2025 年夏季《移动软件开发》实验报告

姓名和学号？	**聂宇航，23170001072**
本实验属于哪门课程？	中国海洋大学 25 夏《移动软件开发》
实验名称？	实验 5：第一个 HarmonyOS 应用
博客地址？	《移动软件开发》第一个 HarmonySO 应用-CSDN 博客
Github 仓库地址？	这个是《移动软件开发》这门课的实验代码与报告

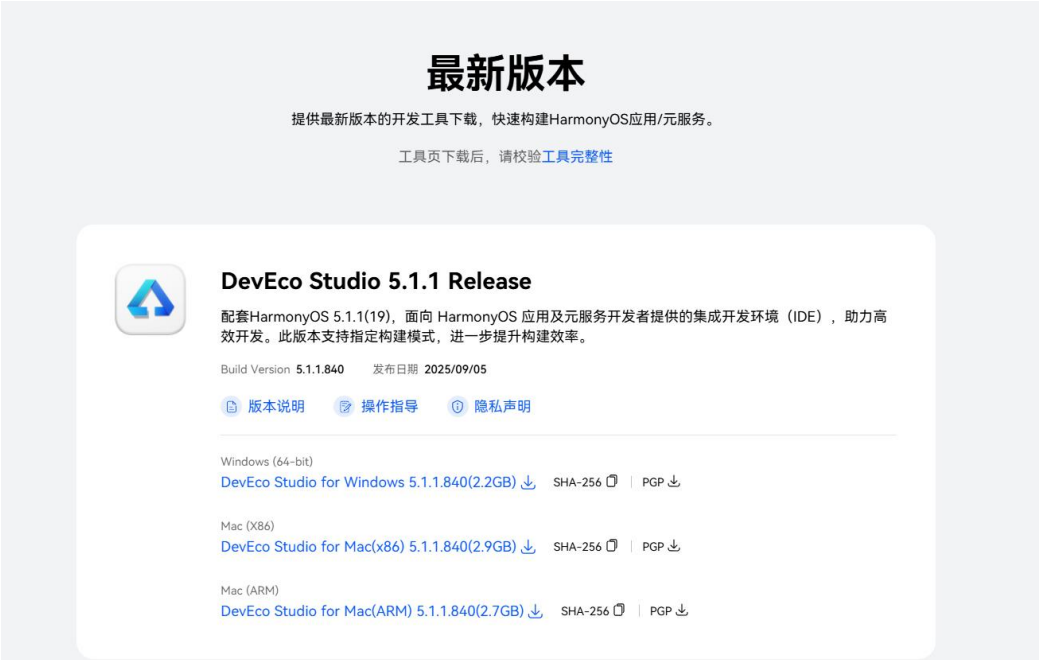
（备注：将实验报告发布在博客、代码公开至 github 是 加分项，不是必须做的）

一、实验目标

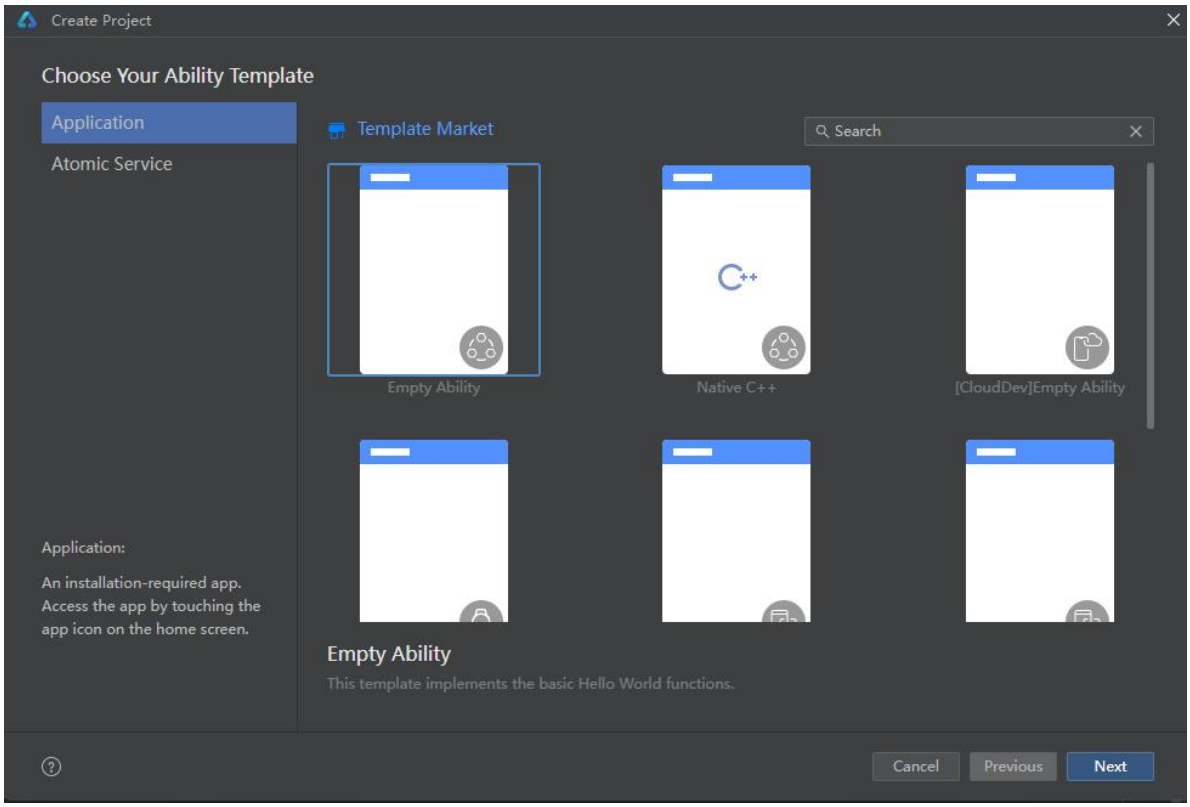
通过这一部分内容的学习和初步实践，开发者可以快速构建出首个 HarmonyOS 应用，掌握应用程序包结构、资源文件的使用以及 ArkTS 的核心功能和语法等基础知识，为后续的应用开发奠定基础。

