Kotlin遇上Gradle

课程内容

- 使用gradle构建工具
- 使用kotlin编写gradle脚本
- 看懂groovy写的gradle脚本

推动人类进步的动力

- 懒

解放你的双手

- 汽车的无人驾驶
- 送饭的百度外卖
- 扫地的扫地机器人
- 电话的语音控制

...

程序员需要解放双手

原始人

- 编译
- 测试
- 手动依赖管理
- 打包
- 上传服务器

现代人

- 自动化构建工具

Java常见构建工具

- Ant
- Maven
- Gradle

Ant
2000年

Maven
2007年

Gradle
2012年

Ant
2000年

编译测试

Maven
2007年

编译测试
依赖管理

Gradle
2012年

编译测试
依赖管理
DSL自定义扩展任务

Gradle是什么?

- 可以用kotlin代码控制的一种智能的自动化构建工具
- 用代码控制工具
- 不是xml配置来控制工具

Gradle是什么?

- 很牛逼的构建工具
- 构建一切可构建的内容

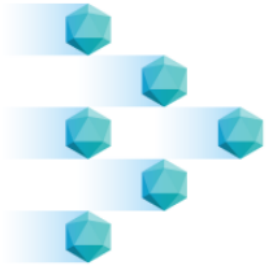
Build Anything

Write in Java, C++, Python or your language of choice. Package for deployment on any platform. Go monorepo or multi-repo. And rely on Gradle's unparalleled versatility to build it all.

Automate Everything

Use Gradle's rich API and mature ecosystem of plugins and integrations to get ambitious about automation. Model, integrate and systematize the delivery of your software from end to end.

Deliver Faster

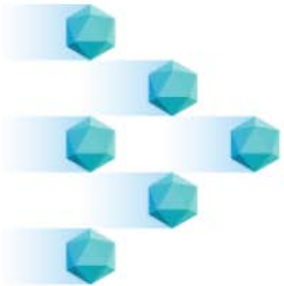


Scale out development with elegant, blazing-fast builds. From compile avoidance to advanced caching and beyond, we pursue performance relentlessly so your team can deliver continuously.

构建一切

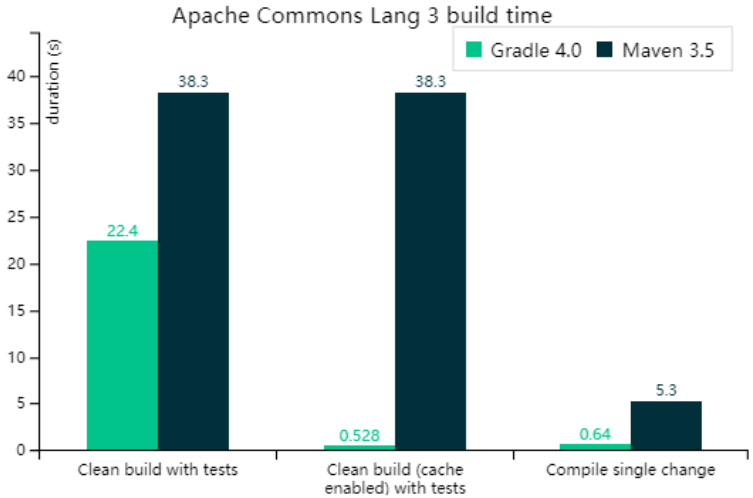
自动化一切

超快交互



Improving build time is one of the most direct ways to *ship faster*. Both Gradle and Maven employ some form of parallel project building and parallel dependency resolution. The biggest differences are Gradle's mechanisms for work avoidance and incrementality. The top 3 features that make Gradle much faster than Maven are:

- Incrementality — Gradle avoids work by tracking input and output of tasks and only running what is necessary, and only processing files that changed when possible.
- Build Cache — Reuses the build outputs of any other Gradle build with the same inputs, including between machines.
- Gradle Daemon — A long-lived process that keeps build information "hot" in memory.



These and more performance features make Gradle at least twice as fast for nearly every scenario (100x faster for large builds using the build cache) in this Gradle vs Maven performance comparison.

搬砖工的哲理:

我双手搬着砖就无法拥抱你,

我双手放开砖就无法养活你.

