

Android 10

Kotlin

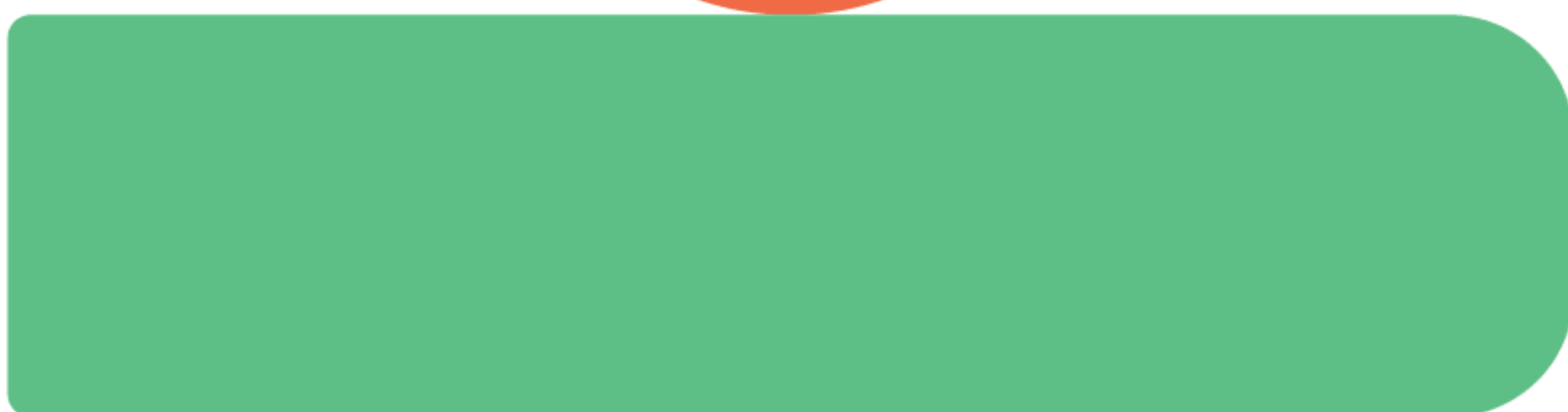
编程通俗演义

牛搞 著

清华大学出版社



掌握Kotlin编程
提高Android应用开发效率



Android 10

Kotlin

编程通俗演义

牛搞 著



清华大学出版社
北京

内 容 简 介

Google 已经将 Kotlin 列为 Android 开发第一开发语言。Kotlin 与 Java 无缝兼容，同时 Kotlin 作为一门新语言，其语法极其简洁精练，稍微熟悉之后，开发效率立即会有明显提升。

本书分为 20 章，严格参考 Android 10 官方开发文档，全面讲解利用 Kotlin 开发 Android 应用的各种技术，章节精心安排、循序渐进，内容准确、翔实、全面而又通俗易懂，绝不是术语的罗列，也绝不是不知所云的翻译。

本书既适合 Android 应用开发初学者、转向 Kotlin 编程的 Android 应用开发人员阅读，也适合高等院校和培训学校计算技术相关专业的师生参考。

本书封面贴有清华大学出版社防伪标签，无标签者不得销售。

版权所有，侵权必究。侵权举报电话：010-62782989 13701121933

图书在版编目 (CIP) 数据

Android 10 Kotlin 编程通俗演义/牛犒著. —北京：清华大学出版社，2020.5

ISBN 978-7-302-55274-1

I. ①A… II. ①牛… III. ①移动终端—应用程序—程序设计 IV. ①TN929.53

中国版本图书馆 CIP 数据核字 (2020) 第 050762 号

责任编辑：夏毓彦

封面设计：王 翔

责任校对：闫秀华

责任印制：杨 艳

出版发行：清华大学出版社

网 址：<http://www.tup.com.cn>，<http://www.wqbook.com>

地 址：北京清华大学学研大厦 A 座 邮 编：100084

社 总 机：010-62770175 邮 购：010-62786544

投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn

质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn

印 装 者：三河市铭诚印务有限公司

经 销：全国新华书店

开 本：190mm×260mm

印 张：26.75

字 数：685 千字

版 次：2020 年 6 月第 1 版

印 次：2020 年 6 月第 1 次印刷

定 价：89.00 元

产品编号：084558-01

前言

写作背景

2020年了，Android 开发的热度怎么样了？学习它，对就业和薪资提升帮助大吗？我想这是大多数人最关心的问题。

一门技术在职场中的需求热度，通过大型求职招聘网站可以很容易分析出结论。大体可以这样说，移动端开发作为软件生态的一部分，从来都有很强的需求。在 2017 年之前，Android 原生开发曾一度进入低谷，因为很多团队都选择基于 JavaScript 的跨平台开发框架。但是，这些框架也存在一些先天缺陷，主要是由于 Android 与 iOS 的巨大差异造成的（这两大系统不可能统一，为了商业利益，必须互相制造壁垒）。事实已经证明了一点，跨平台开发始终绕不开原生开发。所以，2017 年之后，Android 原生开发重新被重视，甚至有国外公司完全回归了原生开发。当前，跨平台开发依然在迅速发展，但是始终绕不过原生开发，而且有些功能只能用原生开发实现。所以，要进行移动开发，必须学习原生开发！

本书作者有 15 年以上软件开发实战经验、5 年以上 IT 实训教学经验，深入了解各种技术、架构、设计模式，对 IT 教育有丰富的体验和深入的思考，对各种技术善于以通俗易懂的语言进行透彻讲解。

本书导读

本书是《Android 9 编程通俗演义》的姊妹篇，作者在其基础上修正部分错误，改进多处设计，将开发语言由 Java 升级为 Kotlin，紧紧追赶 Google 的步伐。

“我有一个梦想，让天下没有难学的技术！”本书与《Android 9 编程通俗演义》一书的写作风格一致：通俗易懂，具体直观，注重实践，以为读者节省脑细胞作为终极目标。

我一直希望能写出一本让读者轻轻松松学编程的书，如果能把学习当作一种休闲方式，那该是多么美好的事情！当然了，众口难调，一本书的风格不可能满足所有人的口味。在本书创作中，作者已尽量做到照顾更多的人，尤其照顾基础差的人，并且尽量少说黑话，努力使它成为一部不那么“反人类”的作品，相信大部分人都很容易接受这种风格。因为从上一本书的读者反馈看来，效果很不错！

本书应该怎么去阅读？答案就一句话：“看就行了！”

如果你是一个勤快人，可以边看边跟着做；如果是一个懒人，那么仅仅停留在“看”上。你可以躺着看、坐着看、趴着看，最好不要走路看，因为对眼睛不好。

本书翔实地讲述一个 Android App 的实现过程，并对很多基础知识进行了专门补齐。实现 App 的每一步都有截图，你不用写代码，也能看到结果。所以，阅读体验是很轻松的。

本书从头至尾讲了一个故事：开发一个 Android 版高仿 QQ App 的故事。本书的内容结构是这样的：

- 第 1 章：Kotlin 语言快速入门。
- 第 2~4 章：Android 开发准备与初步体验。
- 第 5~14 章：Android 基本功能与界面开发。
- 第 15、16 章：实现仿 QQ App 单机版。
- 第 17~19 章：Android 多线程、网络开发。
- 第 20 章：实现仿 QQ App 网络聊天版。

示例源码下载

第 14 章之前讲解基础知识，示例项目为 FirstCotlinApp，其 Git 仓库地址是 <https://gitee.com/nnn/FirstCotlinApp.git>。

第 15 章和第 16 章的项目为无网络通信的仿 QQApp，项目名为 QQApp，其 Git 仓库地址为 <https://gitee.com/nnn/qqAppCotlin.git>。

第 20 章的项目为带网络通信功能的仿 QQApp，是从 QQAppCotlin 改进而来的，因此项目名和包名皆与 QQAppCotlin 相同，其 Git 仓库地址为 <https://gitee.com/nnn/qqAppCotlinHttp.git>。

另外，为了模仿 QQApp 中的树状显示效果，作者还创建了一个开源项目 RecyclerViewTreeView，托管于 GitHub，现已被多人用于商业项目。在本书中亦有对其用法的详细介绍，地址为 <https://github.com/niugao/RecyclerViewTreeView>。

对本书内容或各项目有任何疑问，可在 gitee 或 GitHub 中的项目仓库页面直接留言，也可在作者的 CSDN 博客 https://blog.csdn.net/niu_gao/ 中留言。

读者对象

- 了解 Java 语言，想学习 Kotlin 语言和 Android 开发的初学者
- 想快速了解 Android 开发模式的资深开发人士
- 有一定 Android 开发基础，想进一步提升实战能力的开发人员
- 需要工程教育实践案例的高校教师

致谢

首先感谢各位读者，你们的肯定给予我笔耕不辍的信心和动力！其次要感谢清华大学出版社夏毓彦编辑的大力支持和指导，让我可以专注于内容，充分体验作为作者的乐趣。再次感谢我的家人和朋友，是你们的鼓励与支持给了我动力。最后感谢我自己，耐住寂寞，坚持不辍，能为世人留下一两部作品，真的感觉人生没有虚度。

