

Huawei DevEco Studio 使用指南

V1.0

鸿蒙学堂 hmxt.org 整理

2020 年 9 月 10 日

目 录

1	工具简介	1
2	快速开始	2
2.1	下载与安装软件.....	2
2.1.1	运行环境要求.....	2
2.1.2	下载和安装 DevEco Studio	2
2.1.3	下载和安装 Node.js.....	3
2.2	配置开发环境	4
2.2.1	npm 设置.....	4
2.2.2	设置 Gradle 代理	5
2.2.3	设置 DevEco Studio 代理.....	6
2.2.4	下载 HarmonyOS SDK	8
2.3	运行 Hello World	11
3	工程管理	14
3.1	HarmonyOS 工程介绍.....	14
3.1.1	HarmonyOS APP 工程结构	14
3.1.2	工程目录结构.....	15
3.2	支持的设备模板和编程语言	17
3.3	创建一个新的工程	18
3.3.1	创建和配置新工程	18
3.3.2	导入现有工程.....	19
3.4	在工程中添加 Module	20
3.4.1	新增 Module.....	20
3.4.2	删除 Module.....	23
4	代码编辑	24
4.1	编辑器使用技巧.....	24
4.1.1	代码高亮.....	25
4.1.2	代码智能补齐.....	25
4.1.3	代码错误检查.....	26
4.1.4	代码自动跳转.....	26
4.1.5	代码格式化	27
4.1.6	代码折叠.....	28
4.1.7	代码快速注释.....	29
4.1.8	代码结构树	29
4.1.9	代码查找.....	30

4.1.10	查看 Java 接口文档.....	30
4.2	在模块中添加 Ability	32
4.2.1	创建 Particle Ability	32
4.2.2	创建 Feature Ability.....	34
4.3	添加 JS Component 和 JS Page	35
4.3.1	添加 JS Component	35
4.3.2	添加 JS Page.....	36
4.4	定义 HarmonyOS IDL 接口	37
4.4.1	HarmonyOS IDL 简介	37
4.4.2	创建 .idl 文件	37
4.4.3	实现 HarmonyOS IDL 接口	42
4.5	使用预览器查看应用效果	42
4.6	将 SVG 文件转换为 XML 文件.....	43
4.7	代码安全检查	45
5	编译构建	47
5.1	编译构建概述	47
5.2	编译构建前配置.....	47
5.2.1	工程级 build.gradle.....	48
5.2.2	模块级 build.gradle.....	49
5.2.3	config.json 清单文件	50
5.3	准备签名文件	51
5.3.1	生成密钥和证书请求文件.....	51
5.3.2	申请证书和 Profile.....	55
5.4	编译构建生成 HAP.....	55
5.4.1	前提条件.....	56
5.4.2	构建类型为 Debug 的 HAP（带调试签名信息）	56
5.4.3	构建类型为 Debug 的 HAP（不带签名）	58
5.4.4	构建类型为 Release 的 HAP（带调试签名信息）	58
5.4.5	构建类型为 Release 的 HAP（不带签名）	60
6	应用运行	61
6.1	使用模拟器运行应用.....	61
6.2	使用真机设备运行应用.....	64

6.2.1	在 TV 中运行应用	64
6.2.2	在 Wearable 中运行应用	65
6.2.3	在 Lite Wearable 中运行应用	66
7	应用调试	68
7.1	基本调试操作	68
7.1.1	选择调试代码类型	68
7.1.2	启动调试	69
7.1.3	断点管理	70
7.2	各语言调试功能	71
7.2.1	JS 调试功能	71
7.2.2	Java 调试功能	74
7.2.3	C/C++ 调试功能	74
8	应用发布	75
8.1	编译构建生成 APP	75
8.1.1	前提条件	75
8.1.2	操作步骤	75
8.2	上架华为应用市场	76

声明：所有内容均来自华为官方网站，如有错误，欢迎指正。

1 工具简介

HUAWEI DevEco Studio（以下简称 DevEco Studio）是基于 IntelliJ IDEA Community 开源版本打造，面向华为终端全场景多设备的一站式集成开发环境（IDE），为开发者提供工程模板创建、开发、编译、调试、发布等 E2E 的 HarmonyOS 应用开发服务。

通过使用 DevEco Studio，开发者可以更高效的开发具备 HarmonyOS 分布式能力的应用，进而提升创新效率。

作为一款开发工具，除了具有基本的代码开发、编译构建及调测等功能外，DevEco

Studio 还具有如下特点：

- **多设备统一开发环境**：支持多种 HarmonyOS 设备的应用开发，包括智慧屏、智能穿戴，轻量级智能穿戴设备。
- **支持多语言的代码开发和调试**：包括 Java、XML（Extensible Markup Language）、C/C++、JS（JavaScript）、CSS（Cascading Style Sheets）和 HML（HarmonyOS Markup Language）。
- **支持 FA（Feature Ability）和 PA（Particle Ability）快速开发**：通过工程向导快速创建 FA/PA 工程模板，一键式打包成 HAP（HarmonyOS Ability Package）。
- **支持多设备模拟器**：提供多设备的模拟器资源，包括智慧屏、智能穿戴等设备的模拟器，方便开发者高效调试。

2 快速开始

2.1 下载与安装软件

2.1.1 运行环境要求

当前 DevEco Studio 只支持 Windows 版本，为保证 DevEco Studio 正常运行，建议您
的电脑配置满足如下要求：

- 操作系统：Windows10 64 位
- 内存：8GB 及以上
- 硬盘：100GB 及以上
- 分辨率：1280*800 像素及以上

2.1.2 下载和安装 DevEco Studio

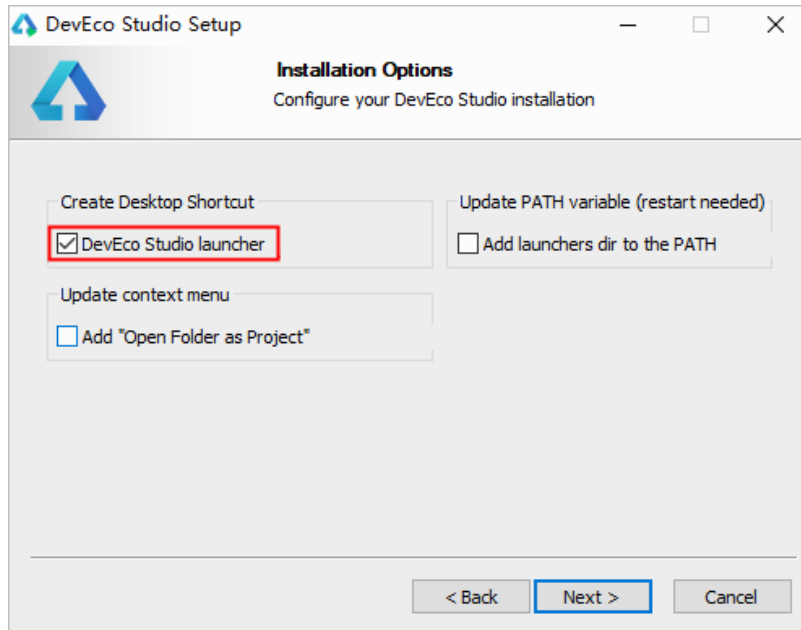
DevEco Studio 的编译构建依赖 JDK，DevEco Studio 预置了 Open JDK，版本为 1.8，
安装过程中会自动安装 JDK。

1. 登录 HarmonsOS 应用开发门户，点击右上角**注册**按钮，注册开发者帐号，注册指导参考注册华为帐号。如果已有华为开发者帐号，请直接点击**登录**按钮。

说明

使用 DevEco Studio 远程模拟器需要华为帐号进行实名认证，建议在注册华为帐号后，立即提交实名认证审核，认证方式包括“个人实名认证”和“企业实名认证”，详情请参考实名认证。

2. 进入 HUAWEI DevEco Studio 产品页，下载 DevEco Studio 安装包。
3. 双击下载的“deveco-studio-xxxx.exe”，进入 DevEco Studio 安装向导，在如下安装选项界面勾选 **DevEco Studio launcher** 后，点击 **Next**，直至安装完成。



2.1.3 下载和安装 Node.js

Node.js 软件仅在使用到 JS 语言开发 HarmonyOS 应用时才需要安装。使用其它语言开发，不用安装 Node.js，请跳过此章节。

1. 登录 [Node.js 官方网站](https://nodejs.org/)，下载 Node.js 软件包。请选择 LTS 版本，Windows 64 位对应的软件包。

Platform	Architecture	Download Link
Windows Installer (.msi)	32-bit	node-v12.18.3-x64.msi
	64-bit	node-v12.18.3-x64.msi
macOS Installer (.pkg)	64-bit	node-v12.18.3.pkg
Source Code	64-bit	node-v12.18.3.tar.gz
Linux Binaries (x64)	64-bit	node-v12.18.3-x64.tar.gz
Linux Binaries (ARM)	ARMv7	node-v12.18.3-armv7.tar.gz
	ARMv8	node-v12.18.3-armv8.tar.gz

2. 点击下载后的软件包进行安装，全部按照默认设置点击 **Next**，直至 **Finish**。安装过程中，Node.js 会自动在系统的 path 环境变量中配置 node.exe 的目录路径。

2.2 配置开发环境

DevEco Studio 开发环境需要依赖于网络环境，需要连接上网络才能确保工具的正常使
用，可以根据如下两种情况来配置开发环境：

- 如果可以直接访问 Internet，只需进行 H2 设置 npm 仓库和下载 HarmonyOS SDK 操作。
- 如果网络不能直接访问 Internet，需要通过代理服务器才可以访问，请根据本章节内容逐条设置开发环境。

2.2.1 npm 设置

2.2.1.1 设置 npm 代理

只有在同时满足以下两个条件时，需要配置 npm 代理，否则，请跳过本章节。

- 需要使用 JS 语言开发 HarmonyOS 应用。
- 网络不能直接访问 Internet，而是需要通过代理服务器才可以访问。这种情况下，配置 npm 代理，便于从 npm 服务器下载 JS 依赖。

打开命令行工具，按照如下方式进行 npm 代理设置和验证。

1. 执行如下命令设置 npm 代理。
- 如果使用的代理服务器需要认证，请按照如下方式进行设置（请将 **user**、**password**、**proxyserver** 和 **port** 按照实际代理服务器进行修改）。

1. `npm config set proxy http://user:password@proxyserver:port`
2. `npm config set https-proxy http://user:password@proxyserver:port`

- 如果使用的代理服务器不需要认证（不需要帐号和密码），请按照如下方式进行设置。

1. `npm config set proxy http:proxyserver:port`
2. `npm config set https-proxy http:proxyserver:port`

2. 代理设置完成后，执行如下命令进行验证。

0. `npm info express`

执行结果如下图所示，则说明代理设置成功。

```
C:\Users\>npm info express

express@4.17.1 | MIT | deps: 30 | versions: 264
Fast, unopinionated, minimalist web framework
http://expressjs.com/

keywords: express, framework, sinatra, web, rest, restful, router, app, api

dist
.tarball: https://registry.npmjs.org/express/-/express-4.17.1.tgz
.shasum: 4491fc38605cf51f8629d39c2b5d026f98a4c134
.integrity: sha512-mHJ9079RqluphRrcw2X/GTh3k9tVv8Yc0yY4Kkh4WDMUYKRZUq0hlo0w2rrrxBqM7VoeUVqgb27xlEMXTnYt4g==
.unpackedSize: 208.1 kB

dependencies:
accepts: ~1.3.7          cookie: 0.4.0            finalhandler: ~1.1.2    path-to-regexp: 0.1.7
array-flatten: 1.1.1     debug: 2.6.9            fresh: 0.5.2            proxy-addr: ~2.0.5
body-parser: 1.19.0      depd: ~1.1.2            merge-descriptors: 1.0.1 qs: 6.7.0
content-disposition: 0.5.3 encodeurl: ~1.0.2        methods: ~1.1.2         range-parser: ~1.2.1
content-type: ~1.0.4     escape-html: ~1.0.3     on-finished: ~2.3.0     safe-buffer: 5.1.2
cookie-signature: 1.0.6  etag: ~1.8.1            parseurl: ~1.3.3        send: 0.17.1
(...and 6 more.)

maintainers:
- dougwilson <doug@somethingdoug.com>
- jasnell <jasnell@gmail.com>
- mikeal <mikeal.rogers@gmail.com>

dist-tags:
latest: 4.17.1      next: 5.0.0-alpha.8

published a year ago by dougwilson <doug@somethingdoug.com>
```

2.2.1.2 设置 npm 仓库

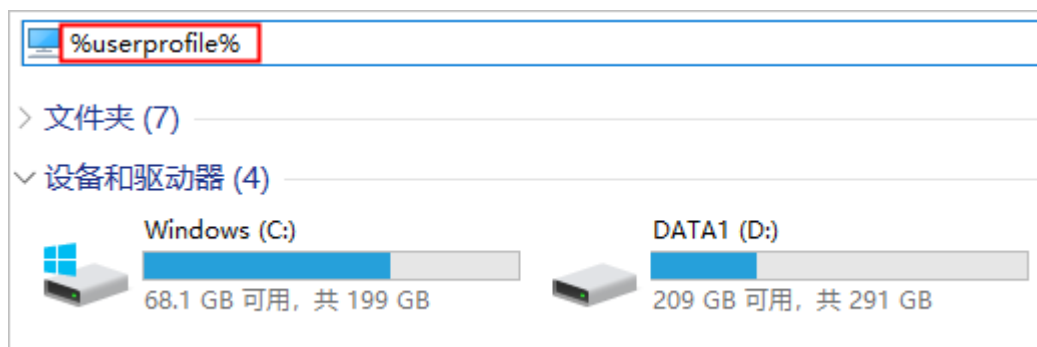
为了提升下载 JS SDK 时，使用 npm 安装 JS 依赖的速度，建议在命令行工具中执行如下命令，重新设置 npm 仓库地址。

1. `npm config set registry https://mirrors.huaweicloud.com/repository/npm/`

2.2.2 设置 Gradle 代理

如果网络不能直接访问 Internet，而是需要通过代理服务器才可以访问，这种情况下，需要设置 Gradle 代理，来访问和下载 Gradle 所需的依赖。否则，请跳过本章节。

1. 打开“此电脑”，在文件夹地址栏中输入 `%userprofile%`，进入个人数据界面。



2. 创建一个文件夹，命令为 **gradle**。如果已有 **gradle** 文件夹，请跳过此操作。
3. 进入 **gradle** 文件夹，新建一个文本文档，命名为 **gradle**，并修改后缀为 **.properties**。
4. 打开 **gradle.properties** 文件中，添加如下脚本，然后保存。
其中代理服务器、端口、用户名、密码和不使用代理的域名，请根据实际代理情况进行修改。其中不使用代理的 “nonProxyHosts” 的配置间隔符是 “|”。

```

1. systemProp.http.proxyHost=proxy.server.com
2. systemProp.http.proxyPort=8080
3. systemProp.http.nonProxyHosts=*.company.com|10.*|100.*
4. systemProp.http.proxyUser=userId
5. systemProp.http.proxyPassword=password
6. systemProp.https.proxyHost=proxy.server.com
7. systemProp.https.proxyPort=8080
8. systemProp.https.nonProxyHosts=*.company.com|10.*|100.*
9. systemProp.https.proxyUser=userId
10. systemProp.https.proxyPassword=password

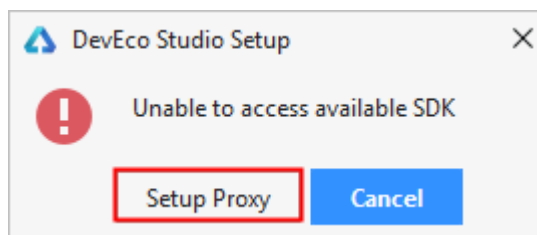
```

2.2.3 设置 DevEco Studio 代理

如果网络不能直接访问 Internet，而需要通过代理服务器才可以访问，这种情况下，需要

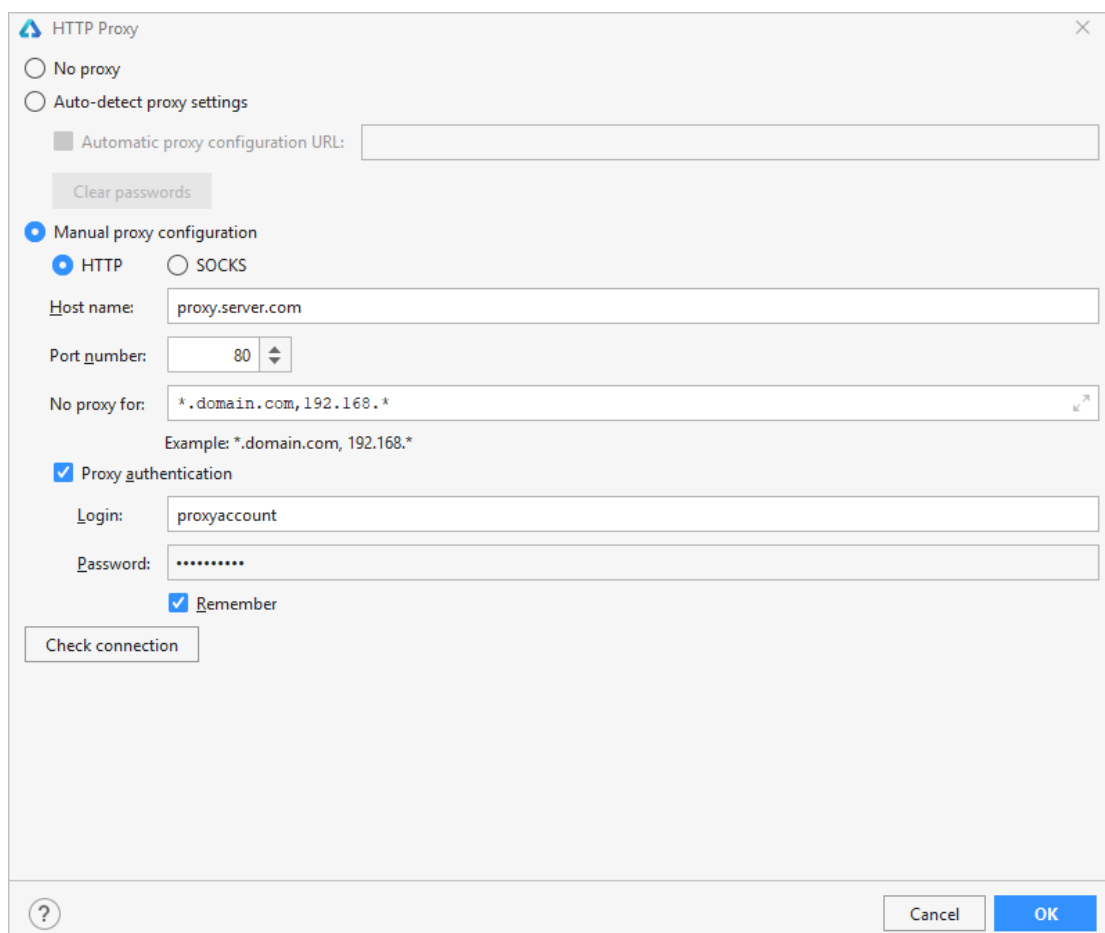
设置 DevEco Studio 代理，来访问和下载外部资源。否则，请跳过本章节。

1. 运行已安装的 DevEco Studio，首次使用，请选择 **Do not import settings**，点击 **OK**。
2. 根据 DevEco Studio 欢迎界面的提示，点击 **Setup Proxy**，或者在欢迎页点击 **Configure > Settings > Appearance&Behavior > System Settings > HTTP Proxy** 进入 HTTP Proxy 设置界面。



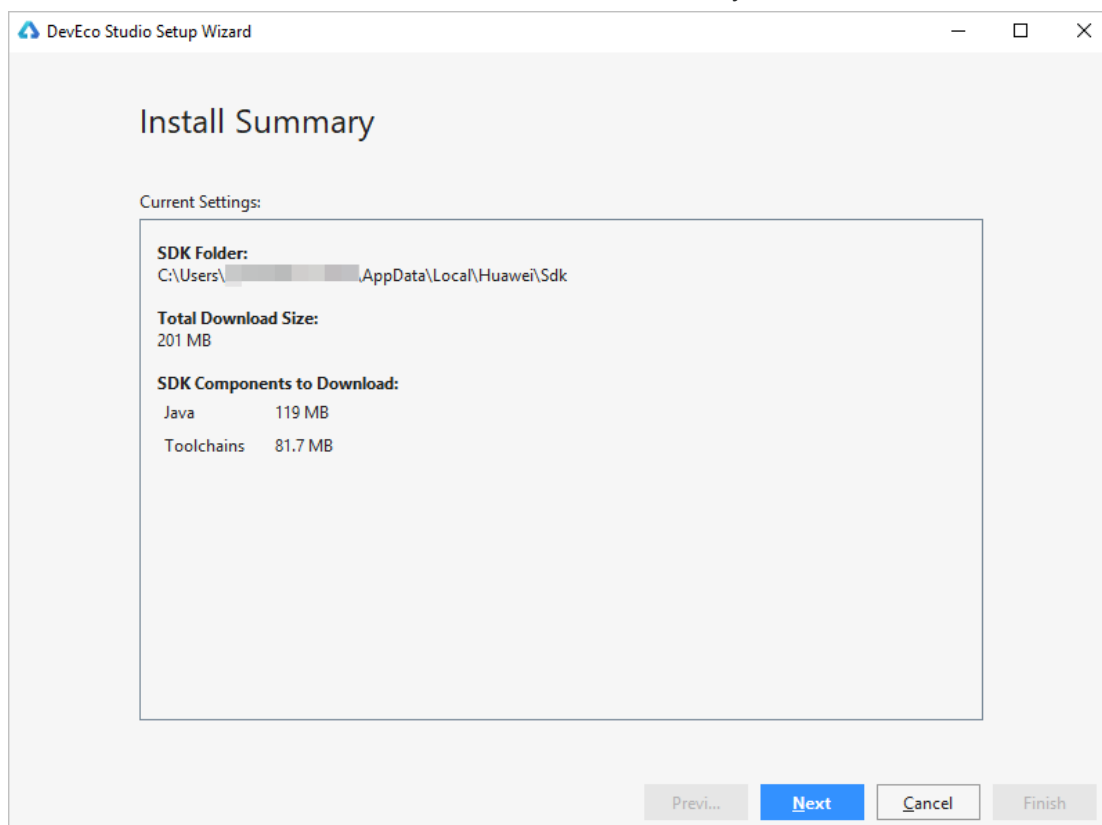
3. 设置 DevEco Studio 的 HTTP Proxy 信息。

- **HTTP** 配置项，设置代理服务器信息。
 - **Host name**：代理服务器主机名或 IP 地址。
 - **Port number**：代理服务器对应的端口号。
 - **No proxy for**：不需要通过代理服务器访问的 URL 或者 IP 地址（地址之间用英文逗号分隔）。
- **Proxy authentication** 配置项，如果代理服务器需要通过认证鉴权才能访问，则需要设置。否则，请跳过该配置项。
 - **Login**：访问代理服务器的用户名。
 - **Password**：访问代理服务器的密码。
 - **Remember**：勾选，记住密码。



4. 配置完成后，点击 **Check connection**，输入网络地址（如：https://developer.harmonyos.com），检查网络连通性。提示“Connection successful”表示代理设置成功。
5. 点击 **OK** 按钮完成 DevEco Studio 代理配置。

6. DevEco Studio 代理设置完成后，会提示安装 HarmonyOS SDK，可以点击 **Next** 下载到默认目录中；如果想更改 SDK 的存储目录，请点击 **Cancel**，并根据下载 HarmonyOS SDK 进行操作。



2.2.4 下载 HarmonyOS SDK

Devco Studio 提供 SDK Manager 统一管理 SDK 及工具链，下载各种编程语言的 SDK 包时，SDK Manager 会自动下载该 SDK 包依赖的工具链。

SDK Manager 提供多种编程语言的 SDK 包，各 SDK 包的说明请参考：

- **Native:** C/C++ 语言 SDK 包，默认不自动下载，需手动勾选下载。对应的接口文档请参考《Native API 参考》。
- **JS:** JS 语言 SDK 包，默认不自动下载，需手动勾选下载。对应的接口文档请参考《JS API 参考》。
- **Java:** Java 语言 SDK 包，首次下载 SDK 时默认下载。对应的接口文档请参考《Java API 参考》。

同时还提供 SDK 对应的工具链（SDK Tools）：

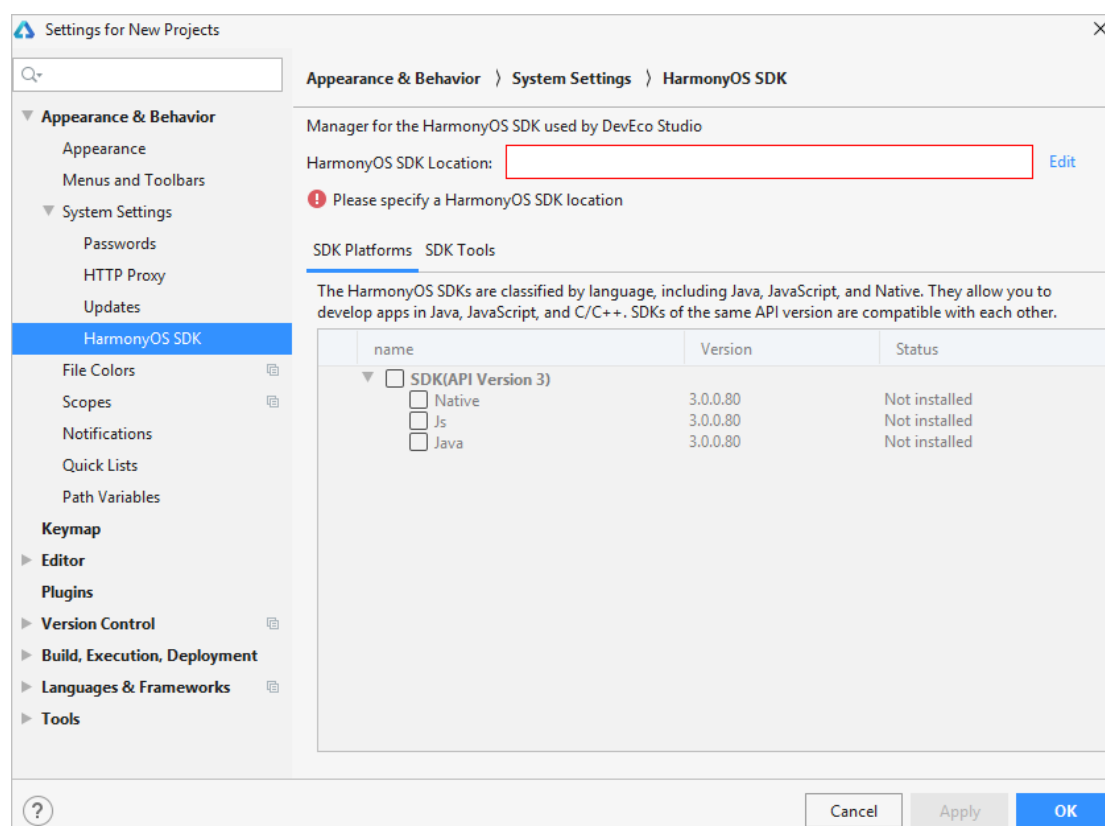
- **Toolchains:** SDK 工具链，HarmonyOS 应用开发必备工具集，包括编译、打包、签名、数据库管理等工具的集合，首次下载 SDK 时默认下载。

