# Gradle实践

## 文档结构

Gradle课程资料

有用软件

配套好书

第2节 gradle快速入门

第3节 开发环境搭建

第4节 Gradle核心语法讲解及实战

第5节 Gradle高级用法实战

第6节 Gradle生命周期探索

第7节 Gradle核心之Project详解及实战

第8节 Gradle核心之Task详解及实战

第9节 Gradle其它模块讲解与自定义Plugin实战

第10节 Gradle持续集成与打包

第11节 课程总结

第2节 gradle快速入门

2-1 本节概述

2-2 什么是领域特定语言DSL？

2-3 groovy初探

第4节 Gradle核心语法讲解及实战

4.1 本节概述

4.10 列表学习(上)

4.11 列表学习(下)

4.12 映射详解(上)

4.13 映射详解(下)

4.16 面向对象学习(中)

4.17 面向对象学习(下)

4.18 本节小结

4.2 基础语法讲解

4.3 String讲解

4.4 字符串方法详解

4.5 逻辑控制

4.6 闭包讲解之基础讲解

4.7 闭包使用讲解

4.8 字符串与闭包结合使用

4.9 闭包进阶讲解

## 第2节Gradle快速入门

### 2.1 什么是领域特定语言DSL

Uml、matlab、xml、html、

### 2.2 groovy初探

是一种基于JVM的敏捷开发语言

结合了Python、Ruby、Smalltalk的许多强大的特性

Groovy可以与java完美结合，而且可以使用java所有的类库

思想：求转不求全，解决特定问题

语法上支持动态类型、闭包等新一代语言

面向对象、面向过程

## 第3节 开发环境搭建

### 3.1 linux下环境搭建

1）安装JDK；

2）下载Groovy SDK

$unzip apache-groovy-binary-2.4.15.zip

$yum install unzip

$sudo yum install unzip

$unzip apache-groovy-binary-2.4.15.zip

$ln -s groovy-2.4.15/ groovy

$rm -rf apache-groovy-binary-2.4.15.zip

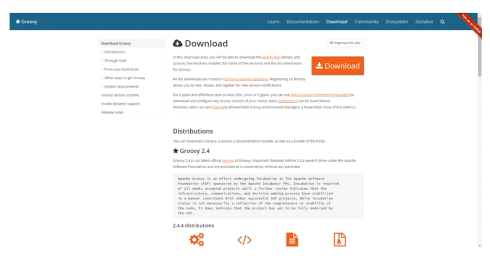
$sudo vi /etc/profile

$source /etc/profile

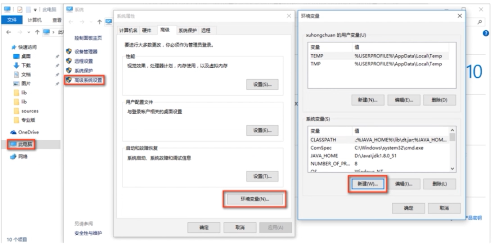
$groovy -v

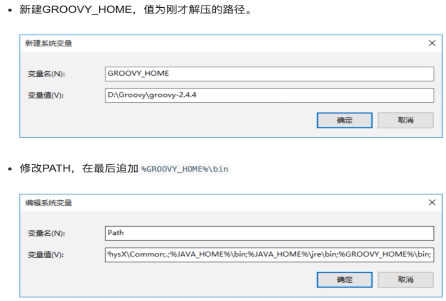
### 3.2 windows下环境搭建

1）下载Groovy SDK

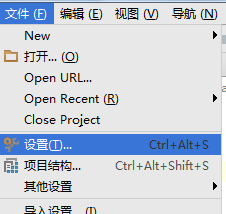


1. 配置classpath环境变量

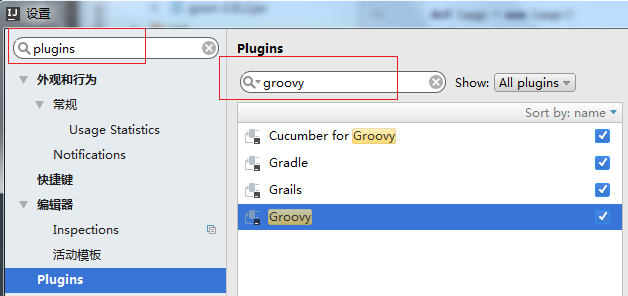




1. 在控制台输入groovy -version，校验是否正确安装
2. IntelliJ中配置groovy
3. 打开idea--preferences--

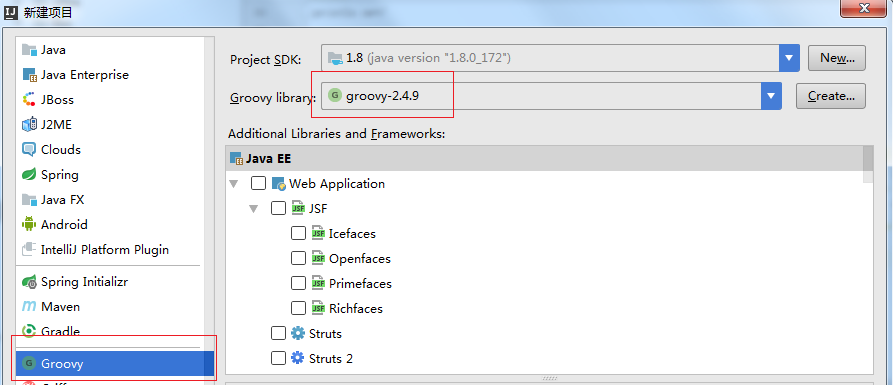


2.搜索plugins，在plugins中搜索groovy

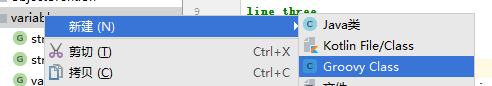


### 3.3 groovy工程创建

1）新建项目，选择Groovy，并选择本地解压的Groovy SDK根目录



1. 新建groovy class



3）输入println “Hello Groovy”，直接运行

在Windows下的cmd窗口中运行grails RunApp命令，但是报错了，错误如上，而且在开发工具中是可以正常启动的，找了很多原因都不对，最后都重新装系统了，后来发现了问题：在cmd的编码会影响命令行的运行，默认情况下cmd的编码是gbk，我改成了utf-8，才导致了这个问题，真的是太纠结了，下次引以为戒吧！附改cmd编码的命令：

ERROR: JAVA\_HOME is set to an invalid directory: D:\\*\*\soft\jdk1.8\jdk

Please set the JAVA\_HOME variable in your environment to match the

location of your Java installation.

Please set the JAVA\_HOME variable in your environment to match the

location of your Java installation.

chcp 查看当前编码

chcp 65001 就是换成UTF-8代码页，还需要点击边框右键-属性-字体-改为Lucida Console

chcp 936 可以换回默认的GBK

chcp 437 是美国英语

## 第4节 Gradle核心语法讲解及实战

### 4.1 基础语法

1）数据类型，def定义弱类型的变量

|  |
| --- |
| **package** variable  **int** x = 10  println x.class  **double** y = 3.14 println y.class  **def** x\_1 = 11 println x\_1.class **def** y\_1 = 3.1415 println y\_1.class **def** name = **'Qndroid'** println name.class  x\_1 = **'Test'** println x\_1.class |
| **class java.lang.Integer**  **class java.lang.Double**  **class java.lang.Integer**  **class java.math.BigDecimal**  **class java.lang.String**  **class java.lang.String** |

2）集合处理

|  |
| --- |
| **def** color = [**red**: **'ff0000'**, **green**: **'00ff00'**, **blue**: **'0000ff'**] color.**yellow** = **'ffff00'** *//添加* println color[**'red'**] println color.**green** println color.getClass() println color.**yellow** *//注意key的取值,key做为变量时特殊处理* **def** pink = **'pink'** color.(pink) = **'ff00ff'** println color.toMapString() **def** map = [**a**: 1, **b**: 2] color.**complex** = map println color.toMapString() |
| **ff0000**  **00ff00**  **class java.util.LinkedHashMap**  **ffff00**  **[red:ff0000, green:00ff00, blue:0000ff, yellow:ffff00, pink:ff00ff]**  **[red:ff0000, green:00ff00, blue:0000ff, yellow:ffff00, pink:ff00ff, complex:[a:1, b:2]]** |

### 4.2 String

1）引号的使用

|  |
| --- |
| **def** name = **'a single \'a\'string'** println name.class  **def** thupleName = **'''\ line one line two line three '''** println thupleName  **def** doubleName = **"this a common String"** println doubleName.class |
| **class java.lang.String**  **line one**  **line two**  **line three**  **class java.lang.String** |

2）可扩展表达式

|  |
| --- |
| **def** name1 = **"Qndroid" def** sayHello = **"Hello:** ${name1}**"** println sayHello println sayHello.class  **def** sum = **"the sum of 2 and 3 equals** ${2 + 3}**"** *//可扩展做任意的表达式* println sum **def** result = echo(sum) println result.class  String echo(String message) {  **return** message } |
| **Hello: Qndroid**  **class org.codehaus.groovy.runtime.GStringImpl**  **the sum of 2 and 3 equals 5**  **class java.lang.String**  **groovy Hello**  **groovy Hello** |

3）字符串方法

|  |
| --- |
| */\* ==================字符串的方法=================== \*/* **def** str = **"groovy Hello"** println str.center(8) println str.padLeft(8, **'a'**) **def** str2 = **'Hello'** println str > str2 println str[0] println str[0..1] println str - str2  println str.reverse() println str.capitalize() |
| **true**  **g**  **gr**  **groovy**  **olleH yvoorg**  **Groovy Hello** |

|  |
| --- |
| **def** str = **"groovy"** println str.center(11,**'a'**) println str.padLeft(11,**'b'**) println str.padRight(11,**'c'**)  **def** str2 = **"oo"** println str < str2  println str.getAt(0) println str[0..2]  println str.minus(str2) println str.reverse() println str.capitalize() |
| **aagroovyaaa**  **bbbbbgroovy**  **groovyccccc**  **true**  **g**  **gro**  **grvy**  **yvoorg**  **Groovy** |

