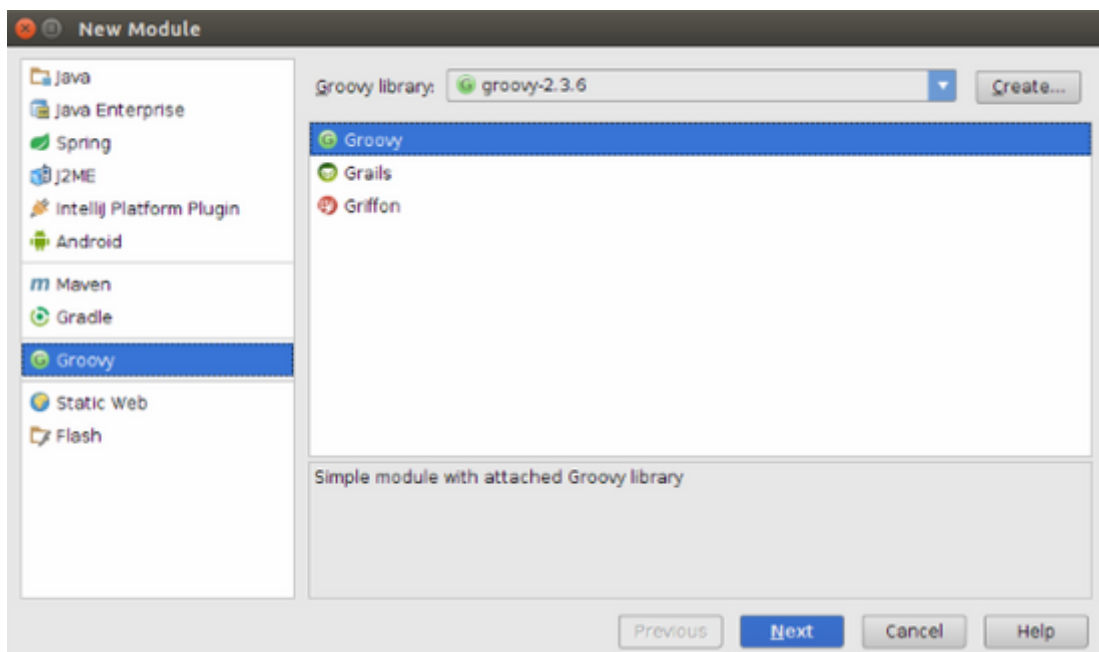


2. Create a new project

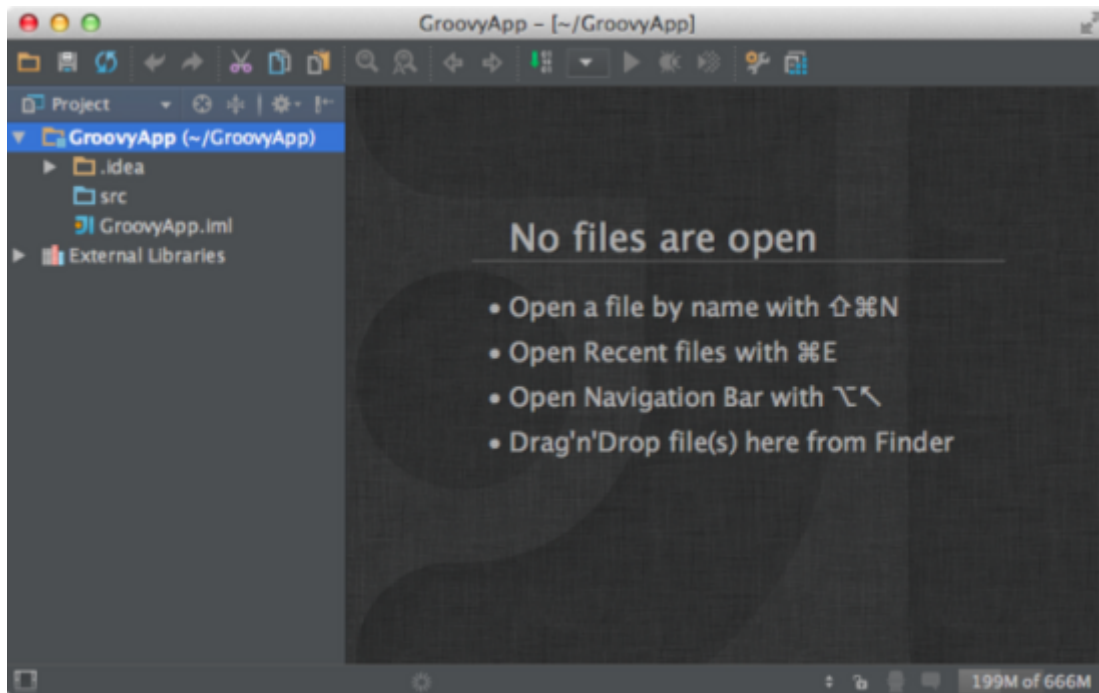
Open Project Wizard and select Scala template. Since Groovy requires Java you have to specify the Project SDK.



If you create a Groovy project for the first time IntelliJ IDEA will offer you to create Groovy SDK library. Press Create button and choose directory with a Groovy SDK.