- **Previewer**: Lite Wearable 预览器，在开发过程中可以动态预览 Lite Wearable 应用的界面呈现效果，默认不自动下载，需手动勾选下载。

首次下载 HarmonyOS SDK 时，只会默认下载 **Java SDK 和 Toolchains**。因此，如果还

需要使用 JS 或 C/C++ 语言开发应用时，需手动下载对应的 SDK 包。

1. 在菜单栏点击 **Configure > Settings** 或者默认快捷键 **Ctrl+Alt+S**，打开 Settings 配置界面。
2. 进入 **Appearance & Behavior > System Settings > HarmonyOS SDK** 菜单界面，点击 **Edit** 按钮，设置 HarmonyOS SDK 存储路径。

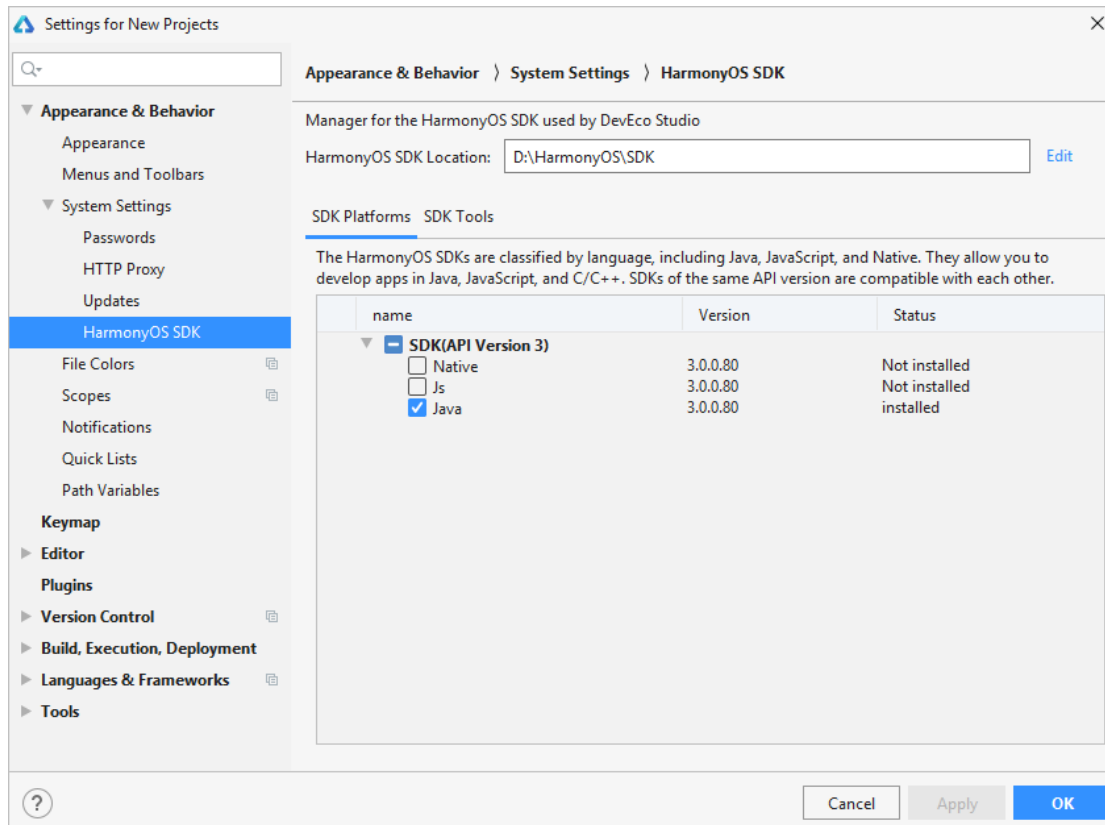


3. 选择 HarmonyOS SDK 存储路径，然后点击 **Next**。在弹出的 **License Agreement** 窗口，点击 **Accept** 开始下载 SDK。

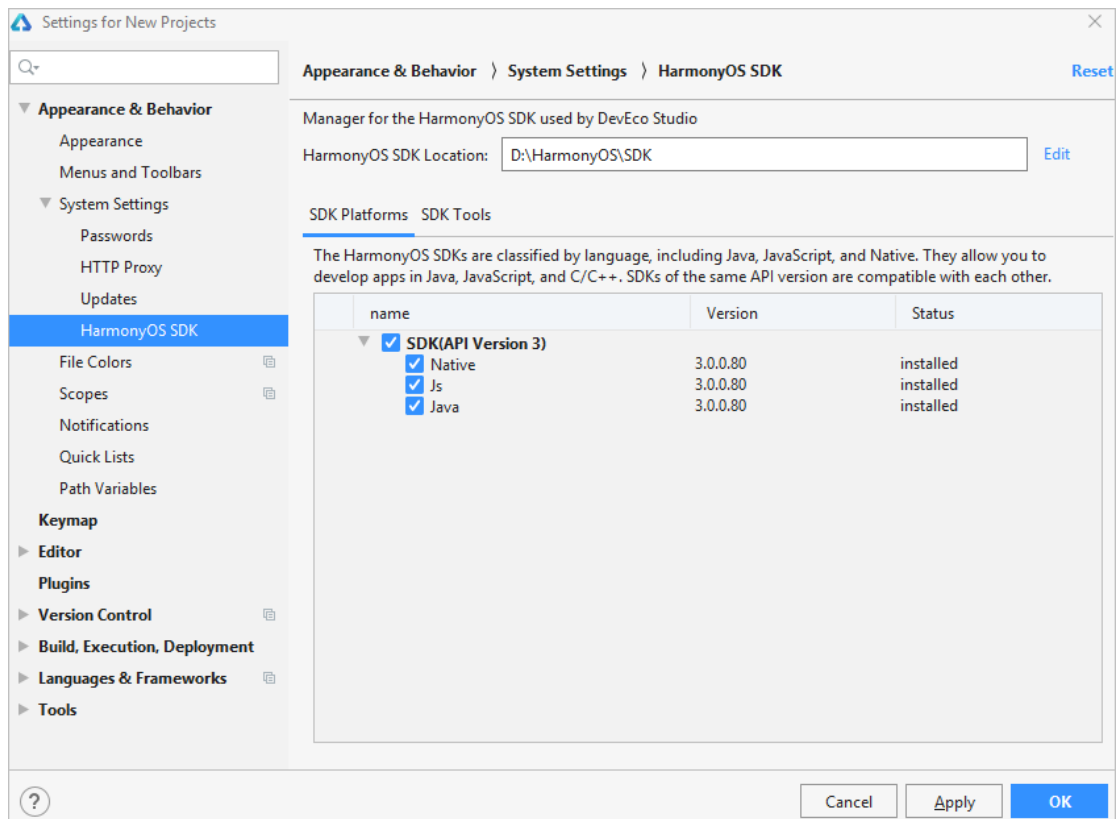
说明

如果本地已有 SDK 包，请选择本地已有 SDK 包的存储路径，DevEco Studio 会增量更新 SDK 及工具链。

4. 等待 HarmonyOS SDK 及工具下载完成，点击 **Finish**，可以看到默认的 SDK Platforms>**Java SDK** 及 SDK Tools>**Toolchains** 已完成下载。



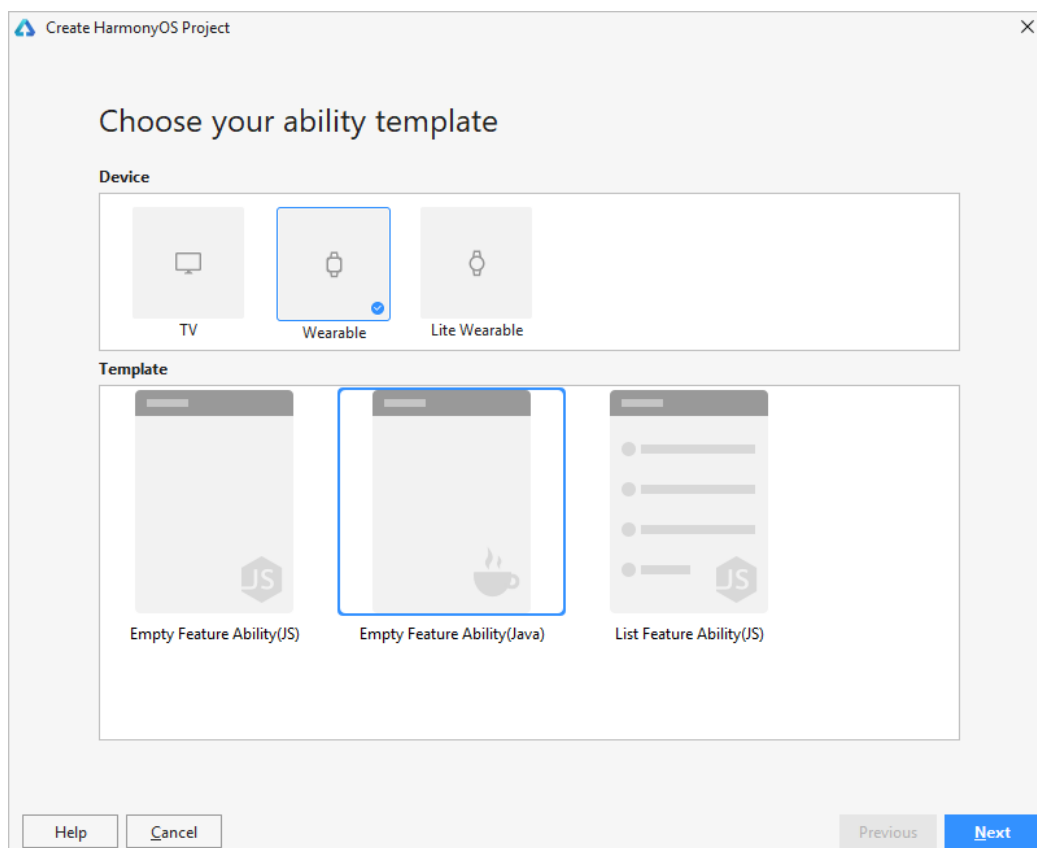
5. 如果工程还会用到 **JS** 或者 **C/C++** 语言，请在 SDK Platform 中，勾选对应的 SDK 包，点击 **Apply**，SDK Manager 会自动将 SDK 包和工具链，下载到 3 中设置的 SDK 存储路径中。



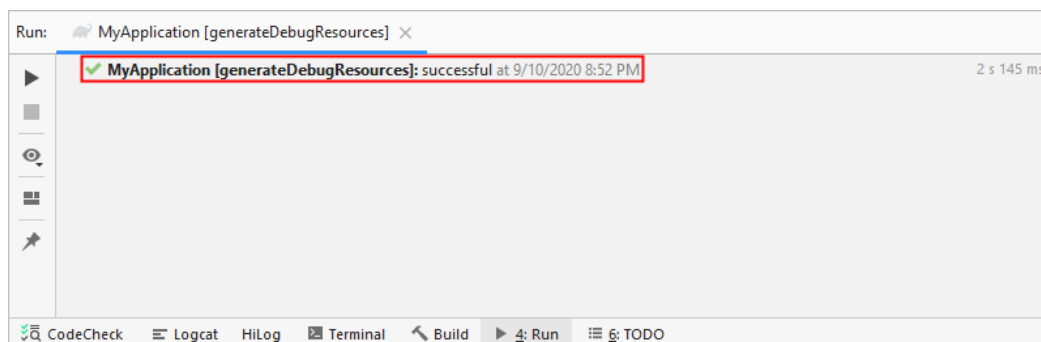
2.3 运行 Hello World

DevEco Studio 开发环境配置完成后，可以通过运行 HelloWorld 工程来验证环境设置是否正确。以 Wearable 工程为例，在 Wearable 远程模拟器中运行该工程。

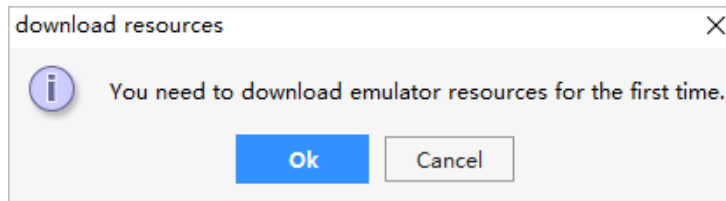
1. 打开 DevEco Studio，在欢迎页点击 **Create HarmonyOS Project**，创建一个新工程。
2. 选择设备类型和模板，以 Wearable 为例，选择 Empty Feature Ability(Java)，点击 **Next**。



3. 填写项目相关信息，保持默认值即可，点击 **Finish**。
4. 工程创建完成后，DevEco Studio 会自动进行工程的同步，同步成功如下图所示。



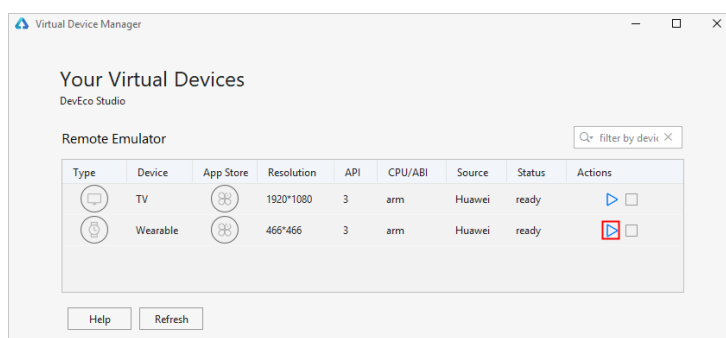
- 在 DevEco Studio 菜单栏，点击 **Tools > HVD Manager**。首次使用模拟器，需下载模拟器相关资源，请点击 **OK**，等待资源下载完成后，点击模拟器界面左下角的 **Refresh** 按钮。



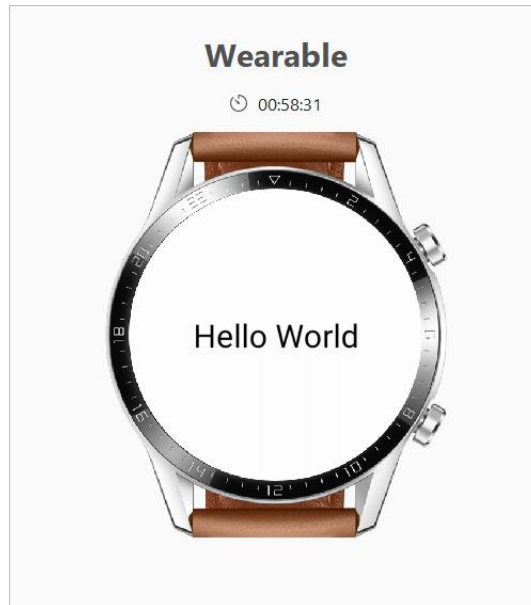
- 在浏览器中弹出华为帐号登录界面，请输入已实名认证的华为帐号的用户名和密码进行登录。
- 登录后，请点击界面的**允许**按钮进行授权。



- 在设备列表中，选择 **Wearable** 设备，并点击按钮，运行模拟器。



- 点击 DevEco Studio 工具栏中的按钮运行工程，或使用默认快捷键 **Shift+F10** 运行工程。
- 在弹出的 Select Deployment Target 界面选择 **Connected Devices**，点击 **OK** 按钮。
- DevEco Studio 会启动应用的编译构建，完成后应用即可运行在 **Remote Device** 上。



3 工程管理

3.1 HarmonyOS 工程介绍

3.1.1 HarmonyOS APP 工程结构

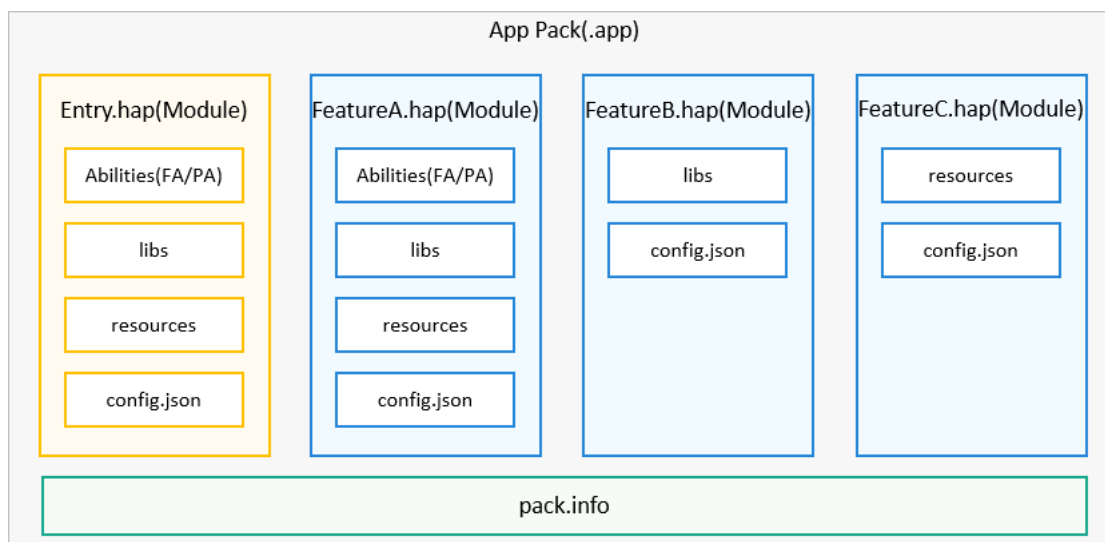
在进行 HarmonyOS 应用开发前，您应该掌握 HarmonyOS 应用的逻辑结构。

HarmonyOS 应用发布形态为 **APP Pack** (Application Package, 简称 APP)，它是由一个或多个 **HAP** (HarmonyOS Ability Package) 包以及描述 APP Pack 属性的 pack.info 文件组成。

一个 HAP 在工程目录中对应一个 Module，它是由代码、资源、第三方库及应用清单文件组成，可以分为 Entry 和 Feature 两种类型。

- **Entry**：应用的主模块。一个 APP 中，对于同一设备类型必须有且只有一个 entry 类型的 HAP，可独立安装运行。
- **Feature**：应用的动态特性模块。一个 APP 可以包含一个或多个 feature 类型的 HAP，也可以不含。

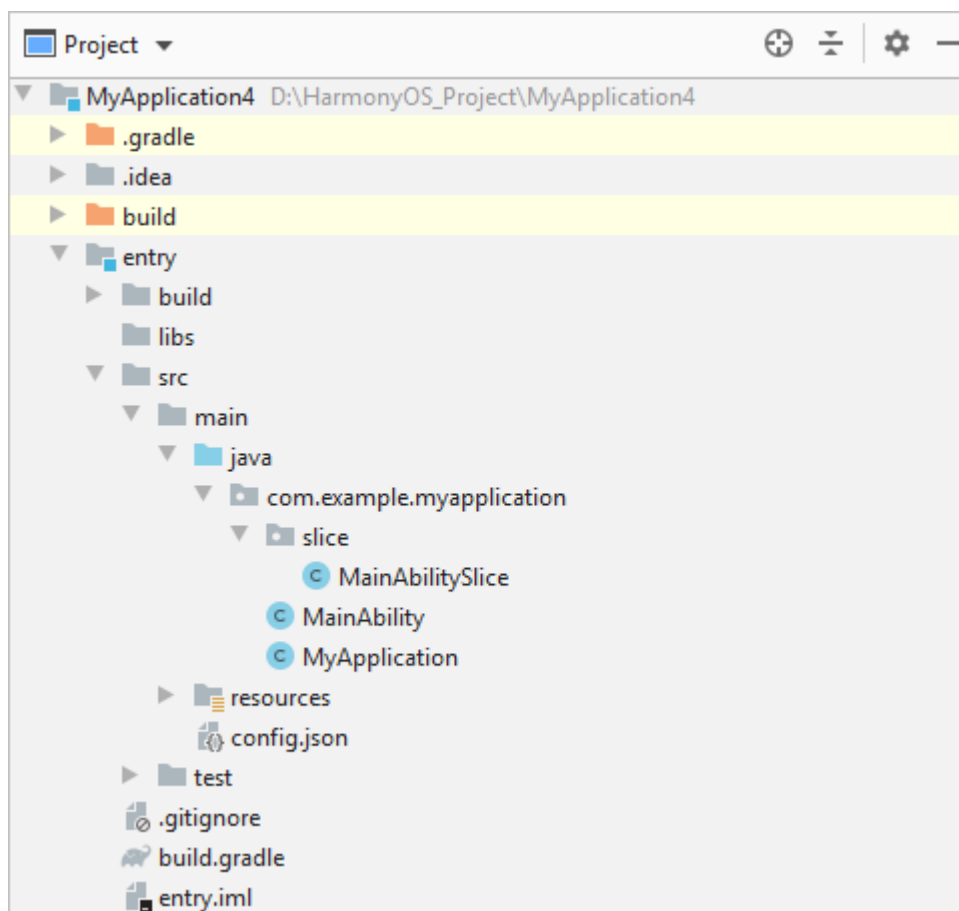
HAP 是 **Ability** 的部署包，HarmonyOS 应用代码围绕 Ability 组件展开，它是由一个或多个 Ability 组成。Ability 分为两种类型：FA (Feature Ability) 和 PA (Particle Ability)。FA/PA 是应用的基本组成单元，能够实现特定的业务功能。FA 有 UI 界面，而 PA 无 UI 界面。



3.1.2 工程目录结构

3.1.2.1 Java 工程目录结构

Java 工程目录结构如下图所示。

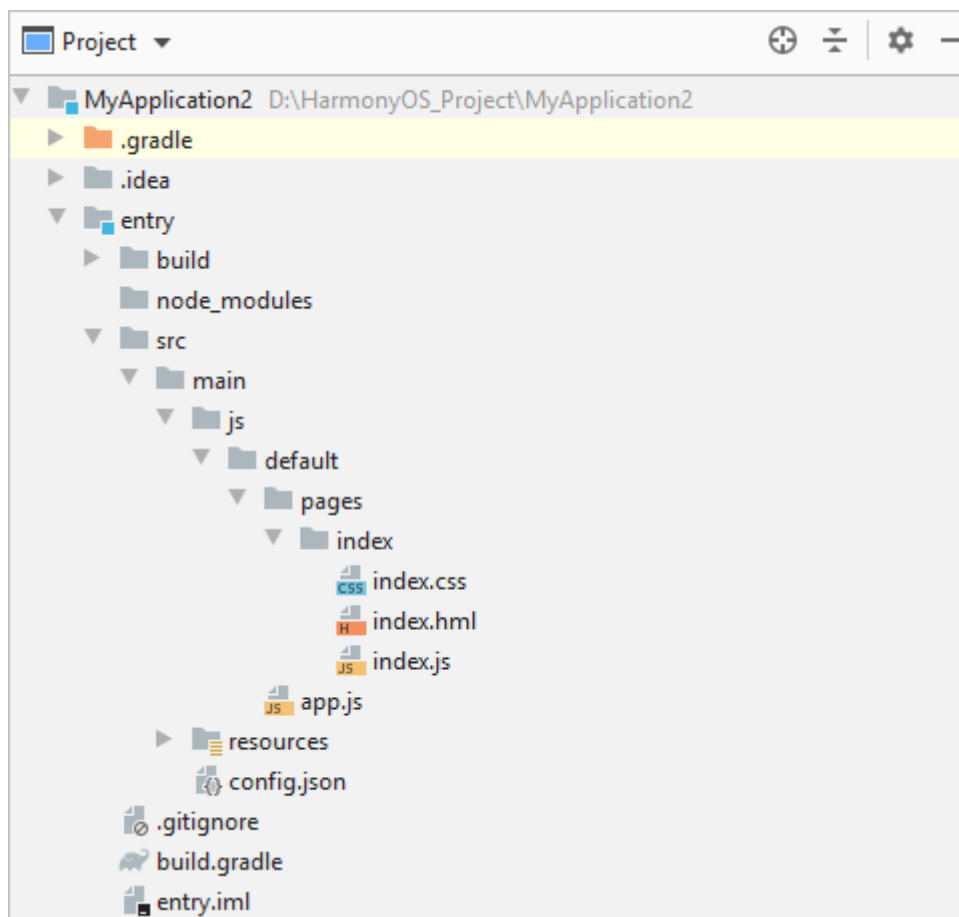


.gradle: Gradle 配置文件，由系统自动生成，一般情况下不需要进行修改。

- **entry**: 默认启动模块（主模块），开发者用于编写源码文件以及开发资源文件的目录。
- **entry>libs**: 用于存放 entry 模块的依赖文件。
- **entry>.gitignore**: 标识 git 版本管理需要忽略的文件。
- **entry>build.gradle**: entry 模块的编译配置文件。
- **entry>src>main>Java**: 用于存放 Java 源码。
- **entry>src>main>resources**: 用于存放资源文件。
- **entry>src>main>config.json**: HAP 清单文件，详细说明请参考 **config.json 清单文件介绍**。
- **entry>src>test**: 编写测试文件的目录。