### 4.3 逻辑控制

1）switch..case

|  |
| --- |
| **def** x = 1.23 **def** result **switch** (x) {  **case 'foo'**:  result = **'found foo'  break  case 'bar'**:  result = **'bar'  break  case** [1.23, 4, 5, 6, **'inlist'**]: *//列表* result = **'list'  break  case** 12..30:  result = **'range'** *//范围* **break  case** Integer:  result = **'integer'  break  case** BigDecimal:  result = **'big decimal'  break  default**: result = **'default'** }  println result |
| **list** |

2）for循环

|  |
| --- |
| *//对范围的for循环* **def** sum = 0 **for** (i **in** 0..9) {  sum += i } println sum  */\*\*  \* 对List的循环  \*/* **for** (i **in** [1, 2, 3, 4, 5, 6, 7, 8, 9]) {  sum += i } |
| *45* |

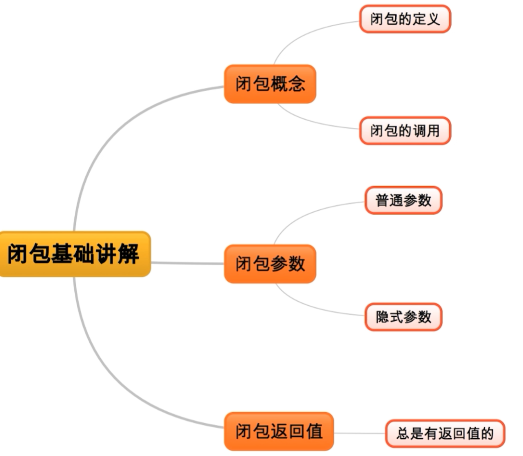
3）Map循环

|  |
| --- |
| */\*\*  \* 对Map进行循环  \*/* **for** (i **in** [**'lili'**: 1, **'luck'**: 2, **'xiaoming'**: 3]) {  sum += i.value } |

4）n的阶层，闭包

|  |
| --- |
| println calculate(5) **def** calculate(**int** number) {  **def** result = 1  1.upto(number) { num ->  result \*= num  }  **return** result } |
| 120 |

### 4.4 闭包基础



1）demo

|  |
| --- |
| **def** clouser = { String name,**int** age -> println **"Hello Groovy!** ${name}**, age is** ${age}**"**} clouser(**"byf"**,23) clouser.call(**'byf'**,23) |
| **Hello Groovy! byf, age is 23**  **Hello Groovy! byf, age is 23** |

2）递归

|  |
| --- |
| **int** x = cal(101) **int** fab(**int** number){  **int** result = 1;  1.upto(number,{num -> result \*= num})  **return** result } println x  **int** fab2(**int** number){  **int** result = 1;  number.downto(1){  num -> result \*=num  }  **return** result } **int** cal(**int** number){  **int** result  number.times {num -> result += num}  **return** result } |
| **5050** |

3）this、owner、delegate

|  |
| --- |
| **def** sscriptClouser = {  println **"this: "** + **this** println **"owner: "**+ owner  println **"delegate: "**+delegate } sscriptClouser.call() |
| **this: variable.HelloGroovy@49d904ec**  **owner: variable.HelloGroovy@49d904ec**  **delegate: variable.HelloGroovy@49d904ec** |

4）闭包嵌套

|  |
| --- |
| **class** Person{  **def Clouser** = {  println **"this: "** + **this** println **"owner: "**+ owner  println **"delegate: "**+delegate  }   **def** say(){  **def** classClouser = {  println **"this: "** + **this** println **"owner: "**+ owner  println **"delegate: "**+delegate  }  classClouser.call()  } } Person p = **new** Person() p.Clouser.call() p.say() **def** nestClouser = {  **def** innerClouser = {  println **"this: "** + **this** println **"owner: "**+ owner  println **"delegate: "**+delegate  }  innerClouser.delegate = p  innerClouser.call() } nestClouser.call() |
| **this: variable.Person@10e41621**  **owner: variable.Person@10e41621**  **delegate: variable.Person@10e41621**  **this: variable.Person@10e41621**  **owner: variable.Person@10e41621**  **delegate: variable.Person@10e41621**  **this: variable.HelloGroovy@3daa422a**  **owner: variable.HelloGroovy$\_run\_closure1@27ce24aa**  **delegate: variable.Person@10e41621** |

5）闭包的委托策略

|  |
| --- |
| */\* 闭包的委托策略  \*/* **class** Student {  String **name  def pretty** = {**"My name is** ${**name**}**"**}  String toString(){  **pretty**.call()  } }  **class** Teacher {  String **name** } **def** stu = **new** Student(**name**: **'Sarash'**) **def** tea = **new** Teacher(**name**:**'Ondriod'**) stu.pretty.delegate = tea *//stu.pretty.resolveStrategy = Closure.DELEGATE\_ONLY //stu.pretty.resolveStrategy = Closure.DELEGATE\_FIRST* stu.pretty.resolveStrategy = Closure.***OWNER\_FIRST*** println stu.toString() |
| *//My name is Ondriod*  *//My name is Ondriod*  *My name is Sarash* |

### 4.5 字符串与闭包结合使用

1）判断、查找

|  |
| --- |
| String str = **'the 2 and 3 is 5'** str.each {  String tmp -> print tmp.multiply(2) } println str.each {}  println str.find{  String s->s.isNumber() }  **def** list = str.findAll{  String s->s.isNumber() } println list.toListString()  **def** result = str.any{  String s->s.isNumber() } println result  result = str.every {  String s->s.isNumber() } println result |
| tthhee 22 aanndd 33 iiss 55the 2 and 3 is 5  2  [2, 3, 5]  true  false |

2）字符串方法

|  |
| --- |
| String str = **'the 2 and 3 is 5' def** list2 = str.collect {  it.toUpperCase() } println list2 |
| [T, H, E, , 2, , A, N, D, , 3, , I, S, , 5] |

3）字符串数组

|  |
| --- |
| **def** list=[1,2,3,4,5] println list.class println list.size() **def** array = [1,2,3,4,5] **as int**[] **int**[] array2 = [1,2,3,4,5] println array.class println array2.class |
| **class java.util.ArrayList**  **5**  **class [I**  **class [I** |

4）排序、遍历

|  |
| --- |
| **def** sortList = [2,3,4,5,2,3,5,5,12,1] Comparator mc = {  a,b -> a==b ?0:Math.abs(a) <Math.abs(b)? 1 : -1 } *//Collections.sort(sortList,mc)* sortList.sort(mc) println sortList  **def** sortStringList = [**'aaaa'**,**'bb'**,**'ccc'**] sortStringList.sort{it -> **return** it.size()} println sortStringList  **int** result = sortList.find {**return** it %2 == 0} println result **def** ret = sortList.findAll {**return** it%2==0} println ret.max() println ret.min() |
| **[12, 5, 5, 5, 4, 3, 3, 2, 2, 1]**  **[bb, ccc, aaaa]**  **12**  **12**  **2** |

### 4.6 列表学习

1）list的定义、添加、删除

|  |
| --- |
| **package** datastruct  *//def list = new ArrayList() //java的定义方式* **def** list = [1, 2, 3, 4, 5] println list.class println list.size() **def** array = [1, 2, 3, 4, 5] **as int**[] **int**[] array2 = [1, 2, 3, 4, 5]  */\*\*  \* list的添加元素  \*/* list.add(6) list.leftShift(7) list << 8 println list.toListString() **def** plusList = list + 9 println plusList.toListString()  */\*\*  \* list的删除操作  \*/* list.remove(7) list.remove((Object) 6) list.removeAt(5) list.removeElement(4) list.removeAll { **return** it % 2 == 0 } println list - [6, 7] println list.toListString() |
| **class java.util.ArrayList**  **5**  **[1, 2, 3, 4, 5, 6, 7, 8]**  **[1, 2, 3, 4, 5, 6, 7, 8, 9]**  **[1, 3, 5]**  **[1, 3, 5]** |

2）列表的排序

|  |
| --- |
| */\*\*  \* 列表的排序  \*/* **def** sortList = [6, -3, 9, 2, -7, 1, 5] Comparator mc = { a, b ->  a == b ? 0 :  Math.abs(a) < Math.abs(b) ? -1 : 1 } Collections.*sort*(sortList, mc) sortList.sort { a, b ->  a == b ? 0 :  Math.*abs*(a) < Math.*abs*(b) ? 1 : -1 } println sortList **def** sortStringList = [**'abc'**, **'z'**, **'Hello'**, **'groovy'**, **'java'**] sortStringList.sort { it -> **return** it.size() } println sortStringList |
| *class java.util.ArrayList*  *5*  *[9, -7, 6, 5, -3, 2, 1]*  *[z, abc, java, Hello, groovy]* |

3）列表的查找

|  |
| --- |
| */\*\*  \* 列表的查找  \*/* **def** findList = [-3, 9, 6, 2, -7, 1, 5] **int** result = findList.find { **return** it % 2 == 0 } *//def result = findList.findAll { return it % 2 != 0 }* println result *//def result = findList.any { return it % 2 != 0 } //def result = findList.every { return it % 2 == 0 }* println result println findList.min { **return** Math.*abs*(it) } println findList.max { **return** Math.*abs*(it) } **def** num = findList.count { **return** it % 2 == 0 } println num |
| *class java.util.ArrayList*  *5*  *6*  *6*  *1*  *9*  *2* |

### 4.7 映射详解

1）映射的定义

|  |
| --- |
| **package** datastruct  *//def map = new HashMap()* **def** colors = [**red** : **'ff0000'**,  **green**: **'00ff00'**,  **blue** : **'0000ff'**] *//索引方式* println colors[**'red'**] println colors.**red** colors.**blue** *//添加元素* colors.**yellow** = **'ffff00'** colors.**complex** = **[a**: 1, **b**: 2**]** println colors.getClass() |
| **ff0000**  **ff0000**  **class java.util.LinkedHashMap** |