提高工作效率,有时间陪陪家人

Gradle快速入门

- 安装 https://gradle.org

下载地址

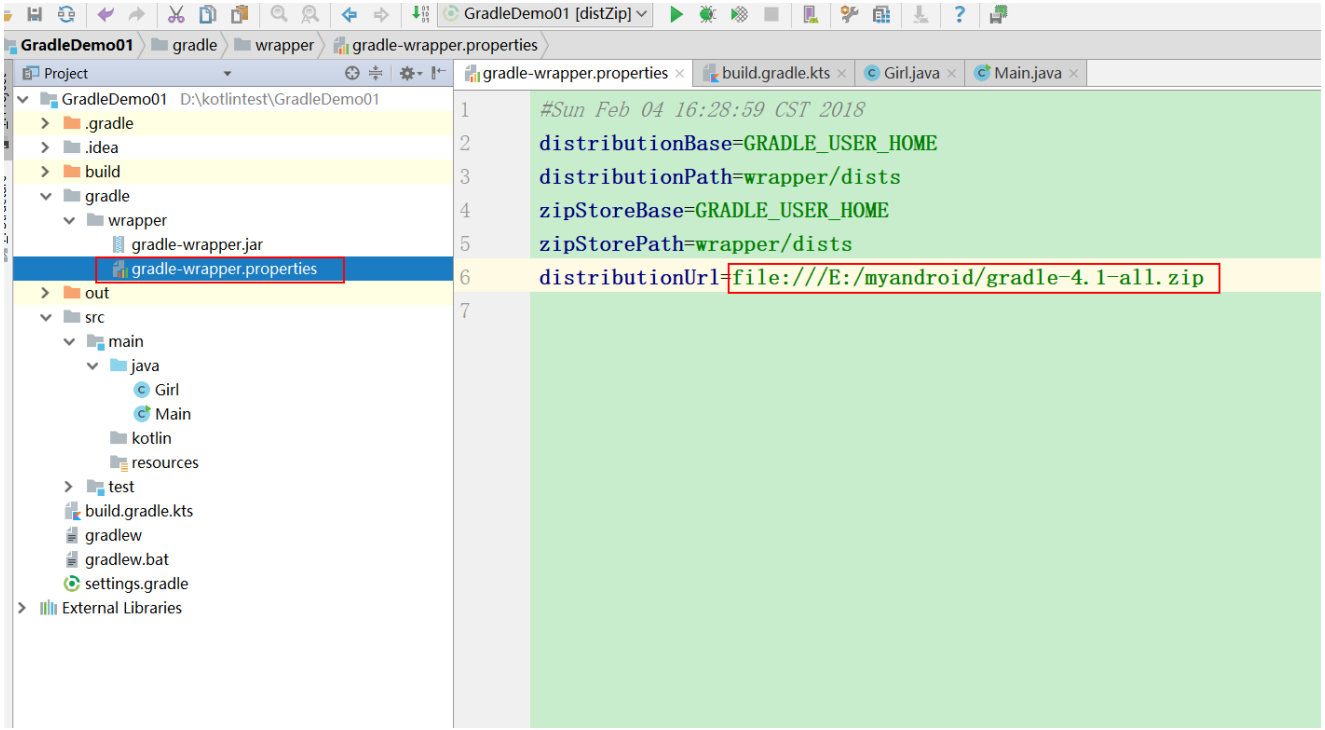
- https://gradle.org/

Accelerate developer productivity

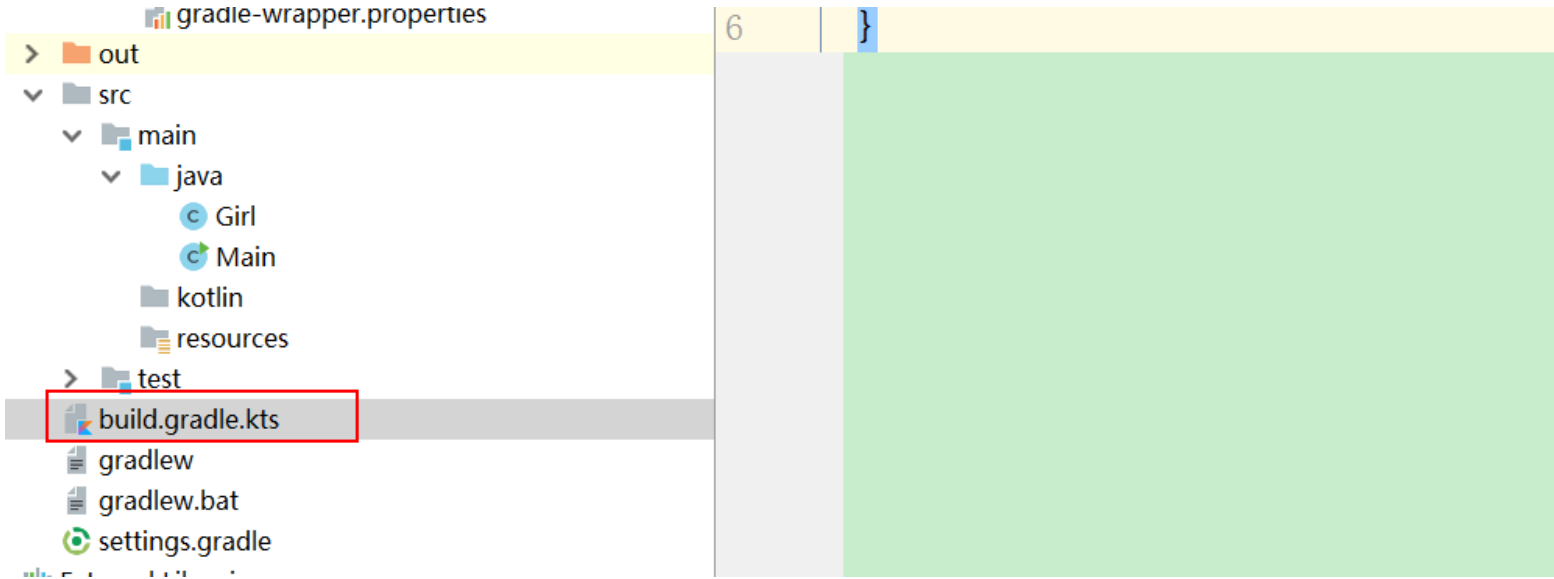
From mobile apps to microservices, from small startups to big enterprises, Gradle helps teams build, automate and deliver better software, faster.

1. Install Gradle → 2. Get Started Guides → 3. Free Training

- 设置gradle版本


GradleDemo01 [distZip]
GradleDemo01
gradle
wrapper
gradle-wrapper.properties
Project
GradleDemo01 D:\kotlintest\GradleDemo01
.gradle
.idea
build
gradle
wrapper
gradle-wrapper.jar
gradle-wrapper.properties
out
src
main
java
Girl
Main
kotlin
resources
test
build.gradle.kts
gradlew
gradlew.bat
settings.gradle
External Libraries
gradle-wrapper.properties
build.gradle.kts
Girl.java
Main.java
#Sun Feb 04 16:28:59 CST 2018
distributionBase=GRADLE_USER_HOME
distributionPath=wrapper/dists
zipStoreBase=GRADLE_USER_HOME
zipStorePath=wrapper/dists
distributionUrl=file:///E:/myandroid/gradle-4.1-all.zip

- 修改build.gradle后缀名为kts



Gradle

- groovy

VS

动态语言vs静态语言

- 动态语言(python groovy javascript…)
- 静态语言(kotlin java)

Kotlin Meets Gradle

May 18, 2016 Chris Beams New Features

Many readers will be familiar with JetBrains' excellent Kotlin programming language. It's been under development since 2010, had its first public release in 2012, and went 1.0 GA earlier this year.

We've been watching Kotlin over the years, and have been increasingly impressed with what the language has to offer, as well as with its considerable uptake—particularly in the Android community.

Late last year, Hans sat down with a few folks from the JetBrains team, and they wondered together: what might it look like to have a Kotlin-based approach to writing Gradle build scripts and plugins? How might it help teams—especially big ones—work faster and write better structured, more maintainable builds?

The possibilities were enticing.

Because Kotlin is a statically-typed language with deep support in both IDEA and Eclipse, it could give Gradle users proper IDE support from auto-completion to refactoring and everything in-between. And because Kotlin is rich with features like first-class functions and extension methods, it could retain and improve on the best parts of writing Gradle build scripts—including a clean, declarative syntax and the ability to craft DSLs with ease.

So we got serious about exploring these possibilities, and over the last several months we've had the pleasure of working closely with the Kotlin team to develop a new, Kotlin-based build language for Gradle.

