Gradle Plugin使用手册

yeungeek

说明

Gradle Plugin User Guide中文版

Gradle Plugin的使用，并结合例子说明

- Gradle Plugin User Guide中文版 正在翻译当中 欢迎大家一起加入
- github: https://github.com/yeungeek/GradlePlugin_UserGuide
- 使用了gitbook进行编辑：http://www.gitbook.io
- 原文地址：http://tools.android.com/tech-docs/new-build-system/user-guide
- 我会开放权限给需要加入的同学，联系我: yeungeek#gmail.com

翻译进度

章节	时间	译者	实例
1	14.09.29	yeungeek	HelloWorld
2	14.09.29	yeungeek	
3.1	14.10.08	yeungeek	
3.2	14.10.09	yeungeek	
baz	baz	baz	

特色

我们是有实例的人

gradle对应的示例代码，可以fork Samples.

简介

本文档适用于0.9版本的Gradle plugin。在我们引入1.0版本之前，内容可能会与之前的版本不兼容。

新构建系统的目标

新构建系统的目标:

- 让重用代码和资源变得更加容易
- 使创建同一个应用程序的多个版本根据容易，不管是多apk的发布还是同一个应用的不同定制版本
- 使构建过程根据容易配置，扩展和自定义
- 优秀IDE的集成

为什么使用Gradle?

Gradle是一个优秀的根据系统和构建工具，它允许通过插件来创建自定义的构建逻辑。

以下的一些特性，让我们选择了Gradle：

- 使用领域专用语言(DSL)来描述和控制构建逻辑
- 构建文件基于Groovy，并允许通过DSL声明和使用代码混合来定义DSL元素和自定义的构建逻辑
- 内置通过Maven和Ivy进行依赖管理
- 相当灵活。允许使用最好的实现，但是不会强制实现的形式。
- 插件提供DSL和API来定义构建文件
- 优秀的API工具与IDE集成

配置

- Gradle1.10 1.11 1.12使用插件0.11.1版本
- SDK Build Tools 版本19.0.0.一些特性需要更高版本。

译者注：gradle目前已经是2.1版本，插件0.12.+ 最新可以关注: http://www.gradle.org/

基础工程

一个Gradle工程的构建描述，定义在工程根目录下的build.gradle文件中.

简单构建文件

一个最简单Gradle纯Java工程的build.gradle文件包含了以下内容:

```
apply plugin: 'java'
```

这是Gradle包装的Java插件。该插件提供了所有构建和测试Java应用程序的东西。 最简单的Android工程的build.gradle描述：

```
buildscript {
   repositories {
      mavenCentral()
   }

   dependencies {
      classpath 'com.android.tools.build:gradle:0.11.1'
   }
}

apply plugin: 'android'

android {
   compileSdkVersion 19
   buildToolsVersion "19.0.0"
}
```

译者注：目前gradle tools版本为0.13.+(2014.10.08)

上述内容包含了Android构建文件的3个主要部分：

buildscript { ... }配置了驱动构建的代码.

在这个例子中，他申明了使用Maven中央库，并且申明了一个Maven构件的依赖classpath。这个构件声明了Gradle的Android插件版本为0.11.1.

注意：这里的配置只影响了构建过程的代码，而不是整个工程的代码.工程本身需要声明它自己的仓库和依赖.这个后面会提到.

然后，跟前面提到的Java插件一样，添加了**android**插件.

最后，**android { ... }**配置了所有android构建的参数.也是Android DSL的入口点. 默认情况下，只有编译的target和build-tools版本是必须的.就是**compileSdkVersion**和**buildtoolsVersion** 两个属性. 编译的target属性相当于在老的构建系统中 `project.properties` 中的target属性.这个新属性和老的target属性一样可以指定一个int(api等级)或者string类型的值.

重要： 你只能使用**android**插件.如果同时使用**java**插件，会导致构建错误.

注意： 你还需要添加*local.properties*文件，使用**sdk.dir**属性，来设置已经存在的SDK路径. 另外，你也可以设置环境变量**ANDROID_HOME**.这两种方式没有什么区别,可以根据你自己的喜好来选择一种.

工程结构

上面提到的基本构建文件需要一个默认的文件结构.Gradle遵循约定优于配置的概念. 在尽可能的情况下提供合理的默认参数. 基本的工程有两个名为"source sets"组件.就是main source code和test code.它们分别位于:

- src/main/
- src/androidTest/

里面的每个文件目录都对应了相应的源组件. 对于Java插件和Android插件，他们对应的Java源代码和Java资源目录：

- java/
- resources/

对于Android插件，有额外的文件和文件目录：

- AndroidManifest.xml
- res/
- assets/
- aidl/
- rs/
- jni/

注意： src/androidTest/AndroidManifest.xml是不需要的，因为它会自动创建.

配置工程结构

当默认的工程结构不适用时，就可能需要去配置它.根据Gradle文档，根据下面的代码可以重新配置Java工程的sourceSets：

```
sourceSets {
   main {
      java {
         srcDir 'src/java'
      }
      resources {
         srcDir 'src/resources'
      }
   }
}
```

> 注意： **srcDir**将会被添加到已存在的源文件目录中(这个在Gradle文档中没有提到，但是实际上确实是这样执行了)

要替换默认的源文件目录，你需要使用一个数组路径的**srcDirs**来替代.下面是使用调用对象的另外一种不同的方法：

```
sourceSets {
   main.java.srcDirs = ['src/java']
   main.resources.srcDirs = ['src/resources']
}
```

想了解更多的信息，可以查看Gradle文档中Java插件部分.

Android插件使用了类似的语法，因为使用了它自己的sourceSets，这些配置会被添加到**android**对象中. 下面这个例子,使用了旧工程结构的main代码，并把**androidTest**的sourceSet映射到tests目录中.

```
android {
   sourceSets {
      main {
         manifest.srcFile 'AndroidManifest.xml'
         java.srcDirs = ['src']
         resources.srcDirs = ['src']
         aidl.srcDirs = ['src']
         renderscript.srcDirs = ['src']
         res.srcDirs = ['res']
         assets.srcDirs = ['assets']
      }

      androidTest.setRoot('tests')
   }
}
```

> 注意：因为旧的结构把所有的源文件(java, aidl, renderscript, and java resources)放在同一个目录中,所以我们需要重新映射所有的sourceSet新组件到同一个**src**目录下.
>
> 注意：**setRoot()**会移动所有的sourceSet(包括它的子目录)到新的目录.例子中把**src/androidTest/***移动到**tests/***

这是在Android中特有的，在Java sourceSets中不起作用.

上述的就是工程迁移的简单示例.

Basic Build Customization

Dependencies, Android Libraries and Multi-project setup

Dependencies on binary packages

Build options

Table of Contents