2）Map的遍历

|  |
| --- |
| */\*\*  \* Map操作详解  \*/* **def** students = [  1: [**number**: **'0001'**, **name**: **'Bob'**,  **score** : 55, **sex**: **'male'**],  2: [**number**: **'0002'**, **name**: **'Johnny'**,  **score** : 62, **sex**: **'female'**],  3: [**number**: **'0003'**, **name**: **'Claire'**,  **score** : 73, **sex**: **'female'**],  4: [**number**: **'0004'**, **name**: **'Amy'**,  **score** : 66, **sex**: **'male'**] ]  *//遍历Entry* students.each { **def** student ->  println **"the key is** ${student.key}**, "** +  **" the value is** ${student.value}**"** } *//带索引的遍历* students.eachWithIndex { **def** student, **int** index ->  println **"index is** ${index}**,the key is** ${student.key}**, "** +  **" the value is** ${student.value}**"** } *//直接遍历key-value* students.eachWithIndex { key, value, index ->  println **"the index is** ${index}**,the key is** ${key}**, "** +  **" the value is** ${value}**"** } |
| *the key is 1, the value is [number:0001, name:Bob, score:55, sex:male]*  *the key is 2, the value is [number:0002, name:Johnny, score:62, sex:female]*  *the key is 3, the value is [number:0003, name:Claire, score:73, sex:female]*  *the key is 4, the value is [number:0004, name:Amy, score:66, sex:male]*  *index is 0,the key is 1, the value is [number:0001, name:Bob, score:55, sex:male]*  *index is 1,the key is 2, the value is [number:0002, name:Johnny, score:62, sex:female]*  *index is 2,the key is 3, the value is [number:0003, name:Claire, score:73, sex:female]*  *index is 3,the key is 4, the value is [number:0004, name:Amy, score:66, sex:male]*  *the index is 0,the key is 1, the value is [number:0001, name:Bob, score:55, sex:male]*  *the index is 1,the key is 2, the value is [number:0002, name:Johnny, score:62, sex:female]*  *the index is 2,the key is 3, the value is [number:0003, name:Claire, score:73, sex:female]*  *the index is 3,the key is 4, the value is [number:0004, name:Amy, score:66, sex:male]* |

2）Map的查找

|  |
| --- |
| *//Map的查找* **def** entry = students.find { **def** student ->  **return** student.value.**score** >= 60 } println entry  **def** entrys = students.findAll { **def** student ->  **return** student.value.**score** >= 60 } println entrys  **def** count = students.count { **def** student ->  **return** student.value.**score** >= 60 &&  student.value.**sex** == **'male'** } println count **def** names = students.findAll { **def** student ->  **return** student.value.**score** >= 60 }.collect {  **return** it.value.**name** } println names.toListString()  **def** group = students.groupBy { **def** student ->  **return** student.value.**score** >= 60 ? **'及格'** : **'不及格'** } println group.toMapString() |
| *2={number=0002, name=Johnny, score=62, sex=female}*  *[2:[number:0002, name:Johnny, score:62, sex:female], 3:[number:0003, name:Claire, score:73, sex:female], 4:[number:0004, name:Amy, score:66, sex:male]]*  *1*  *[Johnny, Claire, Amy]*  *[不及格:[1:[number:0001, name:Bob, score:55, sex:male]], 及格:[2:[number:0002, name:Johnny, score:62, sex:female], 3:[number:0003, name:Claire, score:73, sex:female], 4:[number:0004, name:Amy, score:66, sex:male]]]* |

3）Map的排序

|  |
| --- |
| */\*\*  \* 排序  \*/* **def** sort = students.sort { **def** student1, **def** student2 ->  Number score1 = student1.value.**score** Number score2 = student2.value.**score  return** score1 == score2 ? 0 : score1 < score2 ? -1 : 1 }  println sort.toMapString() |
| *[1:[number:0001, name:Bob, score:55, sex:male], 2:[number:0002, name:Johnny, score:62, sex:female], 4:[number:0004, name:Amy, score:66, sex:male], 3:[number:0003, name:Claire, score:73, sex:female]]* |

### 4.8范围详解

|  |
| --- |
| **package** datastruct  **def** range = 1..10 println range[0] println range.contains(10) println range.from println range.to |
| 1  true  1  10 |

|  |
| --- |
| **def** result = getGrade(75) println result  **def** getGrade(Number number) {  **def** result  **switch** (number) {  **case** 0..<60:  result = **'不及格'  break  case** 60..<70:  result = **'及格'  break  case** 70..<80:  result = **'良好'  break  case** 80..100:  result = **'优秀'  break** }   **return** result } |
| **良好** |

### 4.9面向对象

定义一个Person

|  |
| --- |
| **package** objectorention  */\*\*  \* 1.groovy中默认都是public  \*/* **class** Person **implements** Serializable {  String **name** Integer **age  def** increaseAge(Integer years) {  **this**.**age** += years  }  */\*\*  \* 一个方法找不到时，调用它代替  \** ***@param*** *name  \** ***@param*** *args  \** ***@return*** *\*/* **def** invokeMethod(String name, Object args) {  **return "the method is** ${name}**, the params is** ${args}**"** }  **def** methodMissing(String name, Object args) {  **return "the method** ${name} **is missing"** } } |

定义一个脚本

|  |
| --- |
| **package** objectorention  *//def person = new Person(name: 'Qndroid', age: 26) //println person.cry()* ExpandoMetaClass.*enableGlobally*() *//为类动态的添加一个属性* Person.metaClass.sex = **'male' def** person = **new** Person(**name**: **'Qndroid'**, **age**: 26) println person.sex person.sex = **'female'** println **"the new sex is:"** + person.sex *//为类动态的添加方法* Person.metaClass.sexUpperCase = { -> sex.toUpperCase() } **def** person2 = **new** Person(**name**: **'Qndroid'**, **age**: 26) println person2.sexUpperCase() *//为类动态的添加静态方法* Person.metaClass.static.createPerson = {  String name, **int** age -> **new** Person(**name**: name, **age**: age) } **def** person3 = Person.createPerson(**'renzhiqiang'**, 26) println person3.name + **" and "** + person3.age |
| **male**  **the new sex is:female**  **MALE**  **renzhiqiang and 26** |

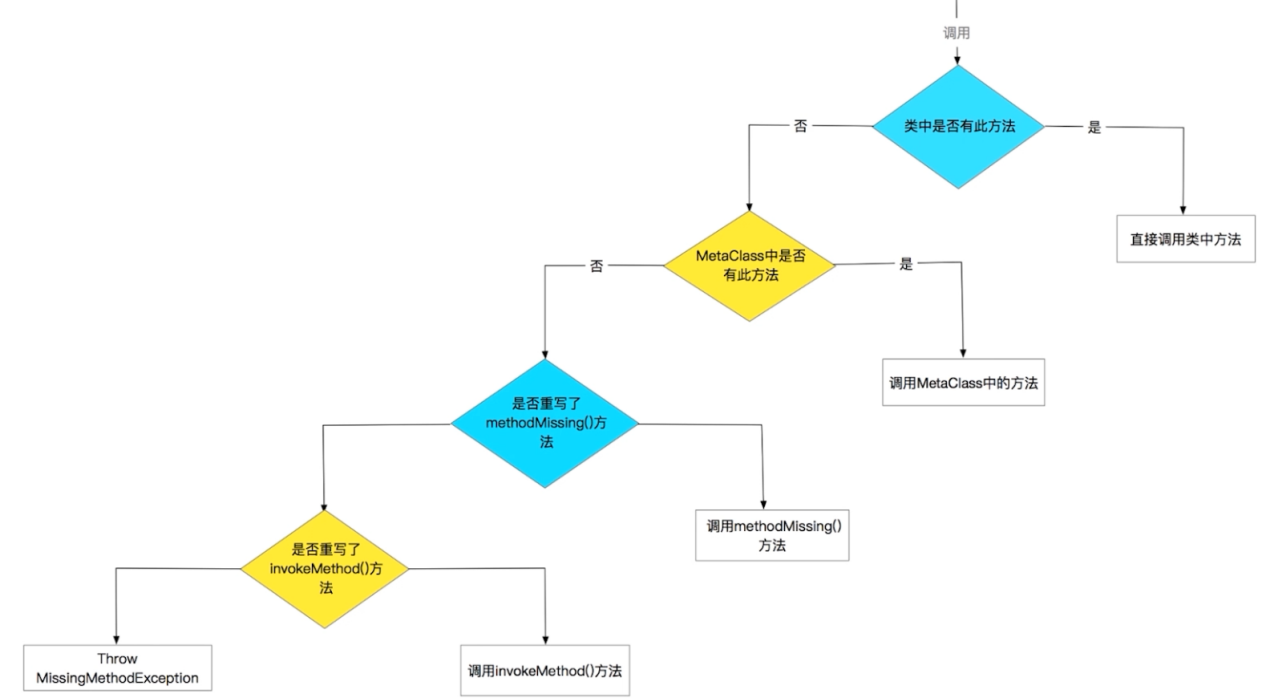
定义一个接口

|  |
| --- |
| **package** objectorention */\*\*  \* 接口中不许定义非public的方法  \*/* **interface** Action {  **void** eat()  **void** drink()  **void** play() } |

定义一个trait接口（适配器模式）

|  |
| --- |
| **trait** DefualtAction {  **abstract void** eat()  **void** play() {  println **' i can play.'** } } |

Groovy的运行时编译



定义一个Person管理器

|  |
| --- |
| **package** objectorention.expand  **import** objectorention.Person  */\*\*  \* Person类的管理器  \*/* **class** PersonManager {  **static** Person createPerson(String name, **int** age) {  **return** Person.createPerson(name, age)  } } |

定义一个应用的init

|  |
| --- |
| **package** objectorention.expand  **import** objectorention.Person  */\*\*  \* 模拟一个应用的管理类  \*/* **class** ApplicationManager {   **static void** init() {  ExpandoMetaClass.*enableGlobally*()  *//为第三方类添加方法* Person.metaClass.static.createPerson = { String name, **int** age ->  **new** Person(**name**: name, **age**: age)  }  } } |

定义一个Main

|  |
| --- |
| **package** objectorention.expand  **class** Entry {  **static void** main(**def** args) {  println **'应用程序正在启动...'** *//初始化* ApplicationManager.*init*()  println **'应用程序初始化完成...'   def** person = PersonManager.  *createPerson*(**'renzhiqiang'**, 26)  println **"the person name is** ${person.name} **"** +  **"and the age is** ${person.age}**"** } } |
| **应用程序正在启动...**  **应用程序初始化完成...**  **the person name is renzhiqiang and the age is 26** |