Project和task

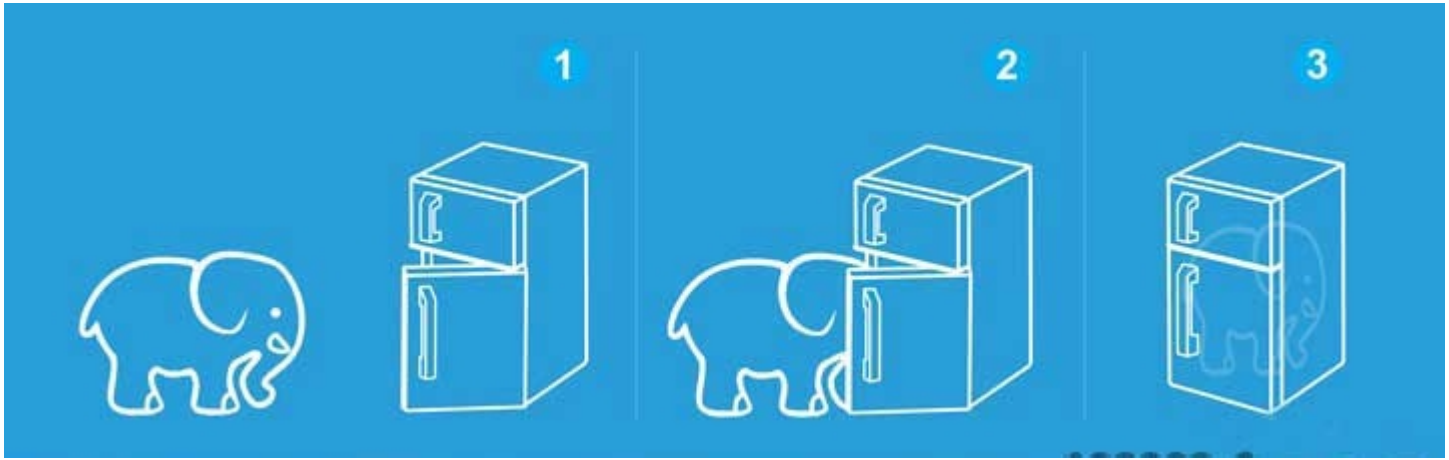
- project和task 是Gradle本身领域主要的两个对象
- Project为task提供了执行的容器和上下文

Task案例

- Helloworld

Task的依赖

- 大象装冰箱



Task的生命周期

- 扫描
- 执行

Task的生命周期

- doFirst
- doLast

Tasks任务集

- 多个任务的合集就是任务集

Gradle默认tasks

- Gradle:tasks

Gradle增量更新

- inputs.dir()
- inputs.file()

- outputs.dir()
- outputs.file()

Gradle常见插件

什么是插件?

- 插件是包含一个或多个任务的合集

application插件

java插件

war插件

插件仓库使用

- 第一种

```
buildscript {
    repositories {
        maven {
            setUrl("https://plugins.gradle.org/m2/")
        }
    }
    dependencies {
        classpath( dependencyNotation: "gradle.plugin.com.hulab.gradle:gradle-plugin:1.0.0")
    }
}
apply {
    plugin( p0: "com.hulab.gradle.gettext-android")
}
```

插件仓库使用

- 第二种

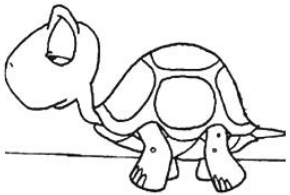
```
plugins{
    id( p0: "com.hulab.gradle.gettext-android") version("1.0.0")
}
```

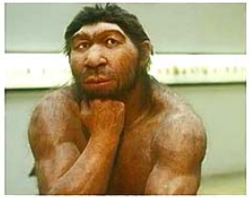
Gradle的依赖管理

- 什么是依赖管理?