牛 搞
2020 年 3 月

目 录

第 1 章 Kotlin 快速入门	1	4.2 虚拟机加速	42
1.1 开发环境配置	1	4.2.1 在 BIOS 中开启虚拟化支持	43
1.1.1 安装 JDK	1	4.2.2 安装 HAXM	43
1.1.2 安装 IDE	2	4.3 App 的样子	44
1.1.3 创建第一个 Kotlin 工程	3	4.4 工程里面有什么	44
1.1.4 工程组织结构	5	第 5 章 UI 资源与 Layout	46
1.1.5 添加代码	6	5.1 Layout	46
1.1.6 运行程序	6	5.2 改动 Layout	49
1.2 大道至简	8	5.2.1 添加图像资源	52
1.3 万变不离其宗	10	5.2.2 文件或文件夹改名	53
1.4 新式语法特征	11	5.2.3 显示自己的图像	53
1.5 Kotlin 独特语法	17	5.2.4 XML 小解	56
1.6 作用域函数	23	5.2.5 Layout 源码解释	57
1.6.1 let()	24	5.3 ConstraintLayout	58
1.6.2 run()	24	5.3.1 ConstraintLayout 的原理	59
1.6.3 apply()	25	5.3.2 子控件在 ConstraintLayout 中居左或居右	60
1.6.4 also()	25	5.3.3 子控件在 ConstraintLayout 中横向居中	61
1.6.5 with()	26	5.3.4 子控件在 ConstraintLayout 中居中偏左	62
1.7 新式语法特点总结	26	5.3.5 子控件 A 在子控件 B 的上面	62
第 2 章 Android 系统简介	27	5.3.6 子控件 A 与子控件 B 左边对齐	63
第 3 章 Android 开发环境搭建	29	5.3.7 设置子控件的宽和高	64
3.1 下载 Android Studio	29	5.3.8 子控件的宽和高保持一定比例	65
3.2 安装 Android Studio	30	5.4 设计登录页面	67
3.3 配置 Android SDK	31	5.4.1 添加用户名输入控件	67
3.4 四项原则	34		
第 4 章 第一个 Kotlin App	35		
4.1 运行 App	37		
4.1.1 在真实设备上调试	38		
4.1.2 配置虚拟机	40		

5.4.2	添加密码输入控件	69	8.2	启动注册页面	100
5.4.3	添加登录按钮	70	8.2.1	修改页面标题	100
5.4.4	完成收工	70	8.2.2	MainActivity 源码	101
5.5	让内容滚动	72	8.3	设计注册页面	102
5.5.1	添加 ScrollView 作为最外层 容器	73	8.4	响应注册按钮进行注册	106
5.5.2	禁止旋转	75	8.5	获取页面返回的数据	107
5.5.3	为横屏和竖屏分别创建 Layout	76	8.5.1	避免常量重复出现	108
5.5.4	让内容居中	77	8.5.2	日志输出	110
5.6	添加新的 Layout 资源	77	8.5.3	将返回的数据设置到 控件中	111
第 6 章	各种 Layout 控件	79	8.6	ActionBar 上的返回图标	111
6.1	FrameLayout	79	8.6.1	原生 Action Bar 与 MaterialDesign Action Bar	112
6.2	LinearLayout	79	8.6.2	登录页面显示返回图标	112
6.2.1	纵向 LinearLayout 中子控件 横向居中	80	8.6.3	注册页面显示返回图标	114
6.2.2	子控件均匀分布	81	8.7	ScrollView 与软键盘	114
6.2.3	子控件按比例分布	81	8.8	源码	115
6.2.4	用 LinearLayout 实现登录 界面	83	8.8.1	MainActivity	115
6.3	GridLayout	85	8.8.2	RegisterActivity.kt	117
6.4	TableLayout	87	第 9 章	Theme	119
第 7 章	操作控件	89	第 10 章	Fragment	121
7.1	在 Activity 中创建界面	89	10.1	弄巧成拙的 Activity	121
7.1.1	类 R	90	10.2	使用 Fragment	123
7.1.2	类 Activity	90	10.3	改造登录页面	125
7.1.3	四大组件	90	10.3.1	添加 layout 文件	125
7.2	在代码中操作控件	91	10.3.2	改变 layout 文件的 内容	126
7.2.1	获取控件	91	10.3.3	添加 Fragment 类	126
7.2.2	响应 View 的事件	93	10.3.4	将 Fragment 放到 Activity 中	130
7.2.3	添加依赖库	93	10.3.5	创建注册 Fragment	132
7.2.4	显示提示	95	10.3.6	显示 RegisterFragment	133
7.2.5	完成收工	97	10.3.7	通过 AppBar 控制页面 导航	133
第 8 章	Activity 导航	98	10.3.8	实现 RegisterFragment 的 逻辑	134
8.1	创建注册页面	98			

10.3.9	从 LoginFragment 中读出 用户名和密码	136	12.7.2	自定义转场动画	169
10.3.10	Fragment 的生命周期	137	第 13 章	自定义控件	174
10.3.11	Fragment 状态保存与 恢复	137	13.1	创建一个 Custom View	175
10.3.12	总结	138	13.2	Custom View 类	176
10.4	对话框	141	13.2.1	构造方法	176
10.4.1	创建子类	142	13.2.2	onDraw()方法	177
10.4.2	显示对话框	143	13.2.3	init()方法	179
10.4.3	响应返回键	144	13.2.4	自定义属性	182
第 11 章	菜单	145	13.2.5	作画	184
11.1	添加菜单资源	145	13.3	创建圆形图像控件	185
11.2	重写 onCreateOptionsMenu()	147	13.3.1	将 Drawable 转成 Bitmap	188
11.3	嵌套菜单	148	13.3.2	变换矩阵	189
11.4	菜单项分组	149	13.3.3	自定义属性的改动	190
11.5	响应菜单项	150	13.3.4	类的所有代码	191
11.6	其他菜单类型	151	第 14 章	RecyclerView	197
第 12 章	动画	152	14.1	基本用法	197
12.1	动画原理	152	14.2	显示多条简单数据	198
12.2	三种动画	153	14.2.1	添加新页面	198
12.3	视图动画	154	14.2.2	创建 Adapter 子类	200
12.3.1	绕着中心转	155	14.2.3	设置 RecyclerView	202
12.3.2	不要反向转	155	14.2.4	用集合保存数据	203
12.3.3	举一反三	156	14.3	让子控件复杂起来	204
12.3.4	动画组	157	14.3.1	创建行 Layout 资源	204
12.4	属性动画	158	14.3.2	应用条目 Layout 资源	206
12.4.1	旋转动画	158	14.3.3	明显区分每一行	207
12.4.2	动画组	159	14.3.4	使用音乐信息类	209
12.5	动画资源	163	14.4	增删改	210
12.6	Layout 动画	165	14.4.1	增加一条数据	210
12.6.1	向 Layout 控件添加 子控件	165	14.4.2	其他操作	212
12.6.2	ViewGroup	167	14.5	局部刷新	212
12.6.3	设置排版动画	167	14.6	响应条目选择	213
12.7	转场动画	169	14.7	显示不同类型的行	214
12.7.1	使用默认转场动画	169	14.7.1	添加新条目数据类	214
			14.7.2	添加条目 Layout	215

14.7.3	创建新的 ViewHolder 类	216	第 16 章	实现聊天界面	313
14.7.4	区分不同的 View Type	216	16.1	原理分析	313
第 15 章 模仿 QQ App 界面			16.2	创建聊天 Activity	313
15.1	创建新的 Android 项目	218	16.2.1	activity_chat.xml	313
15.2	设计登录页面	218	16.2.2	类 ChatActivity	316
15.2.1	创建登录 Fragment	219	16.2.3	显示消息的 Layout	318
15.2.2	设计登录界面	220	16.3	启动 ChatActivity	320
15.2.3	UI 代码	221	16.4	模拟聊天	321
15.2.4	显示登录历史	224	第 17 章 多线程		
15.2.5	设计历史菜单项	228	17.1	线程与进程的概念	323
15.2.6	实现显示历史的代码	229	17.2	创建线程	324
15.2.7	selector 资源	229	17.3	创建线程的另一种方式	325
15.2.8	layer_list 资源	230	17.4	多个线程操作同一个对象	326
15.2.9	定制控件背景	231	17.5	单线程中异步执行	329
15.2.10	动画显示菜单	231	17.6	多线程间同步执行	330
15.2.11	让菜单消失	233	17.7	在其他线程中操作界面	330
15.2.12	响应选中菜单项	234	17.8	HandlerThread	333
15.3	QQ 主页面设计	235	17.9	线程的退出	333
15.3.1	设置导航栏	237	第 18 章 网络通信		
15.3.2	设置 Tab 栏	239	18.1	网络基础知识	336
15.3.3	改变 Tab Item 图标	241	18.1.1	IP 地址与域名	336
15.3.4	为 ViewPager 添加内容	242	18.1.2	TCP 与 UDP	337
15.3.5	ViewPager 与 TabLayout 联动	245	18.1.3	HTTP 协议	337
15.3.6	使用 SpannableString 显示图像	247	18.2	Android HTTP 通信	338
15.3.7	禁止 ViewPager 滑动翻页	251	18.3	使用“异步任务”	341
15.3.8	创建“消息”页	252	18.3.1	定义异步任务类	341
15.3.9	显示气泡菜单	258	18.3.2	使用异步任务类	342
15.3.10	抽屉效果	271	18.3.3	完善异步任务类	344
15.3.11	创建“联系人”页	286	18.3.4	异步任务的退出	349
15.3.12	创建“动态”页	303	18.4	使用 OkHttp 进行网络通信	351
15.3.13	实现搜索功能	304	18.4.1	使用 OkHttp 下载图像	352
			18.4.2	创建 Web 服务端	354
			18.4.3	使用 OkHttp 下载数据	355
			18.4.4	JSON 转对象	357
			18.4.5	使用 OkHttp 上传文件	358