## 第5节 Gradle高级用法实战

### 5.1JSON操作详解

1）JsonOutput、JsonSplur

|  |
| --- |
| **package** file  **import** groovy.json.JsonOutput **import** groovy.json.JsonSlurper **import** objectorention.Person   **def** list = [**new** Person(**name**:**'xiaohong'**,**age**: 23),  **new** Person(**name**:**'xiaoming'**,**age**: 24) ] **def** json = JsonOutput.*toJson*(list)  **def** jsonSlpur = **new** JsonSlurper() println jsonSlpur.parseText(json) println json println JsonOutput.*prettyPrint*(json) |
| **[[age:23, name:xiaohong], [age:24, name:xiaoming]]**  **[{"age":23,"name":"xiaohong"},{"age":24,"name":"xiaoming"}]**  **[**  **{**  **"age": 23,**  **"name": "xiaohong"**  **},**  **{**  **"age": 24,**  **"name": "xiaoming"**  **}**  **]** |

2）从jsonplaceholder获取json格式数据

|  |
| --- |
| **package** file  **import** groovy.json.JsonSlurper  **def** reponse =  getNetworkData(  **'http://jsonplaceholder.typicode.com/albums/6'**)  println reponse println JsonOutput.*prettyPrint*(JsonOutput.*toJson*(reponse))  **def** getNetworkData(String url) {  *//发送http请求* **def** connection = **new** URL(url).openConnection()  connection.setRequestMethod(**'GET'**)  connection.connect()  **def** response = connection.content.text  *//将json转化为实体对象* **def** jsonSluper = **new** JsonSlurper()  **return** jsonSluper.parseText(response) } |
| **[userId:1, id:6, title:natus impedit quibusdam illo est]**  **[userId:1, id:6, title:natus impedit quibusdam illo est]**  **{**  **"userId": 1,**  **"id": 6,**  **"title": "natus impedit quibusdam illo est"**  **}** |

### 5.2xml文件解析详解

1. 解析一个xml文件

|  |
| --- |
| **package** file  **import** groovy.xml.MarkupBuilder  **final** String xml = **'''  <response version-api="2.0">  <value>  <books id="1" classification="android">  <book available="20" id="1">  <title>疯狂Android讲义</title>  <author id="1">李刚</author>  </book>  <book available="14" id="2">  <title>第一行代码</title>  <author id="2">郭林</author>  </book>  </books>  <books id="2" classification="web">  <book available="10" id="1">  <title>Vue从入门到精通</title>  <author id="4">李刚</author>  </book>  </books> </value>  </response> '''** *//开始解析此xml数据* **def** xmlSluper = **new** XmlSlurper() **def** response = xmlSluper.parseText(xml)  println response.value.books[0].book[0].title.text() println response.value.books[0].book[0].author.text() println response.value.books[1].book[0].@available |
| **疯狂Android讲义**  **李刚**  **10** |

遍历xml上的节点

|  |
| --- |
| **def** list = [] response.value.books.each { books ->  *//下面开始对书结点进行遍历* books.book.each { book ->  **def** author = book.author.text()  **if** (author.equals(**'李刚'**)) {  list.add(book.title.text())  }  } }  println list.toListString() |
| **[疯狂Android讲义, Vue从入门到精通]** |

深度遍历xml数据.depthFirst(). 等价于 .'\*\*'.

|  |
| --- |
| *//深度遍历xml数据.depthFirst(). 等价于 .'\*\*'.* **def** titles = response.depthFirst().findAll { book ->  **return** book.author.text() == **'李刚'** ? **true** : **false** } println titles.toListString() |
| **[疯狂Android讲义李刚, Vue从入门到精通李刚]** |

广度遍历xml上的节点 .children(). 等价于 .’\*’.

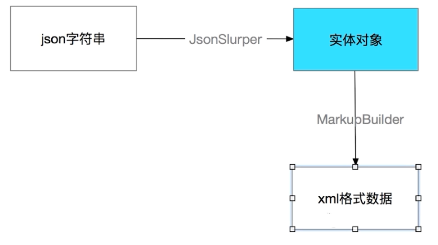
|  |
| --- |
| *//广度遍历xml数据* **def** name = response.value.books.children().findAll { node ->  node.name() == **'book'** && node.@id == **'2'** }.collect { node ->  **return** node.title.text() }  println name |
| **[第一行代码]** |

1. 创建一个xml文件

|  |
| --- |
| */\*\*  \* 生成xml格式数据  \* <langs type='current' count='3' mainstream='true'>  <language flavor='static' version='1.5'>Java</language>  <language flavor='dynamic' version='1.6.0'>Groovy</language>  <language flavor='dynamic' version='1.9'>JavaScript</language>  </langs>  \*/* **def** sw = **new** StringWriter() **def** xmlBuilder = **new** MarkupBuilder(sw) *//用来生成xml数据的核心类 //根结点langs创建成功* xmlBuilder.langs(**type**: **'current'**, **count**: **'3'**,  **mainstream**: **'true'**) {  *//第一个language结点* language(**flavor**: **'static'**, **version**: **'1.5'**) {  age(**'16'**)  }  language(**flavor**: **'dynamic'**, **version**: **'1.6'**) {  age(**'10'**)  }  language(**flavor**: **'dynamic'**, **version**: **'1.9'**, **'JavaScript'**) }  println sw |
| **<langs type='current' count='3' mainstream='true'>**  **<language flavor='static' version='1.5'>**  **<age>16</age>**  **</language>**  **<language flavor='dynamic' version='1.6'>**  **<age>10</age>**  **</language>**  **<language flavor='dynamic' version='1.9'>JavaScript</language>**  **</langs>** |

对象转xml

|  |
| --- |
| **def** langs = **new** Langs() xmlBuilder.langs(**type**: langs.type, **count**: langs.count,  **mainstream**: langs.mainstream) {  *//遍历所有的子结点* langs.languages.each { lang ->  language(**flavor**: lang.flavor,  **version**: lang.version, lang.value)  } } println sw *//对应xml中的langs结点* **class** Langs {  String **type** = **'current'  int count** = 3  **boolean mainstream** = **true  def languages** = [  **new** Language(**flavor**: **'static'**,  **version**: **'1.5'**, **value**: **'Java'**),  **new** Language(**flavor**: **'dynamic'**,  **version**: **'1.3'**, **value**: **'Groovy'**),  **new** Language(**flavor**: **'dynamic'**,  **version**: **'1.6'**, **value**: **'JavaScript'**)  ] } *//对应xml中的languang结点* **class** Language {  String **flavor** String **version** String **value** } |
| **<langs type='current' count='3' mainstream='true'>**  **<language flavor='static' version='1.5'>Java</language>**  **<language flavor='dynamic' version='1.3'>Groovy</language>**  **<language flavor='dynamic' version='1.6'>JavaScript</language>**  **</langs>** |



### 5.3groovy文件处理详解

1. 读取文件，闭包

|  |
| --- |
| **package** file  **import** objectorention.Person  **def** file = **new** File(**'hello.txt'**)  file.eachLine { line ->  println line } |
| **hello groovy**  **hello gradle**  **hello world** |

1. 读取文件，getText

|  |
| --- |
| **def** file = **new** File(**'hello.txt'**) **def** text = file.getText() println text |
| **hello groovy**  **hello gradle**  **hello world** |

1. 读取部分文件

|  |
| --- |
| *//读取文件部分内容* **def** reader = file.withReader { reader ->  **char**[] buffer = **new char**[20]  reader.read(buffer)  **return** buffer } println reader |
| *hello groovy*  *hello* |

1. 文件拷贝

|  |
| --- |
| **def** file = **new** File(**'hello.txt'**) **def** result1 = copy(**'hello.txt'** , **'hello2.txt'**) println result1  **def** copy(String sourcePath, String destationPath) {  **try** {  *//首先创建目标文件* **def** desFile = **new** File(destationPath)  **if** (!desFile.exists()) {  desFile.createNewFile()  }   *//开始copy* **new** File(sourcePath).withReader { reader ->  **def** lines = reader.readLines()  desFile.withWriter { writer ->  lines.each { line ->  writer.append(line + **"\r\n"**)  }  }  }  **return true** } **catch** (Exception e) {  e.printStackTrace()  }  **return false** } |
| **Hello2.txt**  hello groovy hello gradle hello world |

1. 保存对象到文件

|  |
| --- |
| **def** file = **new** File(**'hello.txt'**) **def** person = **new** Person(**name**: **'Qndroid'**, **age**: 26) saveObject(person, **'person.bin'**)   **def** saveObject(Object object, String path) {  **try** {  *//首先创建目标文件* **def** desFile = **new** File(path)  **if** (!desFile.exists()) {  desFile.createNewFile()  }  desFile.withObjectOutputStream { out ->  out.writeObject(object)  }  **return true** } **catch** (Exception e) {  }  **return false** } |
| **person.bin** |

1. 从person.bin中读取对象

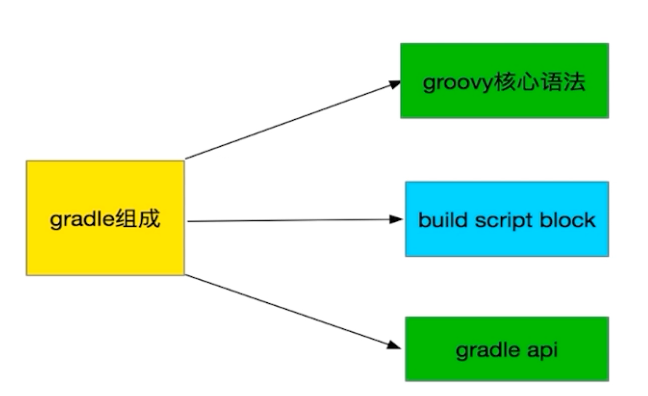
|  |
| --- |
| **def** file = **new** File(**'hello.txt'**)  **def** result3 = (Person) readObject(**'person.bin'**) println **"the name is** ${result3.name} **and the age is** ${result3.age}**"  def** readObject(String path) {  **def** obj = **null  try** {  **def** file = **new** File(path)  **if** (file == **null** || !file.exists()) **return null** *//从文件中读取对象* file.withObjectInputStream { input ->  obj = input.readObject()  }  } **catch** (Exception e) {   }  **return** obj } |
| **the name is Qndroid and the age is 26** |