  在什么地方以什么形式引入外部代码

普通程序员VS优秀程序员





华为用人准则

- 一个程序员拿两个程序员的钱,干三个程序员的活
- 一个优秀的程序员,开发效率相当于2~3个普通的程序员

小案例

- HttpClient的依赖管理

疼讯技术有限公司(深圳分公司)

工作任务单

编号:9527

<table>
<tr><td>任务提出人</td><td>马化疼</td><td>提出部门</td><td>总裁办</td><td>任务承担部门</td><td>黑马程序员</td></tr>
<tr><td>重要程度</td><td colspan="3">☐重要 ■紧急 ☐一般 ☐酌情安排</td><td>要求完成日期</td><td>越快越好</td></tr>
<tr><td>任务内容及要求</td><td colspan="5">1. 采用开源项目 httpclient 编写爬虫用于下载美女图片
2. 美女图片要高质量的,样本见附件
3. httpclient需采用3.1版本, 参考文档见附件</td></tr>
<tr><td></td><td colspan="5">（无法完成记录时，请附页说明）</td></tr>
<tr><td>分管领导会签</td><td>任务提出部门
分管领导核签</td><td colspan="2"></td><td>任务承担部门
分管领导核签</td><td></td></tr>
<tr><td rowspan="3">任务承担部门
人员安排</td><td colspan="5"></td></tr>
<tr><td>预完成时间</td><td colspan="2">（工作日）</td><td>部门主管签字</td><td></td></tr>
<tr><td>开始执行日期</td><td colspan="2"></td><td>任务执行人签字</td><td></td></tr>
<tr><td colspan="6">任务完成情况</td></tr>
<tr><td>任务完成日期</td><td></td><td colspan="4">☐提前完成 ☐按时完成 ☐延期完成 ☐未能完成</td></tr>
<tr><td rowspan="2">任务处理结果</td><td colspan="3" rowspan="2"></td><td>任务承担人</td><td></td></tr>
<tr><td>部门主管</td><td></td></tr>
<tr><td>任务提出部门
验收</td><td colspan="3"></td><td>任务提出部门
核签</td><td></td></tr>
<tr><td colspan="6">（若任务延期或中止，任务承担人要及时与任务提出人协商，并注明延期或中止原因）</td></tr>
<tr><td>延期中止
原因说明</td><td colspan="5"></td></tr>
<tr><td>技术文件或档
案清单</td><td colspan="3">（可附页）</td><td>文件/档案签收</td><td></td></tr>
</table>