二、实验步骤

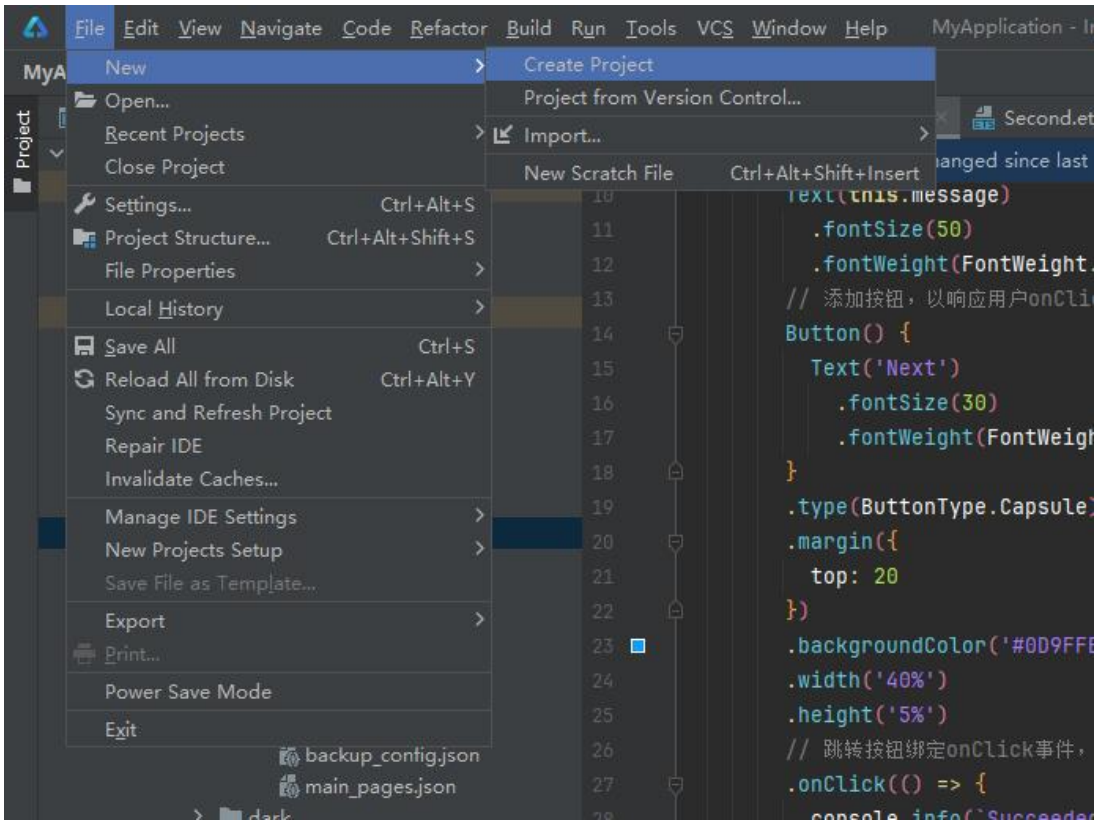
在最新版本 - 下载中心 - 华为开发者联盟中下载最新版的 DevEco Studio



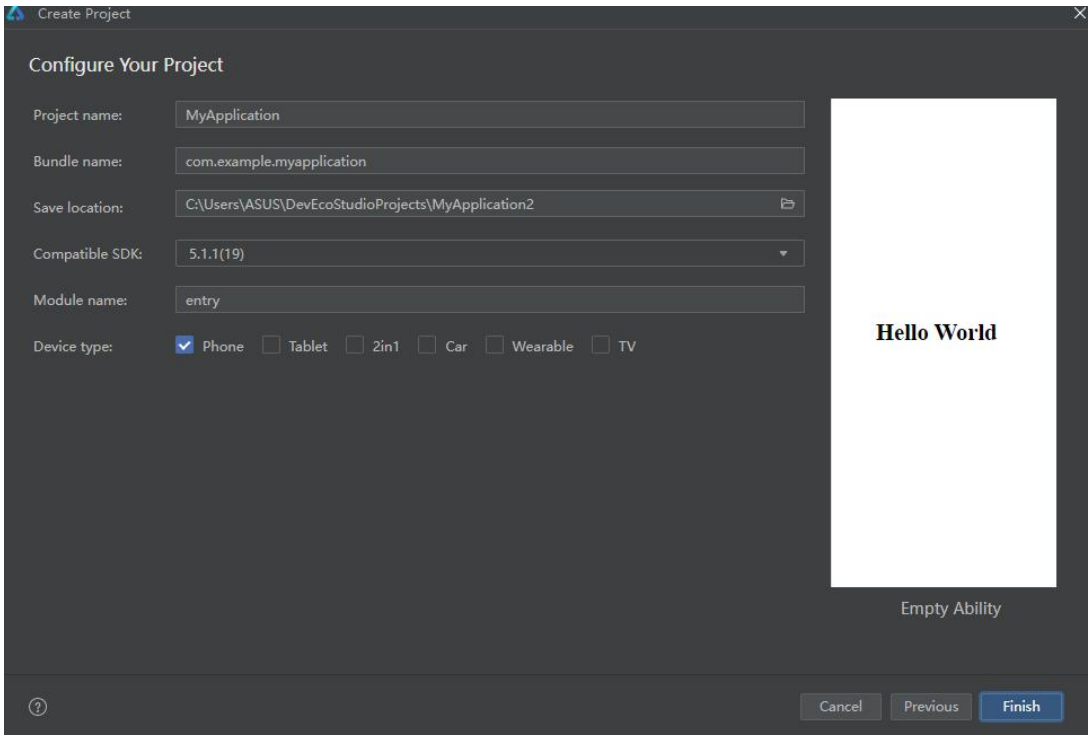
首次打开 DevEco Studio，可以直接在 Application 里选择 Empty Ability 创建工程。



如果已经打开了一个工程，可以在菜单栏选择 File > New > Create Project 来创建一个新工程



选择 Application 应用开发（本文以应用开发为例，Atomic Service 对应为元服务开发），选择模板 Empty Ability，单击 Next 进行下一步配置