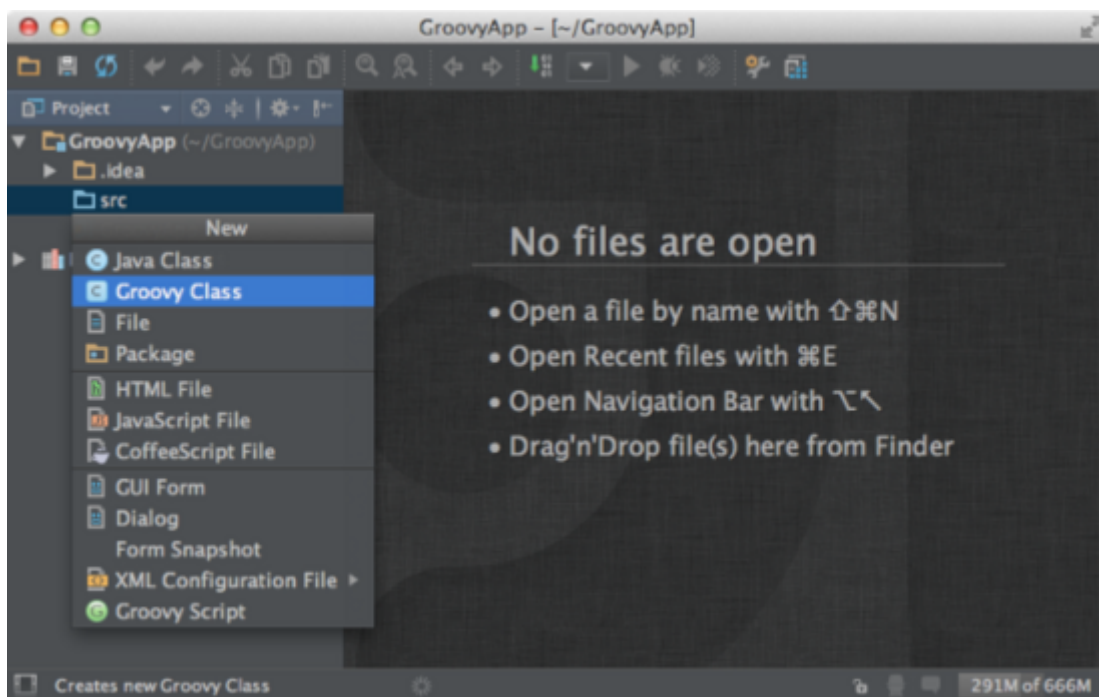


The IDE will create an empty project.