3.1.2.2 JS 工程目录结构

JS 工程目录结构如下图所示。



- **pages 目录**: pages 文件夹下可以包含 1 个或多个页面，每个页面都需要创建一个文件夹（如图中的 index）。页面文件夹下主要包含 3 种文件类型：css、js 和 html 文件。
- **pages > index > index.html 文件**: html 文件定义了页面的布局结构，使用到的组件，以及这些组件的层级关系。

- **pages > index > index.css 文件：**css 文件定义了页面的样式与布局，包含样式选择器和各种样式属性等。
- **pages > index > index.js 文件：**js 文件描述了页面的行为逻辑，此文件里定义了页面里所用到的所有的逻辑关系，比如数据、事件等。
- **app.js 文件：**全局的 JavaScript 逻辑文件和应用的生命周期管理。

3.2 支持的设备模板和编程语言

DevEco Studio 支持包括智慧屏、智能穿戴和轻量级智能穿戴的 HarmonyOS 应用开发，可以根据工程向导轻松创建适应于各类设备的工程，并自动生成对应的代码和资源模板。

同时，DevEco Studio 还提供了多种编程语言供开发者进行 HarmonyOS 应用开发，包括 Java、JS 和 C/C++ 三种编程语言，并支持多种语言的混合开发场景。因此，在创建对应设备的工程时，工具会预置多种 Ability 的模板，并推荐您使用适合的开发语言。

支持的各设备类型工程模板和对应开发语言的对应关系，如下表所示。

Device	工程模板
TV	Empty Feature Ability (JS)
	Empty Feature Ability (Java)
	List Container Ability (Java)
	List Feature Ability (JS)
	Split Panel Ability (Java)
	Tab Feature Ability (JS)
Wearable	Empty Feature Ability (JS)
	Empty Feature Ability (Java)

Device	工程模板
	List Feature Ability (JS)
Lite Wearable	Empty Feature Ability
	List Feature Ability

表 1 各设备开发模板

3.3 创建一个新的工程

当开始开发一个 HarmonyOS 应用时，首先需要根据工程创建向导，创建一个新的工程，工具会自动生成对应的代码和资源模板。

如果创建的工程包含 JS 语言，请确保已经下载了 JS SDK 包，具体可参考下载

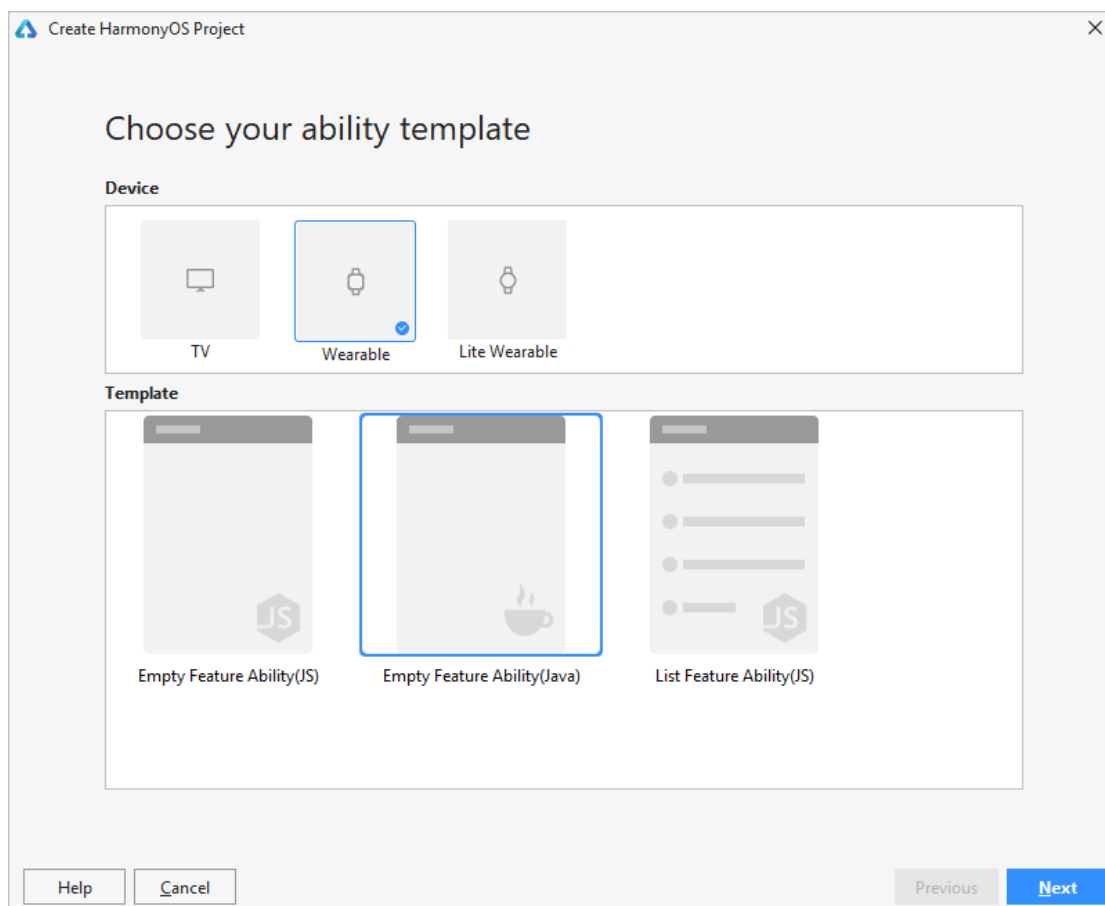
HarmonyOS SDK。

说明

在运行 DevEco Studio 工程时，建议每一个运行窗口有 2GB 以上的可用内存空间。

3.3.1 创建和配置新工程

- 通过如下两种方式，打开工程创建向导界面。
 - 如果当前未打开任何工程，可以在 DevEco Studio 的欢迎页，选择 **Create HarmonyOS Project** 开始创建一个新工程。
 - 如果已经打开了工程，可以在菜单栏选择 **File > New > New Project** 来创建一个新工程。
- 根据工程创建向导，选择需要进行开发的设备类型，然后选择对应的 Ability 模板。



3. 点击 **Next**，进入到工程配置阶段，需要根据向导配置工程的基本信息。
 - **Project name**：工程的名称，可以自定义。
 - **Package name**：软件包名称，默认情况下，应用 ID 也会使用该名称，应用发布时，应用 ID 需要唯一。
 - **Save location**：工程文件本地存储路径。
 - **Compatible SDK**：兼容的 SDK 版本。
4. 点击 **Finish**，工具会自动生成示例代码和相关资源，等待工程创建完成。

3.3.2 导入现有工程

导入现有工程分为以下两种情况：

- 导入 DevEco Studio 创建的 HarmonyOS 应用工程：
 - 如果当前未打开任何工程，可以在 DevEco Studio 的欢迎页，选择 **Open Project** 打开现有工程。
 - 如果已经打开了工程，可以在菜单栏选择 **File > Open** 来打开现有工程。

- 导入其它 IDE 创建的工程，比如导入 Visual Studio Code 创建的轻量级智能穿戴 HarmonyOS 应用工程：
- 如果当前未打开任何工程，可以在 DevEco Studio 的欢迎页，选择 **Import Project** 导入现有工程。
- 如果已经打开了工程，可以在菜单栏选择 **File > Import Project** 导入现有工程。

导入现有工程时，DevEco Studio 会提醒您可以选择在新的窗口打开工程，或者选择在当前窗口打开工程。

3.4 在工程中添加 Module

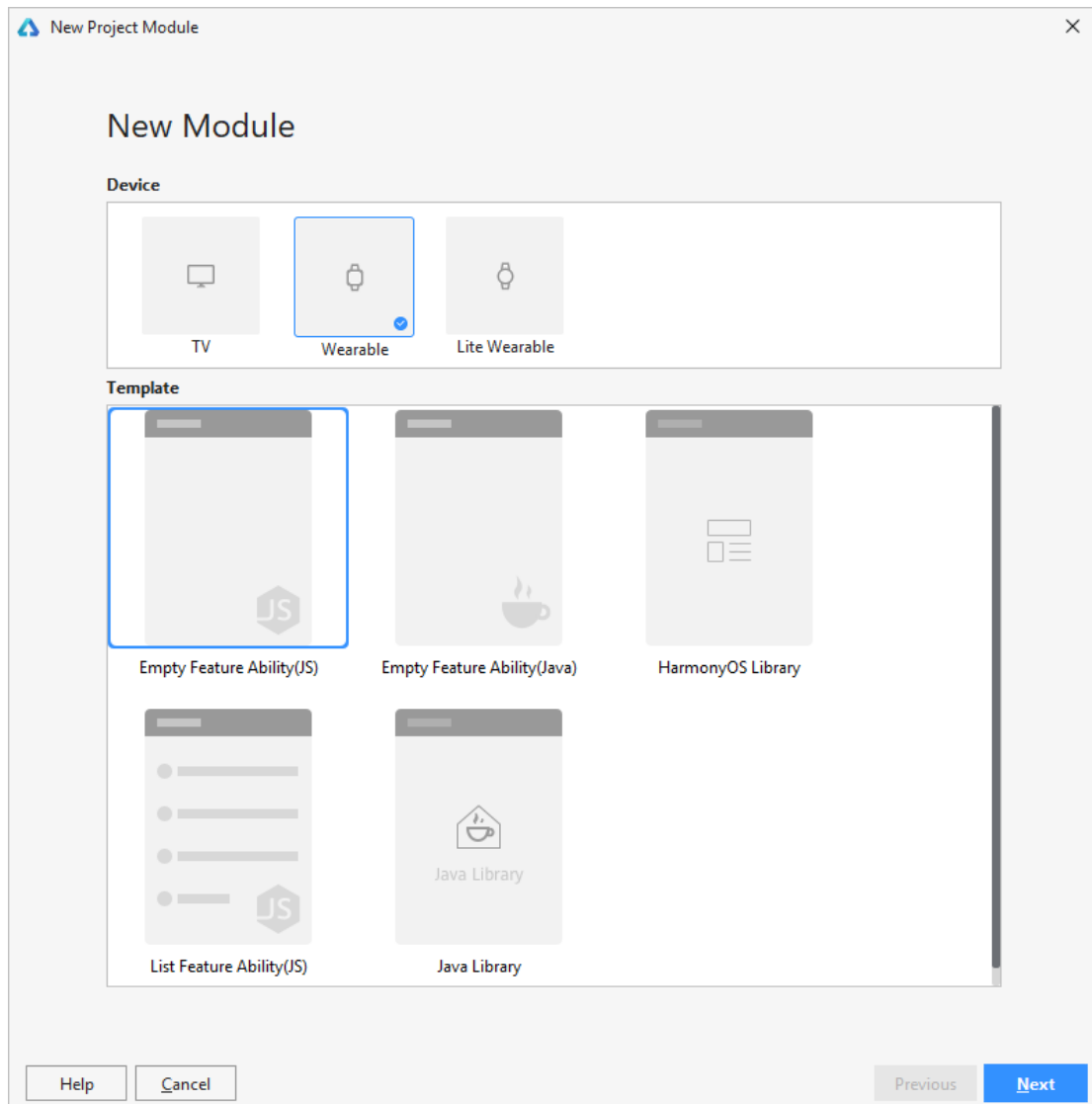
Module 是 HarmonyOS 应用的基本功能单元，包含了源代码、资源文件、第三方库及应用清单文件，每一个 Module 都可以独立进行编译和运行。一个 HarmonyOS 应用通常会包含一个或多个 Module，因此，可以在工程中，创建多个 Module，每个 Module 分为 Ability 和 Library（HarmonyOS Library 和 Java Library）两种类型。

如 HarmonyOS 工程介绍，在一个 APP 中，对于同一类型设备有且只有一个 Entry Module，其余 Module 的类型均为 Feature。因此，在创建一个类型为 Ability 的 Module 时，遵循如下原则：

- 若新增 Module 的设备类型为已有设备时，则 Module 的类型将自动设置为“Feature”。
- 若新增 Module 的设备类型为当前还没有创建 Module，则 Module 的类型将自动设置为“Entry”。

3.4.1 新增 Module

1. 通过如下两种方法，在工程中添加新的 Module。
 - 方法 1：鼠标移到工程目录顶部，点击鼠标右键，选择 **New>Module**，开始创建新的 Module。
 - 方法 2：在菜单栏选择 **File > New > Module**，开始创建新的 Module。
2. 在 New Project Module 界面中，选择 Module 对应的设备类型和模板。



3. 点击 **Next**，在 Module 配置页面，设置新增 Module 的基本信息。
 - Module 类型为 Ability 或者 HarmonyOS Library 时，请根据如下内容进行设置，然后点击 **Next**。
 - **Application/Library name**: 新增 Module 所属的类名称。
 - **Module name**: 新增模块的名称。
 - **Module Type**: 仅 Module 类型为 Ability 时存在，工具自动根据设备类型下的模块进行设置。
 - **Package name**: 软件包名称，可以点击 **Edit** 修改默认包名称，需全局唯一。
 - **Compatible SDK**: 兼容的 SDK 版本。

New Project Module

Configure the New Module

Application/Library name
MyApplication

Module name
myapplication

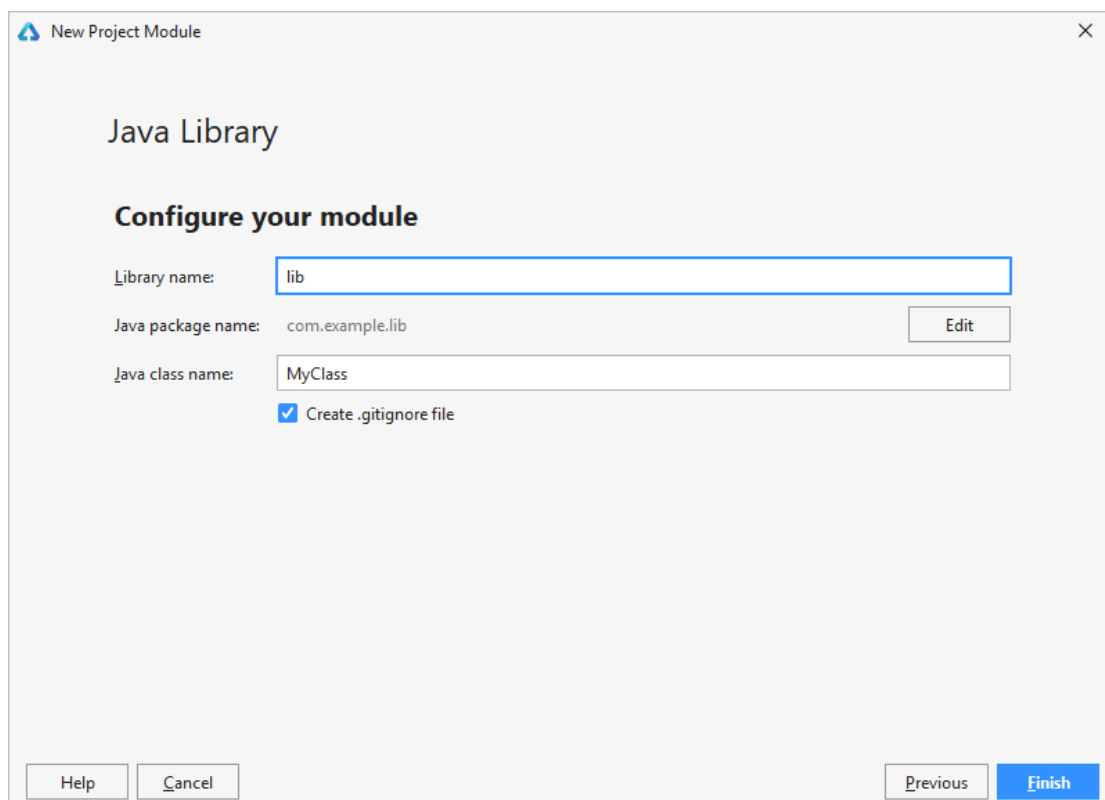
Module Type ?
Feature

Package name
com.example.myapplication Edit

Compatible SDK
SDK: API Version 3

Help Cancel Previous Next

- Module 类型为 Java Library 时，请根据如下内容进行设置，然后点击 **Finish** 完成创建。
- **Library Name**: Java Library 类名称。
- **Java package name**: 软件包名称，可以点击 **Edit** 修改默认包名称，需全局唯一。
- **Java class name**: class 文件名称。
- **Create.gitignore file**: 是否自动创建.gitignore 文件，勾选表示创建。



4. 设置新增 Ability 或 HarmonyOS Library 的 Page Name。

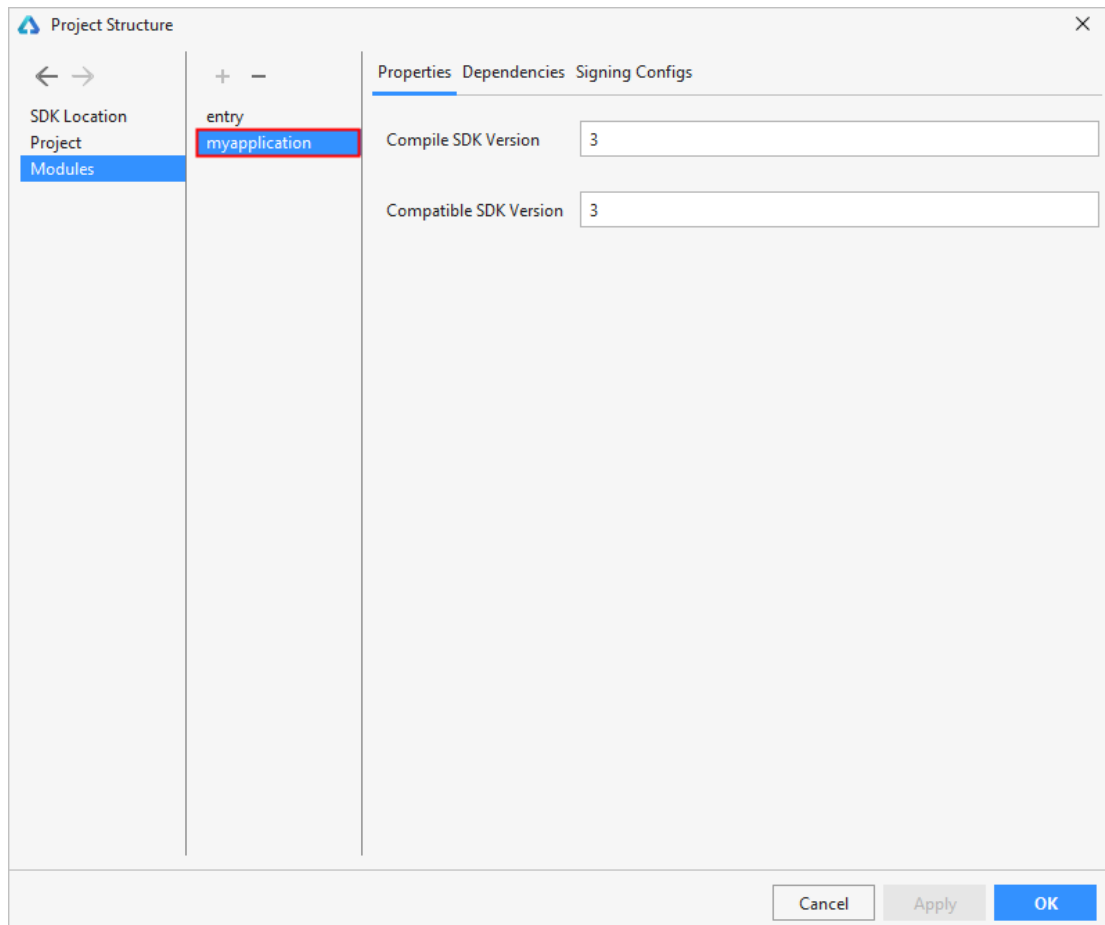
若该 Module 类型为 Ability，需要设置 Visible 参数，表示该 Ability 是否可以被其它应用所调用。

- 勾选 (true)：可以被其它应用调用。
 - 不勾选 (false)：不能被其它应用调用。
5. 点击 **Finish**，等待创建完成后，可以在工程目录中查看和编辑新增的 Module。

3.4.2 删除 Module

为防止开发者在删除 Module 的过程中，误将其它的模块删除，DevEco Studio 提供统一的模块管理功能，需要先在模块管理中，移除对应的模块后，才允许删除。

1. 在菜单栏中选择 **File > Project Structure > Modules**，选择需要删除的 Module，点击按钮，并在弹出的对话框中点击 **Yes**。



2. 在工程目录中选中该模块，点击鼠标右键，选中 **Delete**，并在弹出的对话框中点击 **Delete**。

4 代码编辑

4.1 编辑器使用技巧

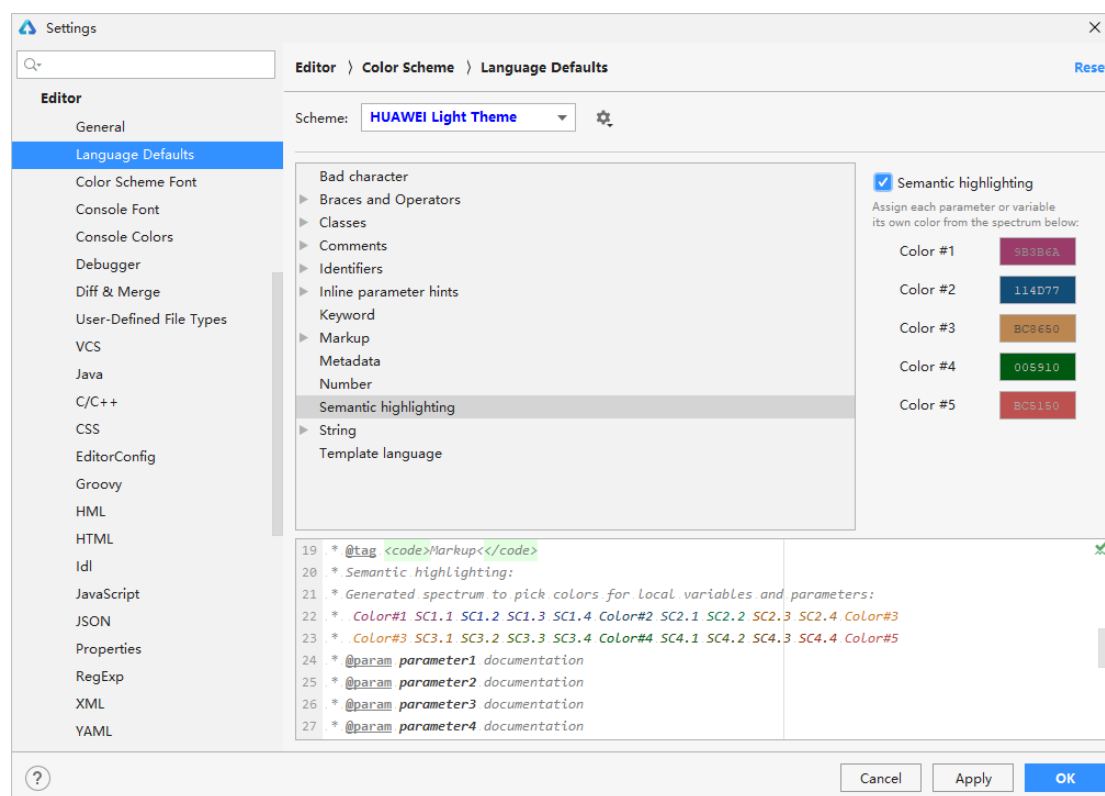
DevEco Studio 支持多种语言进行 HarmonyOS 应用的开发，包括 Java、JS 和 C/C++。

在编写应用阶段，您可以通过掌握各种代码编写的各种常用技巧，来提升编码效率。

4.1.1 代码高亮

支持对代码关键字、运算符、字符串、类名称、接口名、枚举值等进行高亮颜色显示，可以在菜单栏打开 **File > Settings**（或快捷键 **Ctrl+Alt+S**）面板，在 **Editor > Color Scheme** **Scheme** 自定义各语言高亮显示颜色。

同时还可以动态的对**变量名**和**参数名**进行语义高亮，默认情况下为关闭状态，可以在菜单栏打开 **File > Settings**（或快捷键 **Ctrl+Alt+S**）面板，在 **Editor > Color Scheme > Language Defaults > Semantic highlighting** 中，打开语义高亮开关。



4.1.2 代码智能补齐

编辑器工具会分析上下文并理解项目内容，并根据输入的内容，提示可补齐的类，方法，字段和关键字的名称等。

```
1 package com.example.myapplication.slice;
2
3 import ...
4
11
12 public class MainAbilitySlice extends AbilitySlice {
13
14     private PositionLayout myLayout = new PositionLayout( context: this);
15
16     @Override
17     public void onStart(Intent intent) {
18         super.onStart(intent);
19         LayoutConfig config = new LayoutConfig(LayoutConfig.MATCH_PARENT, LayoutConfig.MATCH_PARENT);
20         myLayout.setLayoutConfig(config);
21         ShapeElement element = new ShapeElement();
22         element.setShape(ShapeElement.RECTANGLE);
23         element.setRgbColor(new RgbColor( red: 255, green: 255, blue: 255));
24         myLayout.setBackground(element);
25
26         Text text = new Text( context: this);
27         text.setText("Hello World");
28         text.setTextColor(Color.BLACK);
29     }
30
31
32     @Override
```