美女地址

- http://www.mm131.com
- http://www.mmonly.cc/wmtp/fjtp/136734.html

优秀的程序员 会低成本 高效率 高质量完成项目

不仅需要技术 还要学会管理 善用工具

案例:盖房子

普通程序员

老板找来大工张三,相当于导入httpclient.jar

发现缺少一个搬砖的,又打电话找来李四 相当于导入codec.jar

最后发现还缺和水泥的,又打电话找来王五 相当于导入logging.jar

人员全部到位才可以开工
老板需要知道每一个干活的人 把他们一一找到
效率低 类似与面向过程编程

高级程序员

只需要引入 互联网配置

compile group: 'commons-httpclient', name: 'commons-httpclient', version: '3.1‘

一切自动由gradle搞定 无需关心jar包的依赖

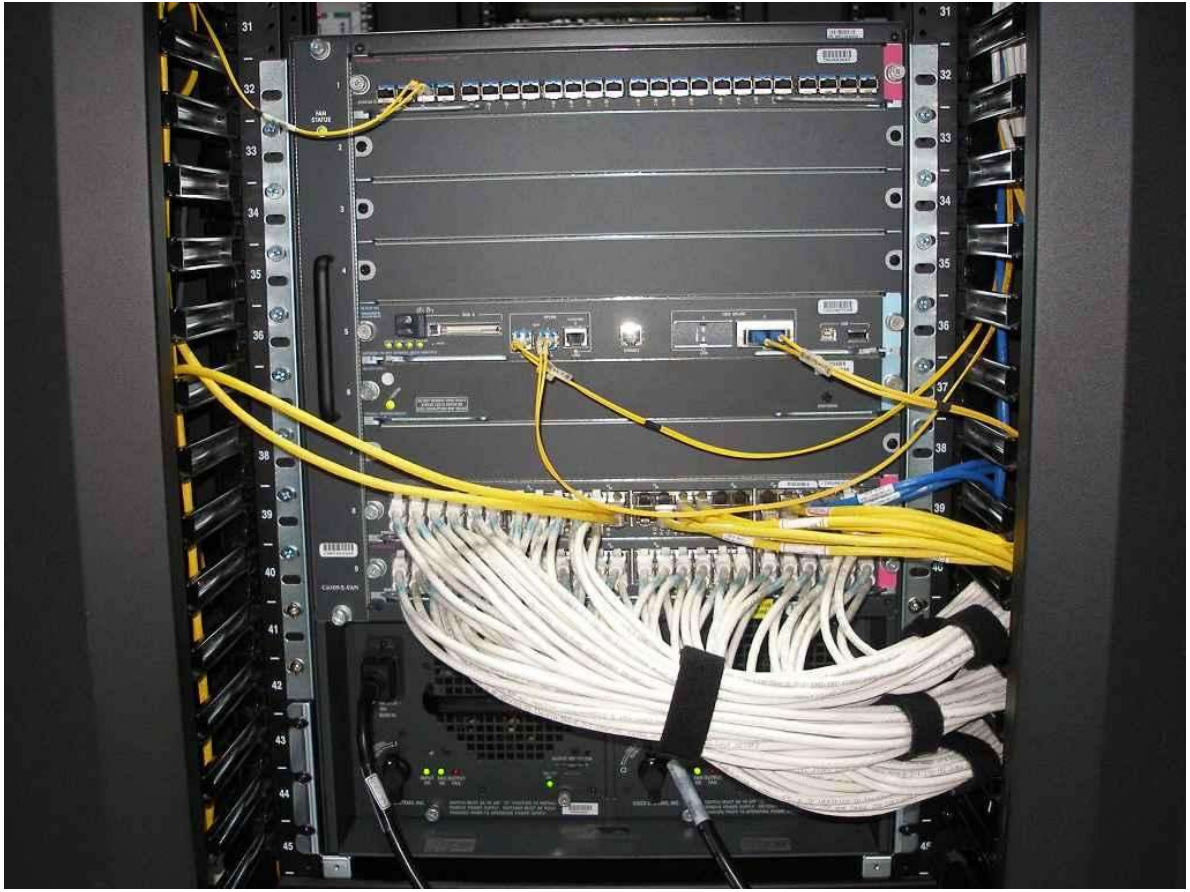
依赖管理

- 指在什么地方以什么形式引入代码
- 告诉kotlin在什么地方以什么形式引入代码

依赖管理

- 所有软件在最开始时都不会很大,随着时间的推移,软件平台不断膨胀,各个组件或模块之间的依赖变的越来越复杂.如果管理不当就会陷入泥潭


快科技
KKJ.CN



代码仓库

- 存放代码的地方

常见仓库

- mavenCentral
- Jcenter
- Local本地仓库(mavenCentral, Jcenter)
- 文件仓库(不建议使用)
- 自定义maven仓库(最常用 maven私服)

依赖库的坐标

- Group(组):用包名来命名,表示由哪一个组织开发 相当于籍贯
- Name(名):项目名 类似于姓名
- Version(版本号):定义jar包的版本 身份证号