注意：在闭包中读写文件，groovy帮助我们把流的关闭做了。

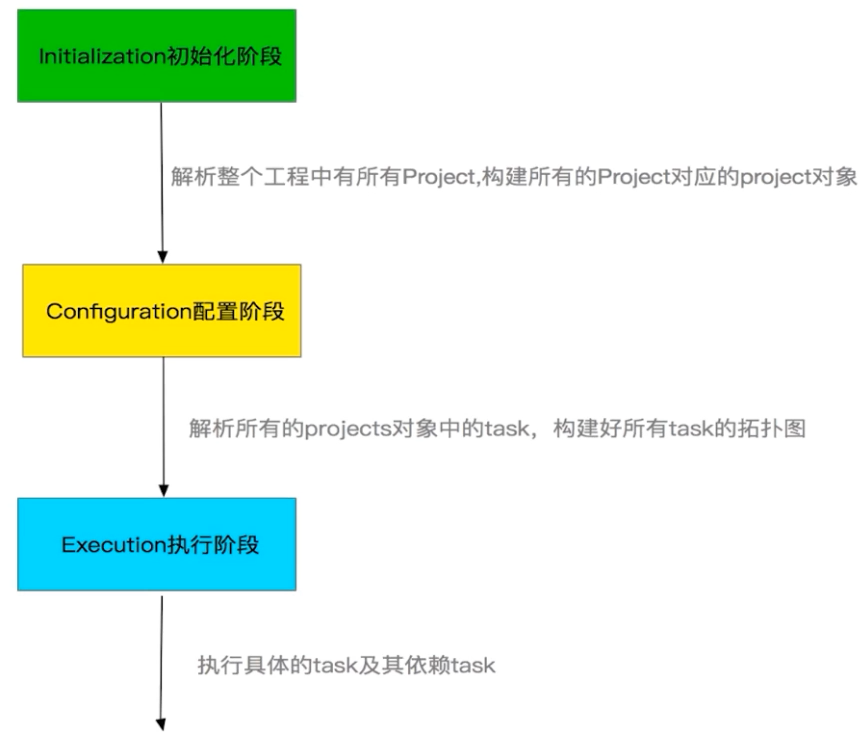
## 第6节 Gradle高级用法实战

### 6.1 gradle概念及优势

* 不仅仅是构建工具，也是一种编程框架
* 优势：灵活；粒度（tinker）；扩展性；兼容性；



### 6.2 gradle执行流程讲解



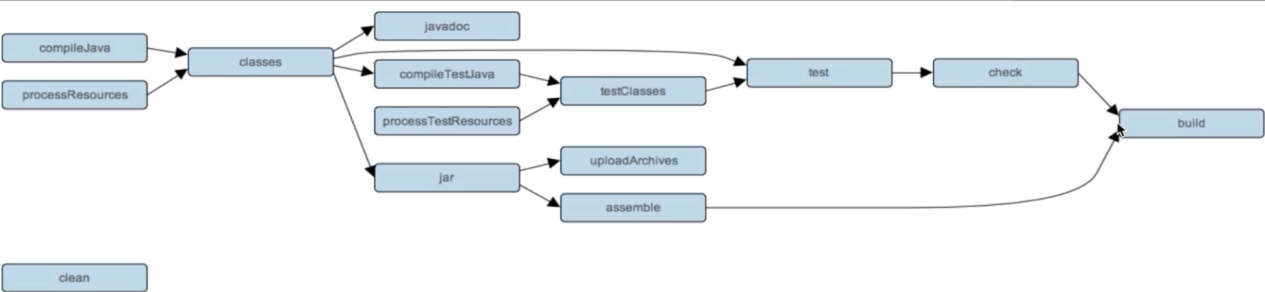
### 6.3 gradle生命周期监听

1）生命周期

|  |
| --- |
| */\*\*  \* 配置阶段开始前的监听回调  \*/* **this**.beforeEvaluate{  println **'beforeEvaluate：配置阶段执行开始...'** } */\*\*  \* 配置阶段完成以后的回调  \*/* **this**.afterEvaluate{  println **'afterEvaluate：配置阶段执行完毕...'** }  */\*\*  \*gradle执行完毕后的回调监听  \*/* **this**.gradle.buildFinished {  println **'buildFinished：执行阶段执行完毕...'** }  **this**.gradle.beforeProject {  println **'beforeProject：项目之前...'** } **this**.gradle.afterProject {  println **'afterProject：项目之后...'** } |
| **F:\groovy\_peojects\gradlelife>gradlew clean**  **settings.gradle：初始化阶段开始...**  **afterProject：项目之后...**  **afterEvaluate：配置阶段执行完毕...**  **:clean UP-TO-DATE**  **BUILD SUCCESSFUL**  **Total time: 1.759 secs**  **buildFinished：执行阶段执行完毕...** |

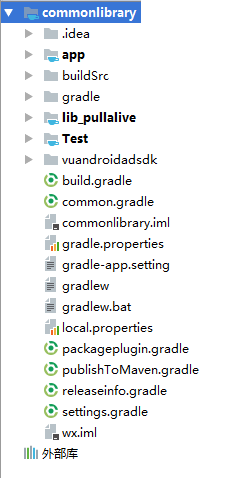
2）执行gradlew build的依赖，有向无环图

|  |
| --- |
| F:\groovy\_peojects\gradlelife>gradlew build  settings.gradle：初始化阶段开始...  afterProject：项目之后...  afterEvaluate：配置阶段执行完毕...  :compileJava UP-TO-DATE  :compileGroovy UP-TO-DATE  :processResources UP-TO-DATE  :classes UP-TO-DATE  :jar  :assemble  :compileTestJava UP-TO-DATE  :compileTestGroovy UP-TO-DATE  :processTestResources UP-TO-DATE  :testClasses UP-TO-DATE  :test UP-TO-DATE  :check UP-TO-DATE  :build  BUILD SUCCESSFUL  Total time: 2.041 secs  buildFinished：执行阶段执行完毕... |



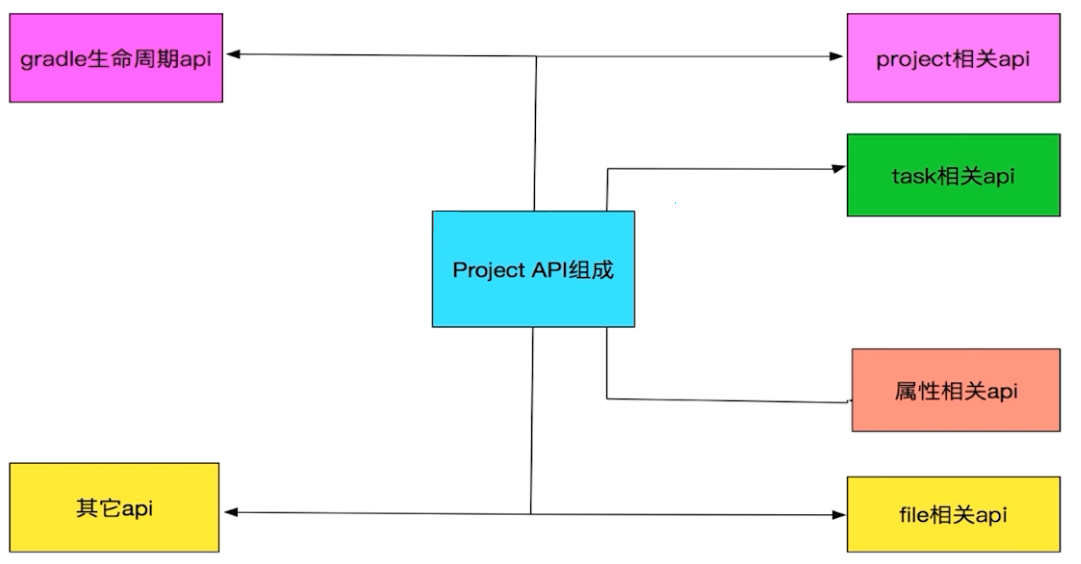
## 第7节 Gradle核心之Project详解及实战

### 7-1 project详解



对于gradle来说，每个带有build.gradle的目录都是一个project

### 7-2 project核心api分解



### 7.3 project相关api详解

1）遍历父子项目名称

|  |
| --- |
| *// project 相关api详解* **this**.getPropjects() **def** getPropjects() {  println **'------------'** println **'ROOT Projects'** println **'-------------'  this**.getAllprojects().eachWithIndex { Project project, **int** index ->  **if** (index == 0) {  println **"Root project ':**${project.name}**'"** } **else** {  println **"+--- project ':**${project.name}**'"** }  } } |
| **F:\groovy\_peojects\gradlelife>gradlew clean**  **settings.gradle：初始化阶段开始...**  **------------**  **ROOT Projects**  **-------------**  **Root project ':gradlelife'**  **+--- project ':app'**  **afterProject：项目之后...**  **afterEvaluate：配置阶段执行完毕...**  **beforeProject：项目之前...**  **afterProject：项目之后...**  **:clean UP-TO-DATE**  **:app:clean UP-TO-DATE**  **BUILD SUCCESSFUL**  **Total time: 2.553 secs**  **buildFinished：执行阶段执行完毕...** |

2）获取子工程的项目名称

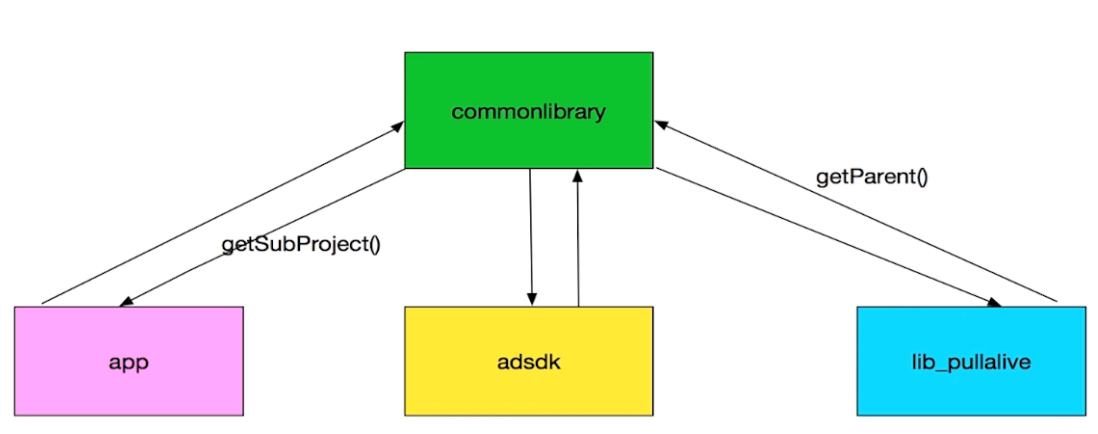
|  |
| --- |
| *// project 相关api详解* **this**.getSubPros() **def** getSubPros() {  println **'------------'** println **'Sub Projects'** println **'-------------'  this**.getSubprojects().eachWithIndex { Project project, **int** index ->  println **"Root project ':**${project.name}**'"** } } |
| **F:\groovy\_peojects\gradlelife>gradlew clean**  **------------**  **Sub Projects**  **-------------**  **Root project ':app'**  **:clean UP-TO-DATE**  **:app:clean UP-TO-DATE**  **BUILD SUCCESSFUL**  **Total time: 1.899 secs** |