从上到下分别是：

Project name（项目名称）

这是整个项目的名字，通常对应于 IDE 中看到的项目工程名。

Bundle name（应用包名）

相当于应用的唯一标识符（类似 Android 的 `packageName`）。

一般遵循 反域名命名规则，例如 b 站：`tv.danmaku.bili`，京东 `com.jingdong.app.mall`。

在系统安装、应用市场上架时会用到。

Save location（保存路径）

你在本地磁盘上存放该项目工程的路径。

Compatible SDK（兼容 SDK 版本）

表示这个项目依赖的 HarmonyOS SDK 版本，决定了项目能调用哪些 API。

通常要选择和目标设备一致或兼容的版本。

Module name（模块名称）

HarmonyOS 项目是 多模块架构的，一个项目可以包含多个模块。

每个模块可以是 应用模块（entry） 或 库模块（feature、har、shared）。

这里填写的是模块名，比如 `entry`（主入口模块）。

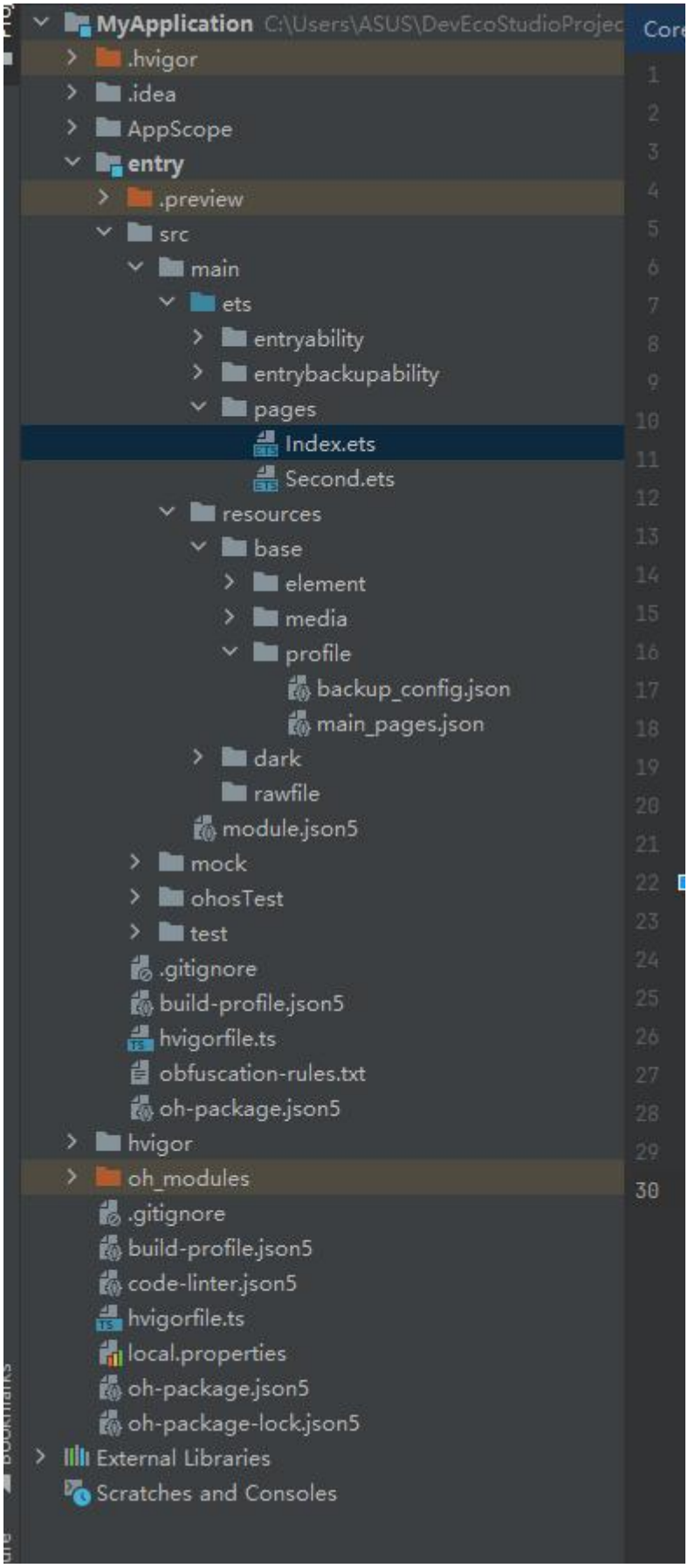
Device type（设备类型）

指定你的应用目标运行的设备类型。

常见选项有：手机（phone）、平板（tablet）、穿戴设备（wearable）、智能家居（liteWearable / tv / car 等）。

决定了应用可以运行在哪些设备上。

在项目左边有个 ArkTS 工程目录结构（Stage 模型）


MyApplication C:\Users\ASUS\DevEcoStudioProjec
.hvigor
.idea
AppScope
entry
.preview
src
main
ets
entryability
entrybackupability
pages
Index.ets
Second.ets
resources
base
element
media
profile
backup_config.json
main_pages.json
dark
rawfile
module.json5
mock
ohosTest
test
.gitignore
build-profile.json5
hvigorfile.ts
obfuscation-rules.txt
oh-package.json5
hvigor
oh_modules
.gitignore
build-profile.json5
code-linter.json5
hvigorfile.ts
local.properties
oh-package.json5
oh-package-lock.json5
External Libraries
Scratches and Consoles

AppScope > app.json5：应用的全局配置信息，详见 app.json5 配置文件。

entry：HarmonyOS 工程模块，编译构建生成一个 HAP 包。

src > main > ets：用于存放 ArkTS 源码。

src > main > ets > entryability：应用/服务的入口。

src > main > ets > entrybackupability：应用提供扩展的备份恢复能力。

src > main > ets > pages：应用/服务包含的页面。

src > main > resources：用于存放应用/服务所用到的资源文件，如图形、多媒体、字符串、布局文件 等。关于资源文件，详见 资源分类与访问。

src > main > module.json5： 模块配置文件。主要包含 HAP 包的配置信息、应用/服务在具体设备上的 配置信息以及应用/服务的全局配置信息。具体的配置文件说明，详见 module.json5 配置文件。

build-profile.json5：当前的模块信息 、编译信息配置项，包括 buildOption、targets 配置等。

hvigorfile.ts：模块级编译构建任务脚本。

obfuscation-rules.txt：混淆规则文件。混淆开启后，在使用 Release 模式进行编译时，会对代码进行 编译、混淆及压缩处理，保护代码资产。详见 开启代码混淆。

oh-package.json5：用来描述包名、版本、入口文件（类型声明文件）和依赖项等信息。

oh_modules：用于存放三方库依赖信息。

build-profile.json5：工程级配置信息，包括签名 signingConfigs、产品配置 products 等。其中 products 中可 配置当前运行环境，默认为 HarmonyOS。

hvigorfile.ts：工程级编译构建任务脚本。

oh-package.json5：主要用来描述全局配置，如：依赖覆盖（overrides）、依赖关系重写 （overrideDependencyMap）和参数化配置（parameterFile）等。