18.5	使用 Retrofit 进行网络通信	360	20.2.2	添加 Fragment 回调接口	400
18.5.1	加入 Retrofit 的依赖项	360	20.2.3	发出登录请求	401
18.5.2	用 Retrofit 下载文本	361	20.2.4	保存自己的信息	403
18.5.3	用 Retrofit 下载图像	363	20.2.5	防止按钮重复单击	403
18.5.4	用 Retrofit 上传图像	364	20.2.6	显示进度条	404
第 19 章	异步调用库 RxJava	366	20.3	获取联系人	406
19.1	小试牛刀	366	20.3.1	修改 Retrofit 接口	407
19.2	精简发送代码	369	20.3.2	使用 RxJava 定时器	407
19.3	精简接收代码	370	20.3.3	添加 Fragment 回调接口	408
19.4	map 与 flatmap	371	20.3.4	获取并显示联系人	408
19.5	并行 map	373	20.3.5	出错重试	410
19.6	RxJava 与 Retrofit 合体	374	20.3.6	停止网络连接	411
19.7	RxJava Retrofit 合体并行执行	376	20.4	发出聊天消息	413
19.8	RxJava 与 Activity 的配合	377	20.4.1	定义承载消息的类	413
第 20 章	实现聊天功能	378	20.4.2	在接口中添加方法	414
20.1	添加注册功能	378	20.4.3	在 ChatActivity 中初始化 Retrofit	414
20.1.1	创建注册 Activity	378	20.4.4	上传消息	415
20.1.2	设计注册页面	379	20.4.5	失败重传	416
20.1.3	显示 Bottom Sheet	381	20.5	获取聊天消息	417
20.1.4	拍照	384	20.5.1	为 ChatService 增加方法	417
20.1.5	提交注册信息	392	20.5.2	发出请求	417
20.2	改进登录功能	399			
20.2.1	创建 Retrofit 相关实例	399			

第 1 章

◀ Kotlin快速入门 ▶

Java 和 Kotlin 都是 Android 的官方开发语言，但是 Kotlin 已上升为第一开发语言，Java 屈居第二。

Kotlin 的官网地址是 <https://kotlinlang.org>。

Kotlin 在底层与 Java 完全兼容，而且 Kotlin 是强类型语言，编译产物是 Java 的 class 文件，要基于虚拟机运行，所以 Kotlin 与 Java 可以说是一体两面、无缝结合。但是，Kotlin 比 Java 更进一步，它编写的程序可以做到不依赖于虚拟机运行，这被称为 Native（原生）方式，就像 C 程序的运行方式，当然比虚拟机快多了，这种运行方式对于移动设备来说意义重大！

如果说 Kotlin 代表了未来开发语言的方向也不算夸张，因为它很新，站在了前人的肩膀上。如果去研究一下各种新出现的语言（比如 Apple 的 Swift），会发现它们的语法规则几乎完全一样。

当前的 Java 使用者大都还停留在第 8 版（JDK 1.8），因为很多库、框架或系统都最高支持到 Java 8。写作此书时，Java 13 就要出世了，Java 8 之后的语法改进有很多。这些改进都体现了新式语法，但是很多人对新式语法不熟悉，甚至看到后感到别扭，然而新式语法思想是每个软件开发者都应该理解和掌握的。

其实要掌握新式语法并不困难，还可以说是一件很轻松的事。万变不离其宗，只要掌握了一门语言，再学另一门就很快，当然要有一个条件：有一本好的、适合的指引手册。本书就是为 Java 开发者提供的一本 Kotlin 快速学习手册。

1.1 开发环境配置

开发环境的配置仅需两步：安装 JDK；安装 IDE。

1.1.1 安装 JDK

在地址“<https://www.oracle.com/technetwork/java/javase/downloads/index.html>”中选择要下载的 JDK，见图 1-1。

单击图 1-1 箭头所指图标，进入新页面，在最下面能看到图 1-2 所示的内容。



图 1-1

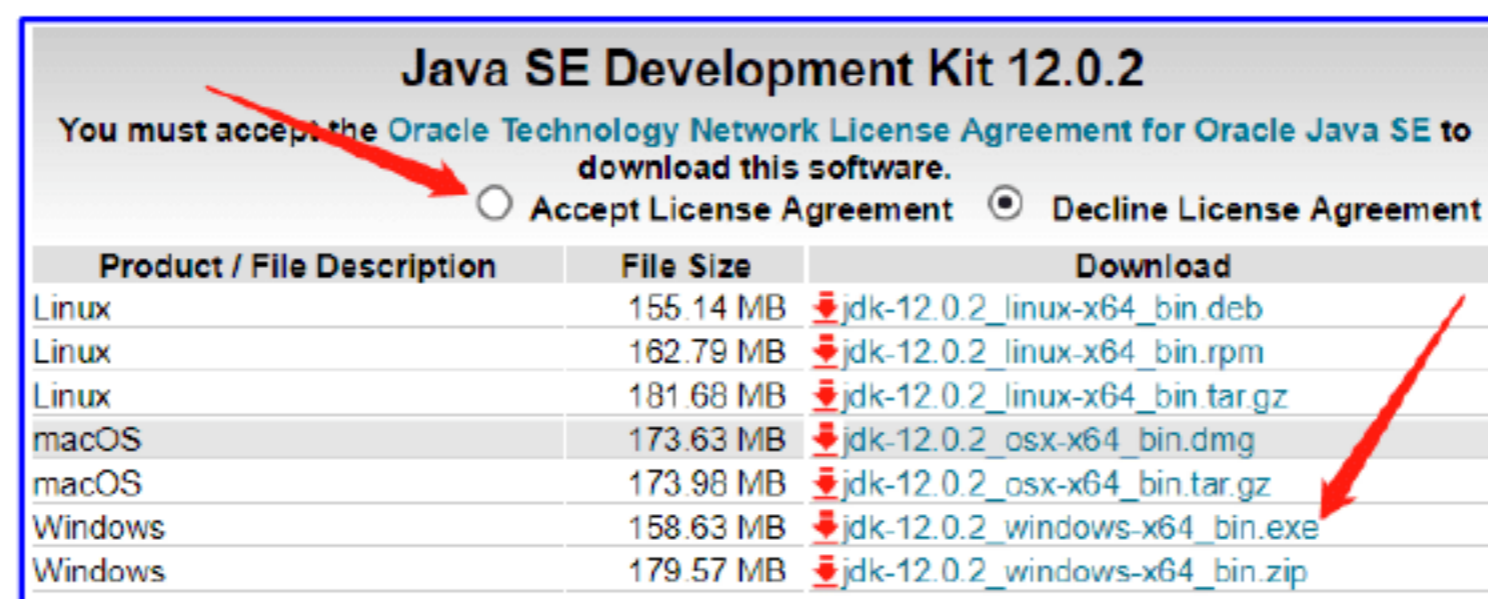


图 1-2

注意一定要选择“Accept License Agreement（同意许可协议）”，才可以用鼠标单击下面的文件链接。

如果是 Windows，建议选择可执行文件（它是安装包，可以自动设置很多配置），并且在安装过程中不要改变默认安装路径，这样不会引起不必要的麻烦。

需要注意的是，JDK 安装到的路径中不能有中文，否则会引起莫名其妙的问题。默认安装位置一般是“系统盘：\Program files”。

1.1.2 安装 IDE

仅有 JDK 虽然可以开发软件，但是要手动维护一切，可以借助开发工具来提高编程效率。Kotlin 是 JetBrains 开发出来的，而 JetBrains 的主业为开发工具，所以选择 JetBrains 家的 IDEA。

IDEA 可以说是 Java 编程的首选 IDE，当然也是 Kotlin 的首选，它的官网地址是“<https://www.jetbrains.com/idea/>”。下载 IDEA 很简单，在页面中单击图 1-3 所指的“DOWNLOAD”图标即可。进入下载页面，如图 1-4 所示。