4.1.3 代码错误检查

如果输入的语法不符合编码规范，或者出现拼写错误，编辑器会实时的进行代码分析，并在代码中突出显示错误或警告，并给出对应的修改建议。

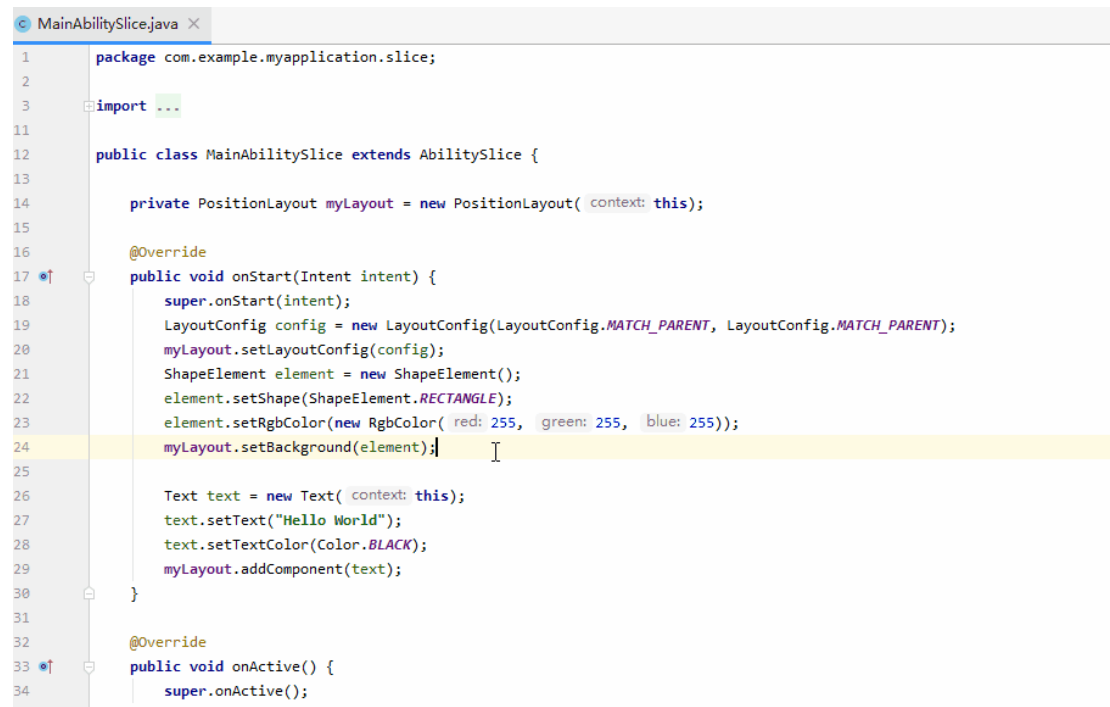
```
1 package com.example.myfirstapp.slice;
2
3 import ...
4
13
14 public class MainAbilitySlice extends AbilitySlice {
15
16     private DirectionalLayout myLayout = new DirectionalLayout( context: this);
17
18     @Override
19     public void onStart(Intent intent) {
20         super.onStart(intent);
21         tyxxx
22         LayoutConfig config = new LayoutConfig(LayoutConfig.MATCH_PARENT, LayoutConfig.MATCH_PARENT);
23         ShapeElement element = new ShapeElement();
24         element.setRgbColor(new RgbColor( red: 255, green: 255, blue: 255));
25         myLayout.setBackground(element);
26     }
```

Cannot resolve symbol 'tyxxx'

Create local variable 'tyxxx' Alt+Shift+Enter More actions... Alt+Enter

4.1.4 代码自动跳转

在编辑器中，可以按住 **Ctrl** 键，鼠标点击代码中的类、方法、参数、变量等名称，可以自动跳转到定义处。



```
1 package com.example.myapplication.slice;
2
3 import ...
4
11
12 public class MainAbilitySlice extends AbilitySlice {
13
14     private PositionLayout myLayout = new PositionLayout( context: this);
15
16     @Override
17     public void onStart(Intent intent) {
18         super.onStart(intent);
19         LayoutConfig config = new LayoutConfig(LayoutConfig.MATCH_PARENT, LayoutConfig.MATCH_PARENT);
20         myLayout.setLayoutConfig(config);
21         ShapeElement element = new ShapeElement();
22         element.setShape(ShapeElement.RECTANGLE);
23         element.setRgbColor(new RgbColor( red: 255, green: 255, blue: 255));
24         myLayout.setBackground(element);
25
26         Text text = new Text( context: this);
27         text.setText("Hello World");
28         text.setTextColor(Color.BLACK);
29         myLayout.addComponent(text);
30     }
31
32     @Override
33     public void onActive() {
34         super.onActive();
35     }
36 }
```

4.1.5 代码格式化

支持对选定范围的代码或者当前整个文件进行代码格式化操作，可以很好的提升代码的美观度和可读性。

- 使用快捷键 **Ctrl + Alt + L** 可以快速对选定范围的代码进行格式化。
- 使用快捷键 **Ctrl + Alt + Shift + L** 可以快速对当前整个文件进行格式化。

如果在进行格式化时，对于部分代码片段不需要进行自动的格式化处理，可以通过如下方式进行设置：

1. **首先**，在 **File>Settings>Editor>Code Style**，点击 “Formatter Control” ，勾选 “Enable formatter markers in comments” 。
2. **其次**，在 Java 或 C/C++ 代码中，对不需要进行格式化操作的代码块前增加 “//@formatter:off” ，对不格式化代码块的最后增加 “//@formatter:on” ，即表示对该范围的代码块不需要进行格式化操作。

```
17 public void onStart(Intent intent) {
18     // @formatter:off
19     super.onStart(intent);
20     LayoutConfig config = new LayoutConfig(LayoutConfig.MATCH_PARENT, LayoutConfig.MATCH_PARENT);
21     myLayout.setLayoutConfig(config);
22     ShapeElement element = new ShapeElement();
23     element.setShape(ShapeElement.RECTANGLE);
24     element.setRgbColor(new RgbColor( red: 255, green: 255, blue: 255));
25     myLayout.setBackground(element);
26     // @formatter:on
27     Text text = new Text( context: this);
28     text.setText("Hello World");
29     text.setTextColor(Color.BLACK);
30     myLayout.addComponent(text);
31     super.setUIContent(myLayout);
32 }
```

4.1.6 代码折叠

支持对代码块的快速折叠和展开，可以使用快捷键 **Ctrl + NumPad+** 快速展开已折叠的

代码；使用快捷键 **Ctrl + NumPad-** 折叠已展开的代码块。

```
14 private PositionLayout myLayout = new PositionLayout( context: this);
15
16 @Override
17 public void onStart(Intent intent) {
18     super.onStart(intent);
19     LayoutConfig config = new LayoutConfig(LayoutConfig.MATCH_PARENT, LayoutConfig.MATCH_PARENT);
20     myLayout.setLayoutConfig(config);
21     ShapeElement element = new ShapeElement();
22     element.setShape(ShapeElement.RECTANGLE);
23     element.setRgbColor(new RgbColor( red: 255, green: 255, blue: 255));
24     myLayout.setBackground(element);
25
26     Text text = new Text( context: this);
27     text.setText("Hello World");
28     text.setTextColor(Color.BLACK);
29     myLayout.addComponent(text);
30     super.setUIContent(myLayout);
31 }
32
33 @Override
34 public void onActive() {
35     super.onActive();
36 }
37
38 @Override
39 public void onForeground(Intent intent) {
40     super.onForeground(intent);
41 }
```


4.1.7 代码快速注释

支持对选择的代码块进行快速注释，使用快捷键 **Ctrl+/**** 快速进行注释。对于已注释的代码块，再次使用快捷键 **Ctrl+/**** 取消注释。

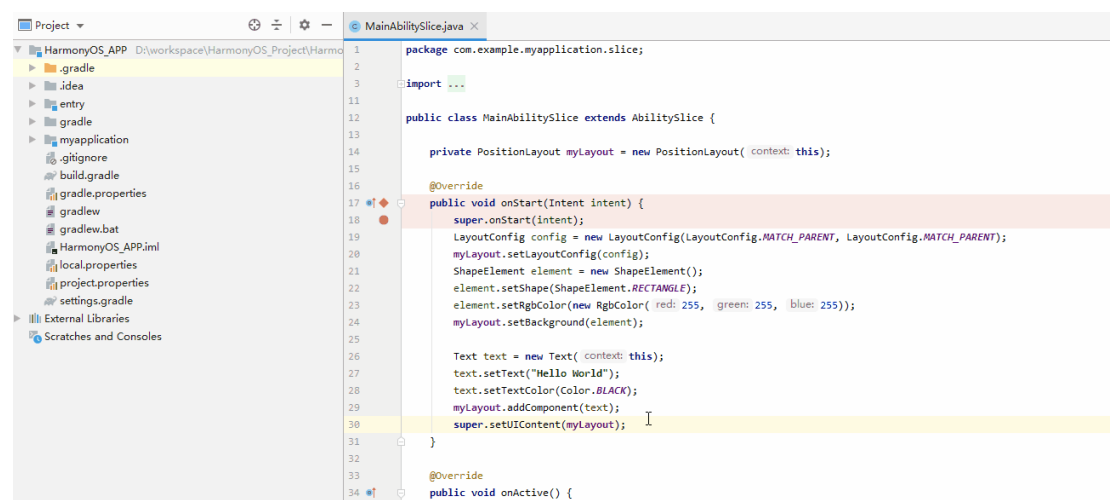
```

11
12 public class MainAbilitySlice extends AbilitySlice {
13
14     private PositionLayout myLayout = new PositionLayout( context: this);
15
16     @Override
17     public void onStart(Intent intent) {
18         super.onStart(intent);
19         LayoutConfig config = new LayoutConfig(LayoutConfig.MATCH_PARENT, LayoutConfig.MATCH_PARENT);
20         myLayout.setLayoutConfig(config);
21         ShapeElement element = new ShapeElement();
22         element.setShape(ShapeElement.RECTANGLE);
23         element.setRgbColor(new RgbColor( red: 255, green: 255, blue: 255));
24         myLayout.setBackground(element);
25
26         Text text = new Text( context: this);
27         text.setText("Hello World");
28         text.setTextColor(Color.BLACK);
29         myLayout.addComponent(text);
30         super.setUIContent(myLayout);
31     }
32
33     @Override
34     public void onActive() { super.onActive(); }
35
36
37
38     @Override
39     public void onForeground(Intent intent) { super.onForeground(intent); }

```

4.1.8 代码结构树

支持快速查看代码文档的结构树，包括全局变量和函数，类成员变量和方法等，并可以跳转到对应代码行。可使用快捷键 **Alt + 7 / Ctrl + F12** 快速打开代码结构树。



```

Project
├── HarmonyOS_APP
│   ├── .gradle
│   ├── .idea
│   ├── entry
│   ├── gradle
│   ├── myapplication
│   ├── .gitignore
│   ├── build.gradle
│   ├── gradle.properties
│   ├── gradlew
│   ├── gradlew.bat
│   ├── HarmonyOS_APP.iml
│   ├── local.properties
│   ├── project.properties
│   ├── settings.gradle
│   └── External Libraries
└── Scratches and Consoles

```

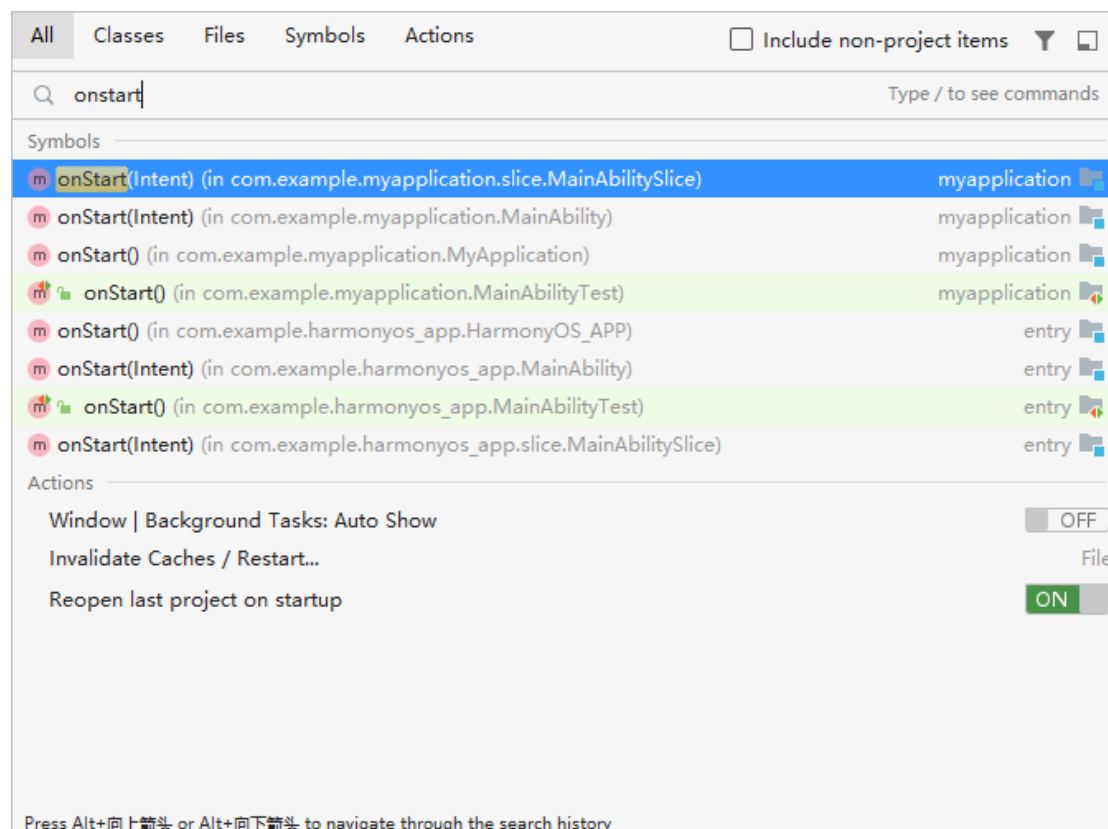
```

1 package com.example.myapplication.slice;
2
3 import ...
4
11 public class MainAbilitySlice extends AbilitySlice {
12
13     private PositionLayout myLayout = new PositionLayout( context: this);
14
15
16     @Override
17     public void onStart(Intent intent) {
18         super.onStart(intent);
19         LayoutConfig config = new LayoutConfig(LayoutConfig.MATCH_PARENT, LayoutConfig.MATCH_PARENT);
20         myLayout.setLayoutConfig(config);
21         ShapeElement element = new ShapeElement();
22         element.setShape(ShapeElement.RECTANGLE);
23         element.setRgbColor(new RgbColor( red: 255, green: 255, blue: 255));
24         myLayout.setBackground(element);
25
26         Text text = new Text( context: this);
27         text.setText("Hello World");
28         text.setTextColor(Color.BLACK);
29         myLayout.addComponent(text);
30         super.setUIContent(myLayout);
31     }
32
33     @Override
34     public void onActive() {

```

4.1.9 代码查找

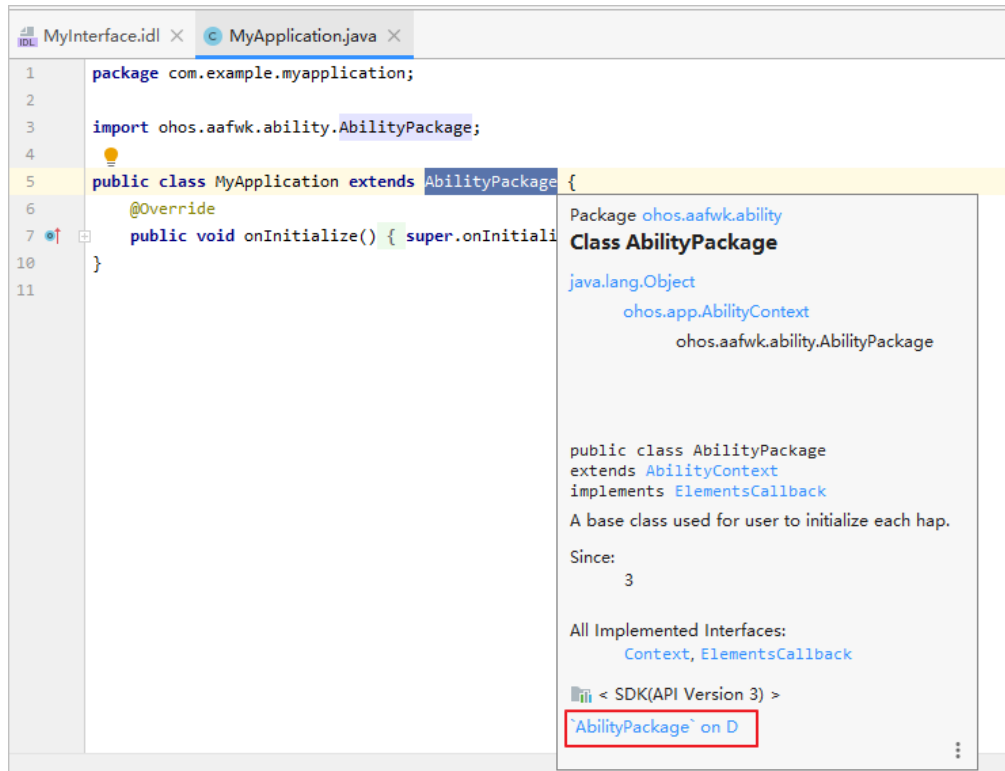
通过对符号、类或文件的即时导航来查找代码。检查调用或类型层次结构，轻松地搜索工程里的所有内容。通过使用连续按压**两次 Shift** 快捷键，打开代码查找界面。



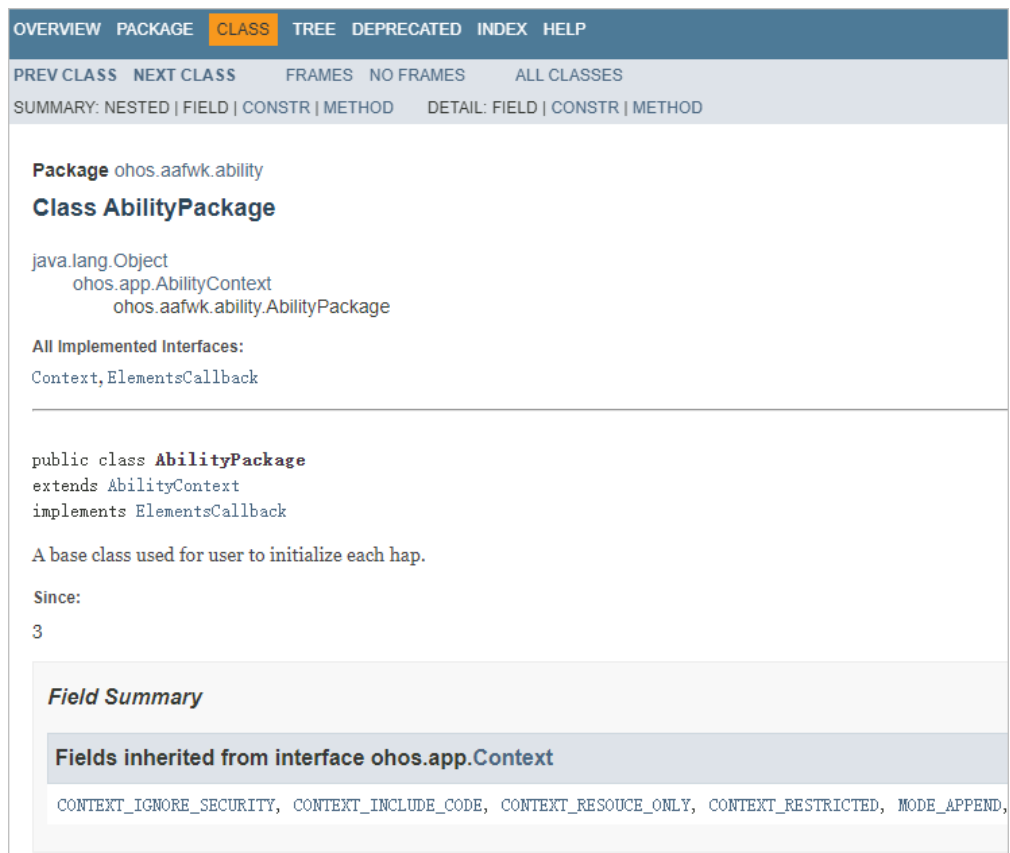
4.1.10 查看 Java 接口文档

在 Java 代码选中 HarmonyOS API 或选中 Java 类时，使用快捷键 **Ctrl+Q**，在弹出的“Documentation”最下方，会显示相应文档的链接。

例如：图示红框中的 “ ‘AbilityPackage’ on D”



点击文档的链接，比如：“‘AbilityPackage’ on D”，将打开详细说明文档。



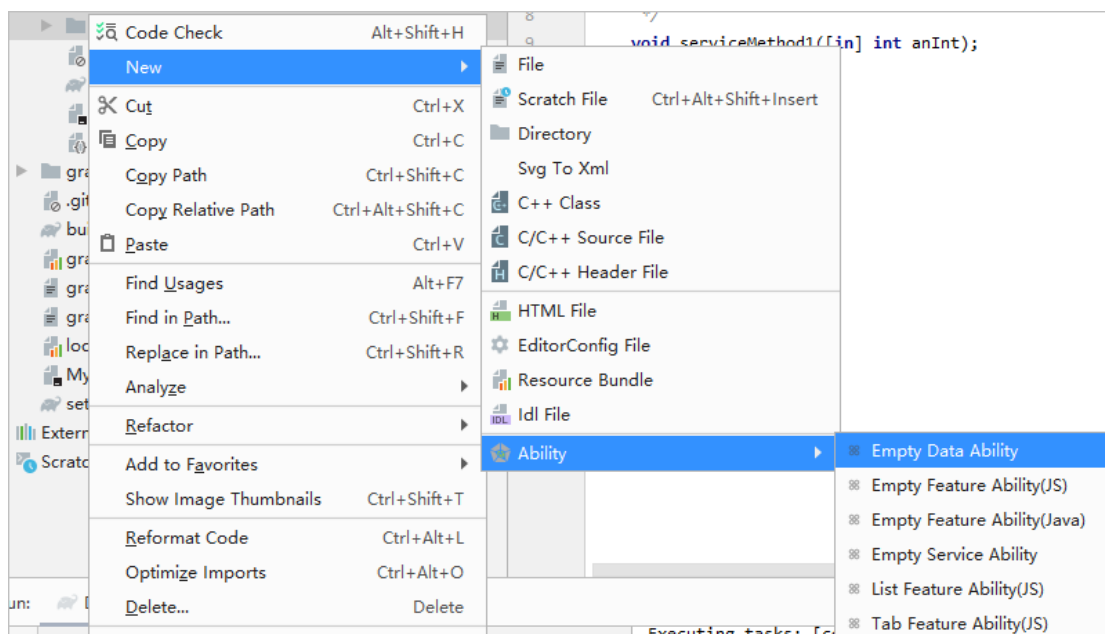
4.2 在模块中添加 Ability

Ability 是应用所具备的能力的抽象，一个 Module 可以包含一个或多个 Ability。Ability 分为两种类型：FA（Feature Ability）和 PA（Particle Ability），DevEco Studio 支持创建的 Ability 模板和应用场景如下表所示。

Ability 类型	Ability 模板	使用场景
Particle Ability	Empty Data Ability	Data Ability 有助于应用管理其自身和其他应用所存储数据的访问，并提供与其他应用共享数据的方法。Data 既可用于同设备不同应用的数据共享，也支持跨设备之间不同应用的数据共享。
	Empty Service Ability	Service Ability 可在后台长时间运行而不提供用户交互界面。Service 可由其他应用或 Ability 启动，即使用户切换到其他应用，Service 仍将在后台继续运行。
Feature Ability	Empty Feature Ability(JS)	用 JS 和 Java 编写带 UI 界面的空模板。
	Empty Feature Ability(Java)	用 Java 和 xml 编写带 UI 界面的空模板。
	List Feature Ability(JS)	用 JS 和 Java 编写带 UI 界面的目录列表模板。
	Tab Feature Ability(JS)	用 JS 和 Java 编写带 UI 界面的表单模板。

4.2.1 创建 Particle Ability

1. 选中对应的模块，点击鼠标右键，选择 **New > Ability**，然后选择 Empty Data Ability 或者 Empty Service Ability。



2. 根据选择的 Ability 模板，设置 Ability 的基本信息。

- **Empty Data Ability** 基本信息设置：
 - **Data Name**：Data Ability 类名称。
 - **Visible**：表示该 Ability 是否可以被其它应用所调用，勾选上则表示允许被调用。
 - **Package name**：新增 Ability 对应的包名称。
- **Empty Service Ability** 基本信息设置：
 - **Service Name**：Service Ability 类名称。
 - **Visible**：表示该 Ability 是否可以被其它应用所调用，勾选上则表示允许被调用。
 - **Package name**：新增 Ability 对应的包名称。
 - **Enable background mode**：指定用于满足特定类型的后台服务，可以将多个后台服务类型分配给特定服务。各服务与 config.json 文件的映射关系如下表所示。

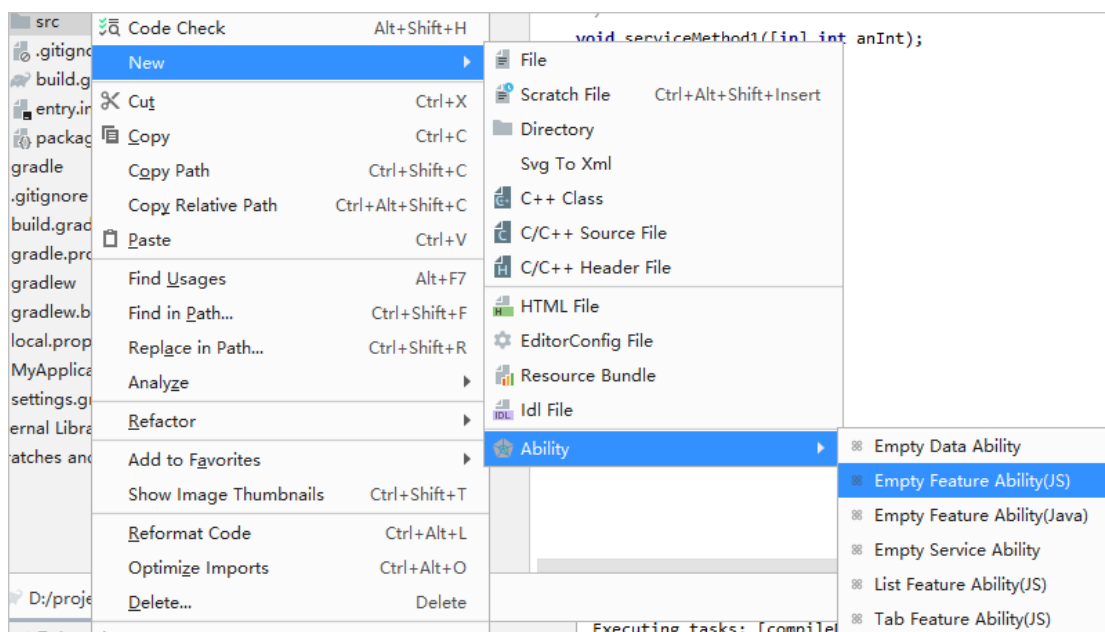
Background modes	对应 config.json 字段名称	描述
Data upload/download, backup/restore	data-transfer	通过网络/对端设备进行数据下载，备份分享，传输等业务
Audio playback	audio-playback	音频输出业务
Audio recording	audio-recording	音频输入业务
Picture-in-picture	picture-in-picture	画中画，小窗口播放视频业务