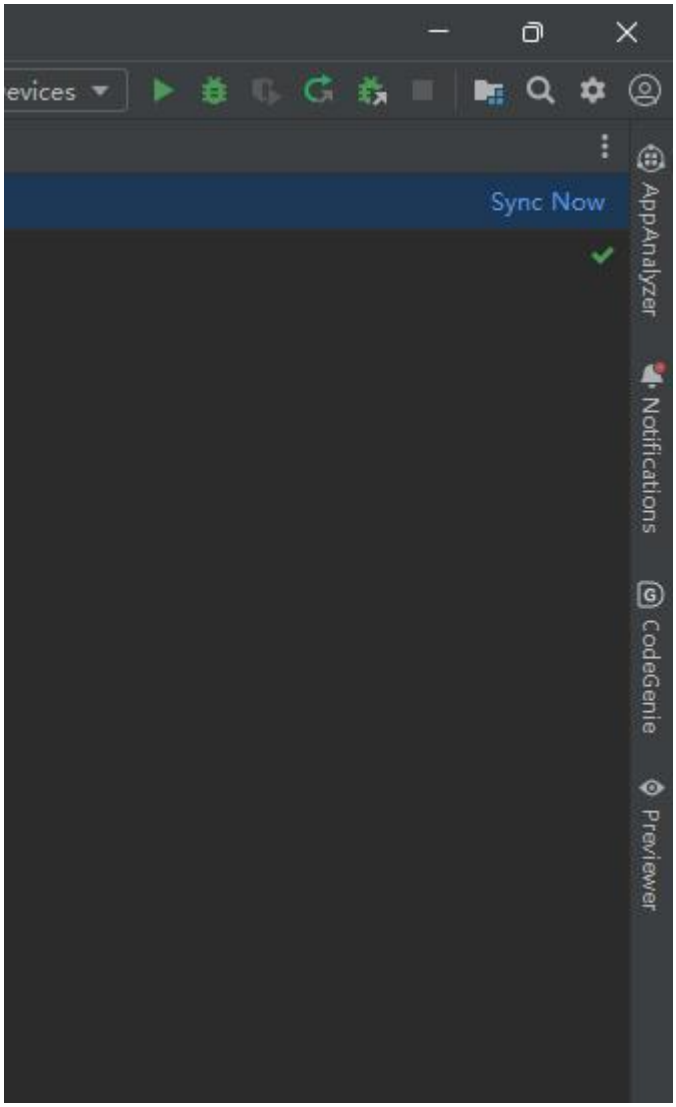
构建第一个文件

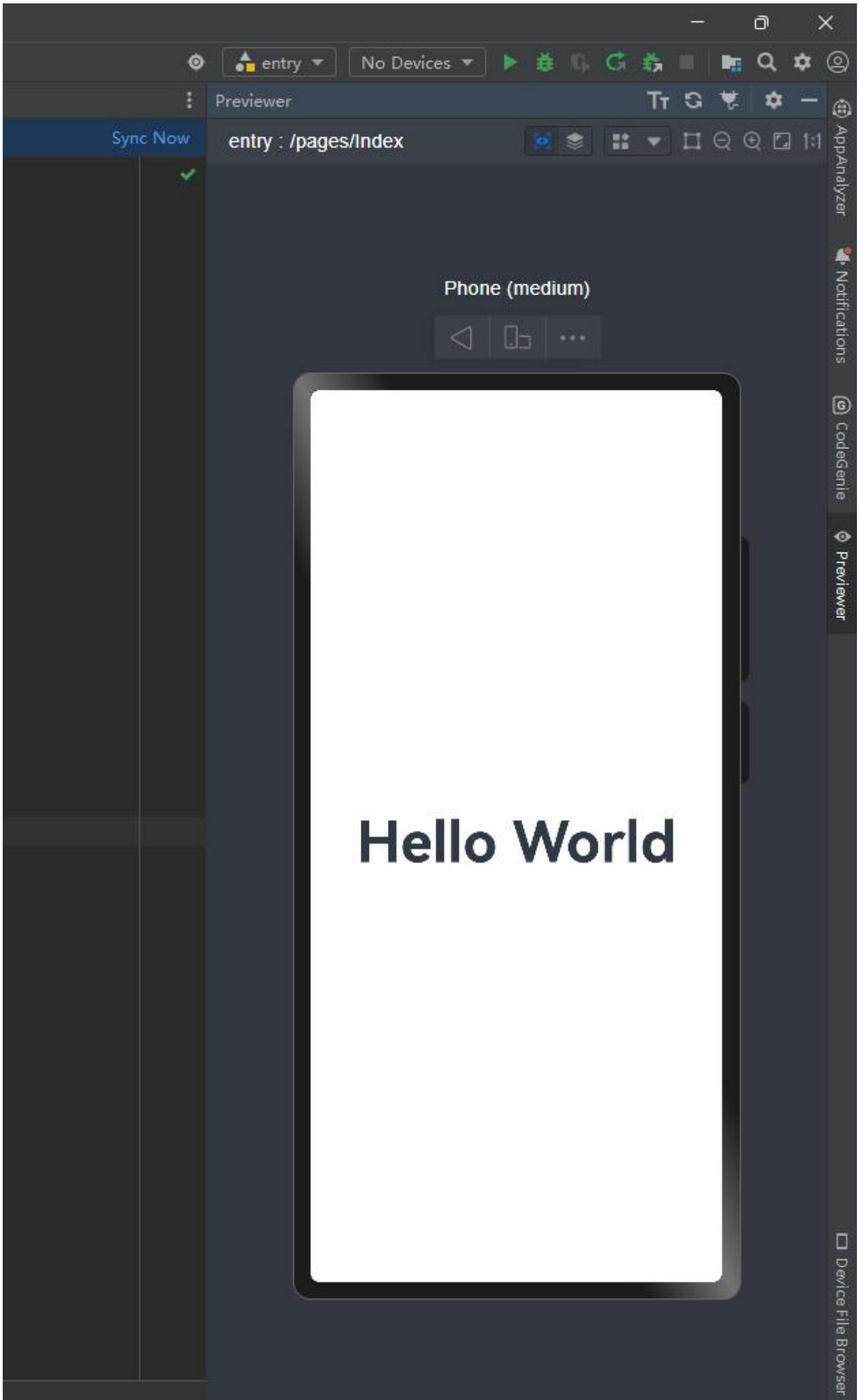
使用文本组件。 工程同步完成后，在 Project 窗口，单击 entry > src > main > ets > pages， 打开 Index.ets 文件， 将页面从 RelativeContainer 相对布局修改成 Row/Column 线性布局。 针对本文中使用文本/按钮来实现页面跳转/返回的应用场景，页面均使用 Row 和 Column 组件来组建布局。对于 更多复杂元素对齐的场景，可选择使用 RelativeContainer 组件进行布局。更多关于 UI 布局的选择和使用，可见 如何选择布局。 Index.ets 文件的示例如下：

```
@Entry

@Component

struct Index {

  @State message: string = 'Hello World';



  build() {

    RelativeContainer() {

      Text(this.message)

        .id('HelloWorld')

        .fontSize($r('app.float.page_text_font_size'))

        .fontWeight(FontWeight.Bold)

        .alignRules({

          center: { anchor: '__container__', align: VerticalAlign.Center },

          middle: { anchor: '__container__', align: HorizontalAlign.Center }

        })

        .onClick(() => {

          this.message = 'Welcome';

        })

    }

    .height('100%')

    .width('100%')

  }
```

```
}
```

点击屏幕右边的 previewer 按钮就可以看到默认的项目是什么样的了




entry
No Devices
Previewer
Sync Now
entry : /pages/Index
Phone (medium)
Hello World
AppAnalyzer
Notifications
CodeGenie
Previewer
Device File Browser

在默认页面基础上，我们添加一个 Button 组件，作为按钮响应用户 onClick 事件，从而实现跳转到另一个页 面。Index.ets 文件的示例如下:

```
// Index.ets
@Entry
@Component
struct Index {
  @State message: string = 'Hello World';
  build() {
    Row() {
      Column() {
        Text(this.message)
          .fontSize(50)
          .fontWeight(FontWeight.Bold)
        // 添加按钮，以响应用户 onClick 事件
        Button() {
          Text('Next')
            .fontSize(30)
            .fontWeight(FontWeight.Bold)
        }
        .type(ButtonType.Capsule)
        .margin({
          top: 20
        })
        .backgroundColor('#0D9FFB')
        .width('40%')
        .height('5%')
```

```
      }
      .width('100%')
    }
    .height('100%')
  }
}
```