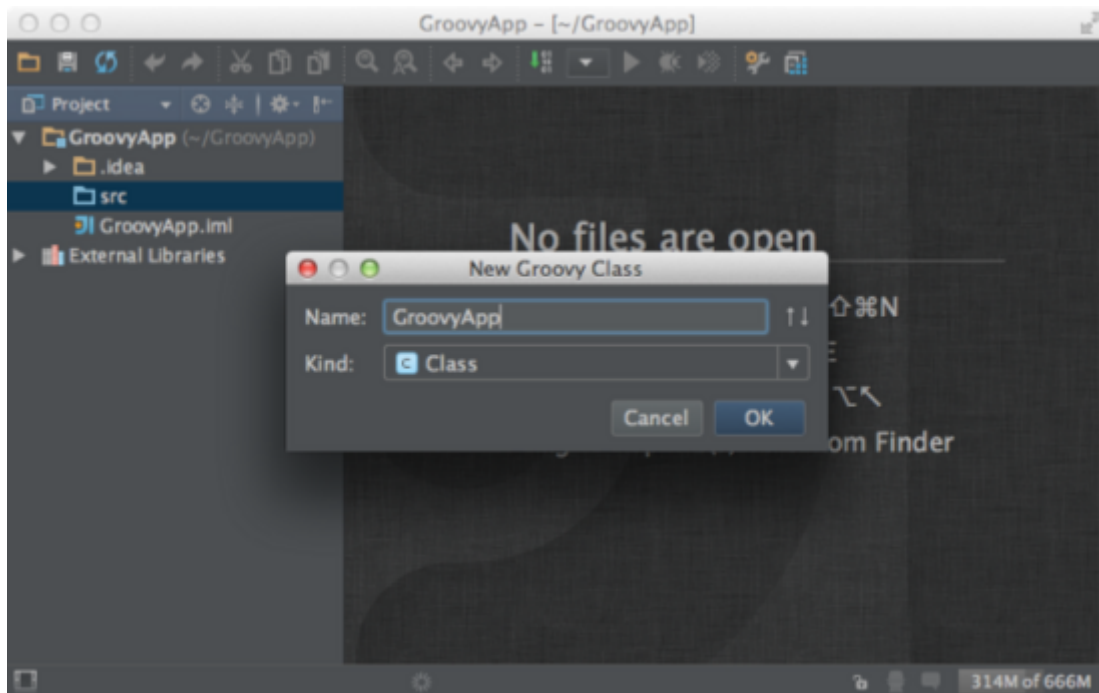


3. Create a new class

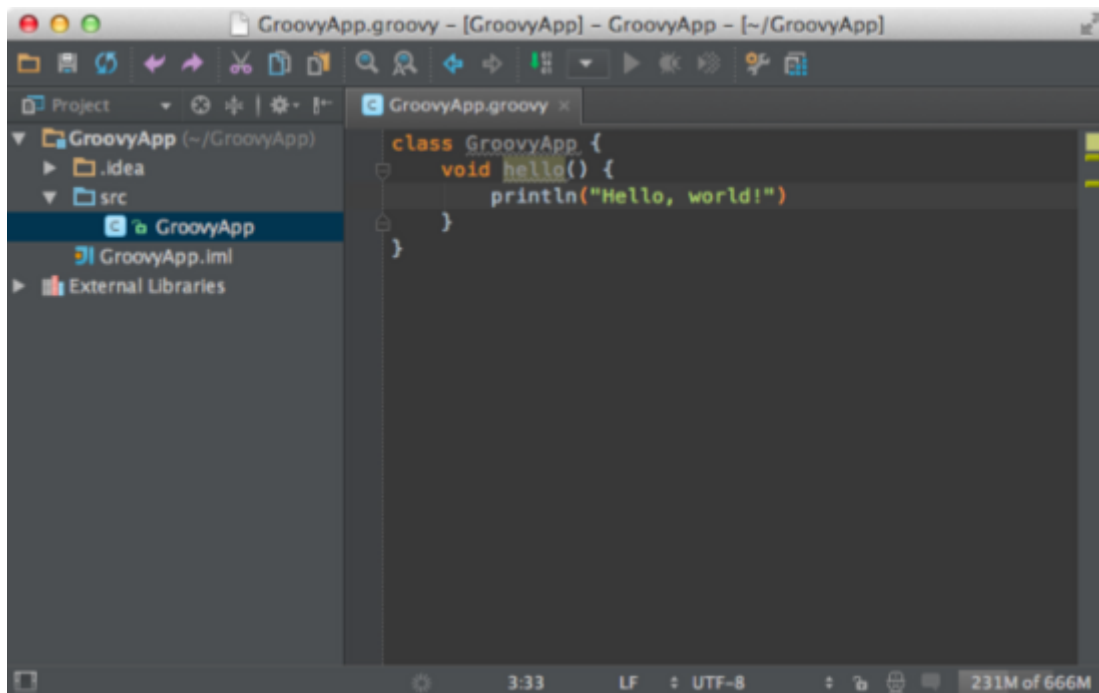
The easiest way to create Groovy class or script is to use `Ctrl + N` shortcut from Project View or Navigation Bar.



Choose between class, interface, enum and annotation with Up and Down arrows.

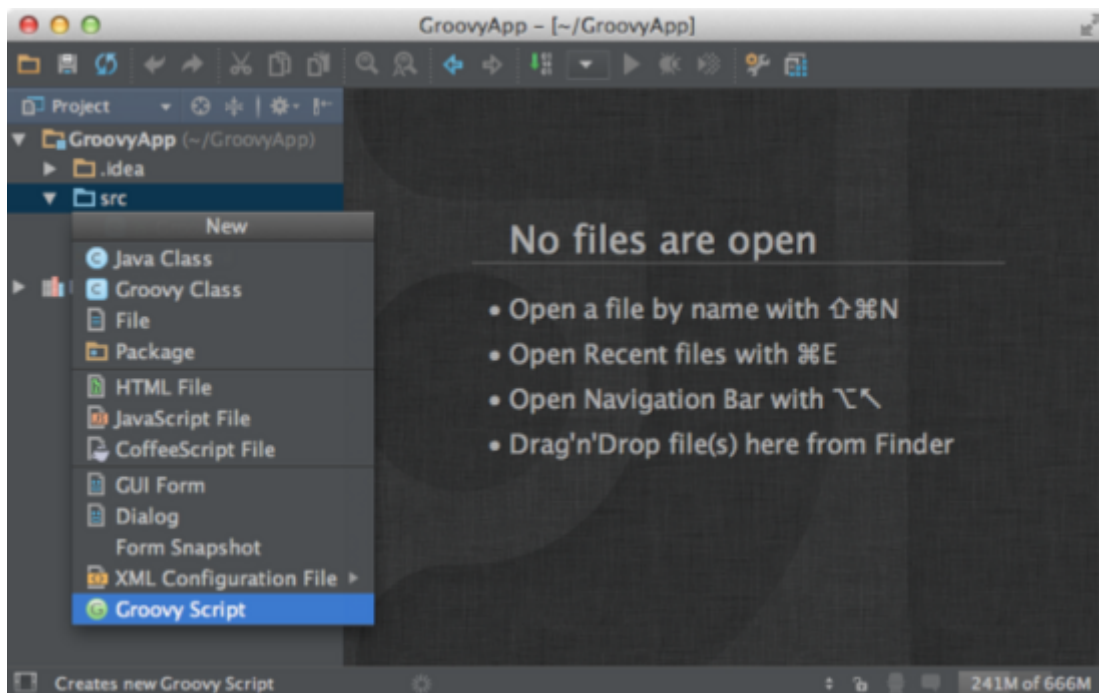


Let's create a class with a method returning "Hello, world!" string.