图 1-3

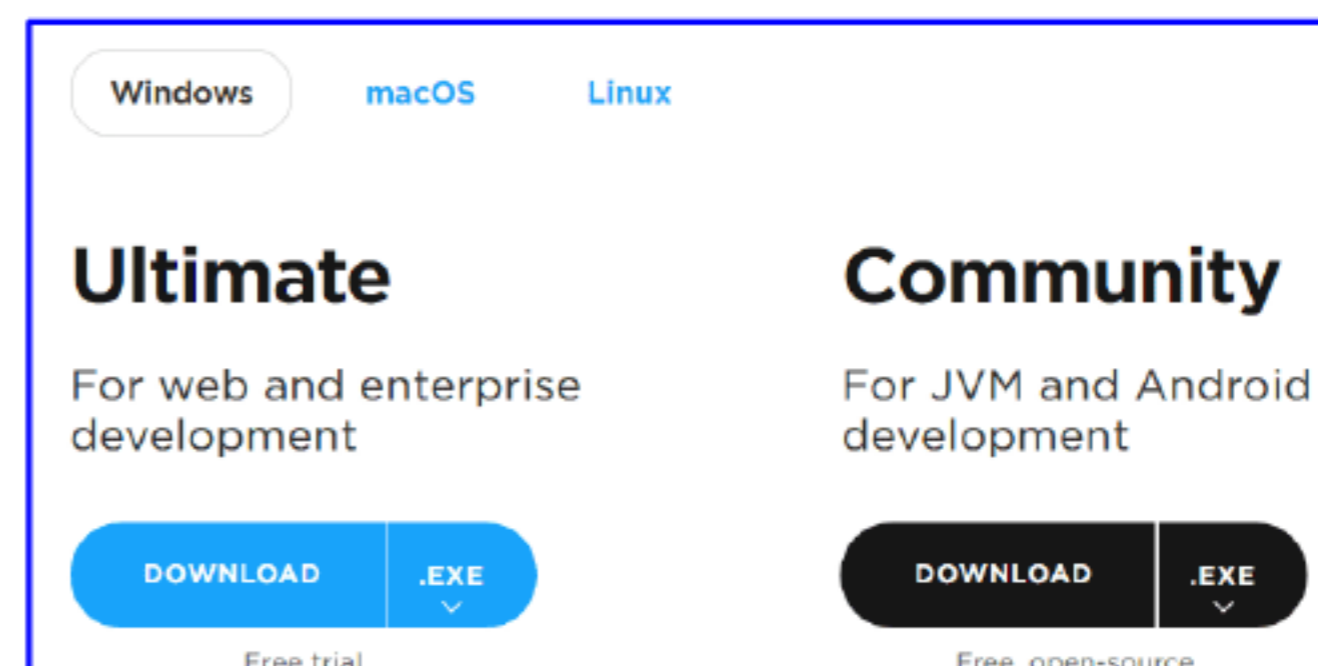


图 1-4

有两个版本可供选择：Ultimate（旗舰版）和 Community（社区版）。旗舰版功能强大，但是要收费；社区版功能少，是免费的。选择社区版来学习 Kotlin 就足够了。

下载的文件是一个安装包，运行它即可安装 IDEA。安装过程最好保持默认设置，安装完就可以用。

跟 JDK 一样，注意 IDEA 所安装到的路径中不能有中文，否则亦会引起莫名其妙的问题，它的默认安装位置一般是“系统盘：\Program files”。

1.1.3 创建第一个 Kotlin 工程

其实 Android 的 IDE Android Studio 就是 IDEA。Google 为 IDEA 开发了 Android 插件，把它和 IDEA 绑定在一起（取名为 Android Studio）供我们下载，所以 Android Studio 的界面与 IDEA 的界面是一样的。

第一次运行 IDEA，会出现如图 1-5 所示的页面。

IDEA 的功能项有“创建新工程”（Create New Project）“引入工程”（Import Project）“打开已有工程”（Open）“从版本控制系统导入工程”（Check out from Version Control）。要创建新工程，选择第一项之后出现创建工程向导，如图 1-6 所示。