3）在子工程app目录下获取父工程的名称

|  |
| --- |
| *// project 相关api详解* **this**.getParentPros() **def** getParentPros() {  println **'------------'** println **'Parent Projects'** println **'-------------'  this**.getParent().eachWithIndex { Project project, **int** index ->  println **"Parent project ':**${project.name}**'"** } } |
| F:\groovy\_peojects\gradlelife\app>gradle clean  > Configure project :  ------------  Sub Projects  -------------  Root project ':app'  > Configure project :app  ------------  Parent Projects  -------------  Parent project ':gradlelife'  BUILD SUCCESSFUL in 1s  1 actionable task: 1 up-to-date |

4）获取根工程的名称

|  |
| --- |
| *// project 相关api详解* **this**.getRootPro() **def** getRootPro() {  **def** name1 = **this**.getRootProject().name  println **"the root project name is :**${name1}**"** } |
| **F:\groovy\_peojects\gradlelife\app>gradle clean**  **> Configure project :**  **------------**  **Sub Projects**  **-------------**  **Root project ':app'**  **> Configure project :app**  **the root project name is :gradlelife**  **BUILD SUCCESSFUL in 1s**  **1 actionable task: 1 up-to-date** |



5）设置项目的属性

|  |
| --- |
| */\*\*  \*project api 详解  \*/* project(**'app'**) { Project project ->  *//println project  //apply plugin: 'com.byf.gradle'* group **'com.byf'** version **'1.0-SNAPSHOT'** dependencies{}  *//andriod{}* } |

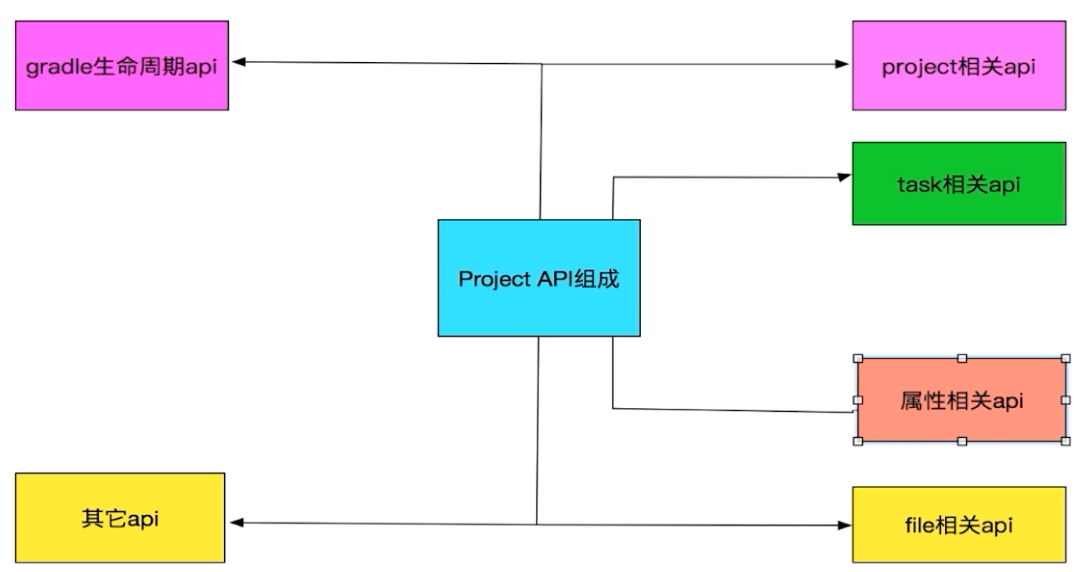
6）为所有子工程设置group

|  |
| --- |
| allprojects {  group **'com.byf'** version **'1.0.0'** } println project(**'test'**).group |
| com.byf  :clean UP-TO-DATE  :app:clean UP-TO-DATE  :test:clean UP-TO-DATE  BUILD SUCCESSFUL  Total time: 2.112 secs |

7）为子工程引入一个写好的脚本

|  |
| --- |
| subprojects { Project project ->  *//引入一个写好的脚本* **if** (project.plugins.hasPlugin(**'com.byf.libary'**))  apply **from**: **'../publishToMaven.gradle'** } |

### 7.4 属性相关api讲解



1）为所有子工程定义全局变量

|  |
| --- |
| **def** mNumber = 25 **def** mLib = **'com.byf.gradle'** *// 定义扩展属性 root模块* subprojects {  ext {  number = 25  lib = **'com.byf.gradle'** } } |
| // 子app模块  println **this**.number println **this**.lib |
| **F:\groovy\_peojects\gradlelife>gradlew clean**  **25**  **com.byf.gradle**  **:clean UP-TO-DATE**  **:app:clean UP-TO-DATE**  **:test:clean UP-TO-DATE**  **BUILD SUCCESSFUL**  **Total time: 1.903 secs** |

2）在父工程定义全局变量，子工程引用

|  |
| --- |
| *// root 模块* ext {  number = 25  lib = **'com.byf.gradle'** } |
| *// 子app模块* println **this**.rootProject.number println **this**.rootProject.lib |
| **F:\groovy\_peojects\gradlelife>gradlew clean**  **25**  **com.byf.gradle**  **:clean UP-TO-DATE**  **:app:clean UP-TO-DATE**  **:test:clean UP-TO-DATE**  **BUILD SUCCESSFUL**  **Total time: 1.848 secs** |

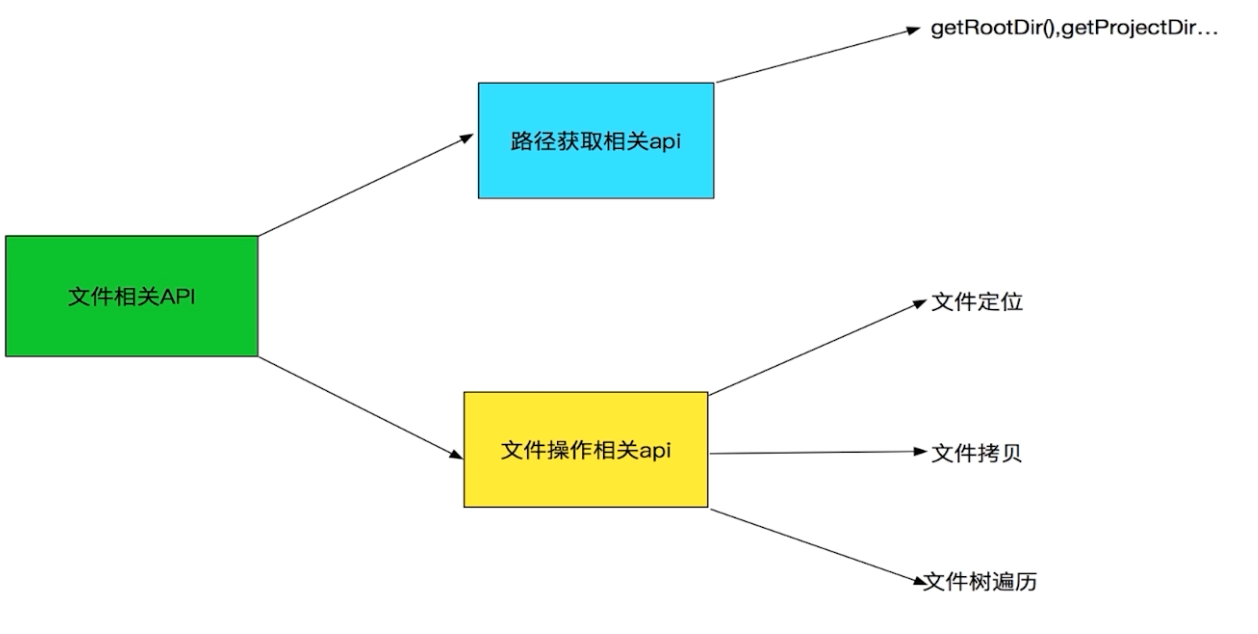
3）定义一个公用全局的变量文件

|  |
| --- |
| *//定义公共common.gradle的变量* ext {  commonname = **'this is common.gradle.'** commonint = 25  angular = [  **'js'** : **'2.0.1'**,  **'bootstrap'** : **'3.0.0'**,  **'ts'** : **'1.0.0'** ] } |
| *// 子test模块* println **this**.commonname println **this**.commonint println **this**.angular.js println **this**.angular.bootstrap println **this**.angular.ts |
| **F:\groovy\_peojects\gradlelife>gradlew clean**  **this is common.gradle.**  **25**  **2.0.1**  **3.0.0**  **1.0.0**  **:clean UP-TO-DATE**  **:app:clean UP-TO-DATE**  **:test:clean UP-TO-DATE**  **BUILD SUCCESSFUL**  **Total time: 2.001 secs** |

2）gradle.properties中定义，hasProperty取参数

|  |
| --- |
| **// 文件gradle.properties isLoadTest**=**false mCompileSdkVersion**=**25** |
| *// 文件settings.gradle* rootProject.name = **'gradlelife'** *//println 'settings.gradle：初始化阶段开始...'* include **'app'**  **// 根据是否在gradle.properties中配置的参数，决定是否引入test模块 if** (hasProperty(**'isLoadTest'**) ? isLoadTest.toBoolean() : **false**) {  include **':test'** } |
|  |