Background modes	对应 config.json 字段名称	描述
Voice/video call over IP	voip	音视频电话、VOIP 业务
Location update	location	定位，导航业务
Bluetooth communication	bluetooth-interaction	蓝牙扫描、连接、传输业务 (穿戴)
Wifi communication	wifi-interaction	WLAN 扫描、连接、传输业务 (多屏，克隆)
Screen recording, screenshot	screen-fetch	录屏，截屏业务

3. 点击 Finish 完成 Ability 的创建，可以在工程目录对应的模块中查看和编辑 Ability。

4.2.2 创建 Feature Ability

1. 选中对应的模块，点击鼠标右键，选择 **New > Ability**，然后选择对应的 Feature Ability 模板。



2. 根据选择的 Ability 模板，设置 Feature Ability 的基本信息。

- **Page Name**: Feature Ability 类名称。

- **Launcher Ability**: 表示该 Ability 在终端桌面上是否有启动图标，一个 HAP 可以有多个启动图标，来启动不同的 FA。
 - **Visible**: 表示该 Ability 是否可以被其它应用所调用，勾选上则表示允许被调用。
 - **JS Component Name**: JS 组件名称，只有涉及 JS 开发语言时才需要设置。
 - **Package name**: 新增 Ability 对应的包名称。
3. 点击 Finish 完成 Ability 的创建，可以在工程目录对应的模块中查看和编辑 Ability。

4.3 添加 JS Component 和 JS Page

在支持 JS 语言的工程中，支持添加新的 JS Component 和 JS Page，在此之前，需要了解它们的基本概念。

- JS Component: 在 JS 工程中，可以存在多个 JS Component（例如 js 目录下的 **default** 文件夹就是一个 JS Component），一个 JS FA 对应一个 JS Component，可以独立编译、运行和调试。

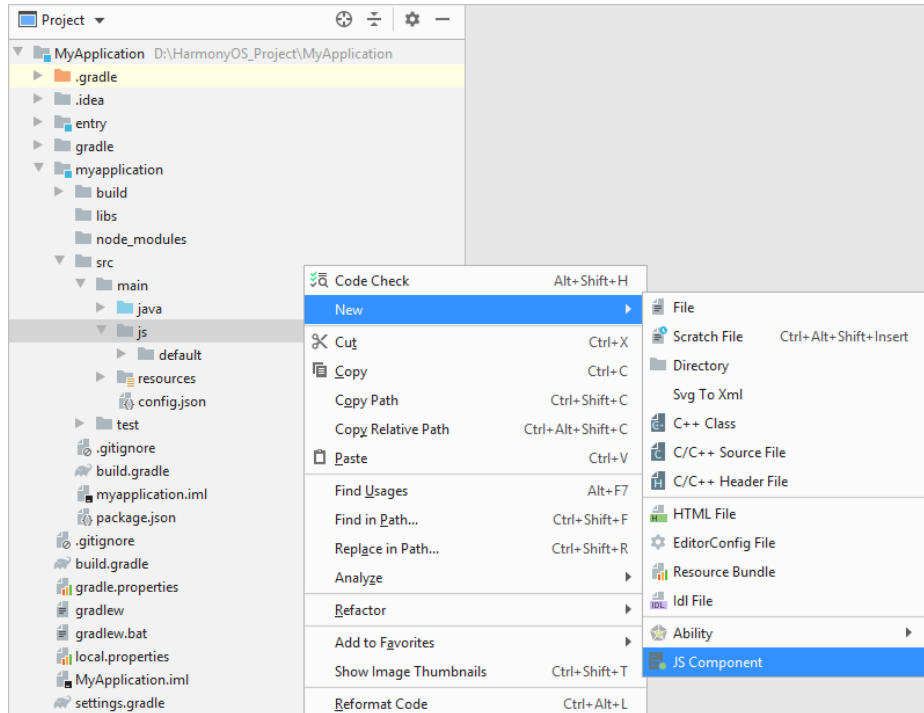
说明

轻量级智能穿戴对应的 JS 工程，只存在一个 JS FA，因此，轻量级智能穿戴的 JS 工程不允许创建新的 JS Component。

- JS Page: Page 是表示 JS FA 的一个前台页面，由 JS、HML 和 CSS 文件组成，是 Component 的最基本单元，构成了 JS FA 的每一个界面。

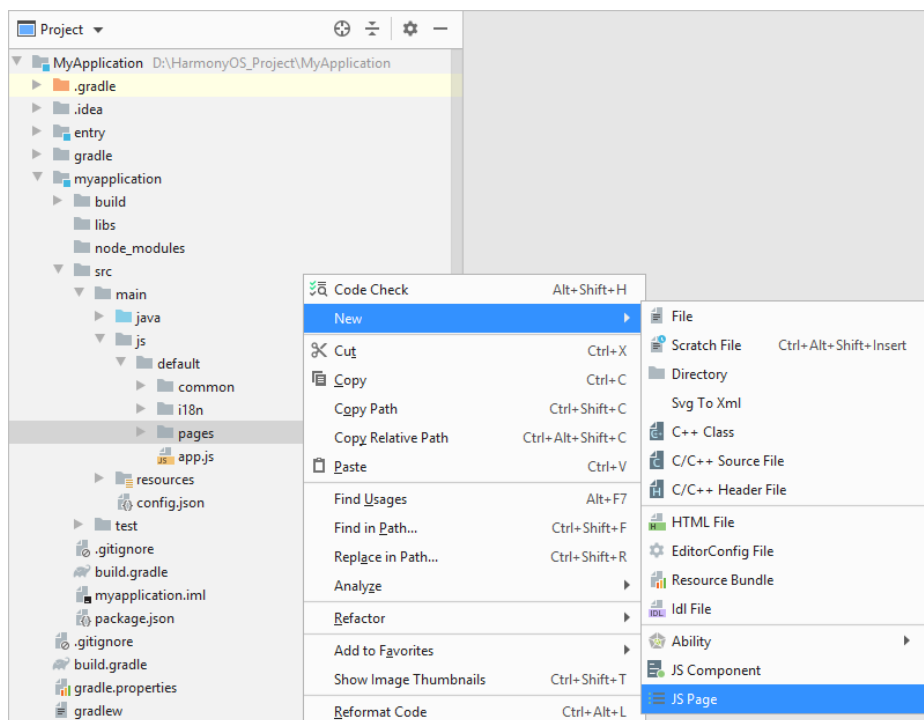
4.3.1 添加 JS Component

在 JS 工程目录中，选中 js 文件夹，然后点击鼠标右键，选择 **New > JS Component**，输入 JS Component Name，点击 **Finish** 完成添加。



4.3.2 添加 JS Page

在 JS 工程目录中，选择需要添加 Page 的 Component 下的 pages 文件夹，然后点击鼠标右键，选择 **New > JS Page**，输入 JS Page Name，点击 **Finish** 完成添加。



4.4 定义 HarmonyOS IDL 接口

4.4.1 HarmonyOS IDL 简介

HarmonyOS Interface Definition Language（简称 HarmonyOS IDL）是 HarmonyOS 的接口描述语言。HarmonyOS IDL 与其他接口语言类似，通过 HarmonyOS IDL 定义客户端与服务端均认可的编程接口，可以实现在二者间的跨进程通信（IPC，Inter-Process Communication）。跨进程通信意味着我们可以在一个进程访问另一个进程的数据，或调用另一个进程的方法。

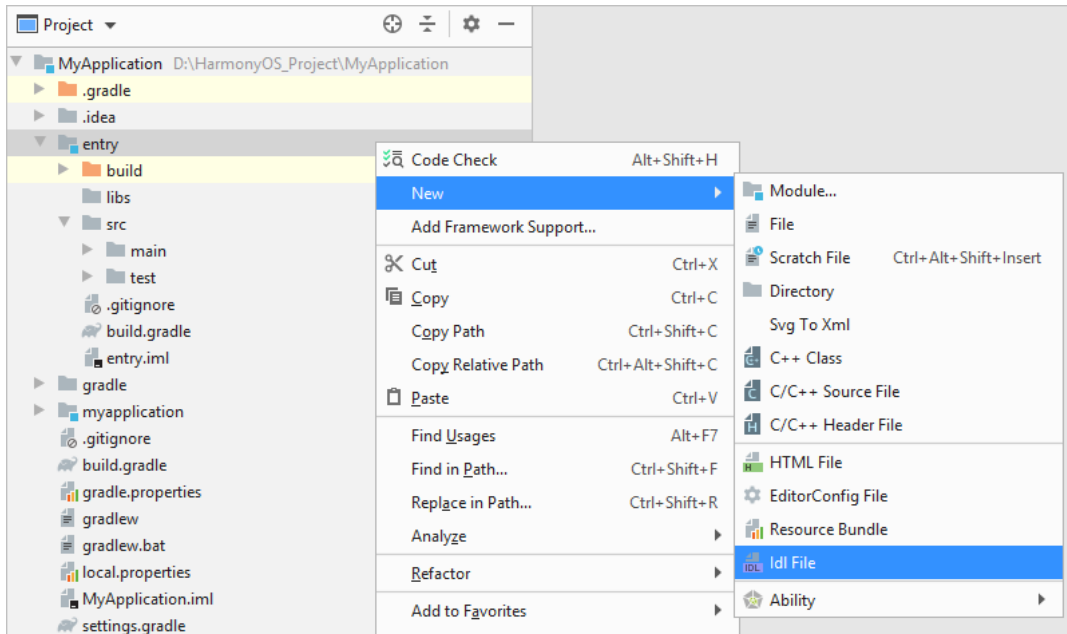
通常我们把应用接口提供方（供调用）称为服务端，调用方称为客户端。客户端通过绑定服务端的 Ability 来与之进行交互，类似于绑定服务。关于 DevEco Studio 接口语言的详细描述请参考 HarmonyOS IDL 接口使用规范。

说明

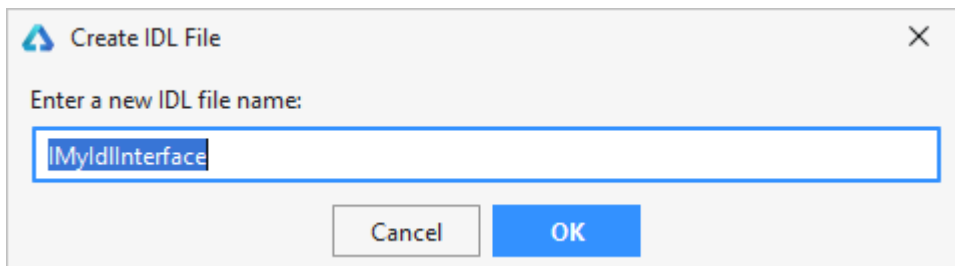
只能使用 Java 或 C++ 语言构建 .idl 文件，因此仅 Java、Java+JS、C/C++ 工程支持 IDL。

4.4.2 创建 .idl 文件

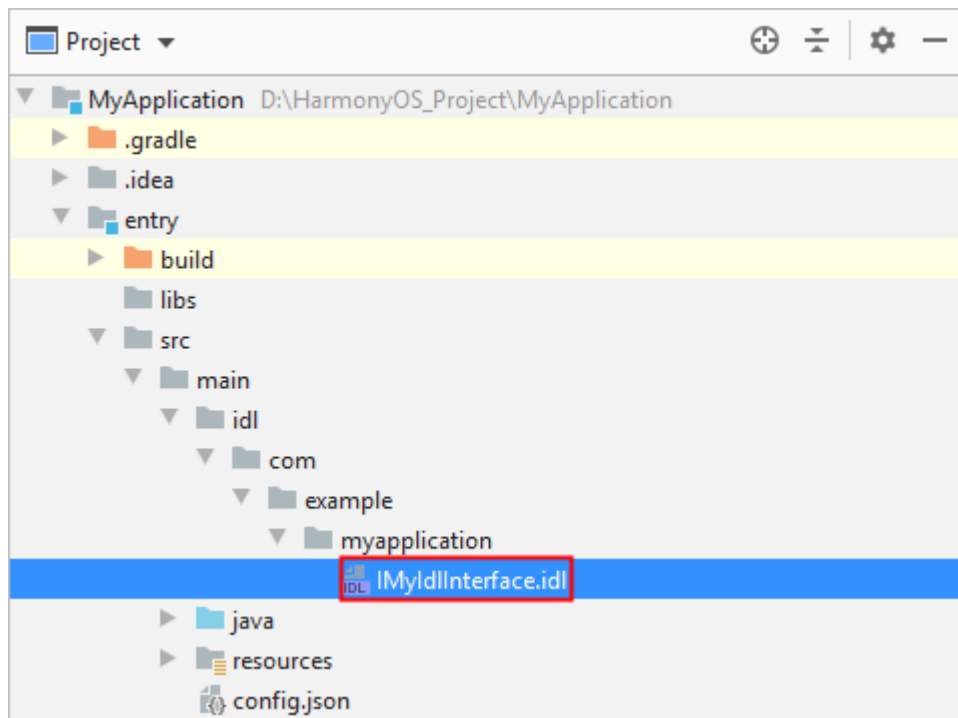
1. 在已经创建/打开的 HarmonyOS 工程中，选择 module 目录或其子目录，点击鼠标右键，选择 **New>Idl File**。



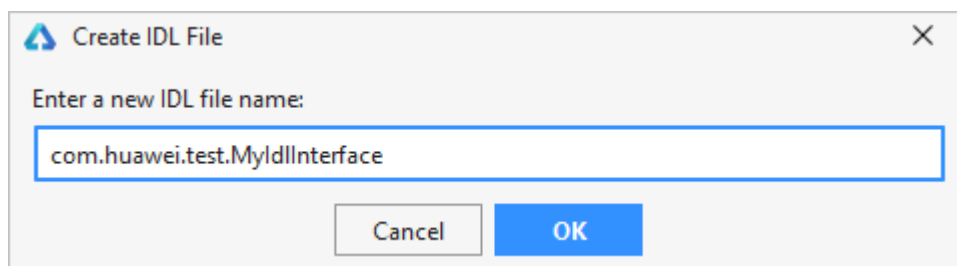
2. 创建 **IDL File**。可以直接输入 IDL 接口名称，也可以通过包名格式定义 idl 接口名称。两种方式的差异仅在于 .idl 文件的文件目录结构。
- 按名称创建，创建 **IDL File** 时，输入接口名称，直接点击 **OK**。



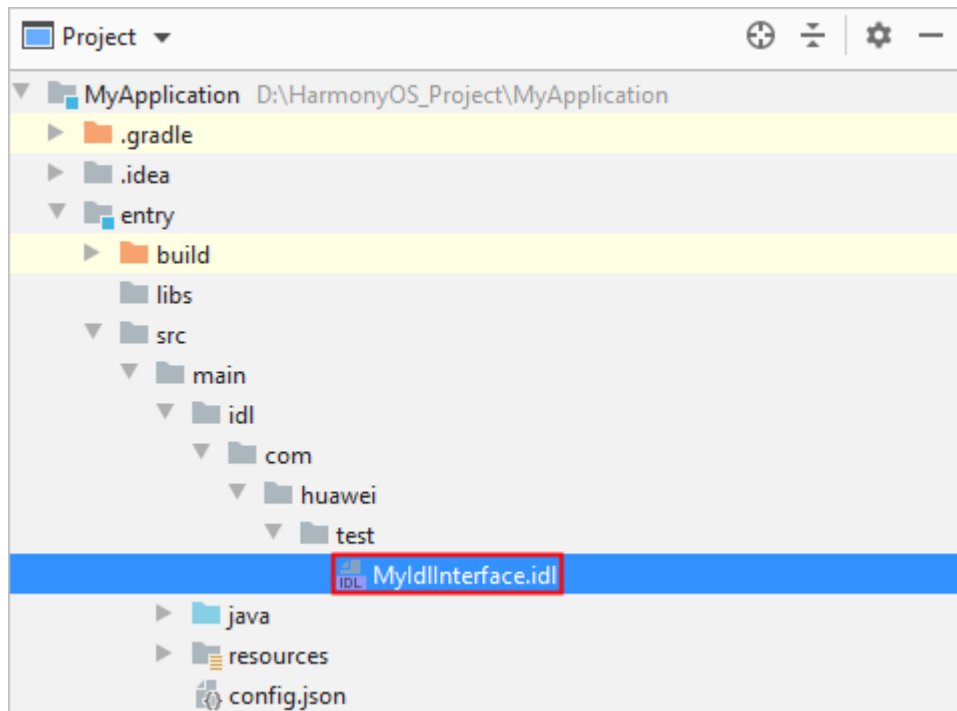
DevEco Studio 在相应 “module” 的 **src>main** 路径下生成 **idl** 文件夹，并按照对应模块的包名生成同样的目录结构及 **IDL** 文件。



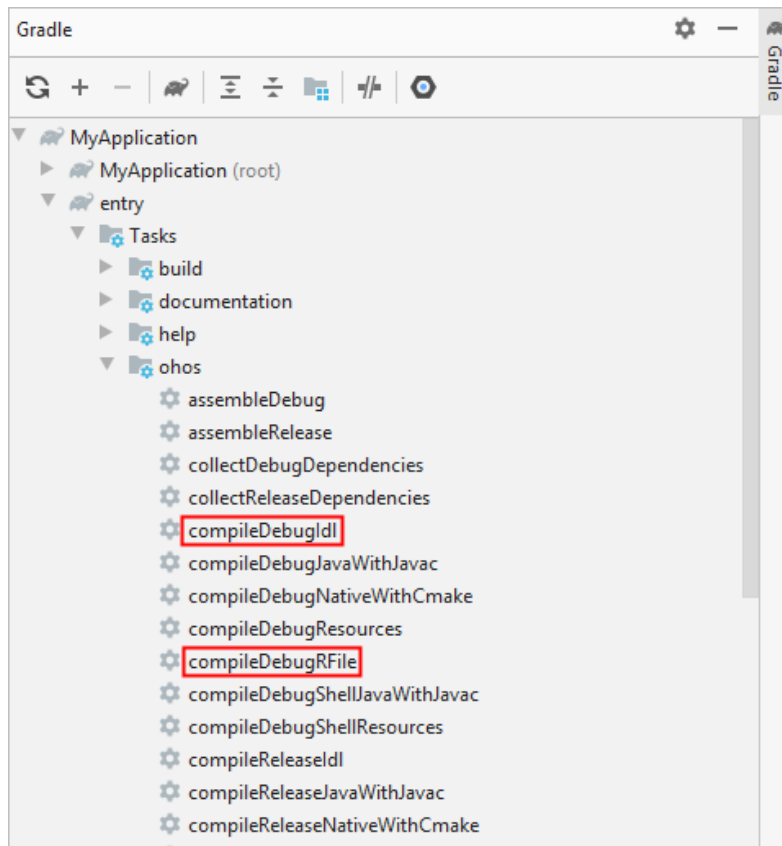
- 按包名创建，自定义.idl 文件存储路径和接口名称。创建“IDL File”时，按照包名创建 IDL 文件。包名利用“.”作为分隔符，如输入“com.huawei.test.MyIdlInterface”。



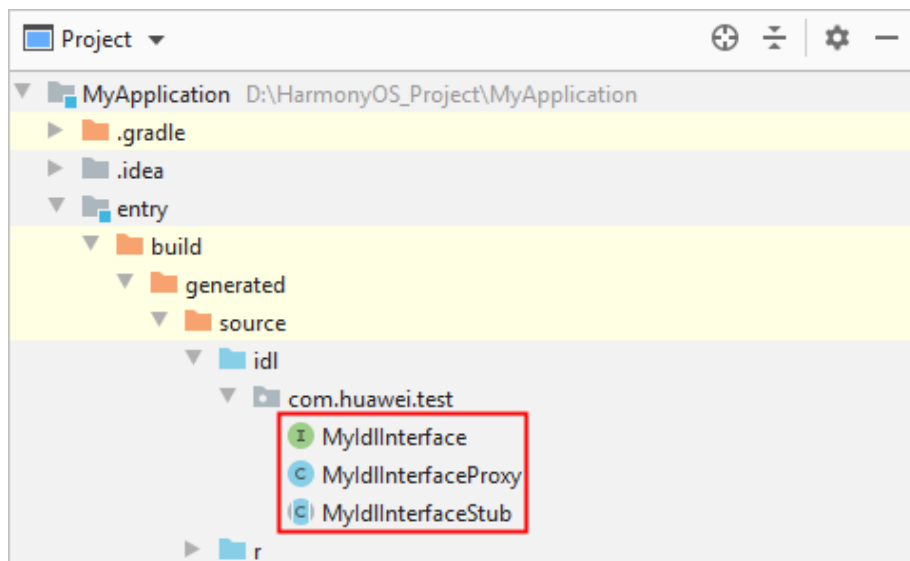
DevEco Studio 在相应“module”的 **src>main** 路径下生成 **idl** 文件夹，并按照输入的包名生成相应目录结构及 **IDL** 文件。可以在此路径继续新增 **IDL** 文件。



3. 点击工程右边栏的 **Gradle**，在 **Tasks > ohos** 中选择 **compileDebugIdl** 或 **compileReleaseIdl**，对模块下的 IDL 文件进行编译。



4. 编译完成后，在 **build > generated > source > Idl > {Package Name}**目录下，生成对应的接口类、桩类和代理类，如下图所示。



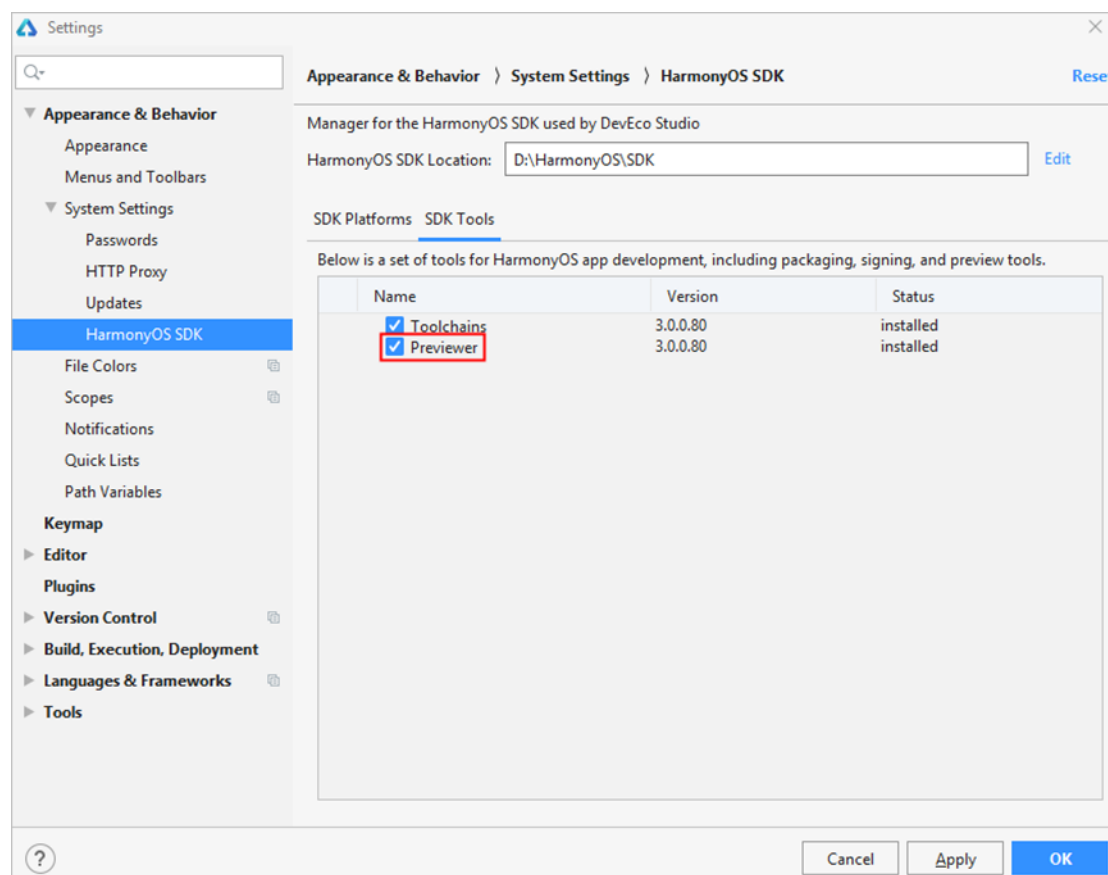
4.4.3 实现 HarmonyOS IDL 接口

开发者可以使用 Java 或 C++ 编程语言构建 .idl 文件，关于 HarmonyOS IDL 接口的实现请参考 IDL 开发指南。

4.5 使用预览器查看应用效果

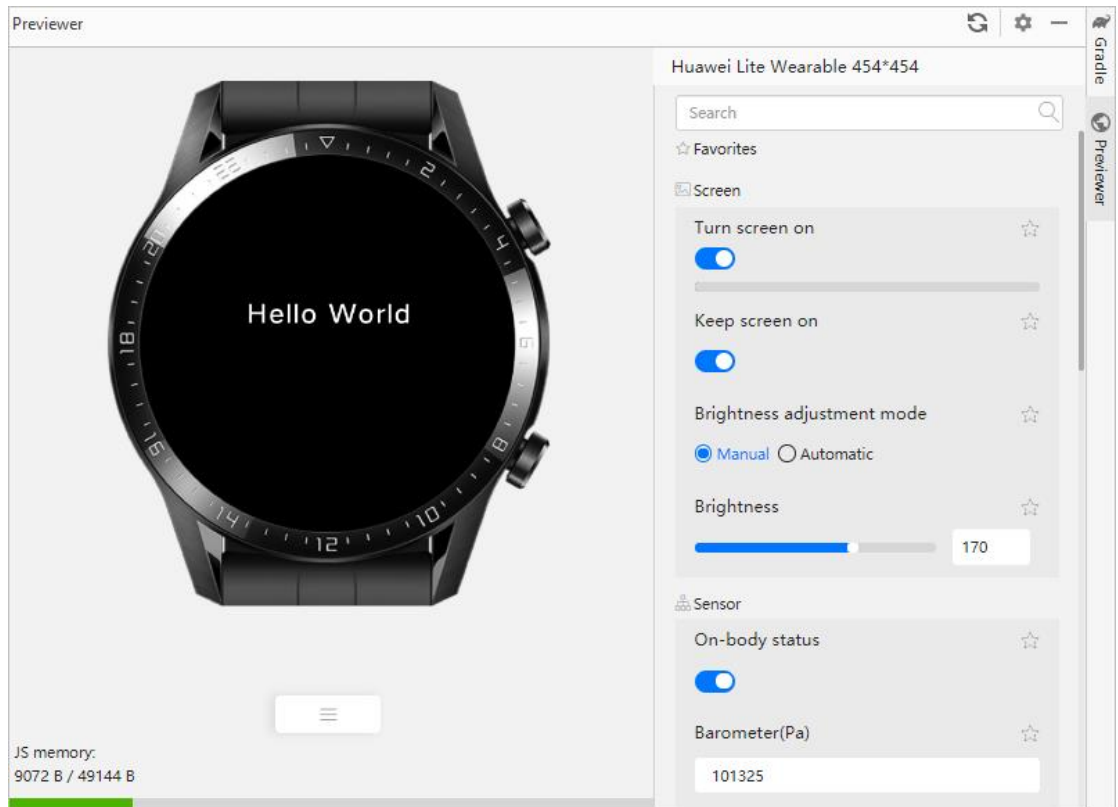
DevEco Studio 仅针对 Lite Wearable 提供预览器的功能。预览器支持代码热加载，在开发应用的同时，只要将开发的代码保存到源码中，即可在预览器中实时查看应用效果，方便开发者随时调整代码。