代码结构分析

`@Entry` 标记这是应用的入口页面（类似 Android 的 `MainActivity`）。程序运行后首先显示这个页面。

`@Component` 表示这是一个 UI 组件，可以在其他页面中复用。这里 `Index` 是一个组件，同时也是入口。

`struct Index` ArkUI 中组件用 `struct` 定义，类似 React 中的函数组件。

`@State message: string = 'Hello World';` `@State` 是一个响应式变量，UI 会随着它的值变化而自动刷新。这里定义了一个字符串 `message`，初始值为 `"Hello World"`。

UI 构建部分 `build()`

`Row()`水平布局容器，里面的子组件会按行排列。

`.height('100%')`：让行容器撑满屏幕高度。

`Column()`垂直布局容器，里面的子组件按列排列。

`.width('100%')`：让列容器占满行容器的宽度。

子组件

`Text(this.message)`

显示 `"Hello World"`。

`.fontSize(50)`：设置字体大小为 50。

`.fontWeight(FontWeight.Bold)`：设置为粗体。

`Button()`定义一个按钮，按钮内容是一个 `Text("Next")`。

按钮样式：

`.type(ButtonType.Capsule)` → 胶囊形按钮。

`.margin({ top: 20 })` → 顶部外边距 20。

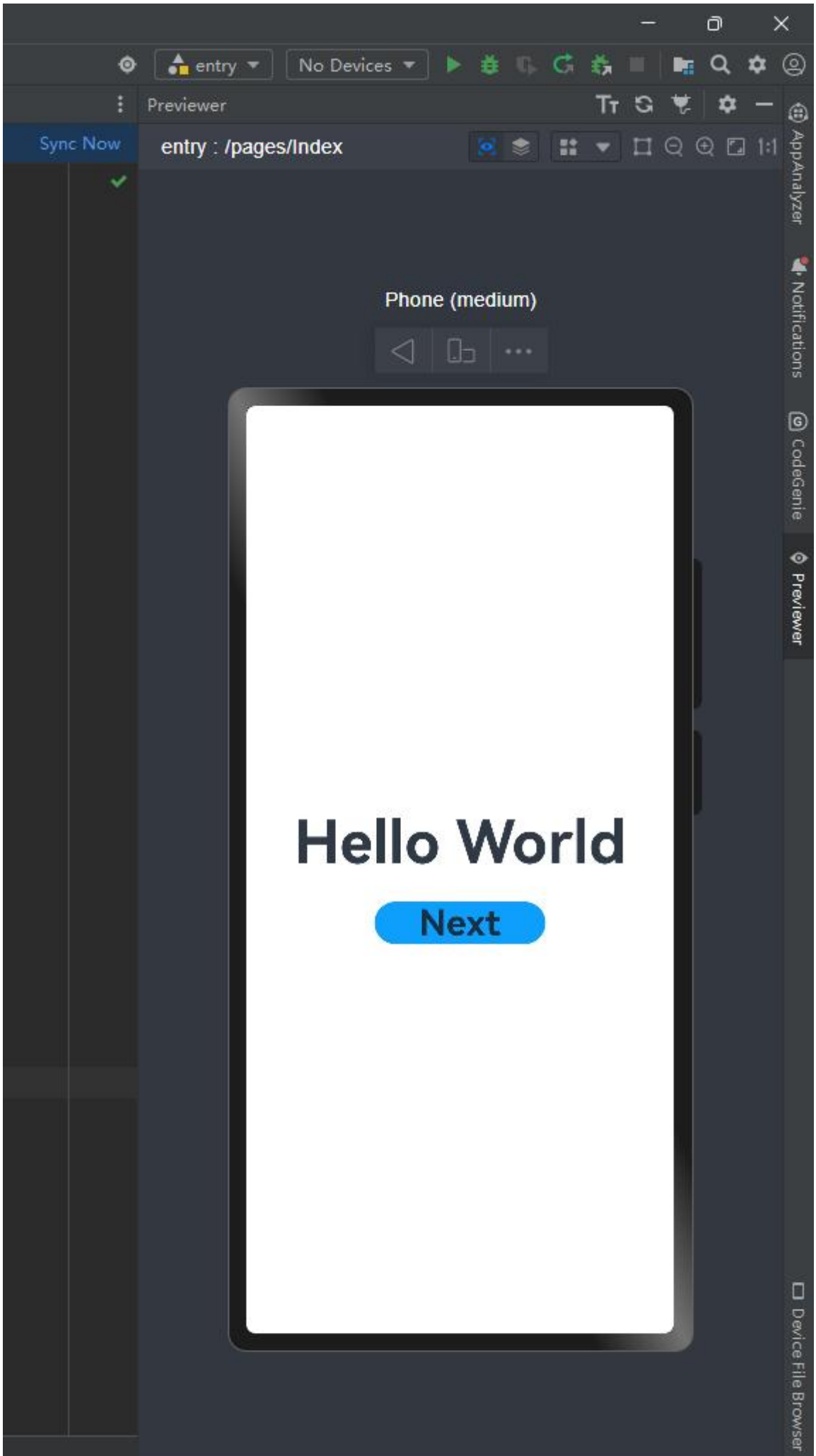
`.backgroundColor('#0D9FFB')` → 蓝色背景。

`.width('40%')` `.height('5%')` → 按钮大小。

按钮内部 `Text('Next')` 字体大小 30，加粗。

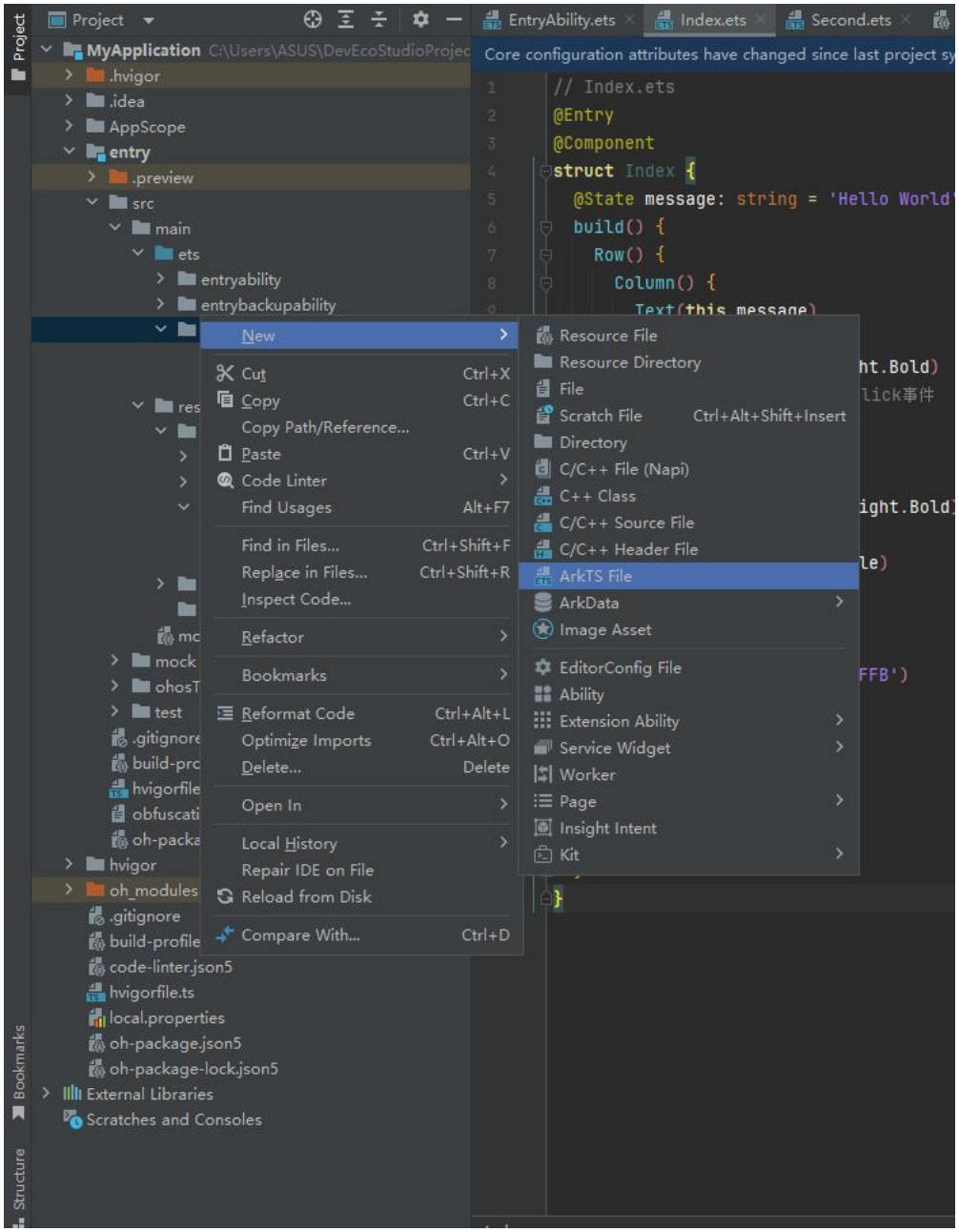
总体效果

顶部显示 “Hello World” 粗体大字。
下方有一个蓝色的 “Next” 按钮，圆角胶囊形。

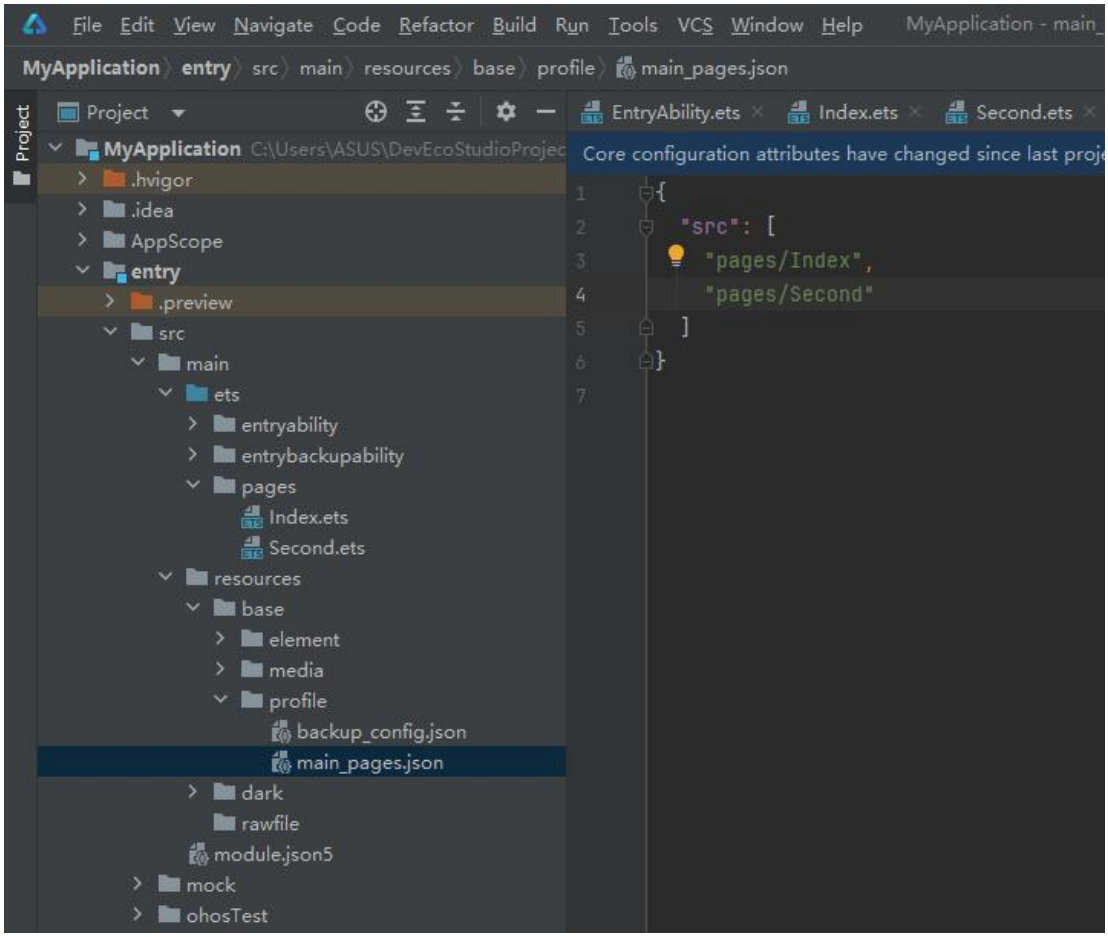

entry
No Devices
Previewer
Sync Now
entry : /pages/Index
Phone (medium)
Hello World
Next
AppAnalyzer
Notifications
CodeGenie
Previewer