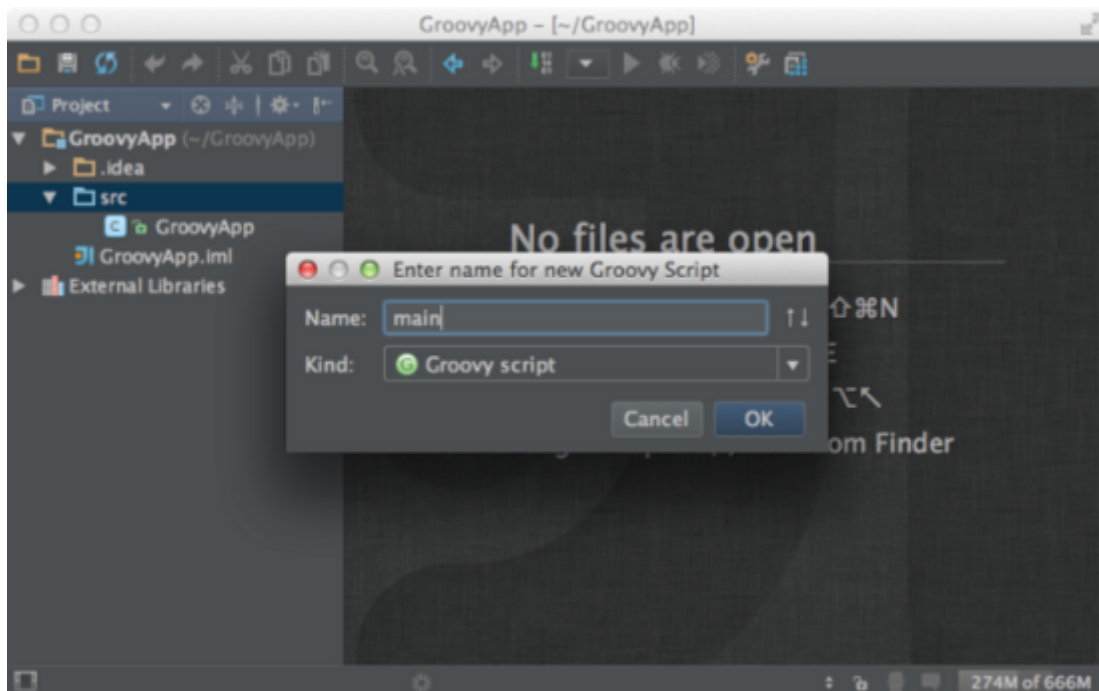


4. Create a new script

Now we can create a script file via Ctrl + N shortcut.



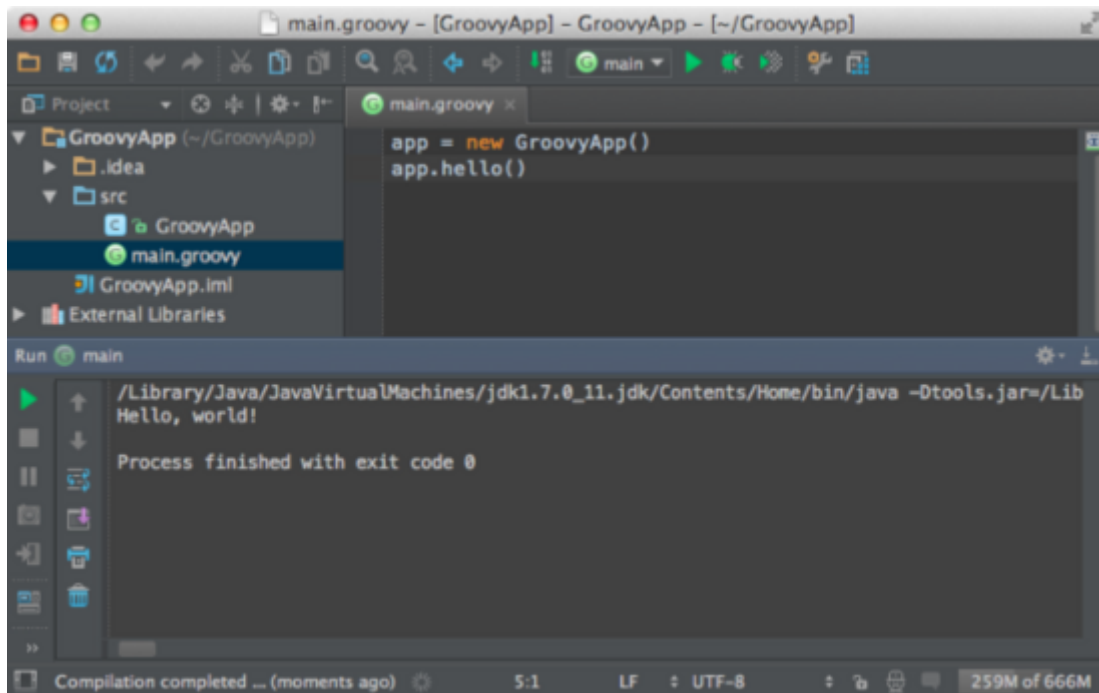
Choose between script and GroovyDSL script with Up and Down arrows.



Now we can create an instance of our class and invoke hello method.