在使用预览器查看应用界面的 UI 效果前，需要确保 **HarmonyOS SDK > SDK Tools** 中，已下载 Previewer 资源，详情请参考下载 HarmonyOS SDK。



打开预览器有两种方式，显示效果如下图所示。

- 通过菜单栏，点击 **View>Tool Windows>Previewer**，打开预览器。
- 在编辑窗口右上角的侧边工具栏，点击 **Previewer**，打开预览器。

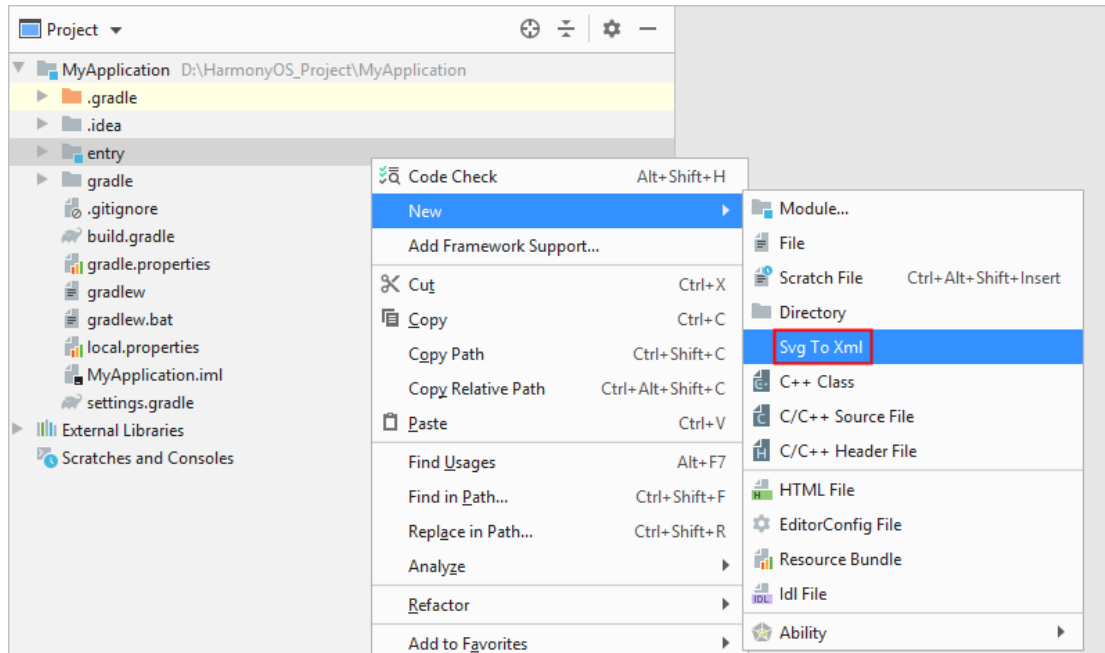


- 预览器还提供 Lite Wearable 的场景化数据注入功能，如点亮/关闭屏幕，调节屏幕亮度，向应用注入步数、心率、经纬度等信息，可以在开发阶段模拟真实的使用场景，便于开发者验证应用使用体验。

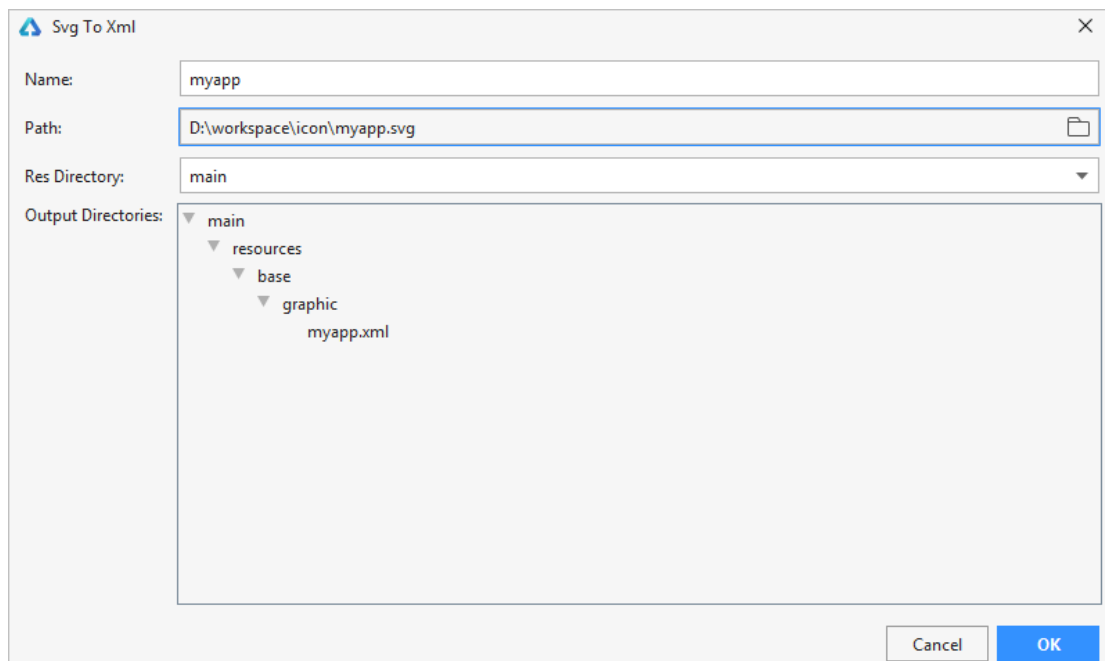
4.6 将 SVG 文件转换为 XML 文件

SVG (Scalable Vector Graphics) 可缩放矢量图形，是一种图像文件格式。目前由于 HarmonyOS 图形渲染引擎不支持 SVG 格式图片的渲染，开发者需要将 SVG 格式的图片文件转为 XML 格式的文件，然后在布局文件中引用转换后的 XML 文件。这样，就可以在模拟器/预览器或者设备上运行应用时，正常的渲染该图像文件。转换方法如下：

1. 选中应用模块，点击鼠标右键，选择 **New>Svg To Xml**。



2. 选择需要转换的 svg 文件，并命名，点击 **OK** 按钮开始转换。



3. 转换成功后，可以在 **resources > base > graphic** 文件下找到转换后的 xml 文件，并在布局文件中，引用该 xml 文件名即可完成对图标文件的引

4.7 代码安全检查

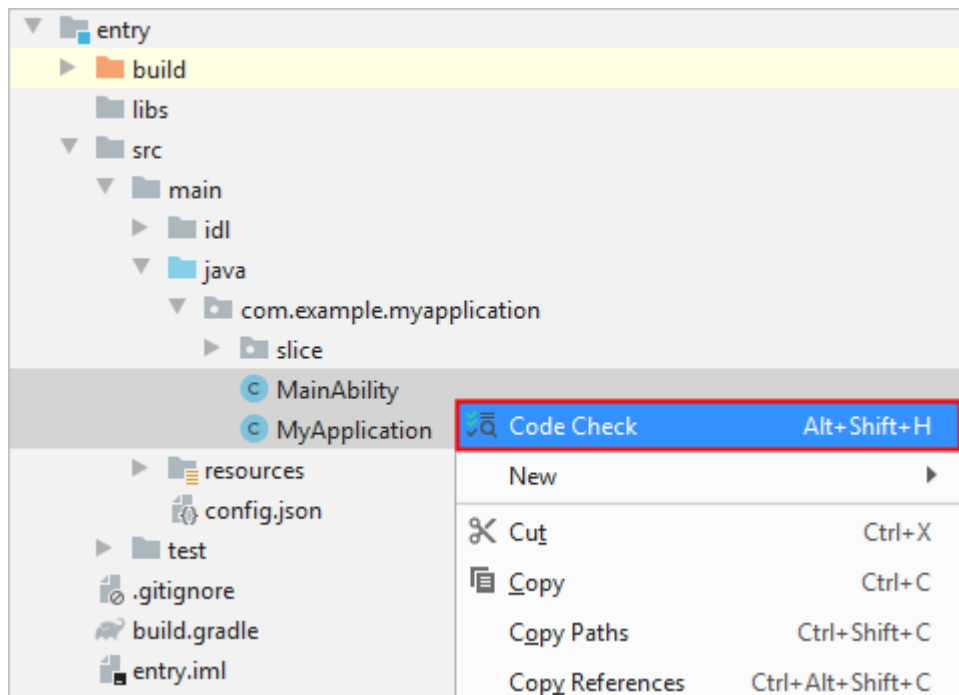
DevEco Studio 针对 Java 语言代码进行安全检查，扫描代码安全问题，并根据扫描结果

提示进行修改，有助于开发提高代码的健壮性。常见的代码安全问题包括如下几类：

- 凭据管理
- 认证问题和会话管理
- 权限控制
- 加密问题
- 信息泄露
- 完整性保护
- 隐私保护
- 不正确输入校验
- 安全编译

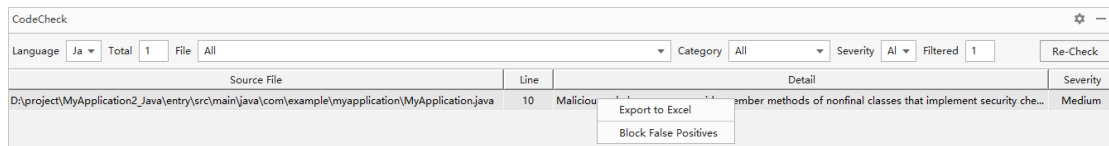
检查方法：鼠标选中已打开的代码编辑文件、或者鼠标点击选中文件或文件夹，或者按

Ctrl+鼠标点击选中多个文件，然后点击鼠标右键，选中 **Code Check**。



扫描完成后：

- 双击某个扫描结果可以跳转到对应代码，可以根据 **Detail** 的建议进行修改。
- 如果某个扫描结果不需要修改，可以对该扫描结果进行屏蔽，屏蔽后再执行 **Code Check** 将不再显示该扫描结果。
- 在扫描结果处，点击鼠标右键，选择 **Block False Positives** 可以屏蔽该行的安全检查。
- 如需恢复安全检查屏蔽的错误信息，可以在.idea>shield_config.xml 中删除某条屏蔽信息，或者直接删除.idea>shield_config.xml 文件来删除全部屏蔽信息。



5 编译构建

5.1 编译构建概述

编译构建是将 HarmonyOS 应用的源代码、资源、第三方库等打包生成 HAP 或者 APP 的过程。其中，HAP 可以直接运行在真机设备或者模拟器中；APP 则是用于应用上架到华为应用市场。HAP 和 APP 的关系说明请参考 HarmonyOS 工程介绍。

为了确保 HarmonyOS 应用的完整性，HarmonyOS 通过数字证书和授权文件来对应用进行管控，只有签名过的 HAP 才允许安装到设备上运行（如果不带签名信息，仅可以运行在模拟器中）；同时，上架到华为应用市场的 APP 也必须通过签名才允许上架。因此，为了保证应用能够发布和安装到设备上，需要提前申请相应的证书与 Profile 文件，详情请参考申请证书和 Profile。

申请证书和 Profile 文件时，用于调试和上架的证书与授权文件不能交叉使用：

- 应用调试证书与应用调试 Profile 文件、应用发布证书与应用发布 Profile 文件具有匹配关系，必须成对使用，不可交叉使用。
- 应用调试证书与应用调试 Profile 文件必须应用于调试场景，用于发布场景将导致应用发布审核不通过；应用发布证书与应用发布 Profile 文件必须应用于发布场景，用于调试场景将导致应用无法安装。

5.2 编译构建前配置

在进行 HarmonyOS 应用的编译构建前，需要对工程和编译构建的 Module 进行设置，请根据实际情况进行修改。

- build.gradle: HarmonyOS 应用依赖 gradle 进行构建，需要通过 build.gradle 来对工程编译构建参数进行设置。build.gradle 分为工程级和模块级两种类型，其中工程根目录下的工程级 build.gradle 用于工程的全局设置，各模块下的 build.gradle 只对本模块生效。
- config.json: 应用清单文件，用于描述应用的全局配置信息、在具体设备上的配置信息和 HAP 的配置信息。

5.2.1 工程级 build.gradle

- apply plugin: 在工程级 Gradle 中引入打包 app 的插件，不需要修改。

```
1. apply plugin: 'com.huawei.ohos.hap'
```

- ohos 闭包：工程配置，包括如下配置项：

- compileSdkVersion: 依赖的 SDK 版本。

```
1. compileSdkVersion 3 //应用编译构建的目标 SDK 版本
2.     defaultConfig {
3.         compatibleSdkVersion 3 //应用兼容的最低 SDK 版本
4.     }
```

- signingConfigs: 发布 APP 时的签名信息，在编译构建生成 APP 中进行设置后自动生成。

- buildscript 闭包：Gradle 脚本执行依赖，包括 Maven 仓地址和插件。

```
0. buildscript {
1.     repositories {
2.         maven {
3.             url 'https://mirrors.huaweicloud.com/repository/maven/'
4.         }
5.         maven {
6.             url 'https://developer.huawei.com/repo/'
7.         }
8.         jcenter()
9.     }
10.     dependencies {
11.         classpath 'com.huawei.ohos:hap:2.0.0.6'
12.     }
13. }
```

- allprojects 闭包：工程自身所需要的依赖，比如引用第三方库的 Maven 仓库和依赖包。

```
0. allprojects {
1.     repositories {
2.         maven {
```

```

3.         url 'https://mirrors.huaweicloud.com/repository/maven/'
4.     }
5.     maven {
6.         url 'https://developer.huawei.com/repo/'
7.     }
8.     jcenter()
9. }
10. }

```

5.2.2 模块级 build.gradle

- apply plugin：在模块级 Gradle 中引入打包 hap 和 library 的插件，无需修改。

```

1. apply plugin: 'com.huawei.ohos.hap' //打包 hap 包的插件
2. apply plugin: 'com.huawei.ohos.library' //将 HarmonyOS Library 打包为 har 的插件
3. apply plugin: 'com.huawei.ohos.java-library' //将 Java Library 打包为 jar 的插件

```

- ohos 闭包：模块配置，包括如下配置项：

- compileSdkVersion：依赖的 SDK 版本。

```

1. compileSdkVersion 3 //应用编译构建的目标 SDK 版本
2.     defaultConfig {
3.         compatibleSdkVersion 3 //应用兼容的最低 SDK 版本
4.     }

```

- signingConfigs：在编译构建生成 HAP 中进行设置后自动生成。

- externalNativeBuild：C/C++ 编译构建代码设置项。

```

1. externalNativeBuild {
2.     path "src/main/cpp/CMakeLists.txt" //CMake 配置入口，提供 CMake 构建脚本的相对路径
3.     arguments "-v" //传递给 CMake 的可选编译参数
4.     abiFilters "arm64-v8a" //用于设置本机的 ABI 编译环境
5.     cppFlags "" //设置 C++ 编译器的可选参数

```

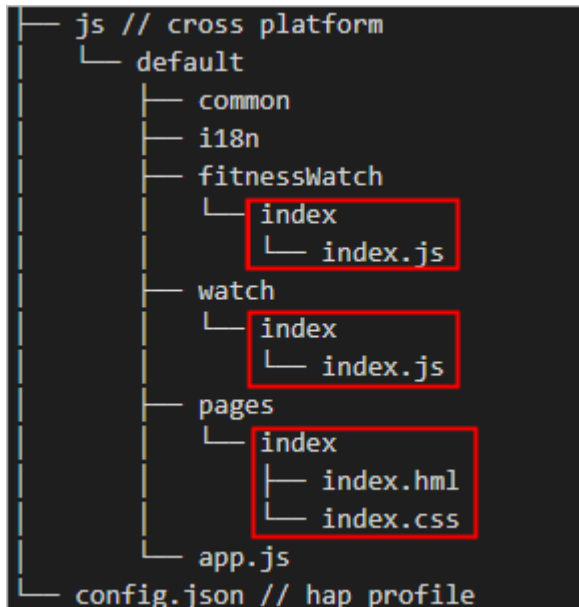
- entryModules：该 Feature 模块关联的 Entry 模块。

```

1. entryModules "entry"

```

- mergeJsSrc：跨设备的应用编译构建，是否需要合并 JS 代码。Wearable 和 Lite Wearable 共用一个工程，如下图所示。当进行编译构建时，将 Wearable/Lite Wearable 目录下的 JS 文件与 pages 目录（Wearable 和 Lite Wearable 共用的源码）下的 JS 文件进行合并打包。



1. mergejsrc **true** //合并 JS 代码打包时，请在 ohos 闭包下手动添加，true 表示需要合并 JS 代码，false 表示不需要合并 JS 代码。

▪ annotationEnabled: 支持数据库注释。

1. compileOptions{
2. annotationEnabled **true** //true 表示支持，false 表示不支持

• dependencies 闭包：该模块所需的依赖项。

```

0. dependencies {
1.   entryImplementation project(':entry') //该 Feature 模块依赖的 Entry 模块
2.   implementation fileTree(dir: 'libs', include: ['*.jar','*.har']) //该模块依赖的本地库，支持 jar 和 har 包
3.   testCompile 'junit:junit:4.12' //测试用例框架，无需修改
4. }
  
```

5.2.3 config.json 清单文件

HarmonyOS 应用的每个模块下包含一个 config.json 清单文件，在编译构建前，需要对照检查和修改 **config.json** 文件，详情请参考 **config.json 清单文件介绍**。

5.3 准备签名文件

5.3.1 生成密钥和证书请求文件

HarmonyOS 应用通过数字证书和授权文件来保证应用的完整性，在申请数字证书和 Profile 文件前，需要通过 DevEco Studio 来生成私钥（存放在.p12 文件中）和证书请求文件（.csr 文件）。同时，也可以使用命令行工具的方式来生成密钥和证书请求文件，用于构筑工程流水线。

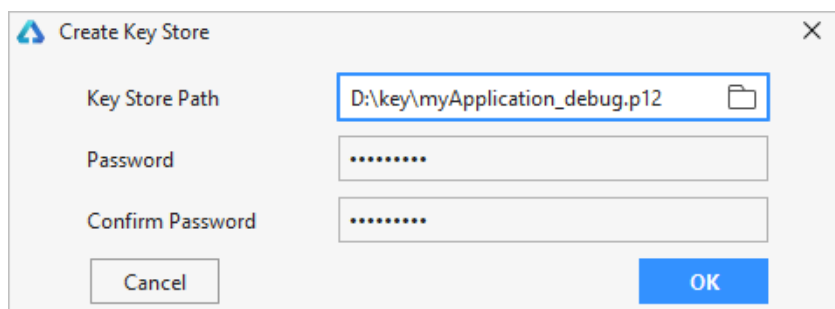
5.3.1.1 使用 DevEco Studio 生成证书请求文件

使用 DevEco Studio 生成证书请求文件的方式有以下两种情况：

- 如果还未生成密钥文件，则可以一键生成密钥和证书请求文件。
- 如果已有密钥文件，则可以使用已有密钥生成证书请求文件。

一键生成密钥和证书请求文件

1. 在主菜单栏点击 **Build > Generate Key**。
 2. 在 **Key Store Path** 中，可以点击 **Choose Existing** 选择已有的密钥库文件；如果没有密钥库文件，点击 **New** 进行创建。下面以新创建密钥库文件为例进行说明。
 3. 在 **Create Key Store** 窗口中，填写密钥库信息后，点击 **OK**。
- **Key Store Path**：选择密钥库文件存储路径。
 - **Password**：设置密钥库密码，必须由大写字母、小写字母、数字和特殊符号中的两种以上字符的组合，长度至少为 8 位。请记住该密码，后续签名配置需要使用。
 - **Confirm Password**：再次输入密钥库密码。



4. 在 **Generate Key** 界面中，继续填写密钥信息后，点击 **Generate Key and CSR**。
 - **Alias**：密钥的别名信息，用于标识密钥名称。请记住该别名，后续签名配置需要使用。
 - **Password**：输入密钥对应的密码，密钥密码需要与密钥库密码保持一致。请记住该密码，后续签名配置需要使用。
 - **Confirm Password**：再次输入密钥密码。
 - **Validity**：证书有效期，建议设置为 25 年及以上，覆盖应用的完整生命周期。
 - **Certificate**：输入证书基本信息，如组织、城市或地区、国家码等。

Generate Key

Key Store Path (*.p12) New Choose Existing

Key Store Password

Create a New Key

Alias ?