### 7.5 文件属性操作讲解



1）获取Root、Build、Project路径

|  |
| --- |
| println **this**.getRootDir().absolutePath println **this**.getBuildDir().absolutePath println **this**.getProjectDir().absolutePath |
| F:\groovy\_peojects\gradlelife>gradlew clean  Starting a Gradle Daemon, 2 incompatible and 1 stopped Daemons could not be reused, use --status for details  F:\groovy\_peojects\gradlelife  F:\groovy\_peojects\gradlelife\build  F:\groovy\_peojects\gradlelife  :clean UP-TO-DATE  :app:clean UP-TO-DATE  :test:clean UP-TO-DATE  BUILD SUCCESSFUL  Total time: 9.71 secs |

2）在当前来的project工程开始查找文件，而不是new的时候填写绝对路径

|  |
| --- |
| println getContent(**'common.gradle'**) **def** getContent(String path){  **try** {  *// file 相对于当前的project工程开始查找* **def** file = file(path)  **return** file.text  } **catch** (GradleException e) {  println **'file not found...'** } } |
| F:\groovy\_peojects\gradlelife>gradlew clean  //定义公共common.gradle的变量  ext {  commonname = 'this is common.gradle.'  commonint = 25  angular = [  'js' : '2.0.1',  'bootstrap' : '3.0.0',  'ts' : '1.0.0'  ]  }  :clean UP-TO-DATE  :app:clean UP-TO-DATE  :test:clean UP-TO-DATE  BUILD SUCCESSFUL |

3）copy闭包文件拷贝

|  |
| --- |
| copy {  from file(**'copy.txt'**)  into getRootProject().getBuildDir() } |
|  |

4）拷贝整个目录

|  |
| --- |
| copy {  from file(**'dir'**)  into getRootProject().getBuildDir().path + **'/app/'** } |
|  |

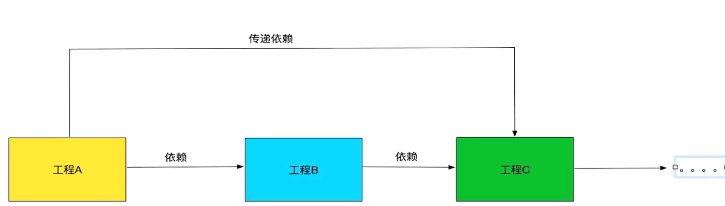
5）copy时指定排除文件，或重命名文件

|  |
| --- |
| copy {  from file(**'dir'**)  into getRootProject().getBuildDir().path + **'/app/'** exclude { details -> details.file.name.endsWith(**'.txt'**) &&  details.file.text.contains(**'test'**) }  rename(**'index.html'**, **'index.jsp'**) } |
|  |

6）文件树的遍历

|  |
| --- |
| *// 对文件树进行遍历* fileTree(**'dir'**) {  FileTree fileTree ->  fileTree.visit {FileTreeElement element ->  println **'the file name is : '** + element.file.name  copy{  from element.file  into getRootProject().getBuildDir().path + **'/test/'** }  } } |
|  |

### 7.5 依赖相关api讲解及实战



不允许使用传递依赖

1）仓库配置

|  |
| --- |
| buildscript{ScriptHandler scriptHandler->  *//配置工程的仓库地址* scriptHandler.repositories {  RepositoryHandle repositoryHandle ->  repositoryHandle.jcenter()  repositoryHandle.mavenCentral()  repositoryHandle.mavenLocal()  repositoryHadnle.ivy{}  repositoryHandle.maven{  name **'byf'** url **'http://localhost:8081:/nexus/repositories'** credentials {  username = **'admin'** password = **'123'** }  }  } } |

2）依赖配置

|  |
| --- |
| *// 为应用添加第三方库依赖* dependencies{  compile fileTree(**include**:[**'\*.jar'**], **dir**:**'libs'**)  compole project(rootProject.ext.lib)  provided(rootProject.ext.dependence)  *//test()  //default()  //runtime()* } |

### 7.6 gradle执行外部命令实战

1）gradle执行外部命名，task移动文件位置

|  |
| --- |
| task appcopy(){  doLast{  *// gradle的执行阶段去执行* **def** sourcePath = **this**.projectDir.path + **'\\dir\\output'  def** destinyPath = **this**.getRootDir().path + **'\\build\\works'  def** command = **"mv -f** ${sourcePath} ${destinyPath}**"** exec {  **try** {  executable **'bash'** args **'-c'**, command  println **'the command is execute success.'** } **catch** (GradleException e){  println **'the command is execyte failed.'** }  }  } } |
|  |

## 第8节 Gradle核心之Task详解及实战

### 8.1 Task定义和配置

1）直接通过task函数去创建

|  |
| --- |
| *// 直接通过task函数去创建* task helloTask {  println **'Hello Task.'** } |
| *F:\groovy\_peojects\gradlelife>gradlew helloTask*  *Hello Task.*  *:helloTask UP-TO-DATE*  *BUILD SUCCESSFUL* |

2）通过TaskContainer容器去创建task

|  |
| --- |
| *//通过TaskContainer容器去创建task* **this**.tasks.create(**name**: **'helloTask2'**) {  println **'hello Task2.'** } |
| *F:\groovy\_peojects\gradlelife>gradlew helloTask2*  *Hello Task.*  *hello Task2.*  *:helloTask2 UP-TO-DATE*  *BUILD SUCCESSFUL* |

3）为task设置属性

|  |
| --- |
| *// 直接通过task函数去创建* task helloTask(**group**: **'byf'**,**description**: **'task study'**) {  println **'Hello Task.'** }  *//通过TaskContainer容器去创建task* **this**.tasks.create(**name**: **'helloTask2'**) {  setGroup(**'byf'**)  setDescription(**'task study'**)  println **'hello Task2.'** } |
|  |

### 8.2 Task执行讲解与实战

1）在task的执行阶段，添加对应的逻辑

|  |
| --- |
| *// 直接通过task函数去创建* task helloTask(**group**: **'byf'**,**description**: **'task study'**) {  println **'Hello Task.'** doFirst{  println **'the task group is :'** + group  }  doFirst{} } helloTask.doFirst{  println **'the task description is :'** + description } |
| *F:\groovy\_peojects\gradlelife>gradlew helloTask*  *Hello Task.*  *hello Task2.*  *:helloTask*  *the task description is :task study*  *the task group is :byf*  *BUILD SUCCESSFUL* |

2）计算编译时长

|  |
| --- |
| *//计算build的执行时长* **def** startBuildTime, endBuildTime *//在配置完成时，找到对应的task* **this**.afterEvaluate{ Project project ->  *// 保证要找的task已经执行完毕* **def** preBuildTask = project.tasks.getByName(**'classes'**)  preBuildTask.doLast{  startBuildTime = System.*currentTimeMillis*();  println **'the start time is : '** + startBuildTime  }  **def** buildTask = project.tasks.getByName(**'build'**)  buildTask.doLast {  endBuildTime = System.*currentTimeMillis*()  println **"the build time is :** ${endBuildTime - startBuildTime}**"** }  } |
| *F:\groovy\_peojects\gradlelife>gradlew build*  *Hello Task.*  *hello Task2.*  *:compileJava UP-TO-DATE*  *:compileGroovy UP-TO-DATE*  *:processResources UP-TO-DATE*  *:classes*  *the start time is : 1544712589385*  *:jar UP-TO-DATE*  *:assemble UP-TO-DATE*  *:compileTestJava UP-TO-DATE*  *:compileTestGroovy UP-TO-DATE*  *:processTestResources UP-TO-DATE*  *:testClasses UP-TO-DATE*  *:test UP-TO-DATE*  *:check UP-TO-DATE*  *:build*  *the build time is : 8*  *:app:compileJava UP-TO-DATE*  *:app:compileGroovy UP-TO-DATE*  *:app:processResources UP-TO-DATE*  *:app:classes UP-TO-DATE*  *:app:jar UP-TO-DATE*  *:app:assemble UP-TO-DATE*  *:app:compileTestJava UP-TO-DATE*  *:app:compileTestGroovy UP-TO-DATE*  *:app:processTestResources UP-TO-DATE*  *:app:testClasses UP-TO-DATE*  *:app:test UP-TO-DATE*  *:app:check UP-TO-DATE*  *:app:build UP-TO-DATE*  *:test:compileJava UP-TO-DATE*  *:test:compileGroovy UP-TO-DATE*  *:test:processResources UP-TO-DATE*  *:test:classes UP-TO-DATE*  *:test:jar UP-TO-DATE*  *:test:assemble UP-TO-DATE*  *:test:compileTestJava UP-TO-DATE*  *:test:compileTestGroovy UP-TO-DATE*  *:test:processTestResources UP-TO-DATE*  *:test:testClasses UP-TO-DATE*  *:test:test UP-TO-DATE*  *:test:check UP-TO-DATE*  *:test:build UP-TO-DATE*  *BUILD SUCCESSFUL* |

### 8.3 Task依赖详解及实战

1）task依赖之dependsOn

|  |
| --- |
| *// dependsOn的task依赖* task taskX {  doLast{  println **'taskX'** } } task taskY {  doLast{  println **'taskY'** } } task taskZ(**dependsOn**:[taskX,taskY]) {  doLast{  println **'taskZ'** } } |
| *F:\groovy\_peojects\gradlelife>gradlew taskZ*  *:app:taskX*  *taskX*  *:app:taskY*  *taskY*  *:app:taskZ*  *taskZ*  *BUILD SUCCESSFUL* |

2）正则匹配依赖

|  |
| --- |
| task lib1 {  doLast{  println **'lib1'** } } task **lib2** << {  println **'lib2'** } task **nolib** << {  println **'nolib'** }  task taskZ {  dependsOn **this**.tasks.findAll { task ->  **return** task.name.startsWith(**'lib'**)  }  doLast{  println **'taskZ'** } } |
| F:\groovy\_peojects\gradlelife>gradlew taskZ  :app:lib1  lib1  :app:lib2  lib2  :app:taskZ  taskZ  BUILD SUCCESSFUL |