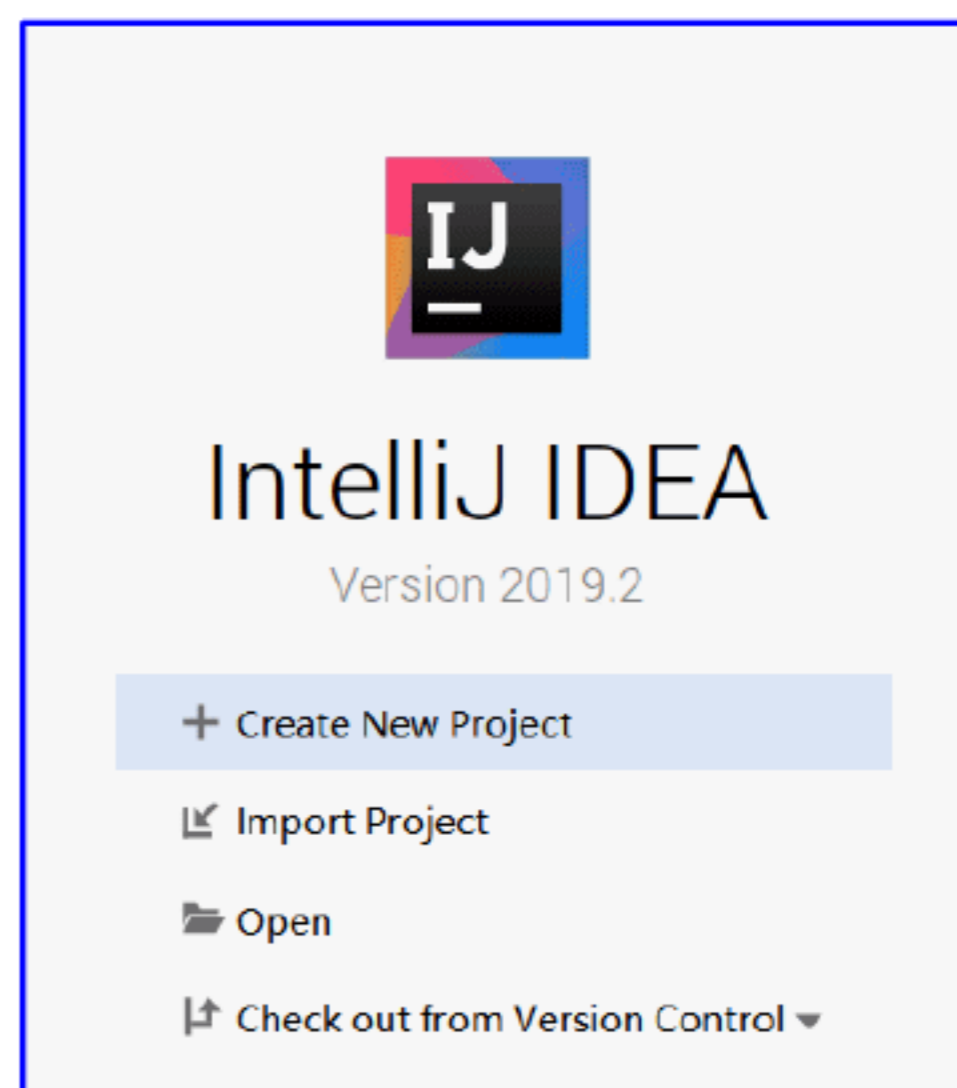


图 1-5

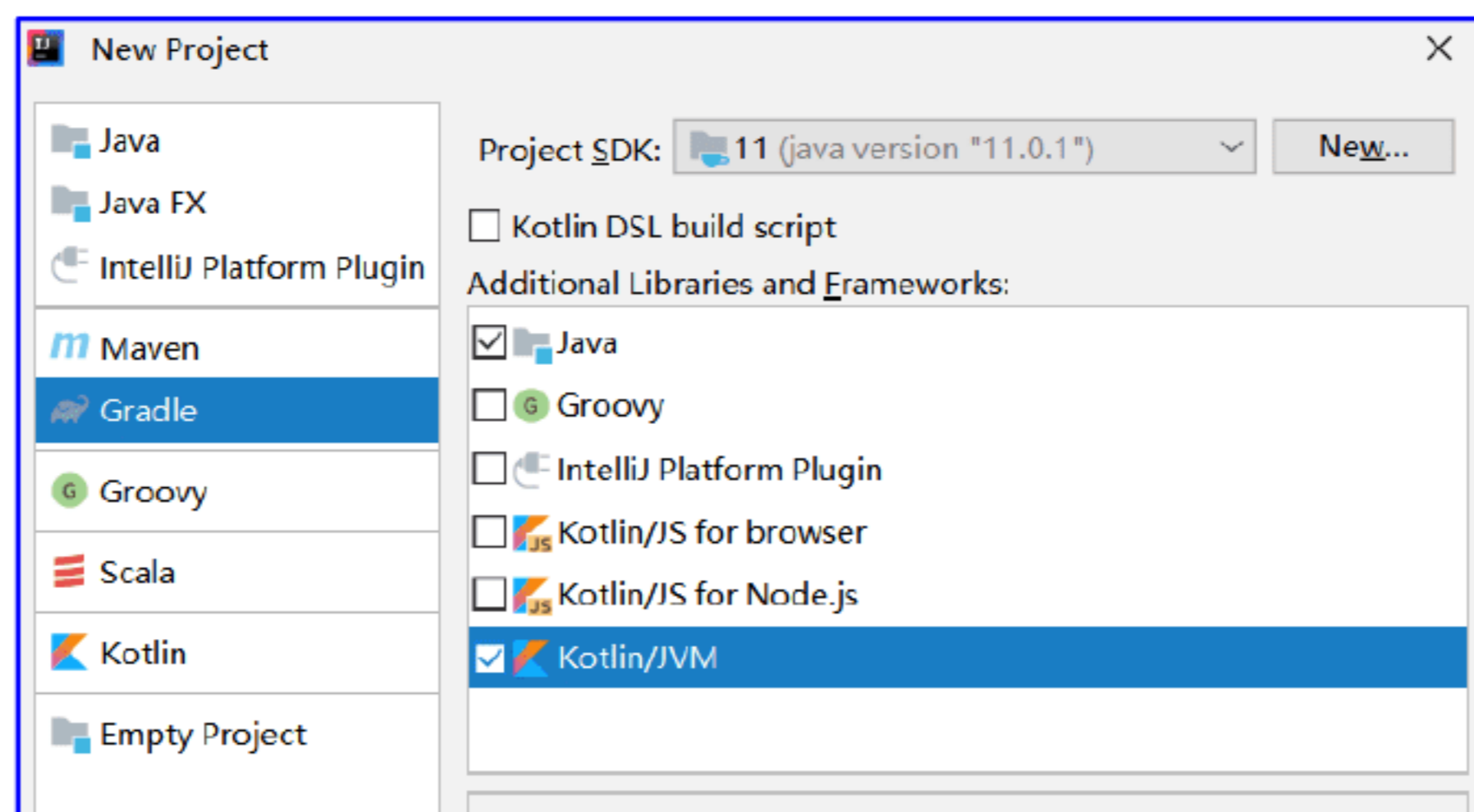


图 1-6

可以创建多种类型的工程，仅 Kotlin 工程就有多种形式，那么选择哪一种呢？推荐创建基于 Gradle 的 Kotlin/JVM 工程。Gradle 是当前如日中天的工程管理软件之一（另一个与它齐名的是 Maven），主要用于管理 Java 工程，但是 Kotlin 与 Java 是同一“种族”，所以也适合使用 Gradle 管理。使用 Kotlin 可以开发多种程序：

- Kotlin/JS 表示用 Kotlin 开发 JavaScript 程序，其实是把 Kotlin 代码翻译成 JavaScript 代码，然后才能在浏览器或 Node.js 环境中执行。
- Kotlin/JVM 表示用 Kotlin 开发基于 JVM 的程序，也就是基于 Java 虚拟机运行的程序，其实就是以 Kotlin 代替 Java 编写代码，编译出的就是 class 文件。

建议选择 Kotlin/JVM 类型的工程，因为 Kotlin 对此类型支持得最好，在此类型的工程中可以使用 Kotlin 的所有特性。选好后单击 Next（下一步）按钮，出现如图 1-7 所示的内容。

在图 1-7 中可以设置程序的名字，在 GroupId（组名）中一般填入的是颠倒的域名，这里主要用于区分不同组织发布的程序，因为域名肯定是唯一的，所以都填写域名。本例中填的是“com.niuedu”，随手写的，只是为了演示一下。

ArtifactId（产品名）指的是程序名，默认与工程同名，所以最好不要用中文，中间也不能用空格等非常规字符，比如取名“HelloKotlin”就合乎规则和习惯。填完后，单击 Next 按钮，进入图 1-8 所示的页面。

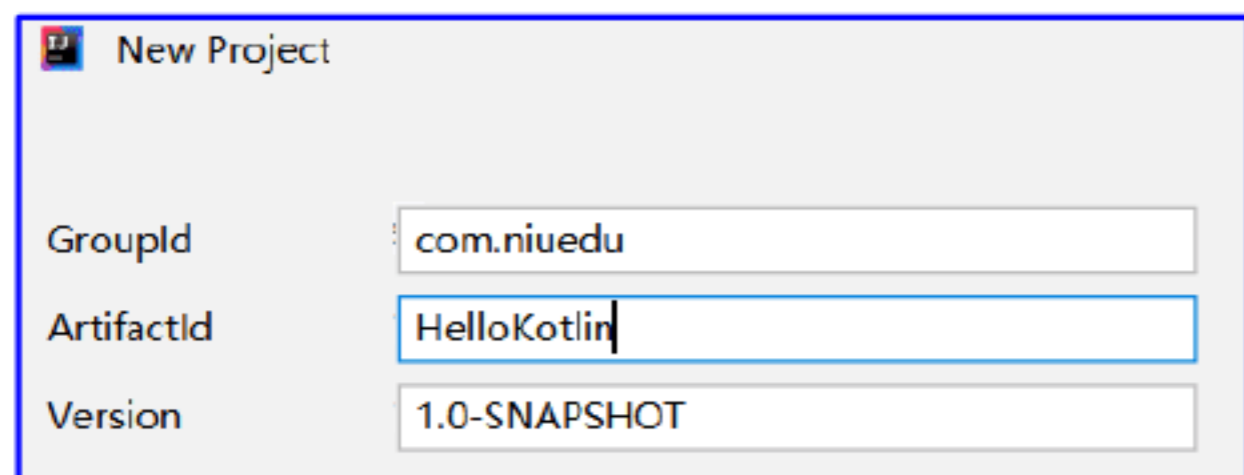


图 1-7

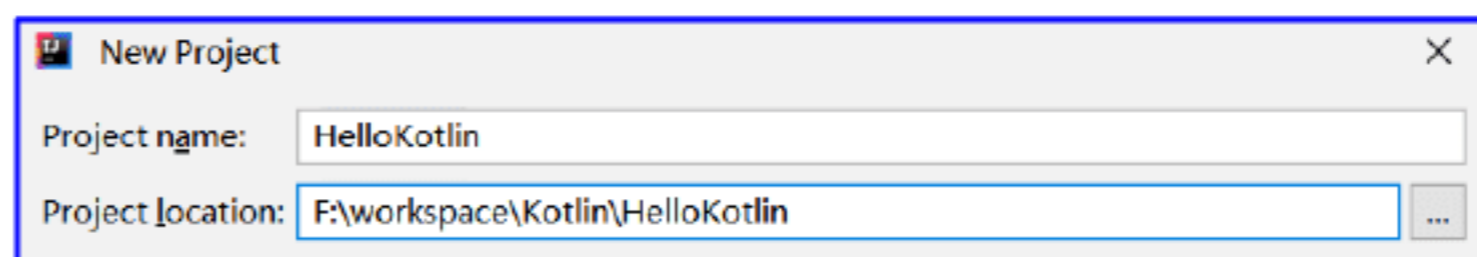


图 1-8

可以修改工程的名字和工程所保存到的路径，这里就用默认的路径，单击 Finish（完成）按钮，Gradle 会根据配置自动生成一个工程，同时 IDEA 会进入编辑模式，如图 1-9 所示。