Device File Browser

构建第二个页面

创建第二个页面。新建第二个页面文件。在 Project 窗口，打开 entry > src > main > ets，右键单击 pages 文件夹，选择 New > ArkTS File，命名为 Second



配置第二个页面的路由。在 Project 窗口，打开 entry > src > main > resources > base > profile，在 main_pages.json 文件中的"src"下配置第二个页面的路由"pages/Second"。



参照第一个页面，在第二个页面添加 Text 组件、Button 组件等，并设置其样式。Second.ets 文件的示例如下：

```
// Second.ets

import { BusinessError } from '@kit.BasicServicesKit';

@Entry

@Component

struct Second {

  @State message: string = 'Hi there';

  build() {

    Row() {

      Column() {

        Text(this.message)

          .fontSize(50)
```

```
            .fontWeight(FontWeight.Bold)
          Button() {
            Text('Back')
              .fontSize(30)
              .fontWeight(FontWeight.Bold)
          }
          .type(ButtonType.Capsule)
          .margin({
            top: 20
          })
          .backgroundColor('#0D9FFB')
          .width('40%')
          .height('5%')
        .width('100%')
      }
      .height('100%')
    }
}
```

实现页面间的跳转

页面间的导航可以通过页面路由 router 来实现。页面路由 router 根据页面 url 找到目标页面，从而实现跳转。使用页面 路由请导入 router 模块。如果需要实现更好的转场动效，推荐使用 Navigation。