Password

Confirm Password

Validity(years) 25

Certificate

First and Last Name

Organizational Unit

Organization

City Or Locality

State Or Province

Country Code(XX)

Cancel Generate Key and CSR Generate Key

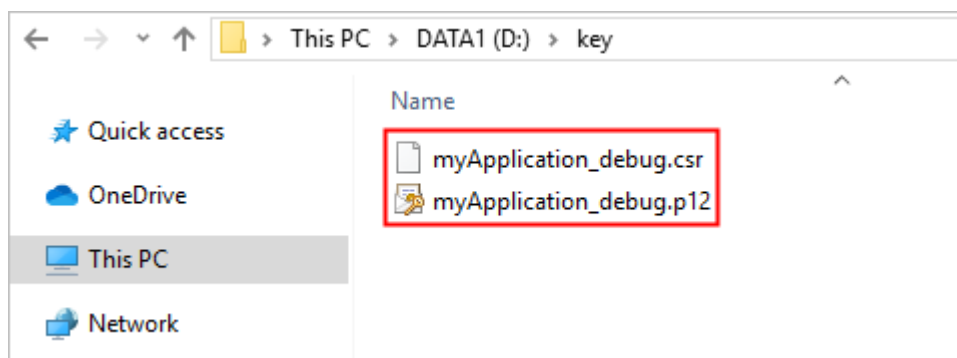
5. 在弹出的窗口中，点击 **CSR File Path** 对应的图标，选择 CSR 文件存储路径。

Generate Certificate Request File

CSR File Path Folder Icon

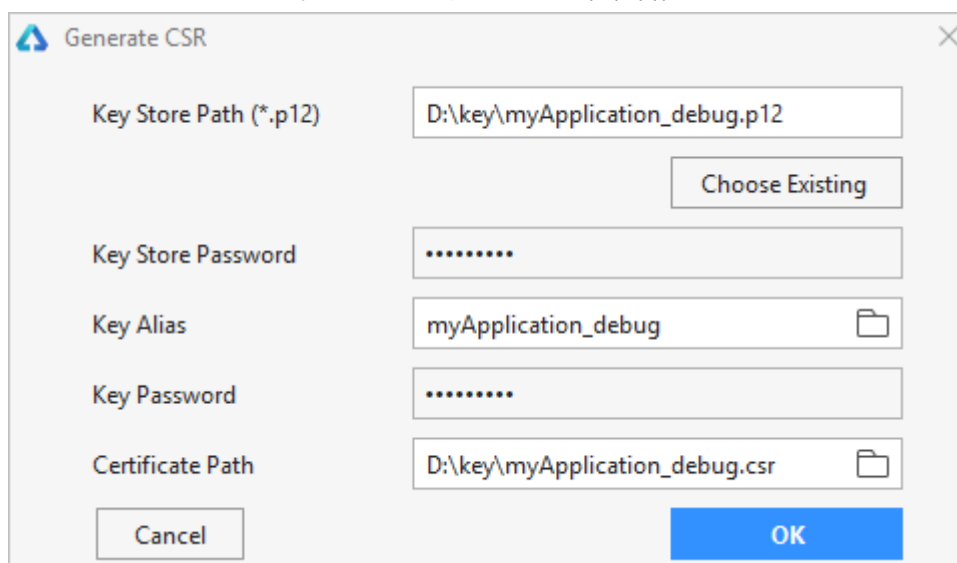
Cancel OK

6. 点击 **OK** 按钮，创建 CSR 文件成功，工具会同时生成密钥文件（.p12）和证书请求文件（.csr）。



使用已有密钥生成证书请求文件

1. 在主菜单栏点击 **Build > Generate Certificate Request File**。
2. 在 Generate CSR 界面，填写证书请求文件生成参数，点击 **OK**。
 - **Key Store Path**：点击 **Choose Existing** 选择已有的密钥库文件，后缀格式为.p12。
 - **Key Store Password**：输入创建密钥时填写的密钥库密码。
 - **Key Alias**：输入创建密钥时填写的别名信息。
 - **Key Password**：输入创建密钥时填写的密钥密码。
 - **Certificate Path**：点击按钮，选择证书请求文件存储路径和名称。

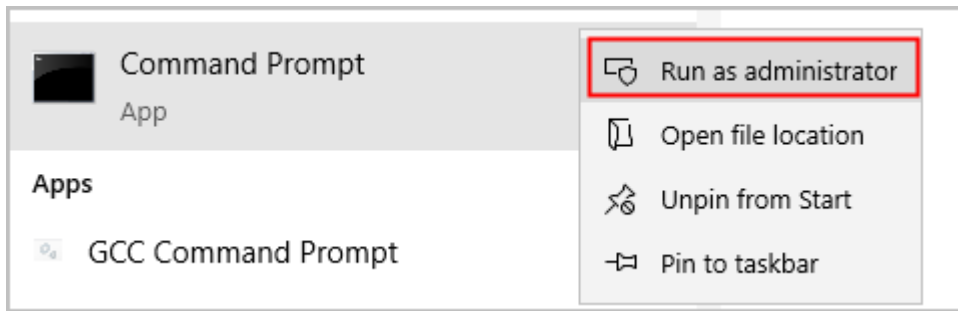


3. 打开证书请求文件存储目录，获取证书请求文件（.csr 文件）。

5.3.1.2 使用命令行工具生成证书请求文件

使用 Open JDK 携带的 Keytool 工具生成证书请求文件。

1. 使用管理员身份运行命令行工具。



2. 切换到 keytool 工具所在路径，实际路径请根据安装目录进行修改。

```
C:\Windows\system32>d:
D:\>cd "\Program Files\Huawei\DevEco Studio\jbr\jre\bin"
D:\Program Files\Huawei\DevEco Studio\jbr\jre\bin>
```

3. 执行如下命令，生成密钥文件。例如，生成的密钥名称为 ide_demo_app.p12，存储到 D 盘根目录下。

```
1. keytool -genkeypair -alias "ide_demo_app" -keyalg EC -sigalg SHA256withECDSA -dname
"C=CN,O=HUAWEI,OU=HUAWEI IDE,CN=ide_demo_app" -keystore d:\idedemokey.p12 -storetype pkcs12 -
validity 9125 -storepass 123456 -keypass 123456
```

生成密钥文件的参数说明如下：

说明

请记录下 **alias**、**storepass** 和 **keypass** 的值，后续编译构建生成 HAP 和编译构建生成 APP 会使用到。

- **alias**：密钥的别名信息，用于标识密钥名称。
 - **sigalg**：签名算法，固定为 **SHA256withECDSA**。
 - **dname**：按照操作界面提示进行输入。
 - C：国家/地区代码，如 CN。
 - O：组织名称，如 HUAWEI。
 - OU：组织单位名称，如 HUAWEI IDE。
 - CN：名字与姓氏，建议与别名一致。
 - **validity**：证书有效期，建议设置为 9125（25 年）。
 - **storepass**：设置密钥库密码。
 - **keypass**：设置密钥的密码，请与 **storepass** 保持一致。
4. 执行如下命令，执行后需要输入 **storepass** 密码，生成证书请求文件，后缀格式为.csr。
 0. keytool -certreq -alias "ide_demo_app" -keystore d:\idedemokey.p12 -storetype pkcs12 -file
 d:\idedemokey.csr

生成证书请求文件的参数说明如下：

- **alias**：与 3 中输入的 alias 保持一致。
- **file**：生成的证书请求文件名称，后缀为.csr。

5.3.2 申请证书和 Profile

目前华为应用市场只支持 Lite Wearable（轻量级智能穿戴）的 HarmonyOS 应用的上架，因此只支持轻量级智能穿戴设备的证书和 Profile 文件申请。智慧屏、智能穿戴等设备的证书、Profile 文件的申请，以及对应应用上架功能，敬请期待。

对于 Lite Wearable 的调试、发布证书和 Profile 的申请，请参考：

- 申请调试证书和 Profile
- 申请发布证书和 Profile

说明

在申请证书和 Profile 前，请根据生成密钥和证书请求文件准备好证书请求文件。

5.4 编译构建生成 HAP

HAP 可以直接在模拟器或者真机设备上运行，用于 HarmonyOS 应用开发阶段的调试和查看运行效果。HAP 按构建类型和是否签名可以分为以下四种形态：

- **构建类型为 Debug 的 HAP（带调试签名信息）**：携带调试签名信息，具备单步调试等调试手段的 HAP，用于开发者在真机或者模拟器中进行应用调试。
- **构建类型为 Debug 的 HAP（不带签名）**：不带调试签名信息，具备单步调试等调试手段的 HAP，仅能运行在模拟器中。
- **构建类型为 Release 的 HAP（带调试签名信息）**：携带调试签名信息，不具备调试能力的 HAP，用于开发者在真机或者模拟器中查看和验证应用运行效果。相对于 Debug 类型的 HAP 包，体积更小，运行效果与用户实际体验一致。
- **构建类型为 Release 的 HAP（不带签名）**：不带调试签名信息，不具备调试能力的 HAP，仅能运行在模拟器中查看和验证应用运行效果。相对于 Debug 类型的 HAP 包，体积更小，运行效果与用户实际体验一致。

根据 HarmonyOS 工程介绍，一个 HarmonyOS 工程下可以存在多个 Module，在编译构建时，可以选择对单个 Module 进行编译构建；也可以对整个工程进行编译构建，同时生成多个 HAP。

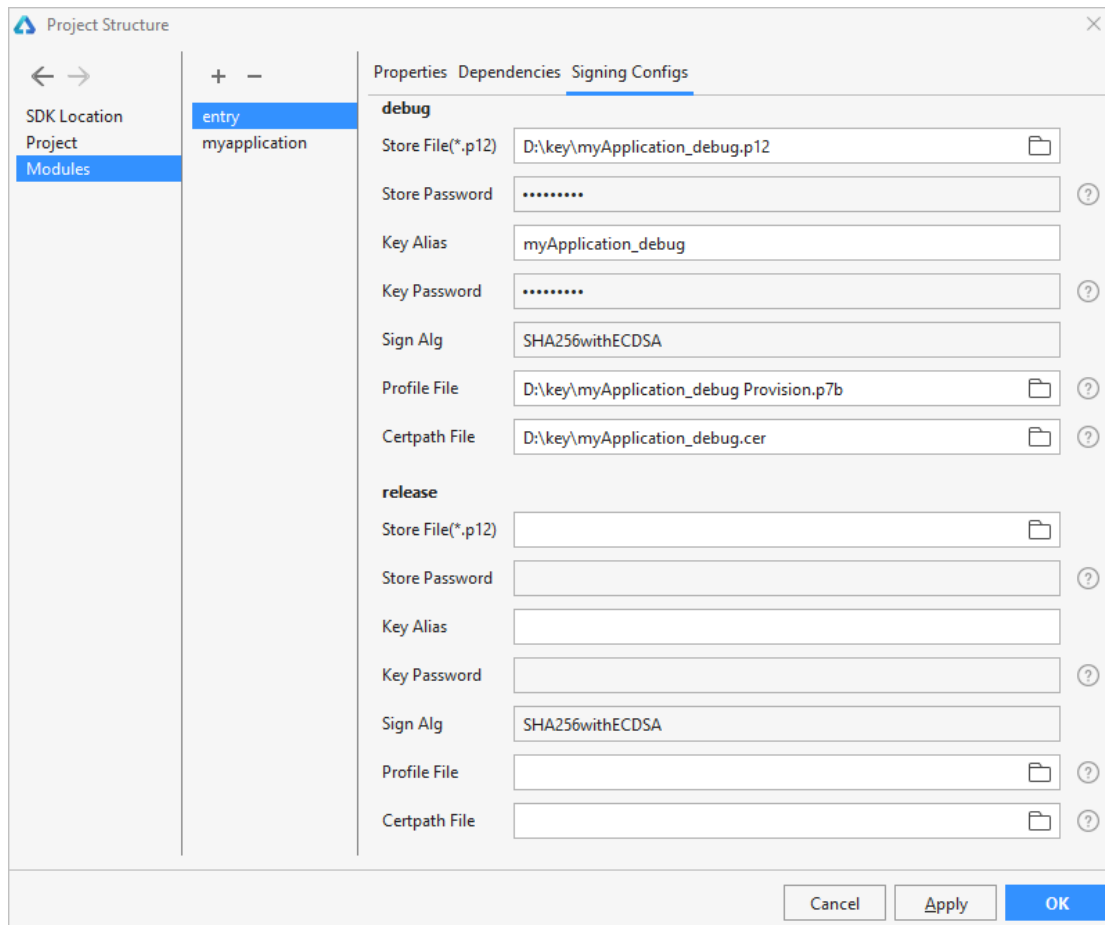
5.4.1 前提条件

- 已完成 build.gradle 和 config.json 的设置，详情请参考编译构建前配置。
- 已完成调试证书和 Profile 文件的申请，详情请参考申请证书和 Profile。

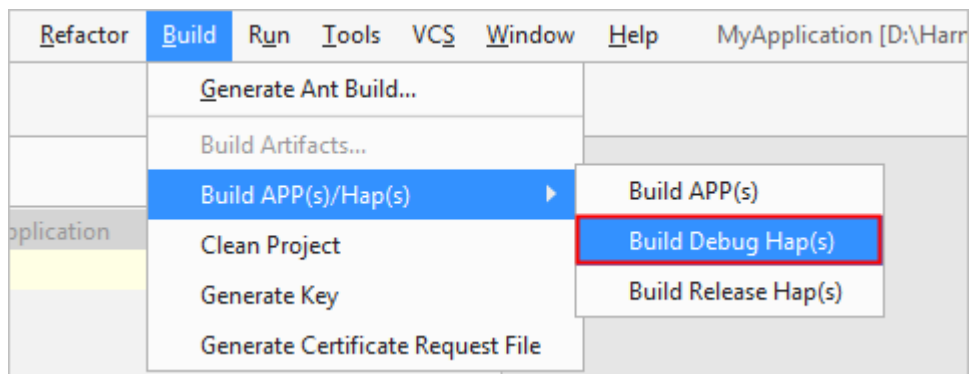
5.4.2 构建类型为 Debug 的 HAP（带调试签名信息）

如果一个工程目录下存在多个 Module，当对单个 Module 进行构建时，只需要对指定的 Module 进行签名；如果对整个工程进行构建，则需要对所有的 Module 进行签名。

1. 打开 **File>Project Structure**，在 **Modules>entry（模块名称）>Signing Configs > debug** 窗口中，配置指定模块的调试签名信息。
 - **Store File**：选择密钥库文件，文件后缀为.p12。
 - **Store Password**：输入密钥库密码。
 - **Key Alias**：输入密钥的别名信息。
 - **Key Password**：输入密钥的密码。
 - **SignAlg**：签名算法，固定为 SHA256withECDSA。
 - **Profile File**：选择申请的调试 Profile 文件，文件后缀为.p7b。
 - **Certpath File**：选择申请的调试数字证书文件，文件后缀为.cer。



- 在主菜单栏，点击 **Build > Build APP(s)/Hap(s) > Build Debug Hap(s)**，生成已签名的 Debug HAP。



5.4.3 构建类型为 Debug 的 HAP（不带签名）

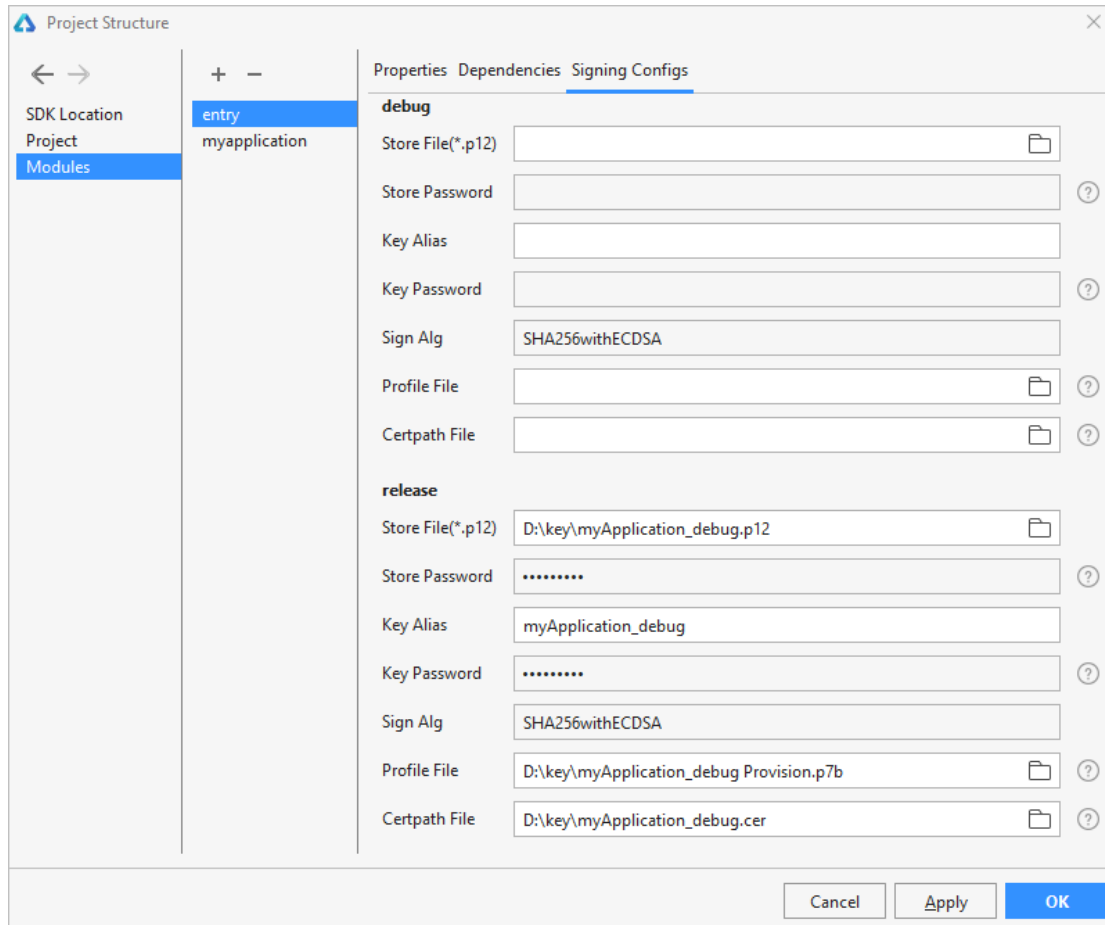
对于构建类型为 Debug 的 HAP，如果没有配置签名参数，则默认不对 HAP 进行签名，该方式生成的 HAP 仅能运行在模拟器上。

在主菜单栏，点击 **Build > Build APP(s)/Hap(s) > Build Debug Hap(s)**，生成不带签名的调试 Debug HAP。

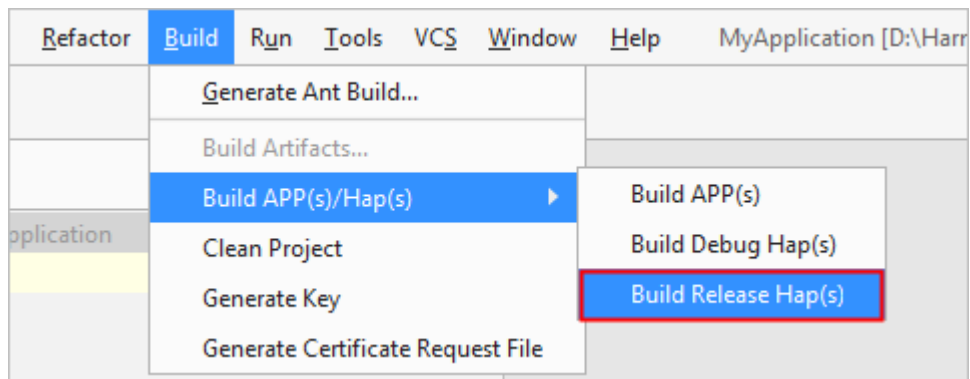
5.4.4 构建类型为 Release 的 HAP（带调试签名信息）

如果一个工程目录下存在多个 Module，当对单个 Module 进行构建时，只需要对指定的 Module 进行签名；如果对整个工程进行构建，则需要对所有的 Module 进行签名。

1. 打开 **File>Project Structure**，在 **Modules>entry（模块名称）>Signing Configs > release** 窗口中，配置指定模块的调试签名信息。
 - **Store File**：选择密钥库文件，文件后缀为.p12。
 - **Store Password**：输入密钥库密码。
 - **Key Alias**：输入密钥的别名信息。
 - **Key Password**：输入密钥的密码。
 - **SignAlg**：签名算法，固定为 SHA256withECDSA。
 - **Profile File**：选择申请的调试 Profile 文件，文件后缀为.p7b。
 - **Certpath File**：选择申请的调试数字证书文件，文件后缀为.cer。



2. 在主菜单栏，点击 **Build > Build APP(s)/Hap(s) > Build Release Hap(s)**，生成已签名的 Release HAP。



5.4.5 构建类型为 Release 的 HAP（不带签名）

对于构建类型为 Release 的 HAP，如果没有配置签名参数，则默认不对 HAP 进行签名，该方式生成的 HAP 仅能运行在模拟器上。

在主菜单栏，点击 **Build > Build APP(s)/Hap(s) > Build Release Hap(s)**，生成不带签名的调试 Release HAP。

6 应用运行

6.1 使用模拟器运行应用

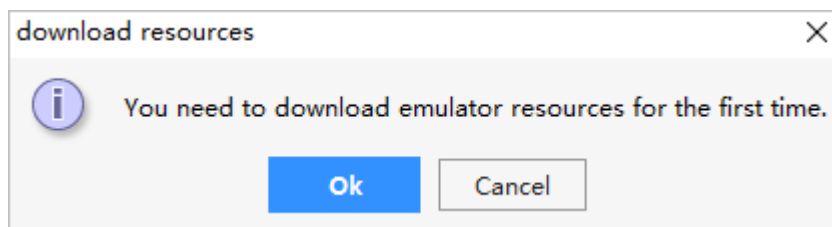
DevEco Studio 提供远程模拟器（**Remote Emulator**）功能，可以将开发的 TV 和 Wearable 应用运行在模拟器上。在模拟器上运行应用不需要签名。

模拟器每次使用时长为 1 小时，到期后模拟器会自动释放资源，请及时完成 HarmonyOS 应用的调试。如果模拟器到期释放后，需重新申请模拟器资源。

说明

Lite Wearable 暂不支持在模拟器中运行，可以选择预览器运行和调试应用，具体请参见使用预览器查看应用效果。

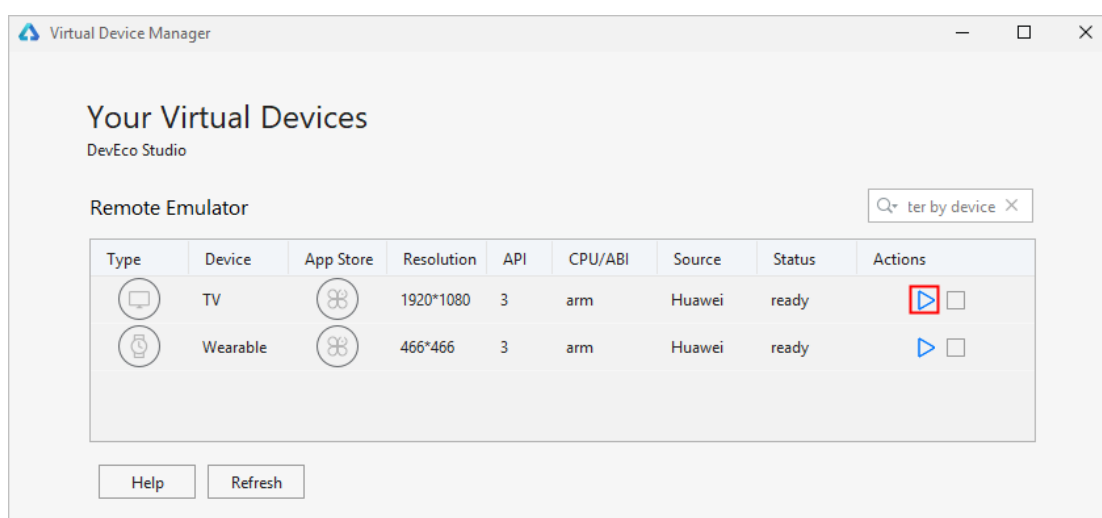
1. 在 DevEco Studio 菜单栏，点击 **Tools > HVD Manager**。首次使用模拟器，请点击 **OK** 按钮下载模拟器相关资源。



2. 在浏览器中弹出华为帐号登录界面，请输入已实名认证的华为帐号的用户名和密码进行登录。
3. 登录后，请点击界面的**允许**按钮进行授权。



4. 点击已经连接的远程模拟设备运行按钮，启动远程模拟设备（同一时间只能启动一个设备）。



5. 点击 DevEco Studio 的 **Run > Run'模块名称'**或，或使用默认快捷键 **Shift+F10**。
6. 在弹出的 Select Deployment Target 界面选择 **Connected Devices**，点击 **OK** 按钮。
7. DevEco Studio 会启动应用的编译构建，完成后应用即可运行在 **Remote Device** 上。



模拟器侧边栏按钮作用：

：释放当前正在使用的模拟器，每台模拟器单次使用时长为 1 小时。

：设置模拟器分辨率。

：返回模拟器主界面。

：后退按钮。

6.2 使用真机设备运行应用

6.2.1 在 TV 中运行应用

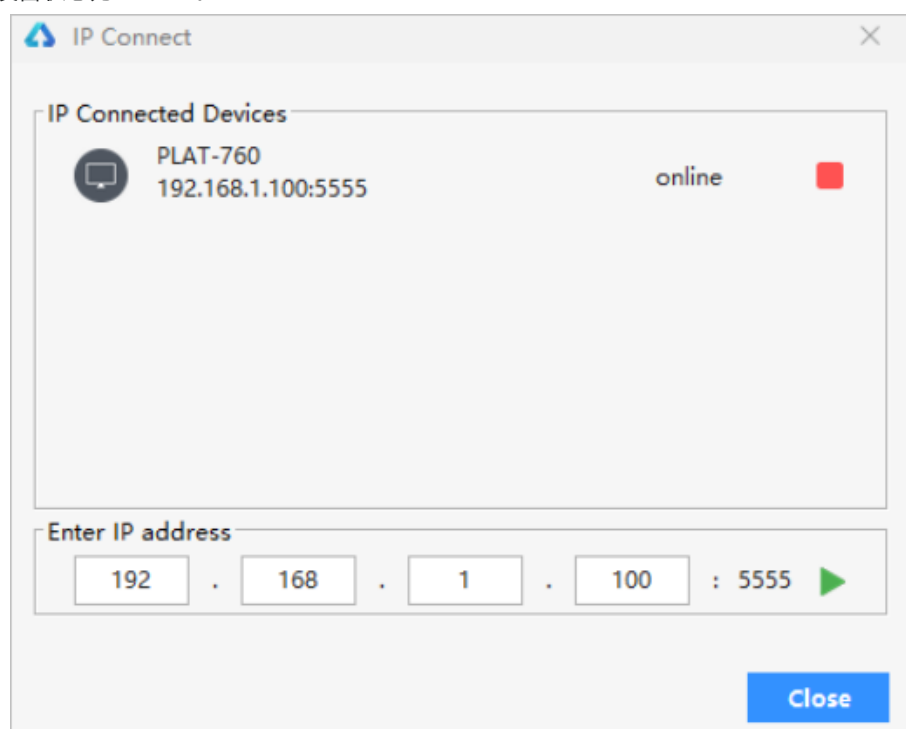
在 TV（智慧屏）中安装和运行 HarmonyOS 应用，采用 **IP Connect** 的连接方式。该连接方式要求 TV 和 PC 端在同一个网段，建议将 TV 和 PC 连接到同一个 WLAN 下；如果采用网线连接，需要手动设置 PC 和 TV 的本地 IP 地址。

6.2.1.1 前提条件

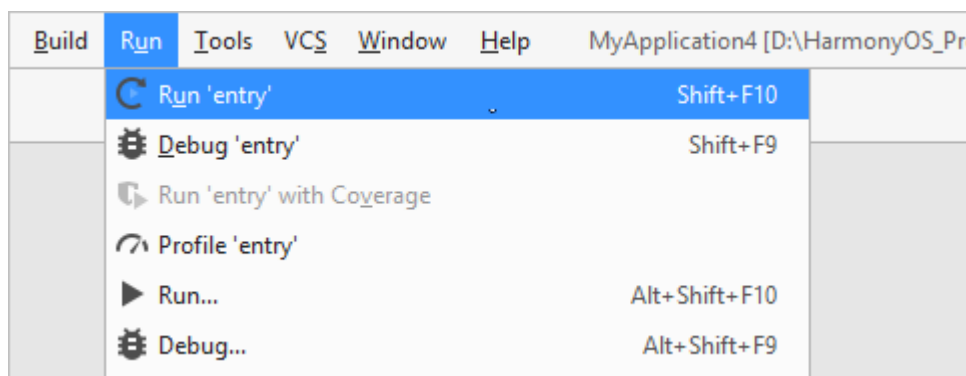
- 已将 TV 和 PC 连接到同一网络或设置为同一个网段。
- 已获取 TV 端的 IP 地址。
- 在 TV 中运行应用，需要提前根据编译构建生成 HAP 完成 HAP 的签名配置。

6.2.1.2 操作步骤

1. 在 DevEco Studio 菜单栏中，点击 **Tools>IP Connect**，输入连接设备的 IP 地址，点击，连接正常后，设备状态为 **online**。



- 在菜单栏中，点击 **Run>Run'模块名称'**或，或使用默认快捷键 **Shift+F10** 运行应用。



- 在弹出的界面，选择已连接的 TV 设备，点击 **OK** 按钮。
- DevEco Studio 启动 HAP 的编译构建和安装。安装成功后，点击 TV 桌面上的应用图标，运行 HarmonyOS 应用。