### 8.4 Task输入输出

1）定义全局变量，输出文件（写文件），输入文件（读取文件）

|  |
| --- |
| ext {  versionName = **'V1.12.1'** versionCode= **'120'** versionDescription=**'这是app的第2个版本，上线了一些基础的功能'** destFile = file(**'releases.xml'**)  **if** (destFile != **null** && !destFile.exists()) {  destFile.createNewFile()  } } |
| task writeTask {  inputs.property(**'versionCode'**, **this**.versionCode)  inputs.property(**'versionName'**, **this**.versionName)  inputs.property(**'versionDescription'**, **this**.versionDescription)   *// 为task指定输出* outputs.file destFile  doLast {  **def** data = inputs.getProperties()  File file = outputs.getFiles().getSingleFile()  *// 将map转化为实体对象* **def** versionMsg = **new** VersionMsg(data)  *// 将实体对象转化为xml格式数据* **def** sw = **new** StringWriter()  **def** xmlBuilder = **new** MarkupBuilder(sw)  **if** (file.text != **null** && file.text.size() <= 0){  *// 文件中没有内容* xmlBuilder.releases{  release{  versionCode(versionMsg.versionCode)  versionName(versionMsg.versionName)  versionDescription(versionMsg.versionDescription)  }  }  file.withWriter {  writer -> writer.append(sw.toString())  }  } **else** {  *// 已有版本信息* xmlBuilder.release {  versionCode(versionMsg.versionCode)  versionName(versionMsg.versionName)  versionDescription(versionMsg.versionDescription)  }  *// 将生成的xml数据插入到根节点之前* **def** lines = file.readLines()  **def** lengths = lines.size()-1  file.withWriter { writer ->  lines.eachWithIndex { String line, **int** index ->  **if** (index != lengths) {  writer.append(line + **'\r\n'**)  } **else if** (index == lengths){  writer.append(**'\r\n'** + sw.toString() + **'\r\n'**)  writer.append(lines.get(lengths))  }  }  }  }    } } |
| task readTask {  inputs.file destFile  doLast {  **def** file = inputs.files.singleFile  println file.text  } } |
| **class** VersionMsg {  String **versionCode** String **versionName** String **versionDescription** } |
| task taskTest {  dependsOn readTask, writeTask  doLast {  println **'输入输出任务结束'** } } |
| F:\groovy\_peojects\gradlelife>gradlew taskTest  :app:readTask  :app:writeTask  :app:taskTest  输入输出任务结束  BUILD SUCCESSFUL |
| F:\groovy\_peojects\gradlelife>gradlew taskTest  :app:readTask  <releases>  <release>  <versionCode>120</versionCode>  <versionName>V1.12.1</versionName>  <versionDescription>这是app的第2个版本，上线了一些基础的功能</versionDescription>  </release>  </releases>  :app:writeTask  :app:taskTest  输入输出任务结束  BUILD SUCCESSFUL |
| <releases>  <release>  <versionCode>120</versionCode>  <versionName>V1.12.1</versionName>  <versionDescription>这是app的第2个版本，上线了一些基础的功能</versionDescription>  </release>  <release>  <versionCode>120</versionCode>  <versionName>V1.13.1</versionName>  <versionDescription>这是app的第2个版本，上线了一些基础的功能</versionDescription>  </release>  </releases> |

### 8.5 挂接自定义Task到构建过程中

1）挂接在一个task后

|  |
| --- |
| task taskX {  println **'this is taskX'** } task taskY {  mustRunAfter taskX  println **'this is taskY'** } task taskZ {  mustRunAfter taskY  println **'this is taskZ'** } |
| F:\groovy\_peojects\gradlelife>gradlew clean  this is taskX  this is taskY  this is taskZ  :clean UP-TO-DATE  :app:clean UP-TO-DATE  :test:clean UP-TO-DATE  BUILD SUCCESSFUL |
| F:\groovy\_peojects\gradlelife>gradlew taskZ taskX taskY  this is taskX  this is taskY  this is taskZ  :app:taskX UP-TO-DATE  :app:taskY UP-TO-DATE  :app:taskZ UP-TO-DATE  BUILD SUCCESSFUL |

2）releaseinfo.gradle

|  |
| --- |
| **import** groovy.xml.MarkupBuilder  ext {  versionName = **'V2.13.1'** versionCode= **'120'** versionDescription=**'这是app的第3个版本，上线了一些基础的功能'** destFile = file(**'releases.xml'**)  **if** (destFile != **null** && !destFile.exists()) {  destFile.createNewFile()  } }  **this**.project.afterEvaluate{ project->  **def** buildTask = project.tasks.getByName(**'build'**)  **if** (buildTask == **null**){  **throw** GradleException(**'the build task is not found'**)  }  buildTask.doLast{  writeTask.execute()  } } task writeTask {  inputs.property(**'versionCode'**, **this**.versionCode)  inputs.property(**'versionName'**, **this**.versionName)  inputs.property(**'versionDescription'**, **this**.versionDescription)   *// 为task指定输出* outputs.file destFile  doLast {  **def** data = inputs.getProperties()  File file = outputs.getFiles().getSingleFile()  *// 将map转化为实体对象* **def** versionMsg = **new** VersionMsg(data)  *// 将实体对象转化为xml格式数据* **def** sw = **new** StringWriter()  **def** xmlBuilder = **new** MarkupBuilder(sw)  **if** (file.text != **null** && file.text.size() <= 0){  *// 文件中没有内容* xmlBuilder.releases{  release{  versionCode(versionMsg.versionCode)  versionName(versionMsg.versionName)  versionDescription(versionMsg.versionDescription)  }  }  file.withWriter {  writer -> writer.append(sw.toString())  }  } **else** {  *// 已有版本信息* xmlBuilder.release {  versionCode(versionMsg.versionCode)  versionName(versionMsg.versionName)  versionDescription(versionMsg.versionDescription)  }  *// 将生成的xml数据插入到根节点之前* **def** lines = file.readLines()  **def** lengths = lines.size()-1  file.withWriter { writer ->  lines.eachWithIndex { String line, **int** index ->  **if** (index != lengths) {  writer.append(line + **'\r\n'**)  } **else if** (index == lengths){  writer.append(**'\r\n'** + sw.toString() + **'\r\n'**)  writer.append(lines.get(lengths))  }  }  }  }    } }  task readTask {  inputs.file destFile  doLast {  **def** file = inputs.files.singleFile  println file.text  } }  **class** VersionMsg {  String **versionCode** String **versionName** String **versionDescription** }  task taskTest {  dependsOn readTask, writeTask  doLast {  println **'输入输出任务结束'** } } |
| apply **from**: **this**.rootProject.file(**'releaseinfo.gradle'**) |
| F:\groovy\_peojects\gradlelife>gradlew build  :compileJava UP-TO-DATE  :compileGroovy UP-TO-DATE  :processResources UP-TO-DATE  :classes UP-TO-DATE  :jar  :assemble  ... |
| <releases>  <release>  <versionCode>120</versionCode>  <versionName>V1.12.1</versionName>  <versionDescription>这是app的第2个版本，上线了一些基础的功能</versionDescription>  </release>  <release>  <versionCode>120</versionCode>  <versionName>V1.13.1</versionName>  <versionDescription>这是app的第2个版本，上线了一些基础的功能</versionDescription>  </release>  <release>  <versionCode>120</versionCode>  <versionName>V2.13.1</versionName>  <versionDescription>这是app的第3个版本，上线了一些基础的功能</versionDescription>  </release>  </releases> |

### 8.6 Task类型

1）Copy类型的task

|  |
| --- |
| *// Copy类型的task* task copyDocs(**type**: Copy) {  from **'dir/copy.txt'** into **'build/'** } |
|  |

2）Delete类型的task

|  |
| --- |
| *//Delete类型的task* task makePretty(**type**: Delete) {  delete **'dir/abc.css'** followSymlinks = **true** } |
|  |

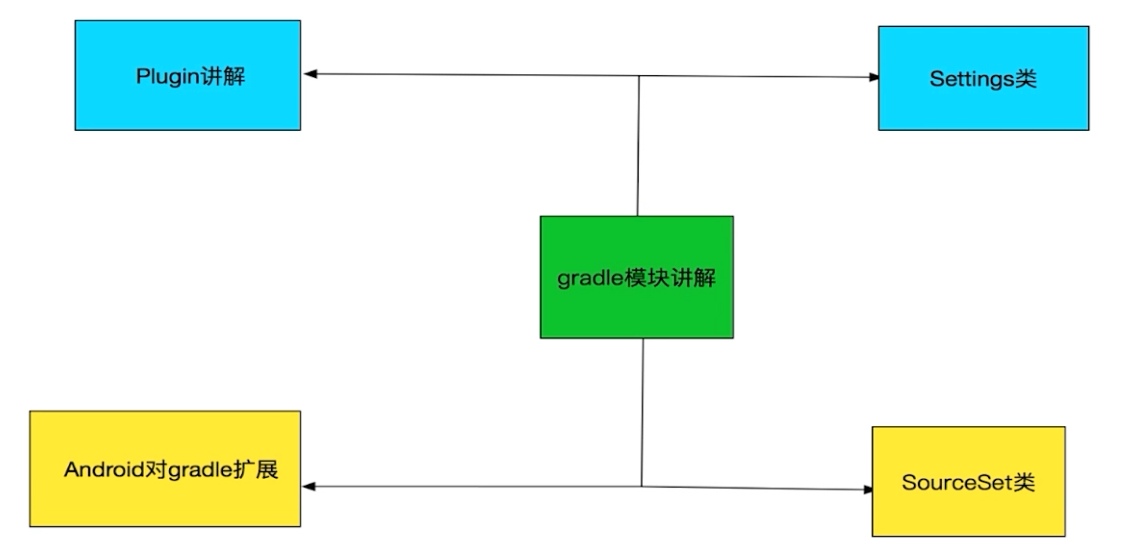
3）gradle task打jar包

|  |
| --- |
| jar {  manifest {  attributes **'Main-Class'**: appMainClass  } }  task clearJar(**type**: Delete) {  delete **'build/libs/lib'** }  task copyJar(**type**: Copy) {  from configurations.**runtime** into(**'build/libs/lib'**) }  task release(**type**: Copy, **dependsOn**: [build, clearJar, copyJar]) |

## 第9节 Gradle其它模块讲解与自定义Plugin实战

### 9.1 Settings类讲解

### 9.2 SourceSet讲解



资源、代码、第三方库

约定大于配置

9.3 自定义Plugin实战

9.4 Android及Java插件对gradle的扩展

9.5 如何迁移到gradle

第10节 Gradle持续集成与打包

10.1 Jenkins环境变量搭建

10.2 Jenkins配置