第一个页面跳转到第二个页面。 在第一个页面中，跳转按钮绑定 onClick 事件，单击按钮时跳转到第二页。Index.ets 文件的示例如下：

```
// Index.ets

import { BusinessError } from '@kit.BasicServicesKit';

@Entry

@Component

struct Index {

  @State message: string = 'Hello World';

  build() {

    Row() {

      Column() {

        Text(this.message)

          .fontSize(50)

          .fontWeight(FontWeight.Bold)

        // 添加按钮，以响应用户 onClick 事件

        Button() {

          Text('Next')

            .fontSize(30)

            .fontWeight(FontWeight.Bold)

        }

        .type(ButtonType.Capsule)

        .margin({

          top: 20

        })

        .backgroundColor('#0D9FFB')

        .width('40%')

        .height('5%')

        // 跳转按钮绑定 onClick 事件，单击时跳转到第二页
```

```
          .onClick(() => {
            console.info(`Succeeded in clicking the 'Next' button.`)
            // 获取 UIContext
            let uiContext: UIContext = this.getUIContext();
            let router = uiContext.getRouter();
            // 跳转到第二页
            router.pushUrl({ url: 'pages/Second' }).then(() => {
              console.info('Succeeded in jumping to the second page.')
            }).catch((err: BusinessError) => {
              console.error(`Failed to jump to the second page. Code is ${err.code},
message is ${err.message}`)
            })
          })
        }
        .width('100%')
      }
      .height('100%')
    }
}
```

第二个页面返回到第一个页面。 在第二个页面中，返回按钮绑定 onClick 事件，单击按钮时返回到第一页。Second.ets 文件的示例如下:

```
// Second.ets
import { BusinessError } from '@kit.BasicServicesKit';
@Entry
```

```
@Component
struct Second {
  @State message: string = 'Hi there';
  build() {
    Row() {
      Column() {
        Text(this.message)
          .fontSize(50)
          .fontWeight(FontWeight.Bold)
        Button() {
          Text('Back')
            .fontSize(30)
            .fontWeight(FontWeight.Bold)
        }
        .type(ButtonType.Capsule)
        .margin({
          top: 20
        })
        .backgroundColor('#0D9FFB')
        .width('40%')
        .height('5%')
        // 返回按钮绑定 onClick 事件，单击按钮时返回到第一页
        .onClick(() => {
          console.info(`Succeeded in clicking the 'Back' button.`)
        // 获取 UIContext
          let uiContext: UIContext = this.getUIContext();
```

```
            let router = uiContext.getRouter();
            try {
              // 返回第一页
              router.back()
              console.info('Succeeded in returning to the first page.')
            } catch (err) {
              let code = (err as BusinessError).code;
              let message = (err as BusinessError).message;
              console.error(`Failed to return to the first page. Code is ${code},
message is ${message}`)
            }
          })
        }
        .width('100%')
      }
      .height('100%')
    }
}
```

