HenCoder Plus 讲义

Gradle Plugin

本质

把逻辑独立的代码抽取和封装

Plugin 的最基本写法

写在 `build.gradle` 里：

> build.gradle:

```
class PluginDemo implements Plugin<Project> {

  @Override
  void apply(Project target) {
    println 'Hello author!'
  }
}

apply plugin: PluginDemo
```

Extension

> build.gradle:

```
class ExtensionDemo {
  def author = 'Kai'
}

class PluginDemo implements Plugin<Project> {
```

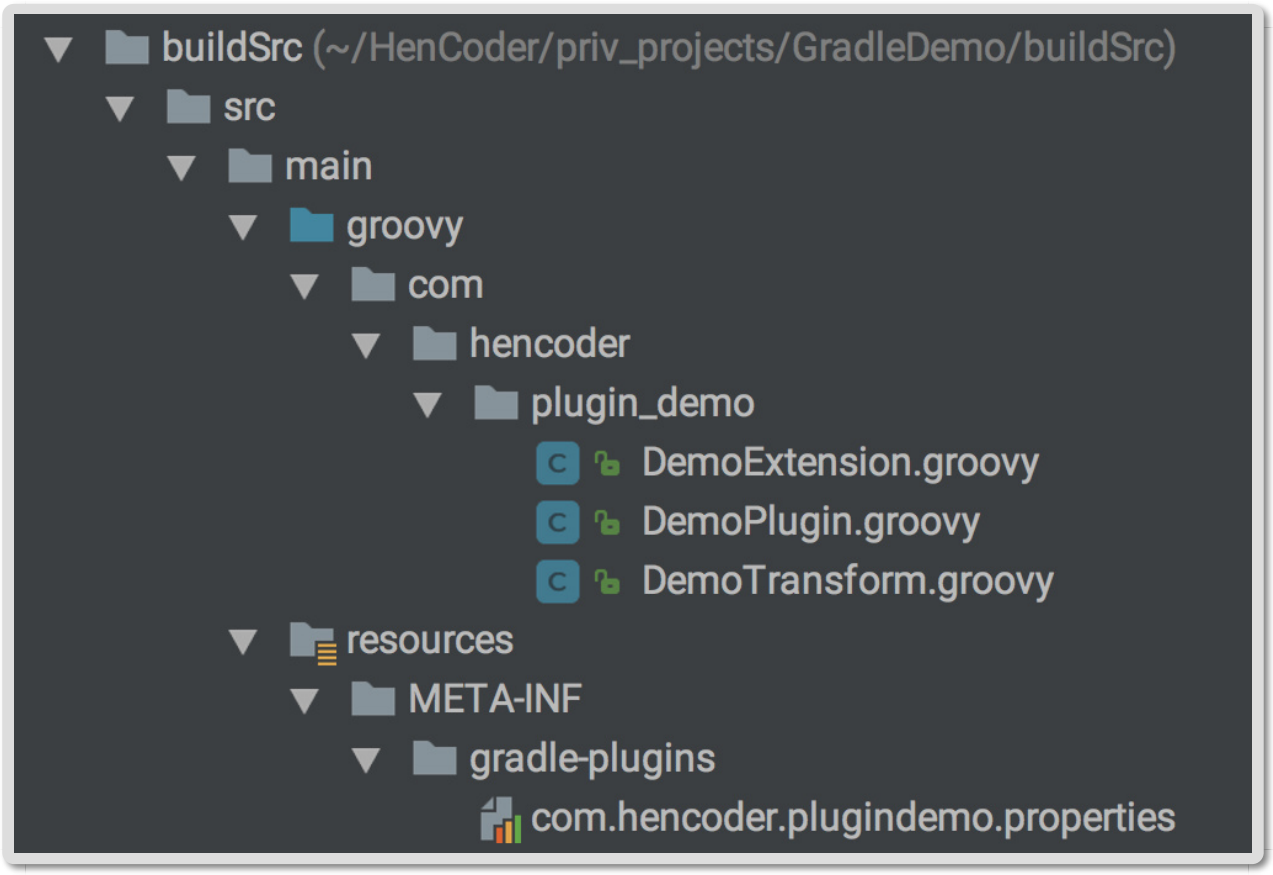
```
  @Override
  void apply(Project target) {
    def extension = target.extensions.create('hencoder',
ExtensionDemo)
    target.afterEvaluate {
      println "Hello ${extension.author}!"
    }
  }
}

apply plugin: PluginDemo

hencoder {
  author 'rengwuxian'
}
```

写在 buildSrc 目录下

- 目录结构：



- `resources/META-INF/gradle-plugins/*.properties` 中的 `*` 是插件的名称，例如 `*.properties` 是 `com.hencoder.plugindemo.properties`，最终在应用插件是的代码就应该是：

  ```
  apply plugin: 'com.hencoder.plugindemo'
  ```