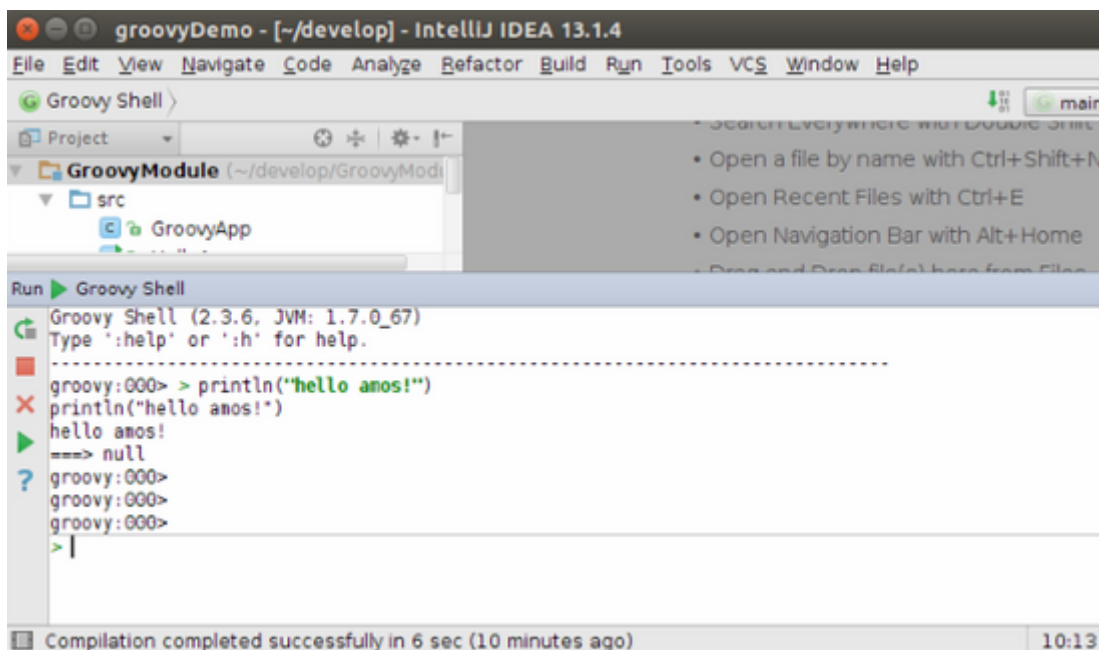
5. Run the project

In order to run the application you can manually create a Run configuration via Run → Edit configurations or run the active script automatically by pressing Ctrl + Shift + F10 shortcut.



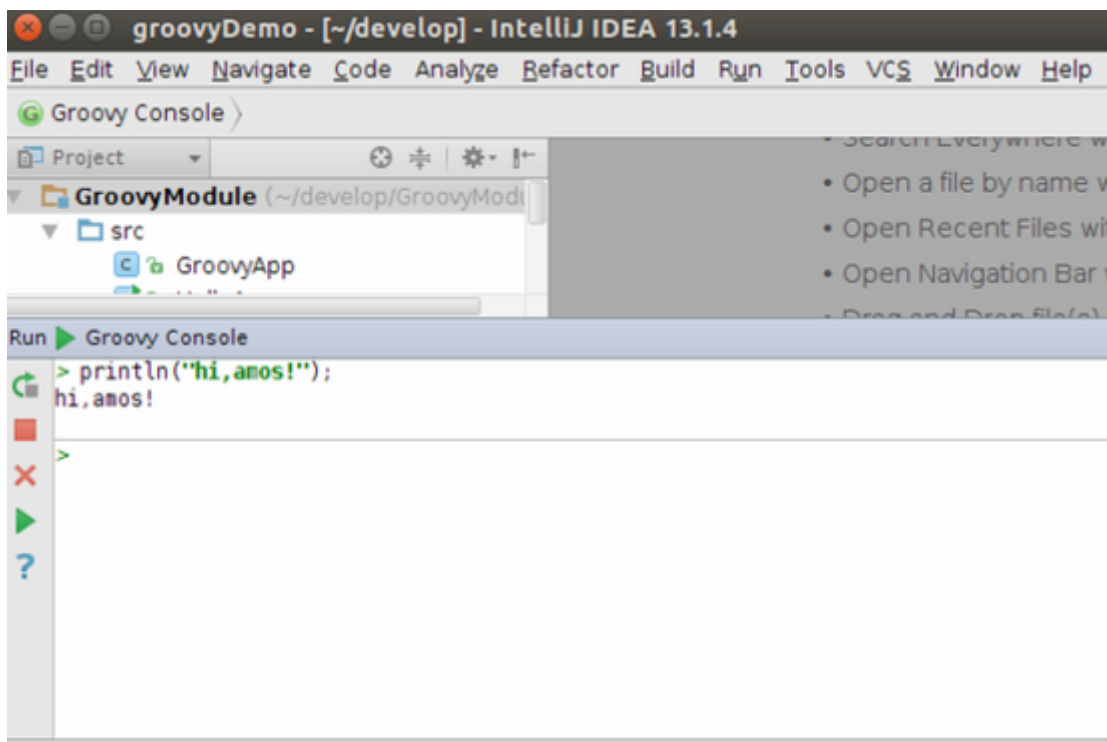
使用Groovy Shell :

打开 Tools-->Groovy Shell...



使用Groovy Console :

打开 Tools-->Groovy Console...



二、Groovy初探

1、Groovy和Java对比

- Groovy 的松散的 Java 语法允许省略分号和修改符。
- 除非另行指定，Groovy 的所有内容都为 `public`。
- Groovy 允许定义简单脚本，同时无需定义正规的 `class` 对象。
- Groovy 在普通的常用 Java 对象上增加了一些独特的方法和快捷方式，使得它们更容易使用。
- Groovy 语法还允许省略变量类型。
- 关于闭包：可以将 闭包 想像为一个代码块，可以现在定义，以后再执行。可以使用这些强大的构造做许多漂亮的事，不过最著名的是简化迭代。使用 Groovy 之后，就有可能再也不需要编写 `Iterator` 实例了。
- 动态的 Groovy：从技术上讲，Groovy 可能是您最近听说过的类型最松散的动态语言之一。从这个角度讲，Groovy 与 Java 语言的区别很大，Java 语言是一种固定类型语言。在 Groovy 中，类型是可选的，所以您不必输入 `String myStr = "Hello";` 来声明 `String` 变量。可以直接使用 `def` 进行不指定类型定义，类似于 js 中的 `var`。
- 与Java互用：用 Groovy 编写的任何内容都可以编译成标准的 Java 类文件并在 Java 代码中重用。类似地，用标准 Java 代码编写的内容也可以在 Groovy 中重用。