6.2.2 在 Wearable 中运行应用

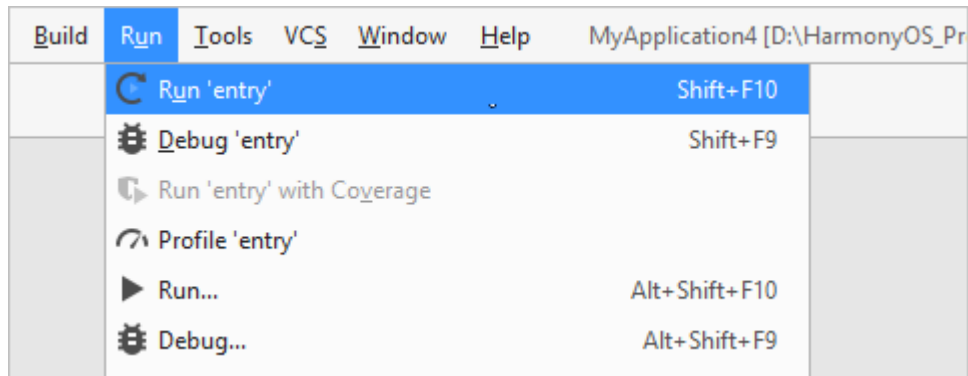
在 Wearable 中安装和运行 HarmonyOS 应用，采用 **USB** 连接或采用 **WLAN** 的连接方式。

6.2.2.1 前提条件

在 Wearable 中运行应用，需要提前根据编译构建生成 HAP 章节，完成 HAP 的签名配置。

6.2.2.2 采用 USB 连接安装应用

- 使用 USB 方式，连接 Wearable 和 PC 端。
- 在菜单栏中，点击 **Run>Run'模块名称'**或，或使用默认快捷键 **Shift+F10** 运行应用。



3. 在弹出的界面，选择已连接的 Wearable 设备，点击 **OK** 按钮。
4. DevEco Studio 启动 HAP 的编译构建和安装。安装成功后，点击 Wearable 中的应用图标，运行 HarmonyOS 应用。

6.2.2.3 采用 WLAN 连接安装应用

方法与在 TV 中运行应用类似，只是连接的设备为 Wearable。

6.2.3 在 Lite Wearable 中运行应用

Lite Wearable 的 HarmonyOS 应用安装，依赖华为手机上的**运动健康**和**应用调测助手** APP 辅助进行。

6.2.3.1 前提条件

- 已将**运动健康 APP**升级至最新版本。
- 从华为应用市场安装**应用调测助手 APP**。
- 在 Lite Wearable 中运行应用，需要提前根据编译构建生成 HAP 完成 HAP 的签名配置。

6.2.3.2 操作步骤

1. 使用 USB 连接线将手机和电脑进行连接，确保连接状态是正常的。
2. 手机与电脑使用 USB 连接时，在手机上选择**传输文件**连接方式。

3. 在工程目录中的 **Build > outputs > hap** 中选择生成的 HAP，通过手工拷贝的方式将 HAP 拷贝至手机中的 `/sdcard/haps/` 目录。

说明

如果在手机存储根目录下没有“hps”文件夹，请手工创建后再拷贝 HAP 到该文件夹下。

4. 将 Lite Wearable 通过蓝牙与华为手机进行连接。
 - a. 进入**运动健康** APP，在设备页签中，点击**添加设备**按钮。



- b. 进入**手表**列表中，选择对应的 Lite Wearable 型号。
 - c. 点击**开始配对**，按照界面指引完成 Lite Wearable 与华为手机之间的连接。
5. 打开**应用调测助手** APP，界面会显示已经与华为手机连接的 Lite Wearable。

说明

如果 Lite Wearable 与华为手机未连接，请点击**应用调测助手** APP 界面的**连接设备**按钮，手机会自动打开**运动健康** APP 添加 Lite Wearable。

6. 点击**应用调测助手** APP 界面中的**安装手表应用**按钮，选择需要安装的 HarmonyOS 安装包进行安装。
7. 安装完成后，点击 Lite Wearable 中的应用图标，运行 HarmonyOS 应用。

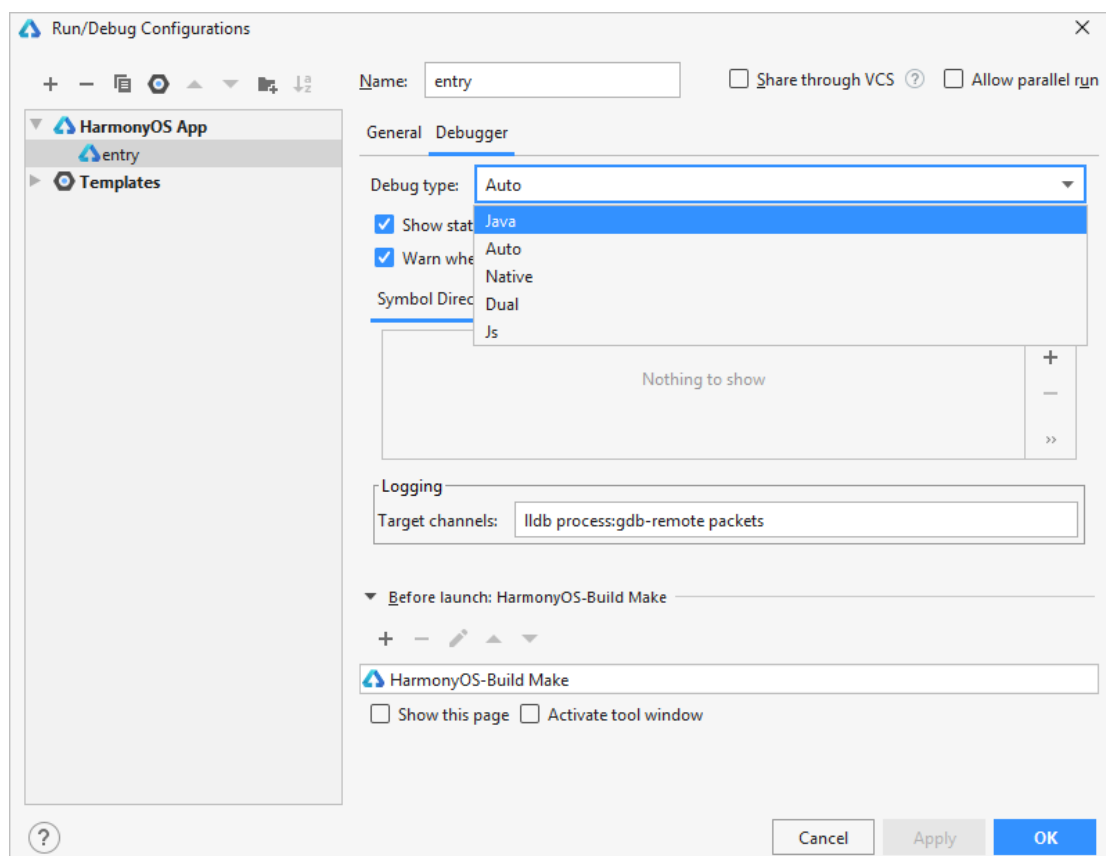
7 应用调试

7.1 基本调试操作

DevEco Studio 提供了基于各种编写代码及不同设备的调试功能，如果使用了多种代码编写应用，请参考选择调试代码类型进行配置后启动调试，调试过程中基于不同的代码进行断点管理。

7.1.1 选择调试代码类型

点击 **Run > Edit Configurations > Debugger**，在 **HarmonyOS App** 中，选择相应模块， 可以进行 Java/JS/C++ 调试配置。



调试类型	调试代码
Java	Java
Auto	Java C/C++ 根据代码自动匹配调试类型
Native	C/C++
Dual	C/C++ Java 同时调试 C/C++ 代码与 Java 代码
Js	JavaScript

表 1 Java/Auto/Native/Dual/Js 调试类型配置项

- 对于 TV 和 Wearable 设备，请根据应用编写的代码来配置调试类型，然后进行调试。
- 对于 Lite Wearable 设备，与调试类型配置无关，可直接进行调试。

7.1.2 启动调试

1. 在工具栏中，点击 **Debug**。
2. 在弹出的界面，选择需要调试的设备。
 - 真实设备：一般为可以用 USB 或 IP 方式连接的实体设备。
 - Remote Device：远程设备模拟器，支持 TV 和 Wearable，请参考使用模拟器运行应用启动连接设备后，方可选择进行调试。
3. 如果需要设置断点调试，则需要选定要设置断点的有效代码行，在行号（比如：24 行）的区域后，单击鼠标左键设置断点（如图示的红点）。

```

1 package com.example.myapplication.slice;
2
3 import ...
4
11
12 public class MainAbilitySlice extends AbilitySlice {
13
14     private PositionLayout myLayout = new PositionLayout( context: this);
15
16     @Override
17     public void onStart(Intent intent) {
18         super.onStart(intent);
19         LayoutConfig config = new LayoutConfig(LayoutConfig.MATCH_PARENT, LayoutConfig.MATCH_PARENT);
20         myLayout.setLayoutConfig(config);
21         ShapeElement element = new ShapeElement();
22         element.setShape(ShapeElement.RECTANGLE);
23         element.setRgbColor(new RgbColor( red: 255, green: 255, blue: 255));
24         myLayout.setBackground(element);
25
26         Text text = new Text( context: this);
27         text.setText("Hello World");
28         text.setTextColor(Color.BLACK);
29         myLayout.addComponent(text);
30         super.setUIContent(myLayout);
31     }

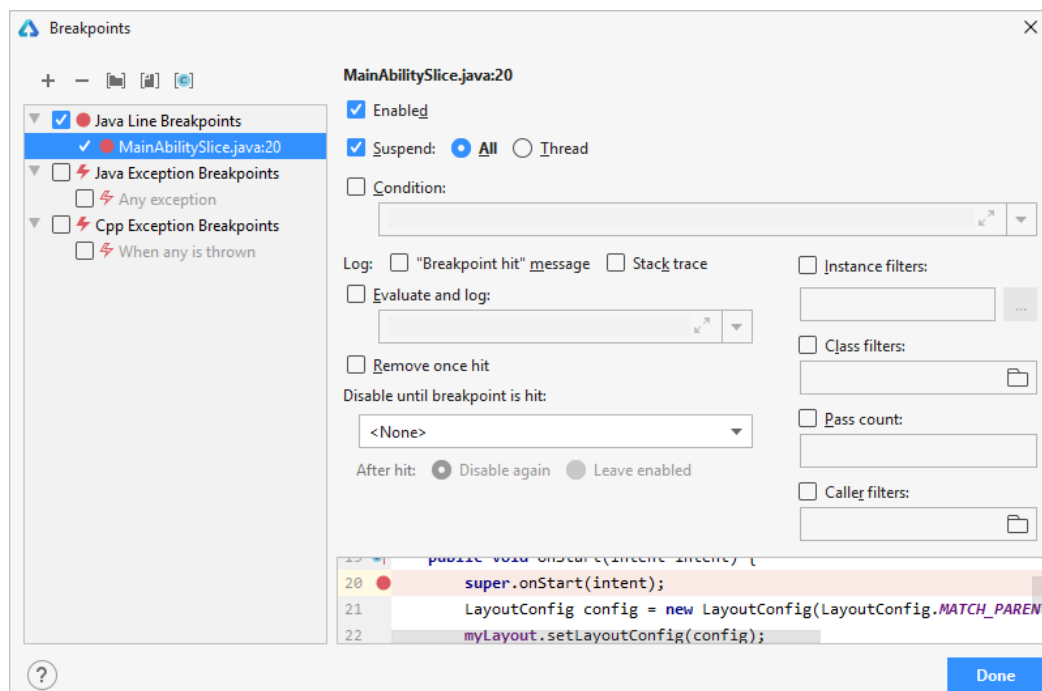
```

设置断点后，调试能够在正确的断点处中断，并高亮显示该行。

7.1.3 断点管理

在设置的程序断点红点处，点击鼠标右键，然后点击 **More**（或按快捷键

Ctrl+Shift+F8），可以管理断点。



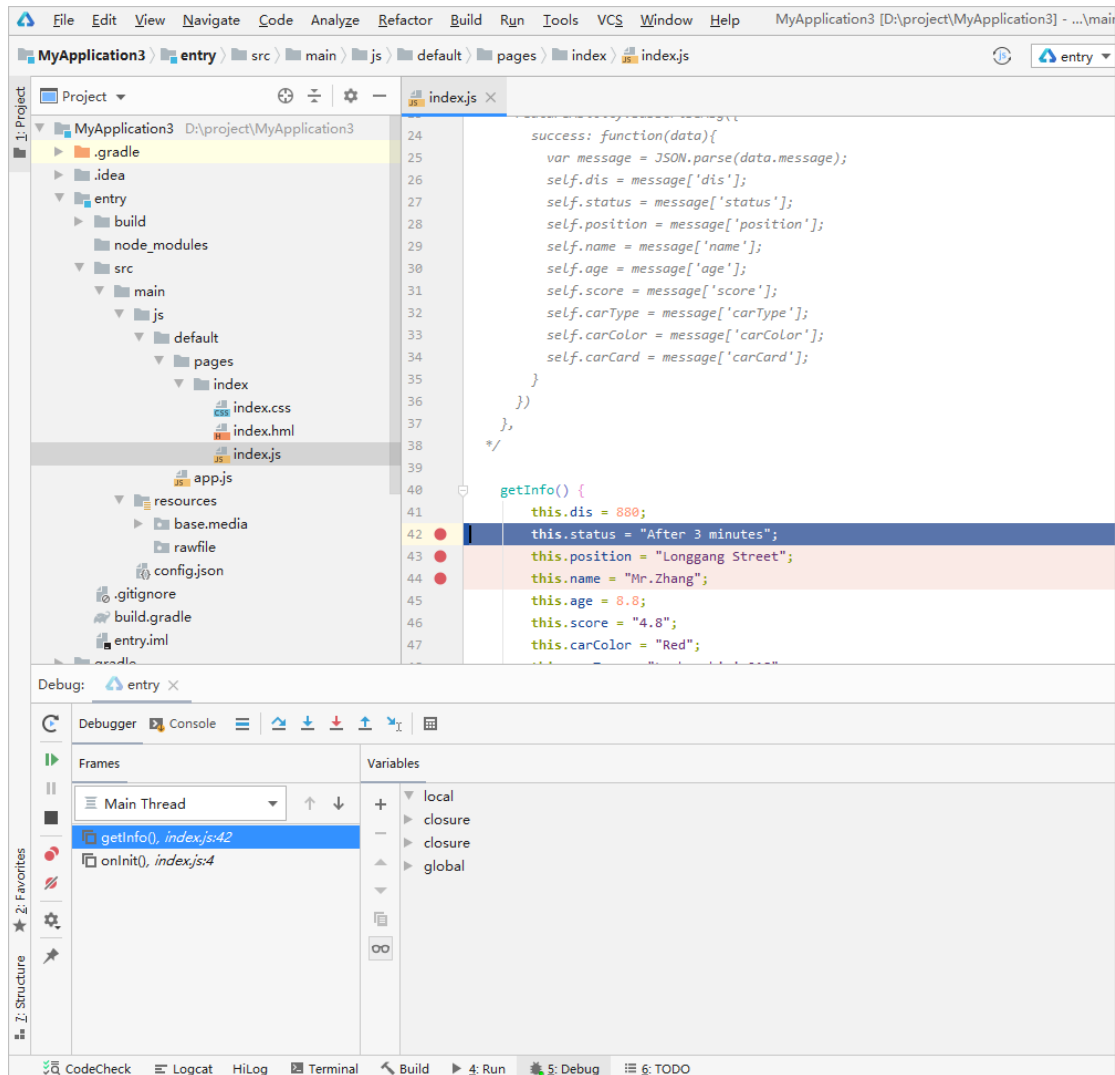
代码类型	断点管理
JS (JavaScript)	普通行断点
Java	普通行断点 Exception (异常) 断点
C/C++	普通行断点 Exception (异常) 断点 Symbolic (符号) 断点 设置 Watchpoint (仅支持 x86、x86_64 架构)

表 2 不同代码类型的断点管理功能

7.2 各语言调试功能

7.2.1 JS 调试功能

对 JS 进行调试的界面如下：

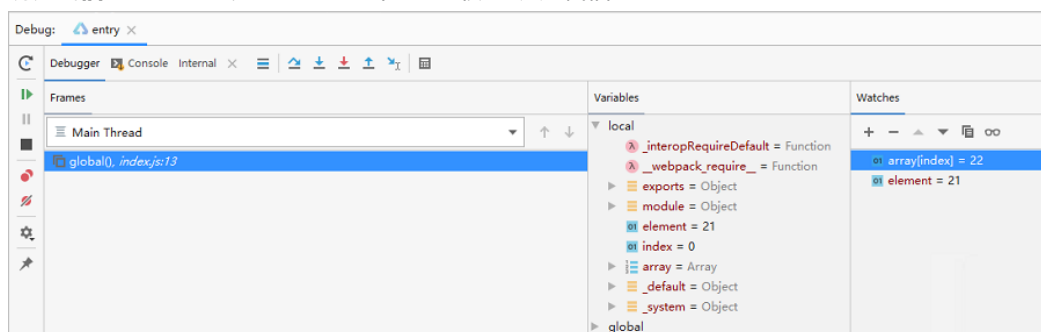


按钮	名称	快捷键	功能
	Resume Program	F9	当程序执行到断点时停止执行，点击此按钮程序继续执行。
	Step Over	F8	在单步调试时，直接前进到下一行（如果在函数中存在子函数时，不会进入子函数内单步执行，而是将整个子函数当作一步执行）。
	Step Into	F7	在单步调试时，遇到子函数后，进入子函数并继续单步执行。

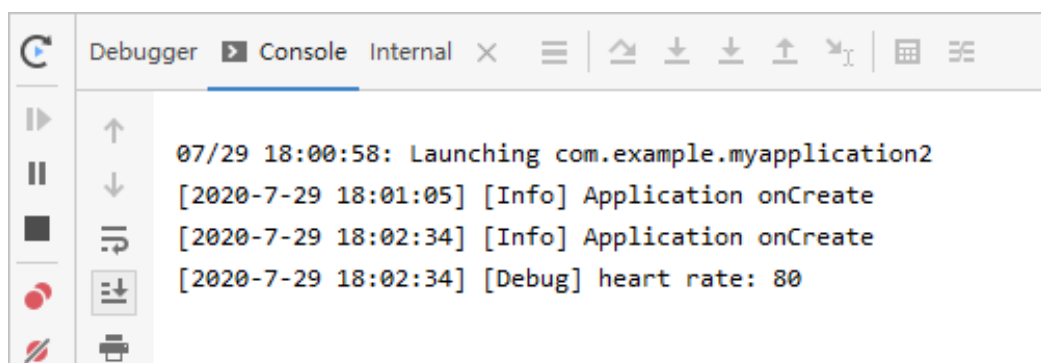
按钮	名称	快捷键	功能
	Force Step Into	Alt+Shift+F7	在单步调试时，强制下一步。
	Step Out	Shift+F8	在单步调试执行到子函数内时，点击 Step Out 会执行完子函数剩余部分，并跳出返回到上一层函数。
	Rerun	Ctrl+F5	重新启动调试。
	Stop	Ctrl+F2	停止调试任务。
	Run To Cursor	-	断点执行到鼠标停留处，仅 TV、Wearable 支持。

表 1 调试器按钮

- 常用的调试功能：
- **变量值查看：**在调试过程中，可以通过调试侧边栏中的 **Variables** 查看已执行程序中包含的变量的当前取值。
- **变量监控：**也可以在 **Watches** 中添加关注的变量，对添加的变量进行监控。
- **调用栈信息查看：**可以在 **Frames** 中查看函数的调用栈信息。



- **调试日志打印：**调试控制台 **Console** 可以打印调试的日志信息。



7.2.2 Java 调试功能

- 通过 **Attach Debugger to Process** 选择进程进行调试，能根据调试类型，在已运行应用的设备上，自动进入相应的调试模式。



- 具备 Step Into, Step Out, Step Over, Force Step Into, Rerun、Run To Cursor 等基本调试能力，详细描述请参考表 调试器按钮。
- 支持 Inline Values，即编辑器显示变量值。
- 调试中断后，能够恢复执行。

7.2.3 C/C++ 调试功能

- 通过 **Attach Debugger to Process** 选择进程进行调试，能根据调试类型，在已运行应用的设备上，自动进入相应的调试模式。



- Native 类型调试器，能启动 Debug Session 和 LLDB Server 运行调试。
- 具备 Step Into, Step Out, Step Over, Force Step Into, Rerun、Run To Cursor 等基本调试能力，详细描述请参考表 1。
- 支持 Force Step Over。
- 支持 Inline Values，即编辑器显示变量值。
- 调试中断后，能够恢复执行。
- LLDB 命令控制台：
 - 支持使用 LLDB 命令自助调试
 - 支持 UI 调试按钮/快捷键多指令输入

■

8 应用发布

8.1 编译构建生成 APP

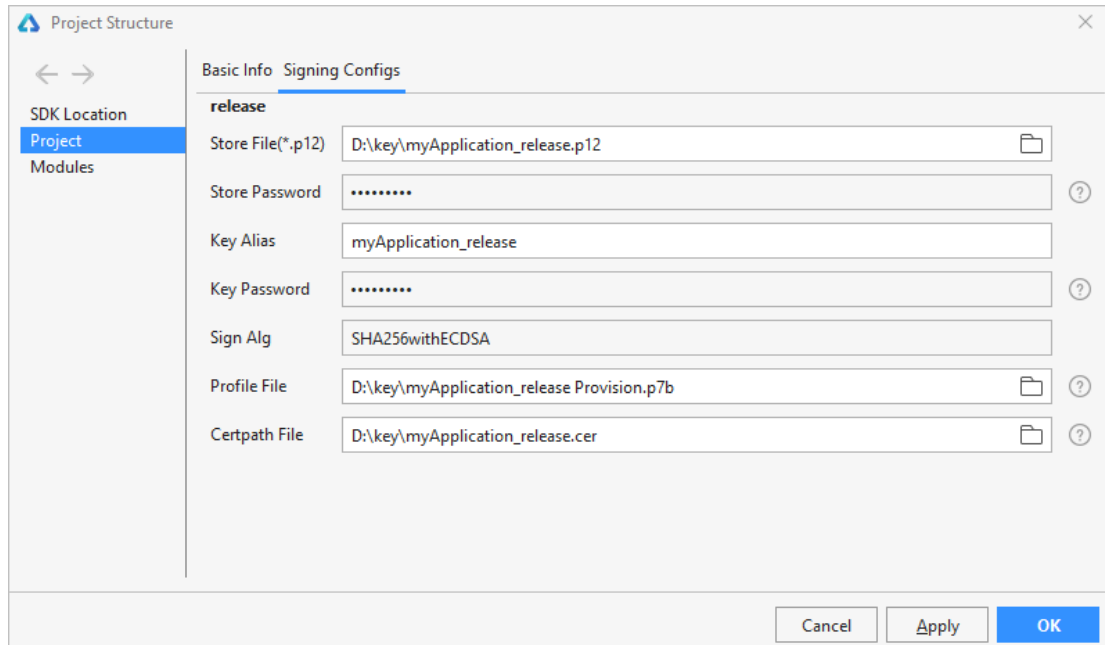
开发者完成 HarmonyOS 应用开发后，需要将应用打包成 APP，用于发布到华为应用市场。打包 APP 时，DevEco Studio 会将工程目录下的所有 HAP 模块打包到 APP 中，因此，如果工程目录中存在不需要打包到 APP 的 HAP 模块，请手动删除后再进行编译构建生成 APP。

8.1.1 前提条件

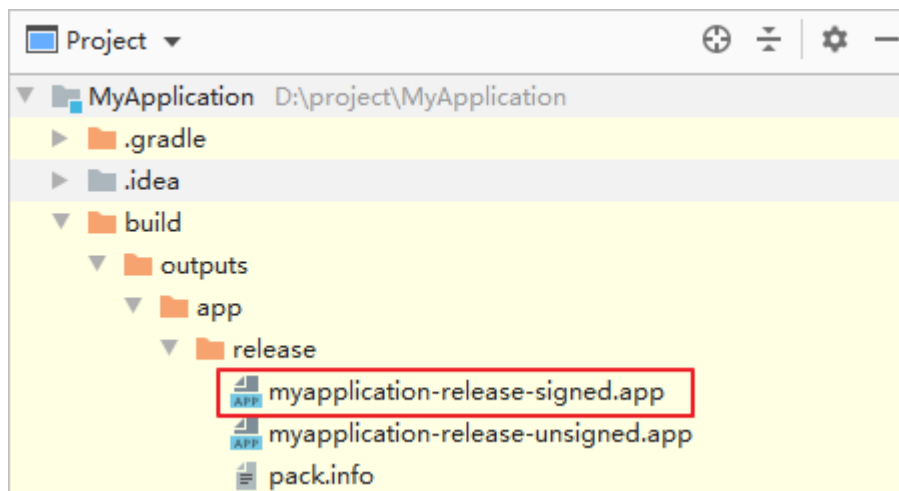
- 已完成发布证书和 Profile 文件的申请，详情请参考申请证书和 Profile。
- 已完成 build.gradle 和 config.json 的设置，详情请参考编译构建前配置。

8.1.2 操作步骤

1. 在 **Project Structure > Project > Signing Configs** 窗口中，配置工程的签名信息，设置完成后，点击 **OK** 按钮。
 - **Store File**：选择密钥库文件，文件后缀为.p12。
 - **Store Password**：输入密钥库密码。
 - **Key Alias**：输入密钥的别名信息。
 - **Key Password**：输入密钥的密码。
 - **SignAlg**：签名算法，固定为 SHA256withECDSA。
 - **Profile File**：选择申请的发布 Profile 文件，文件后缀为.p7b。
 - **Certpath File**：选择申请的发布数字证书文件，文件后缀为.cer。



2. 点击 **Build > Build APP(s)/Hap(s) > Build APP(s)**，等待编译构建完成已签名的 APP。
3. 编译构建完成后，可以在 **build > outputs > app > release** 目录下，获取带签名的 APP。



8.2 上架华为应用市场

将 HarmonyOS 应用打包成 APP 后，通过 AppGallery Connect 将 HarmonyOS 应用分发到不同的设备上。您可以根据发布 HarmonyOS 应用指导将 APP 上架到华为应用市场。