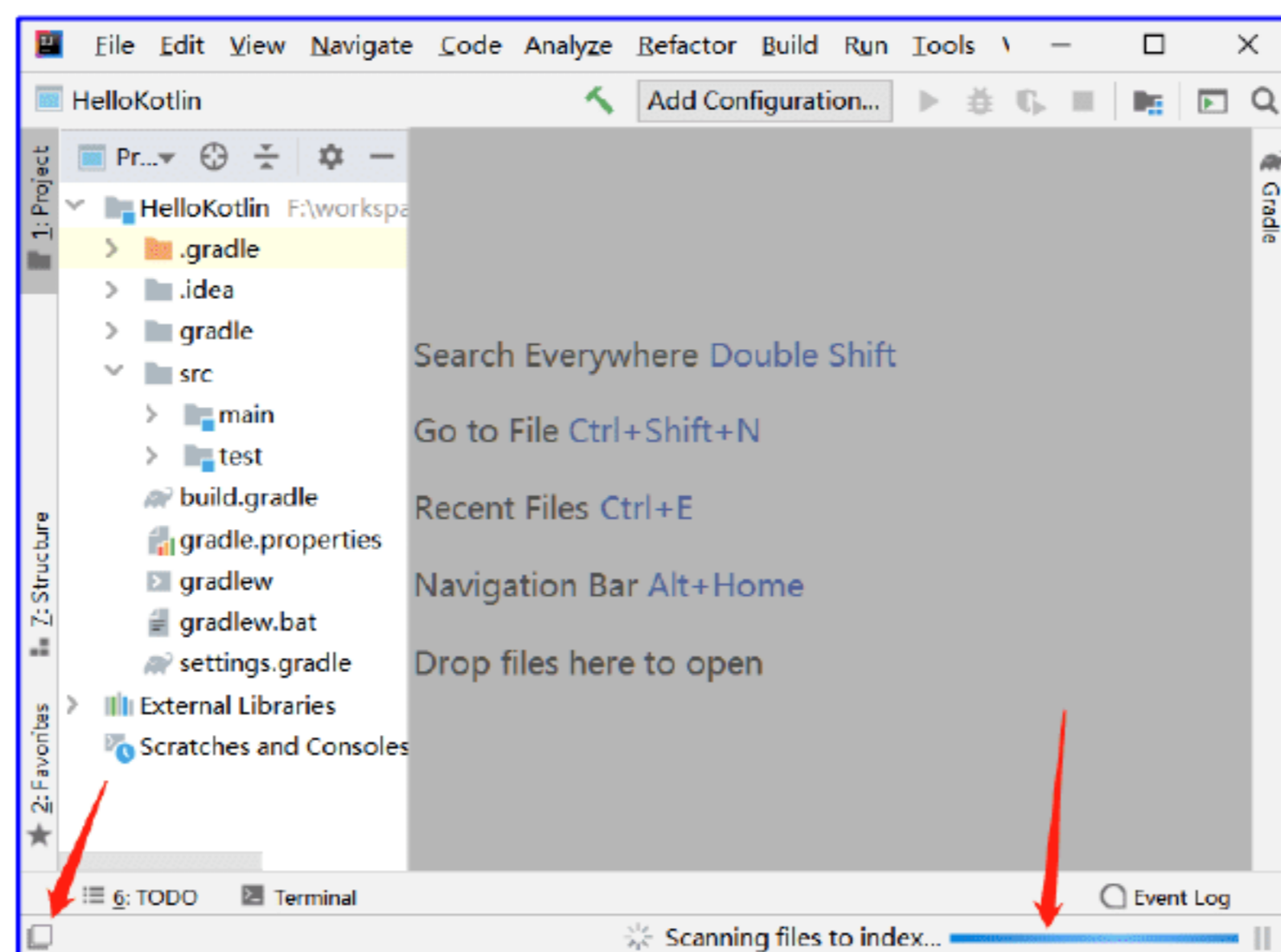


图 1-9

整个工作区的构成非常主流化，左边是工程目录，右边是内容编辑区（现在没有打开任何文件）。注意下面的状态栏，左边这个小图标是切换左右竖向工具条的开关，初用者如果不小心点，就会找不着某些窗口。右边是进度条，要十分注意这个地方，如果此处有进度条或文字出现，则表示 IDEA 正在忙着做什么事，此时最好不要动工程中的文件，等 IDEA 忙完了，再编辑文件。尤其是第一次创建工程，可能需要很长时间，因为 Gradle 工程会严重依赖网络，自动下载很多文件，包括管理工程的插件以及工程所依赖的库，如果网速慢（这是很可能的，因为文件仓库服务器在国外），就可能需要漫长的等待。如果工程创建不成功，十有八九是因为网络问题导致某些文件没有下载成功，这时就需要重试（IDEA 会提示重试）下载。

下面了解一下工程的组织结构。

1.1.4 工程组织结构

当工程创建成功后，可以看到图 1-10 这样的目录结构。

稍微解释一下：`.gradle`、`.idea`、`gradle` 这三个文件夹是 IDEA 自己产生的，用于工程管理，我们不用它们。

`src` 下是工程源码和非源码文件的保存地，但是不能随便放这些文件，`src/main` 下存放的是源码，`java` 下存放的是 Java 源码，`kotlin` 下存放的是 kotlin 源码，`resources` 下存放的是非源码文件，比如图片、配置文件等。`test` 与 `main` 的目录结构相同，`test` 起什么作用呢？它下面存放的是单元测试代码。

注意！这种目录结构是固定的，不要试想通过一些配置来改变目录的名字或作用，这种理念名曰“约定大于配置”。

根目录下的 `build.gradle` 文件是整个工程管理的核心文件，因为它是工程的配置文件，当前它的内容是这样的：

```
plugins {
    id 'java'
    id 'org.jetbrains.kotlin.jvm' version '1.3.41'
}

group 'com.niuedu'
version '1.0-SNAPSHOT'

sourceCompatibility = 1.8

repositories {
    mavenCentral()
}

dependencies {
    implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk8"
    testCompile group: 'junit', name: 'junit', version: '4.12'
}

compileKotlin {
    kotlinOptions.jvmTarget = "1.8"
}

compileTestKotlin {
    kotlinOptions.jvmTarget = "1.8"
}
```

这些代码是用 Groovy 语言编写的，配置了工程的一些工具或参数，比如工程管理所需插件(Plugins)、源码兼容性(Source Compatibility)、仓库(Repositories)、依赖的库(Dependencies)等。改动比较多的是依赖库。