2、实例演示Java和Groovy的区别

用 Java 编写的 Hello World

用 Java 编写的典型的 Hello World 示例如下所示：

```
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello World!");  
    }  
}
```

编译和运行 Java 示例

在这个简单的 HelloWorld 类中，我省略了包，而且向控制台输出的时候没有使用任何多余的编码约定。下一步是用 `javac` 编译这个类，如下所示：

```
c:>javac HelloWorld.java
```

最后，运行经过编译的类：

```
c:>java HelloWorld
```

迄今为止还不错 — 很久以前就会编这么基础的代码了，所以这里只是回顾一下。下面，请看用 Groovy 编码的相同过程。

用 Groovy 编写的 Hello World

就像前面提到过的，Groovy 支持松散的 Java 语法 — 例如，不需要为打印 “Hello World!” 这样的简单操作定义类。

而且，Groovy 使日常的编码活动变得更容易，例如，Groovy 允许输入 `println`，而无需输入 `System.out.println`。当您输入 `println` 时，Groovy 会非常聪明地知道您指的是 `System.out`。

所以，用 Groovy 编写 Hello World 程序就如下面这样简单：

```
println "Hello World!"
```

请注意，在这段代码周围没有类结构，而且也没有方法结构！我还使用 `println` 代替了 `System.out.println`。

运行 Groovy 示例

假设我将代码保存在文件 `/home/amosli/developsoft/language/groovy/test/Hello.groovy` 内，只要输入以下代码就能运行这个示例：

```
amosli@amosli-ThinkPad:~/developsoft/language/groovy/groovy-2.3.6/bin$ ./groovy ../../test/Hello.groovy
Hello World!
```

如果已经配置了 groovy 的环境变量，那么只需要输入以下命令即可：

```
root@amosli-ThinkPad:/home/amosli/developsoft/language/groovy/test# groovy Hello.groovy
Hello World!
```

在控制台上输出 “Hello World!” 所需的工作就这么多。

更快捷的方式

```
amosli@amosli-ThinkPad:~/developsoft/language/groovy/groovy-2.3.6/bin$ ./groovy -e "println 'helloworld'"
helloworld
```

如果已经配置了 groovy 的环境变量，那么只需要输入以下命令即可：

```
root@amosli-ThinkPad:/home/amosli/developsoft/language/groovy/test# groovy -e "println 'helloworld'"
helloworld
```

这会生成相同的结果，而且甚至 无需定义任何文件 ！

3、Groovy 是没有类型的 Java 代码

很可能将 Groovy 当成是没有规则的 Java 代码。但实际上，Groovy 只是规则少一些。这一节的重点是使用 Groovy 编写 Java 应用程序时可以不用考虑的一个 Java 编程的具体方面：类型定义。

为什么要有类型定义？

在 Java 中，如果要声明一个 String 变量，则必须输入：

```
String value = "Hello World";
```

但是，如果仔细想想，就会看出，等号右侧的字符已经表明 value 的类型是 String。所以，Groovy 允许省略 value 前面的 String 类型变量，并用 def 代替。

```
def value = "Hello World"
```

实际上，Groovy 会根据对象的值来判断它的类型。

运行程序！

将 HelloWorld.groovy 文件中的代码编辑成下面这样：

```
String message = "Hello World"
println message.class //class java.lang.String
```

4、通过 Groovy 进行循环

方式1：

这里可以定义i为int 或者 def，或者不定义其类型

```
for(i = 0; i < 5; i++){
    println i
}
```

方式2：

使用in进行循环，其中..表示“到”，0..5 表示0到5，类似于0<=5;这里循环6次；

```
for(i in 0..5){
    println i
}
```

可以使用0..<5进行限定，类似于0<5, 循环5次；

5、Groovy中的集合

1)、Groovy 中的List

```
def range = 0..4
println range.class
assert range instanceof List
```

请注意，assert 命令用来证明范围是 java.util.List 的实例。接着运行这个代码，证实该范围现在是类型 List 的集合。

Groovy 的语法：

```
def coll = ["Groovy", "Java", "Ruby"]
assert coll instanceof Collection
```

```
assert coll instanceof ArrayList
```

你将会注意到，

coll

对象看起来很像 **Java** 语言中的数组。实际上，它是一个

Collection

。要在普通的 **Java** 代码中得到集合的相同实例，必须执行以下操作：

```
Collection<String> coll = new ArrayList<String>();
coll.add("Groovy");
coll.add("Java");
coll.add("Ruby");
```

在 **Java** 代码中，必须使用 `add()` 方法向 `ArrayList` 实例添加项。

而Groovy中则提供了3种方法：

```
coll.add("Python")
coll << "Smalltalk"
coll[5] = "Perl"
```

查找元素：

如果需从集合中得到某个特定项，可以通过像上面那样的位置参数获取项。例如，如果想得到第二个项 “Java”，可以编写下面这

```
assert coll[1] == "Java"
Groovy 还允许在集合中增加或去掉集合，如下所示：
def numbers = [1,2,3,4]
assert numbers + 5 == [1,2,3,4,5]
assert numbers - [2,3] == [1,4]
```

Groovy中的特殊方法：

```
def numbers = [1,2,3,4]
assert numbers.join(",") == "1,2,3,4"
assert [1,2,3,4,3].count(3) == 2
```