- *.properties 中只有一行，格式是：

  ```
  implementation-
  class=com.hencoder.plugin_demo.DemoPlugin
  ```

  其中等号右边指定了 Plugin 具体是哪个类

- Plugin 和 Extension 写法和在 `build.gradle` 里的写法一样
- 关于 buildSrc 目录
  - 这是 gradle 的一个特殊目录，这个目录的 build.gradle 会自动被执行，即使不配置进 settings.gradle。（实际上在 gradle 的 6.0 之后，`buildSrc` 已经成为了一个保留字，你在 `settings.gradle` 里配置的项目已经不允许叫 `buildSrc`）
  - buildSrc 所配置出来的 Plugin 会被自动添加到编译过程中的每一个 project 的 classpath，因此它们才可以直接使用 `apply plugin: 'xxx'` 的方式来便捷应用这些 plugin

Transform

- 是什么：是由 Android 提供了，在项目构建过程中把编译后的文件（jar 文件和 class 文件）添加自定义的中间处理过程的工具
- 怎么写
  - 先加上依赖：

```
// 因为 buildSrc 需要自己添加仓库
repositories {
  google()
  jcenter()
}

dependencies {
  implementation
'com.android.tools.build:gradle:4.0.0-beta04'
}
```

- 然后继承 `com.android.build.api.transform.Transform`，创建一个子类：

```
class DemoTransform extends Transform {

  // 构造方法
  DemoTransform() {
  }

  // 对应的 task 名
  @Override
  String getName() {
    return 'hencoderTransform'
  }

  // 你要对那些类型的结果进行转换(是字节码还是资源文件？)
  @Override
  Set<QualifiedContent.ContentType>
getInputTypes() {
    return TransformManager.CONTENT_CLASS
  }

  // 适用范围包括什么(整个 project 还是别的什么？)
  @Override
  Set<? super QualifiedContent.Scope> getScopes()
{
```

```
    return TransformManager.SCOPE_FULL_PROJECT
  }

  @Override
  boolean isIncremental() {
    return false
  }

  // 具体的「转换」过程
  @Override
  void transform(TransformInvocation
transformInvocation) throws TransformException,
InterruptedException, IOException {
    def inputs = transformInvocation.inputs
    def outputProvider =
transformInvocation.outputProvider

    inputs.each {
      // jarInputs: 各个依赖所编译成的 jar 文件
      it.jarInputs.each {
        // dest:
        //
./app/build/intermediates/transforms/hencoderTrans
form/...
        File dest =
outputProvider.getContentLocation(it.name,
it.contentTypes, it.scopes, Format.JAR)
        FileUtils.copyFile(it.file, dest)
      }

      // derectoryInputs: 本地 project 编译成的多个
class 文件存放的目录
      it.directoryInputs.each {
        // dest:
        //
./app/build/intermediates/transforms/hencoderTrans
form/...
```

```
            File dest =
outputProvider.getContentLocation(it.name,
it.contentTypes, it.scopes, Format.DIRECTORY)
            FileUtils.copyDirectory(it.file, dest)
        }
    }
  }
}
```

- 还能做什么：修改字节码 上面的这段代码只是把编译完的内容原封不动搬运到目标位置，没有实际用处。要修改字节码，需要引入其他工具，例如 javassist。 javassist 的使用教程在网上有很多，可以搜索一下。

问题和建议?

课上技术相关的问题，都可以去群里和大家讨论，对于比较通用的、有价值的问题，可以去我们的知识星球提问。

具体技术之外的问题和建议，都可以找丢物线（微信：diuwuxian），丢丢会为你解答技术以外的一切。

觉得好?

如果你觉得课程很棒，欢迎给我们好评呀！https://ke.qq.com/comment/index.html?cid=381952

> 一定要是你真的觉得好，再给我们好评。不要仅仅因为对扔物线的支持而好评（报名课程已经是你最大的支持了，再不够的话 B 站多来点三连我也很开心），另外我们也坚决不做好评返现等任何的交易。我们只希望，在课程对你有帮助的前提下，可以看到你温暖的评价。

更多内容：

- 网站：https://hencoder.com；https://kaixue.io
- 各大搜索引擎、微信公众号、微博、知乎、掘金、哔哩哔哩、YouTube、西瓜视频、抖音、快手、微视：统一账号「**扔物线**」，我会持续输出优质的技术内容，欢迎大家关注。
- 哔哩哔哩快捷传送门：https://space.bilibili.com/27559447

大家如果喜欢我们的课程，还请去扔物线的哔哩哔哩，帮我素质三连，感谢大家！