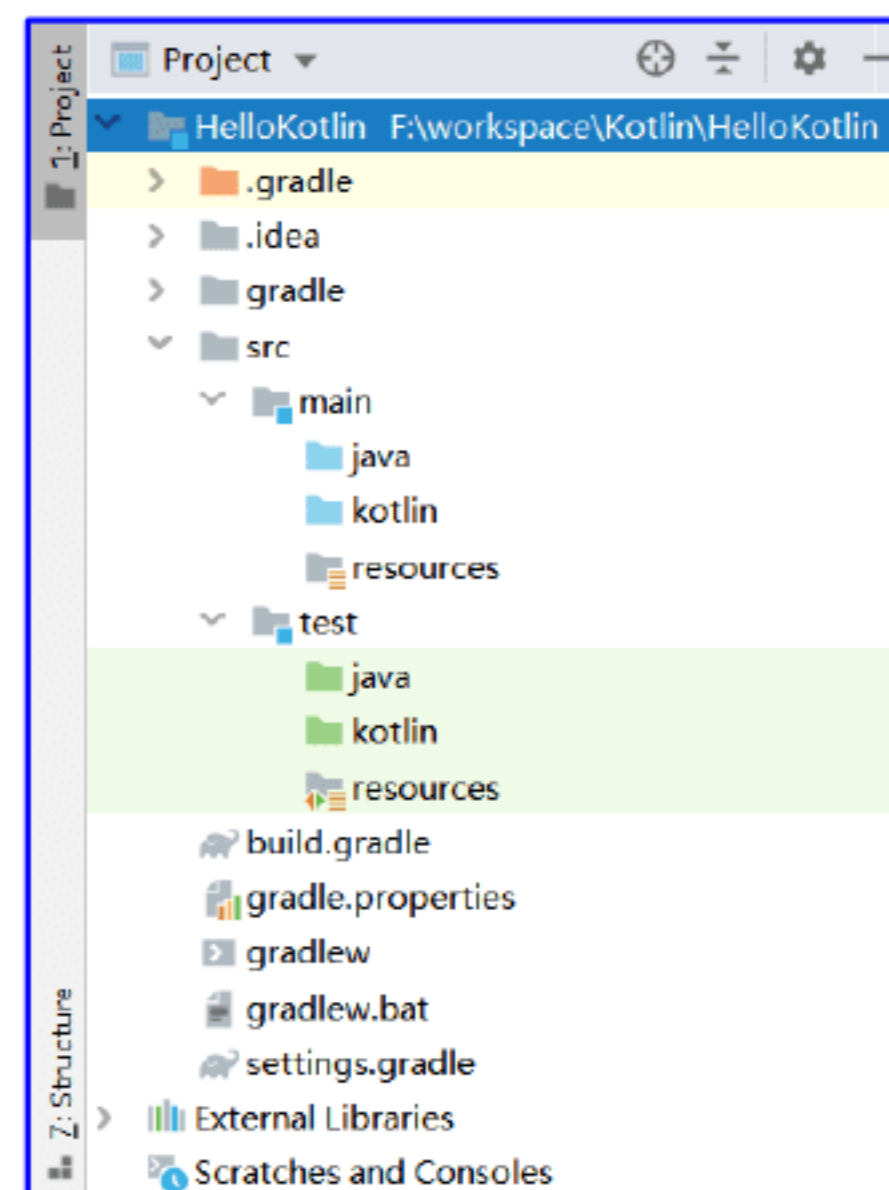


图 1-10

至于根目录下的其他文件，都是与 Gradle 相关的配置文件或脚本工具，也是自动生成的，我们不需要关心。

虽然工程下有这么多文件，但是这个工程是空工程，因为没有实质的程序代码，下面就来添加代码完成第一个程序。

1.1.5 添加代码

与 Java 相似，先创建一个包，在包下再创建文件，与 Java 不同，main 函数不用写在类中，直接作为全局函数即可。

首先创建一个包，见图 1-11。

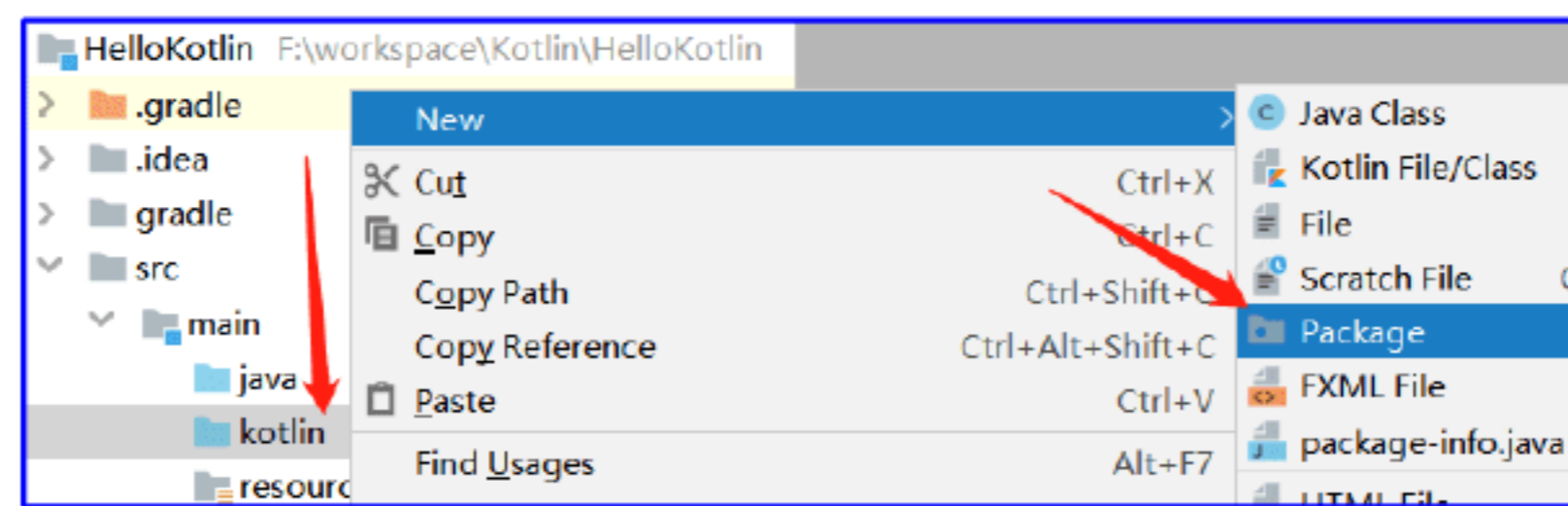


图 1-11

用鼠标右击“kotlin”目录，在弹出的快捷菜单中选择“New”→“Package”，出现图 1-12 所示的界面。在这里填上包名，单击“OK”按钮，会在 main/java 下创建 com.niuedu 包，在包上右击，弹出如图 1-13 所示的快捷菜单。

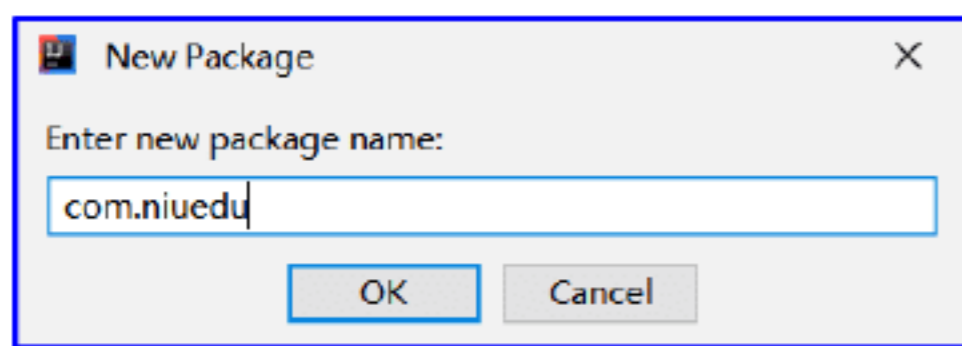


图 1-12

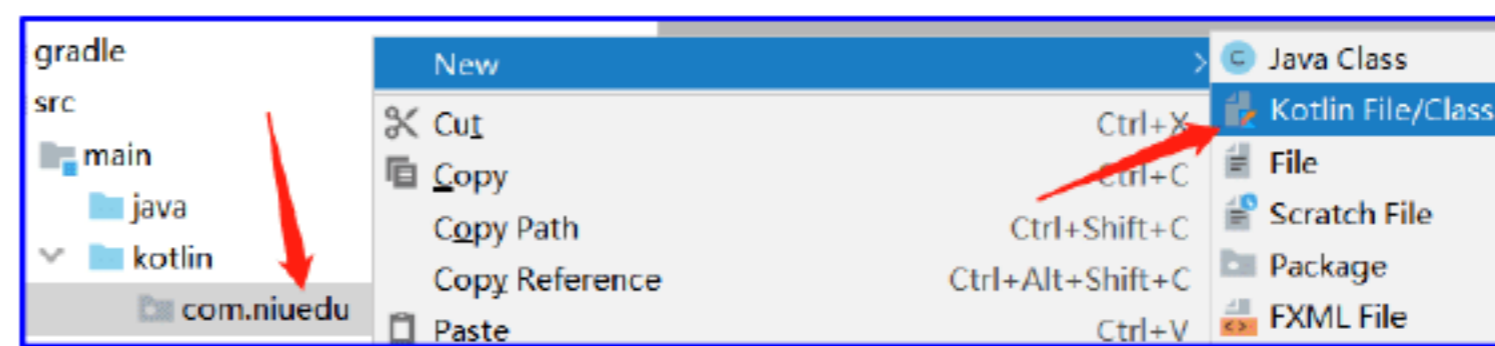


图 1-13

选择“Kotlin File/Class”之后，出现图 1-14 所示的界面。接下来选择所创建文件的类型，填入文件名“HelloApp”，选择“File”并双击，会在 main/kotlin/com.niuedu 下添加文件 HelloApp.kt。编辑此文件，最终内容如下：