`join()` 和 `count()` 只是在任何项List上都可以调用的众多方便方法中的两个。分布操作符（spread operator）是个特别方便的工具，使用这个工具不用在集合上迭代，就能够调用集合的每个项上的方法。

假设有一个 `String` 列表，现在想将列表中的项目全部变成大写，可以编写以下代码：

```
assert ["JAVA", "GROOVY"] ==
["Java", "Groovy"]*.toUpperCase()
```

请注意 `*` 标记。对于以上列表中的每个值，都会调用 `toUpperCase()`，生成的集合中每个 `String` 实例都是大写的。

2) Groovy中的Map

Java 语言中的映射是名称-值对的集合。所以，要用 **Java** 代码创建典型的映射，必须像下面这样操作：

```
Map<String, String>map = new HashMap<String, String>();
map.put("name", "Andy");
map.put("VPN-#", "45");
```

Groovy 使得处理映射的操作像处理列表一样简单 — 例如，可以用 Groovy 将上面的 Java 映射写成

```
def hash = [name:"Andy", "VPN-#":45]
```

请注意，Groovy 映射中的键不必是 `String`。在这个示例中，`name` 看起来像一个变量，但是在幕后，Groovy 会将它变成 `String`。

验证hash格式：

```
assert hash.getClass() == java.util.LinkedHashMap
```

Groovy 中Hash的Set/Get

```
//方法1
hash.put("id", 23)
assert hash.get("name") == "Andy"

//方法2
hash.dob = "01/29/76"
//. 符号还可以用来获取项。例如，使用以下方法可以获取 dob 的值：
assert hash.dob == "01/29/76"

//方法3
assert hash["name"] == "Andy"
hash["gender"] = "male"
assert hash.gender == "male"
assert hash["gender"] == "male"
```

请注意，在使用 `[]` 语法从映射获取项时，必须将项作为 `String` 引用。

6、Groovy 中的闭包

Java 的 `Iterator` 实例，用它在集合上迭代，就像下面这样：

```
def acoll = ["Groovy", "Java", "Ruby"]

for(Iterator iter = acoll.iterator(); iter.hasNext();){
    println iter.next()
}
```

请注意，`each` 直接在 `acoll` 实例内调用，而 `acoll` 实例的类型是 `ArrayList`。在 `each` 调用之后，引入了一种新的语法 — `{`，然后是一些代码，然后是 `}`。由 `{}` 包围起来的代码块就是闭包。

```
def acoll = ["Groovy", "Java", "Ruby"]

acoll.each{
    println it
}
```

闭包中的 `it` 变量是一个关键字，指向被调用的外部集合的每个值 — 它是默认值，可以用传递给闭包的参数覆盖它。下面的代码执行同样的操作，但使用自己的项变量：

```
def acoll = ["Groovy", "Java", "Ruby"]

acoll.each{ value ->
    println value
}
```

在这个示例中，用 `value` 代替了 Groovy 的默认 `it`。

```
def hash = [name:"Andy", "VPN-#":45]
hash.each{ key, value ->
```

```
println "${key} : ${value}"
}
```

请注意，闭包还允许使用多个参数 — 在这个示例中，上面的代码包含两个参数（`key` 和 `value`）。

请记住，凡是集合或一系列的内容，都可以使用下面这样的代码进行迭代。

```
> "amosli".each{
println it.toUpperCase();
}
```

```
A
M
O
S
L
I
```

```
def excite = {
word-> return "this is ${word} "
};
```

这段代码是名为 `excite` 的闭包。这个闭包接受一个参数（名为 `word`），返回的 `String` 是 `word` 变量加两个感叹号。请注意在 `String` 实例中替换 的用法。在 `String` 中使用 `${value}` 语法将告诉 Groovy 替换 `String` 中的某个变量的值。可以将这个语法当成 `return word + "!!"` 的快捷方式。

```
//可以通过两种方法调用闭包：直接调用或者通过 call() 方法调用。
excite("Java");
excite.call("Groovy")
```

输出：this is Groovy

7、Groovy 中的类

新建一个类song：

```
class Song {
    def name
    def artist
    def genre
}
```

```
class SongExample {
    static void main(args) {
        def sng = new Song(name:"Le Freak",
            artist:"Chic", genre:"Disco")
    }
}
```

Groovy 自动提供一个构造函数，构造函数接受一个名称-值对的映射，这些名称-值对与类的属性相对应。这是 Groovy 的一项开箱即用的功能 — 用于类中定义的任何属性，Groovy 允许将存储了大量值的映射传给构造函数。映射的这种用法很有意义，例如，您不用初始化对象的每个属性。

也可以添加下面这样的代码：

```
def sng2 = new Song(name:"Kung Fu Fighting", genre:"Disco")
```

也可以像下面这样直接操纵类的属性：

```
def sng3 = new Song()
sng3.name = "Funkytown"
sng3.artist = "Lipps Inc."
```



```
sng3.setGenre("Disco")

assert sng3.getArtist() == "Lipps Inc."
```

在 Song 类中，添加以下代码：

```
String toString() {
    "${name}, ${artist}, ${genre}"
}
```

8、Groovy中的单元测试

```
@Test
public void test() {

    def sng2 = new Song(name:"Kung Fu Fighting", genre:"Disco")

    println sng2.getArtist();

}
```