- 河北那个身份证为1.0的张三,带着你的兄弟过来

依赖配置阶段

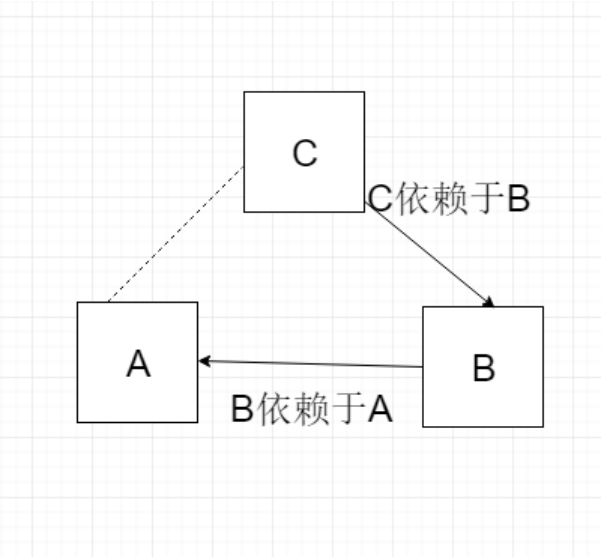
- 编译时依赖 compile
- 测试时依赖 testcompile

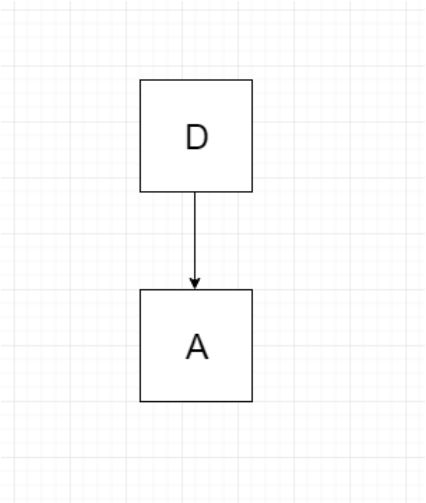
依赖配置阶段

- TDD(test driven development) 测试驱动开发

依赖的冲突

- 依赖的传递性





什么是冲突

项目需要C和D两个依赖
C需要依赖A(haha.jar) 1.0版本
D需要依赖A(haha.jar)2.0版本

调用A里面的sayHello方法
究竟执行的是1.0版本的还是2.0版本的

解决冲突

- Gradle自动依赖最高版本的依赖
- 排除传递性依赖
- 强制指定一个版本

修改默认构建解决方案

```
configurations.all {
    resolutionStrategy{
        failOnVersionConflict()
    }
}
```

强制指定版本号

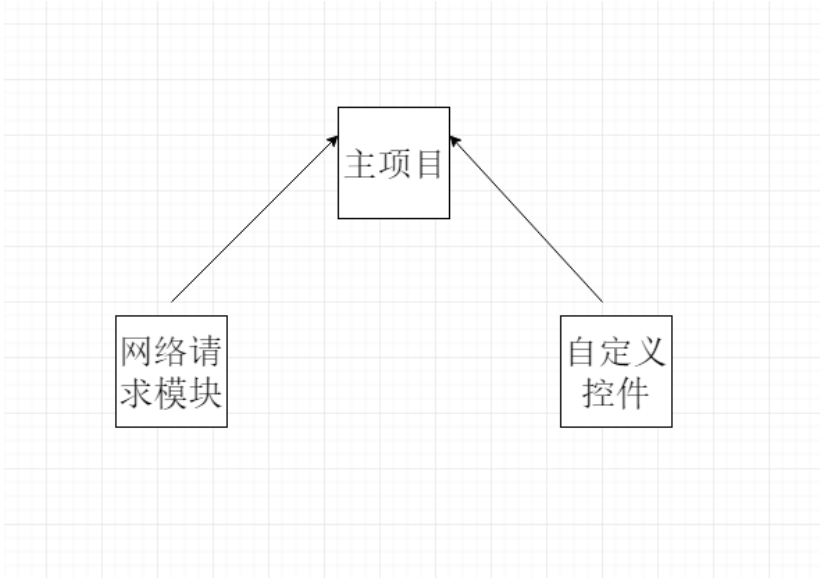
```
configurations.all {
    resolutionStrategy{
        failOnVersionConflict()
        force("commons-logging:commons-logging:1.0.1")
    }
}
```

Gradle扩展任务

```
//扩展gradle的copy任务
task( p0: "mycopy",Copy::class){
    from( ...sourcePaths: "src/main/java")
    into( destDir: "temp")
}
```

Gradle多项目构建

分模块开发



多模块开发

- 软件设计原则:高内聚，低耦合
- 分模块开发

分模块开发

- setting.gradle:module的全局配置
- 工程下build.gradle:整个工程的构建
- module下的build.gradle:当前module的gradle构建

buildscript

- gradle构建需要依赖的插件

```
buildscript {
    repositories {
        maven {
            setUrl("https://plugins.gradle.org/m2/")
        }
    }
    dependencies {
        classpath( dependencyNotation: "org.jetbrains.kotlin:kotlin-gradle-plugin:1.2.31")
    }
}
```

subprojects

- subprojects:所有子模块的配置

```
subprojects {
    group = "com.itcast.gradle"
    version = "1.0-SNAPSHOT"
    //引入kotlin插件
    apply {
        plugin( p0: "org.jetbrains.kotlin.jvm")
    }
    //引入仓库
    repositories {
        mavenCentral()
    }
}
```

allprojects

- allprojects :所有子模块的配置

```
allprojects {
    group = "com.itcast.gradle"
    version = "1.0-SNAPSHOT"
    //引入kotlin插件
    apply {
        plugin( p0: "org.jetbrains.kotlin.jvm")
    }
    //引入仓库
    repositories {
        mavenCentral()
    }
}
```