其中：

`.onClick(() => {...})`

给按钮绑定点击事件。

每次点击都会执行回调函数。

`console.info(...)`

在日志中输出点击事件是否成功触发，方便调试。

`let uiContext: UIContext = this.getUIContext();`

获取当前页面的 `UIContext`（UI 上下文对象）。

UIContext` 提供页面路由、窗口管理等能力。

```
let router = uiContext.getRouter();
```

从 `UIContext` 中获取 页面路由对象，用来进行页面跳转或回退。

```
router.back()
```

回退到上一页，默认会返回到栈顶的前一个页面。

如果你是从第一页跳到第二页，这里执行就会回到第一页。

异常捕获 `try...catch`

如果 `router.back()` 执行失败（比如当前页面就是第一页，没有可回退的页面），会进入 `catch`。

`err as BusinessError`：把错误对象转为 `BusinessError` 类型，从而能读取 `.code` 和 `.message`。

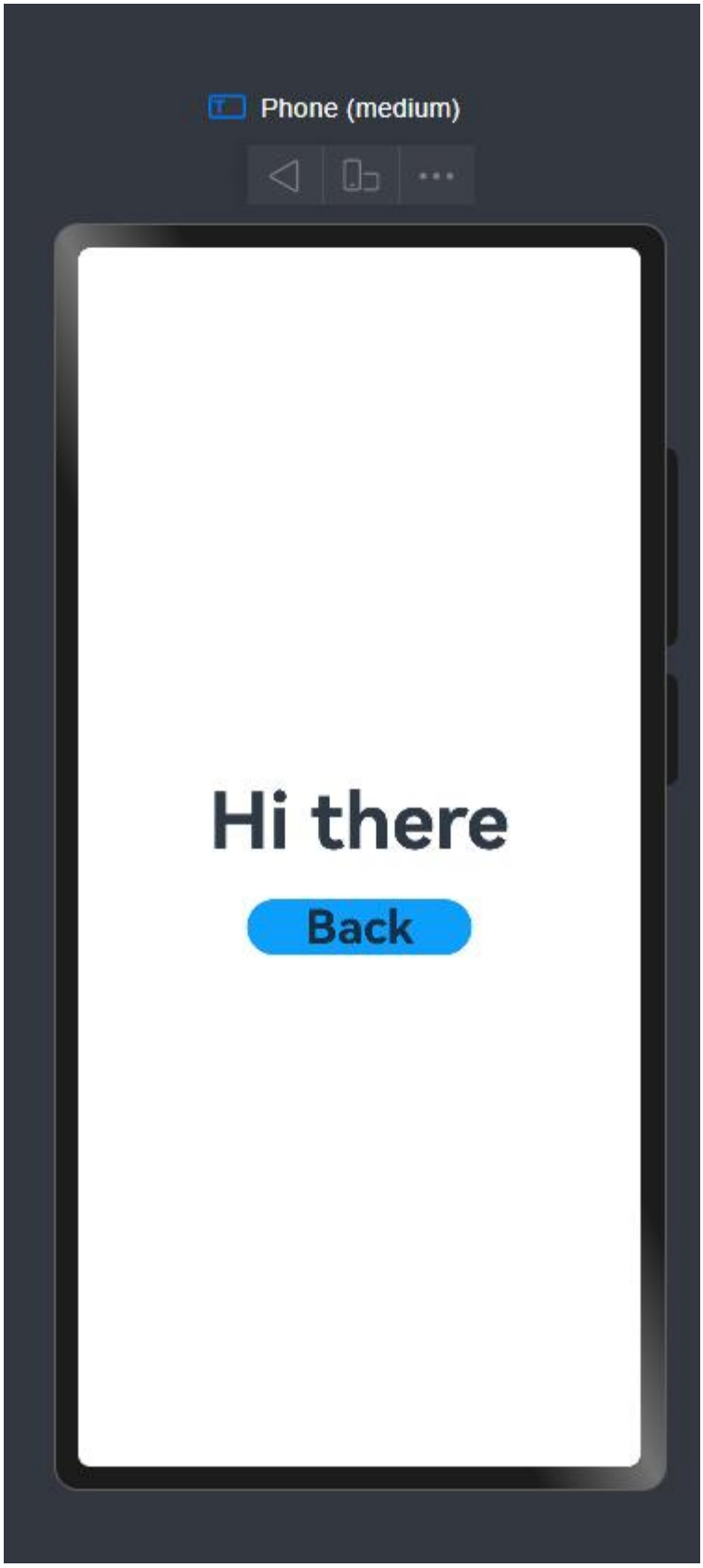
输出错误日志，方便定位问题。

三、程序运行结果

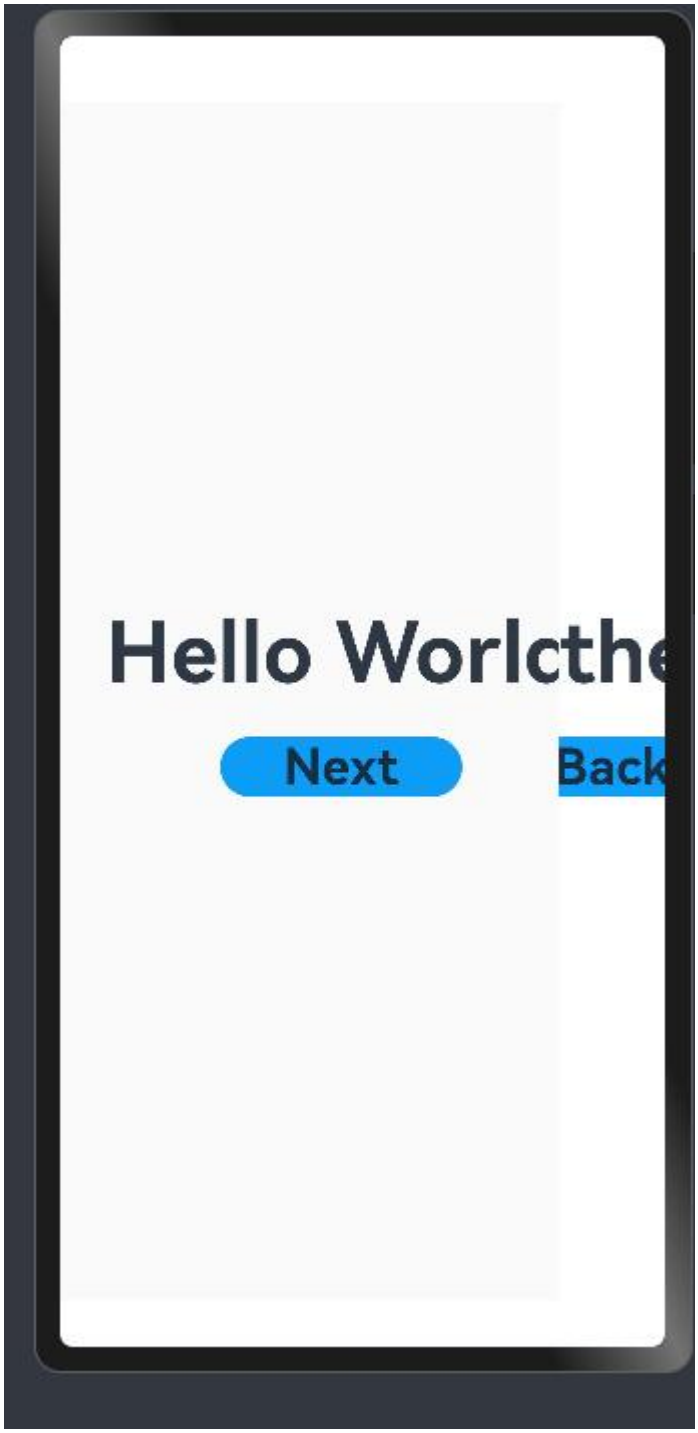
列出程序的最终运行结果及截图。

重新点击 previewer

点击 `next` 可以进入第二个页面

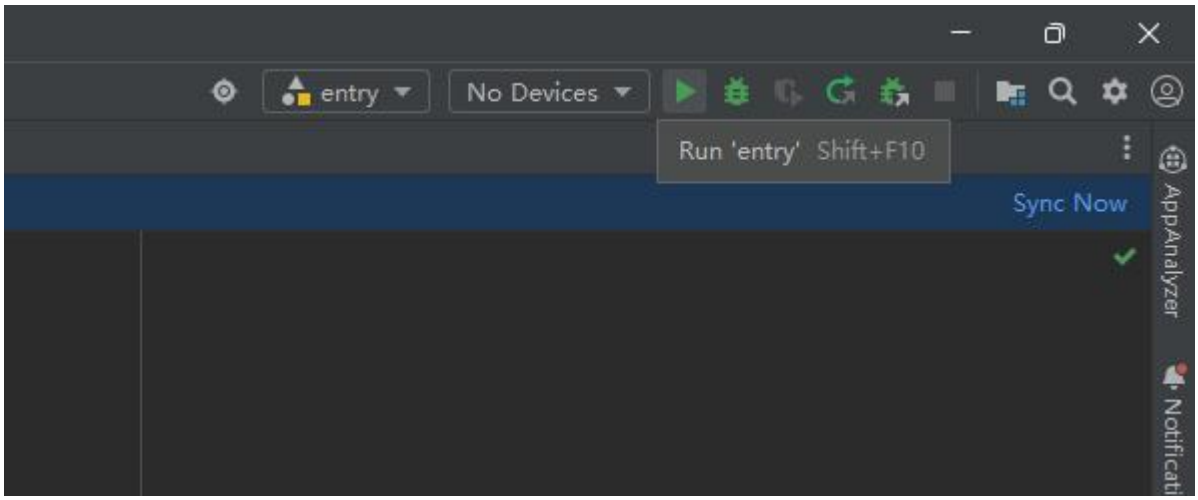


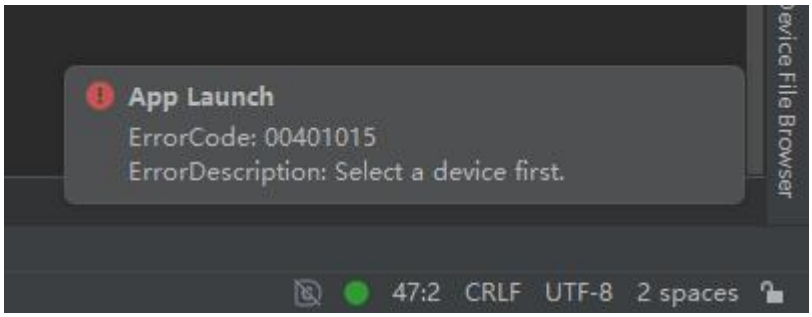
点击 `back` 可以返回第一个页面



四、问题总结与体会

一开始一直在点上面的 run 和 debug 一直，没用





问了 gpt，它也无能为力。

然后我就想着先不用 run，先把代码写好，之后再看可不可以 run

写完第一个按钮之后才知道是要点旁边的 previwer 才行