在IntelliJ 中只需要加入@Test注解就可以使用JUnit 测试

加个?可以防止空指针的错误：

```
def getArtist() {
    artist?. toUpperCase();
}
```

9、扩展

如果需要用Groovy做web 项目的话可以去了解一下Grails框架。

本文源码：<https://github.com/amosli/groovy>

参考：

1. <http://confluence.jetbrains.com/display/IntelliJIDEA/Groovy>
2. <http://www.ibm.com/developerworks/cn/education/java/j-groovy/j-groovy.html>
3. <http://groovy.codehaus.org/>
4. <http://confluence.jetbrains.com/display/IntelliJIDEA/Getting+Started+with+Grails>



分享

收藏

纠错



专访：听道哥聊互联网江湖

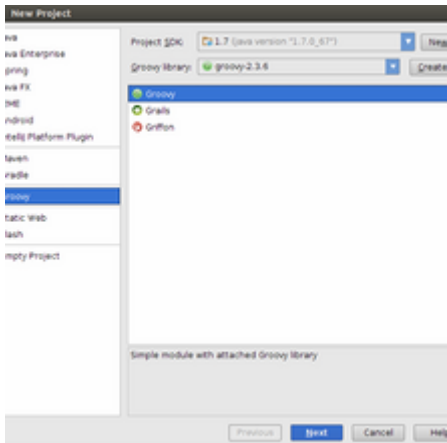
“根据判断，下一个被黑产盯上的行业很有可能是直播行业。现在来看，这件事情已经发生了。”

推荐文章

- 1. [无意中发 Android 5.0 LruCache 的 bug](#)

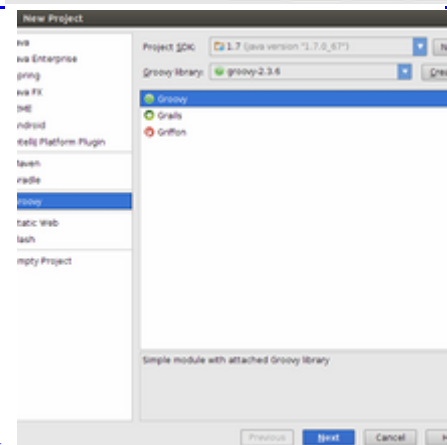
- 2. [高并发Java（1）：前言](#)
- 3. [Java代码优化](#)
- 4. [【译】Brave Clojure 第十二章:与JVM共事](#)
- 5. [记一次ClassLoader死锁，及近距离感受R大](#)
- 6. [Java注解处理器从入门到出门](#)

相关推刊

- 


by [tonyZ14](#)

[《Groovy编程》](#)

6
- 

by [netman44](#)

[《groovy》](#)

1
- 

by [oldratlee](#)

[《默认推刊》](#)

136

我来评几句

请输入评论内容...

登录后评论

已发表评论数 (0)

相关站点

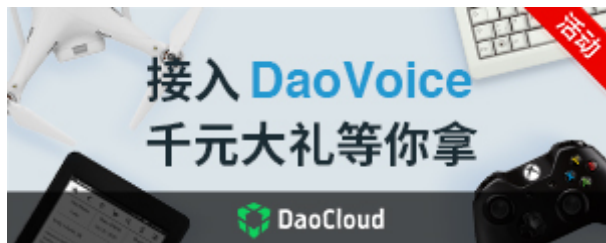
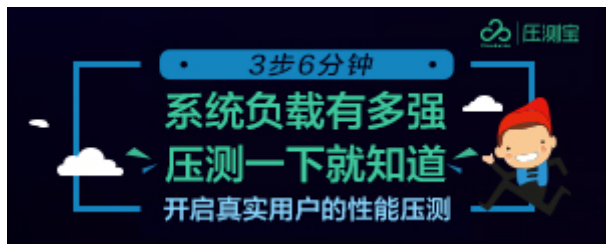


[博客园-原创精华区](#)

[+ 订阅](#)

[热门文章](#)

- 1. [无意中发現 Android 5.0 LruCache 的 bug](#)
- 2. [高并发Java \(1\)：前言](#)
- 3. [Java代码优化](#)
- 4. [【译】Brave Clojure 第十二章:与JVM共事](#)
- 5. [记一次ClassLoader死锁，及近距离感受R大](#)



收藏到推刊

[创建推刊](#)

收藏取消

推刊名(必填)

请填写推刊名

推刊描述

描述不能大于100个字符!

权限设置：☒ 公开 ☐ 仅自己可见

创建取消

x

文章纠错

邮箱地址

错误类型

正文不准确 ▼

补充信息

提交

网站相关

- [关于我们](#)
- [移动应用](#)
- [建议反馈](#)

关注我们

[推酷网](#)

tuicool2012

友情链接

- [人人都是产品经理](#) [PM256](#) [移动信息化](#) [行晓网](#) [智城外包网](#) [虎嗅](#) [IT耳朵](#) [创媒工场](#) [经理人分享](#) [市场部网](#) [砍柴网](#) [CocoaChina](#) [北风网](#) [云智慧](#) [我赢职场](#) [大数据时代](#) [奇笛网](#) [咕噜网](#) [红联](#) [linux](#) [Win10之家](#) [鸟哥笔记](#) [爱游戏](#) [投资潮](#) [31会议网](#) [极光推送](#) [Teambition](#) [硅谷网](#) [leangoo](#) [ZEALER中国](#) [OpenSNS](#) [小牛学堂](#) [handone](#) [Scrum中文网](#) [比戈大牛](#) [又拍云](#) [更多链接>>](#)