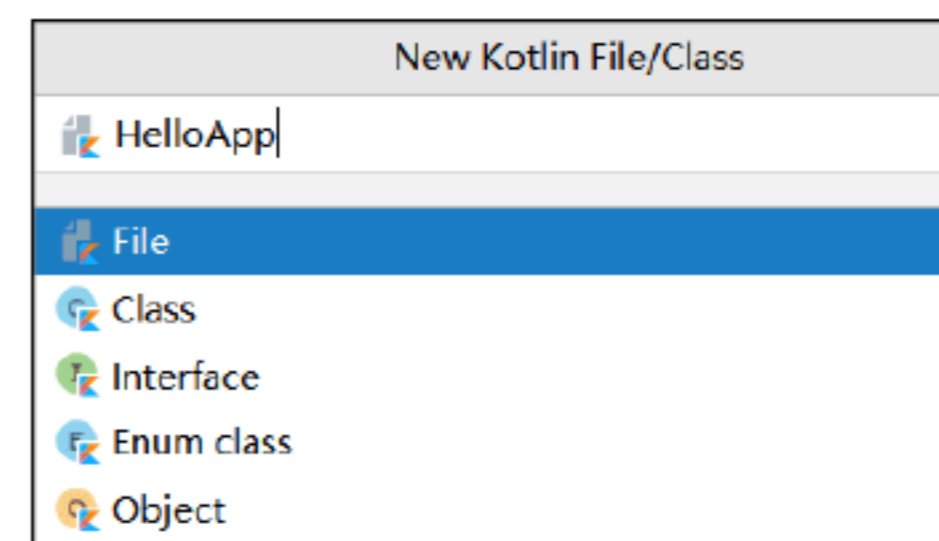


图 1-14

```
package com.niuedu

fun main() {
    println("Hello world!")
}
```

Main()函数很简单，其内容是打印一条文本。代码有了，如何运行呢？请看下节讲解。

1.1.6 运行程序

与 Java 工程一样，需要先配置运行方式。单击图 1-15 所示的位置，进入配置页面。

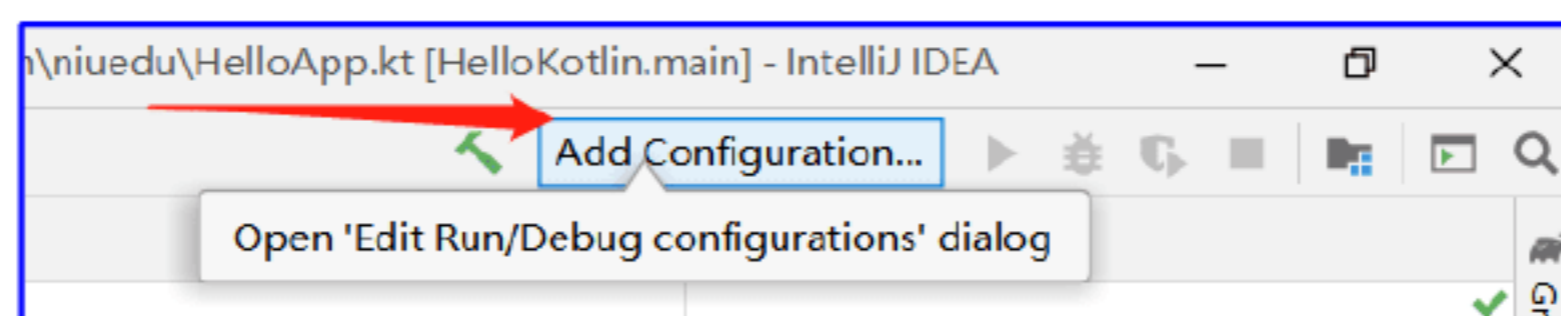


图 1-15

图 1-16 是运行方式的配置页面。

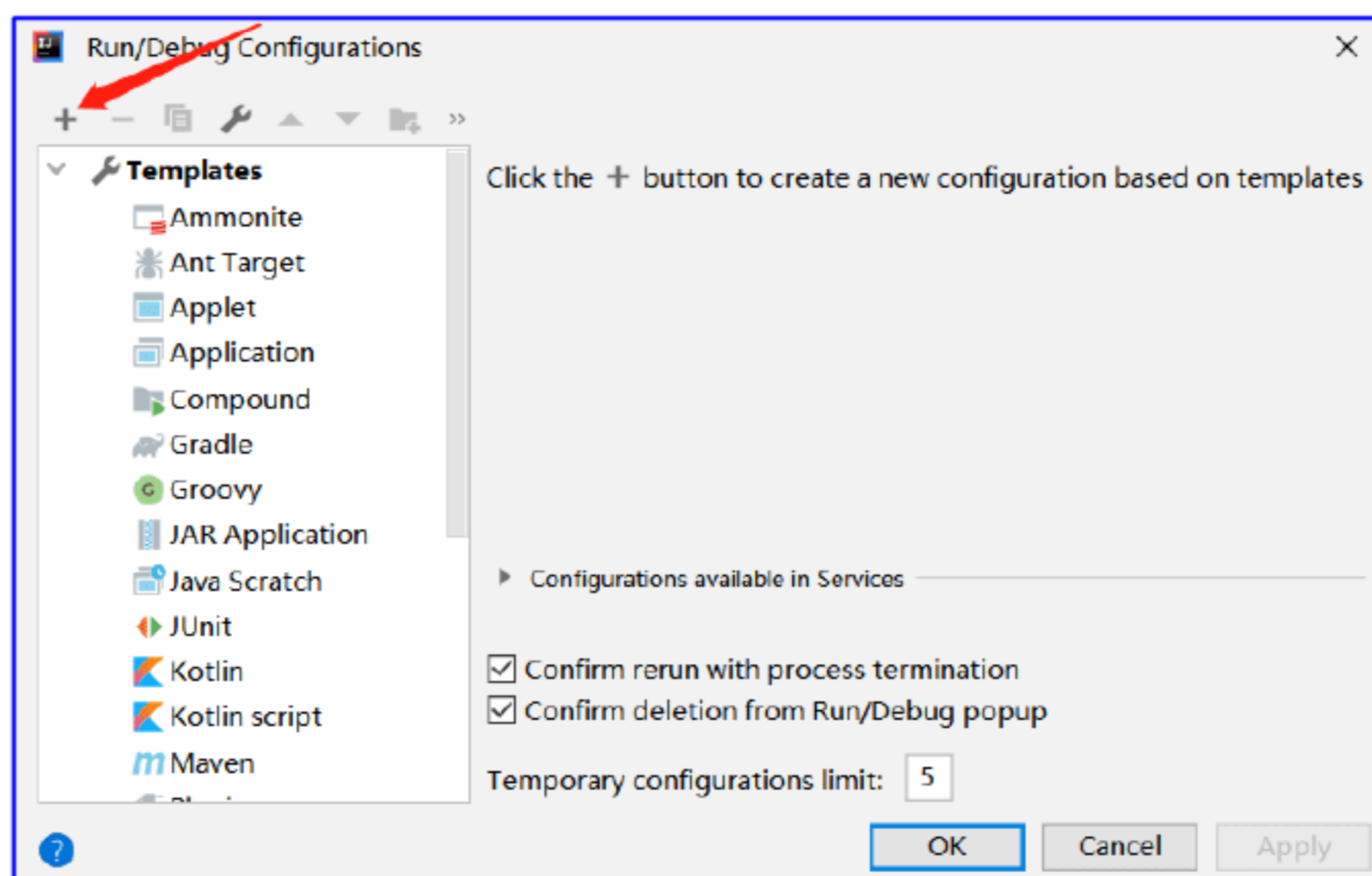


图 1-16

单击左上角的“+”图标，添加一条运行方式。选择正确的方式，这里应选“Kotlin”，如图 1-17 所示。

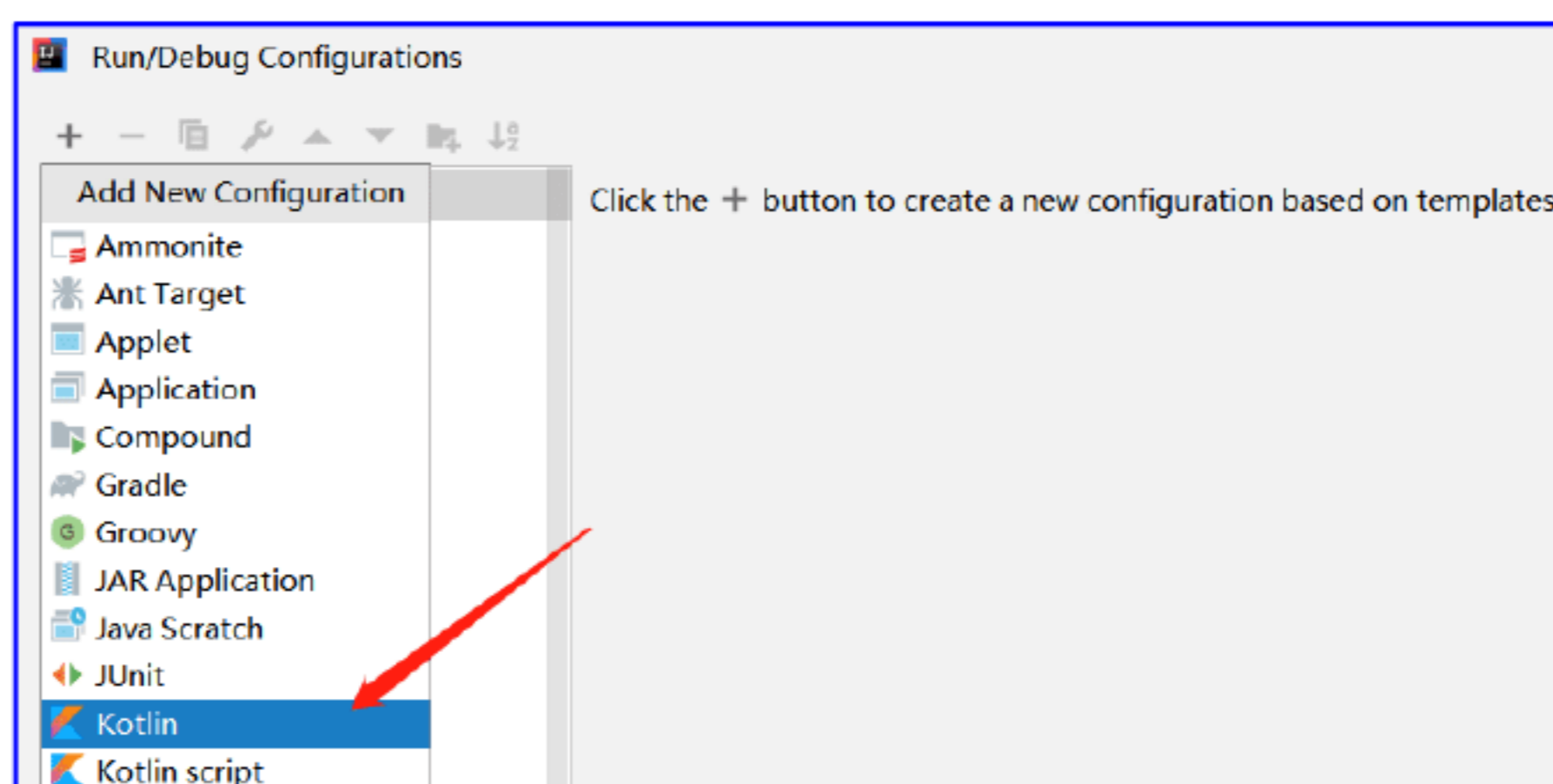


图 1-17

选择后则会出现如图 1-18 所示的界面。

将运行方式取名为“app”。在“Main class”字段填入的类名对应着 HelloApp.kt 文件，虽然 main 函数是全局函数，但是因为 Kotlin 代码最终要转成 class，所以它必然会有一个主类，从这里的类名可以看出 Kotlin 文件是如何与 Java 类对应的。

最后要注意的是，“Use classpath of module”字段需选择“HelloKotlin.main”，否则找不到 HelloAppKt 类。完成后单击“OK”按钮，回到主页面，此时可以看到运行方式旁边的图标变成绿色了，如图 1-19 所示。