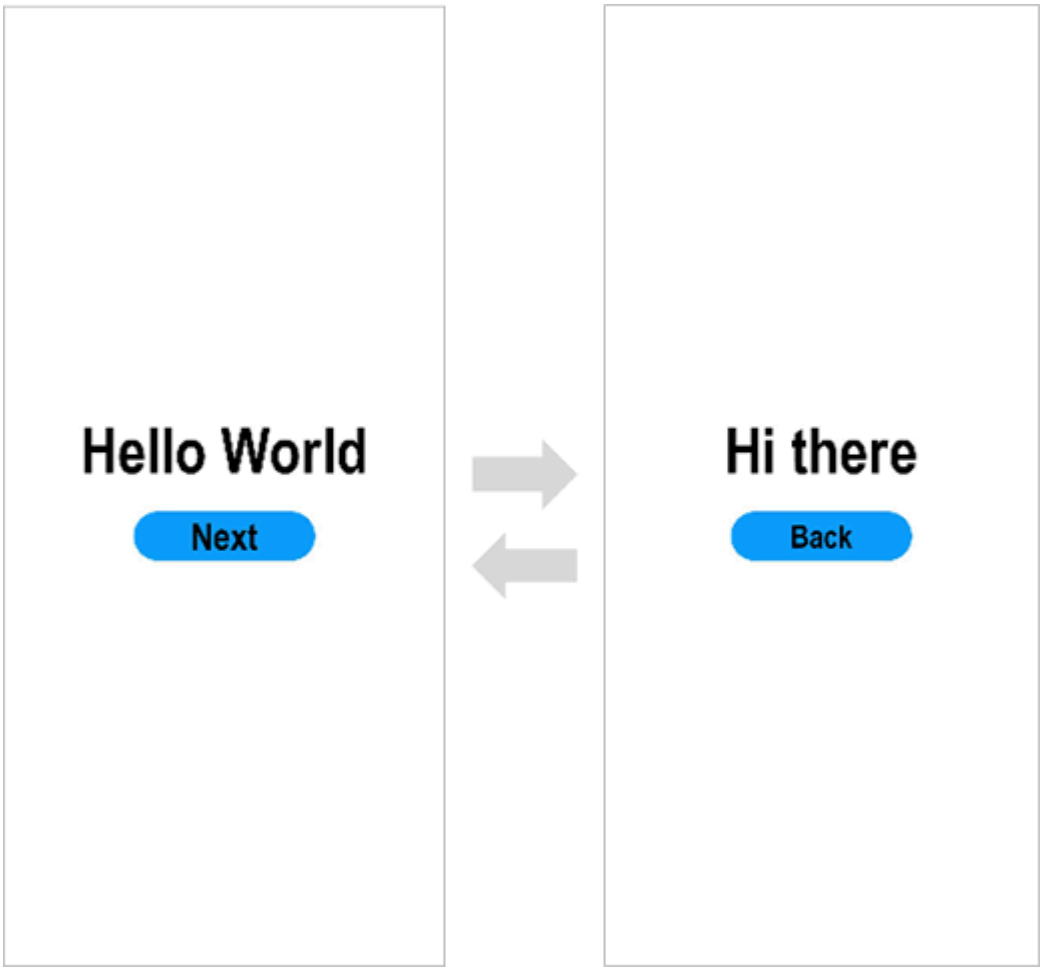
实验5：第一个 HarmonyOS 应用

> 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。

开发准备

本文档适用于HarmonyOS应用开发的初学者。通过构建一个简单的具有页面跳转/返回功能的应用（如下图所示），快速了解工程目录的主要文件，熟悉HarmonyOS应用开发流程。



在开始之前，您需要了解有关HarmonyOS应用的一些基本概念：UI框架的简单说明、应用模型的基本概念。

基本概念

UI框架

HarmonyOS提供了一套UI开发框架，即方舟开发框架（ArkUI框架）。方舟开发框架可为开发者提供应用UI开发所必需的能力，比如多种组件、布局计算、动画能力、UI交互、绘制等。

方舟开发框架针对不同目的和技术背景的开发者提供了两种开发范式，分别是基于ArkTS的声明式开发范式（简称“声明式开发范式”）和兼容JS的类Web开发范式（简称“类Web开发范式”）。以下是两种开发范式的简单对比。

开发范式名称	语言生态	UI更新方式	适用场景	适用人群
声明式开发范式	ArkTS语言	数据驱动更新	复杂度较大、团队合作度较高的程序	移动系统应用开发人员、系统应用开发人员
类Web开发范式	JS语言	数据驱动更新	界面较为简单的程序应用和卡片	Web前端开发人员

工具准备

请下载并安装最新版DevEco Studio 。

DevEco Studio 5.1.1 Release

配套HarmonyOS 5.1.1(19)，面向 HarmonyOS 应用及元服务开发者提供的集成开发环境（IDE），助力高效开发。此版本支持指定构建模式，进一步提升构建效率。

Build Version 5.1.1.840　　发布日期 2025/09/05

版本说明　操作指导　隐私声明

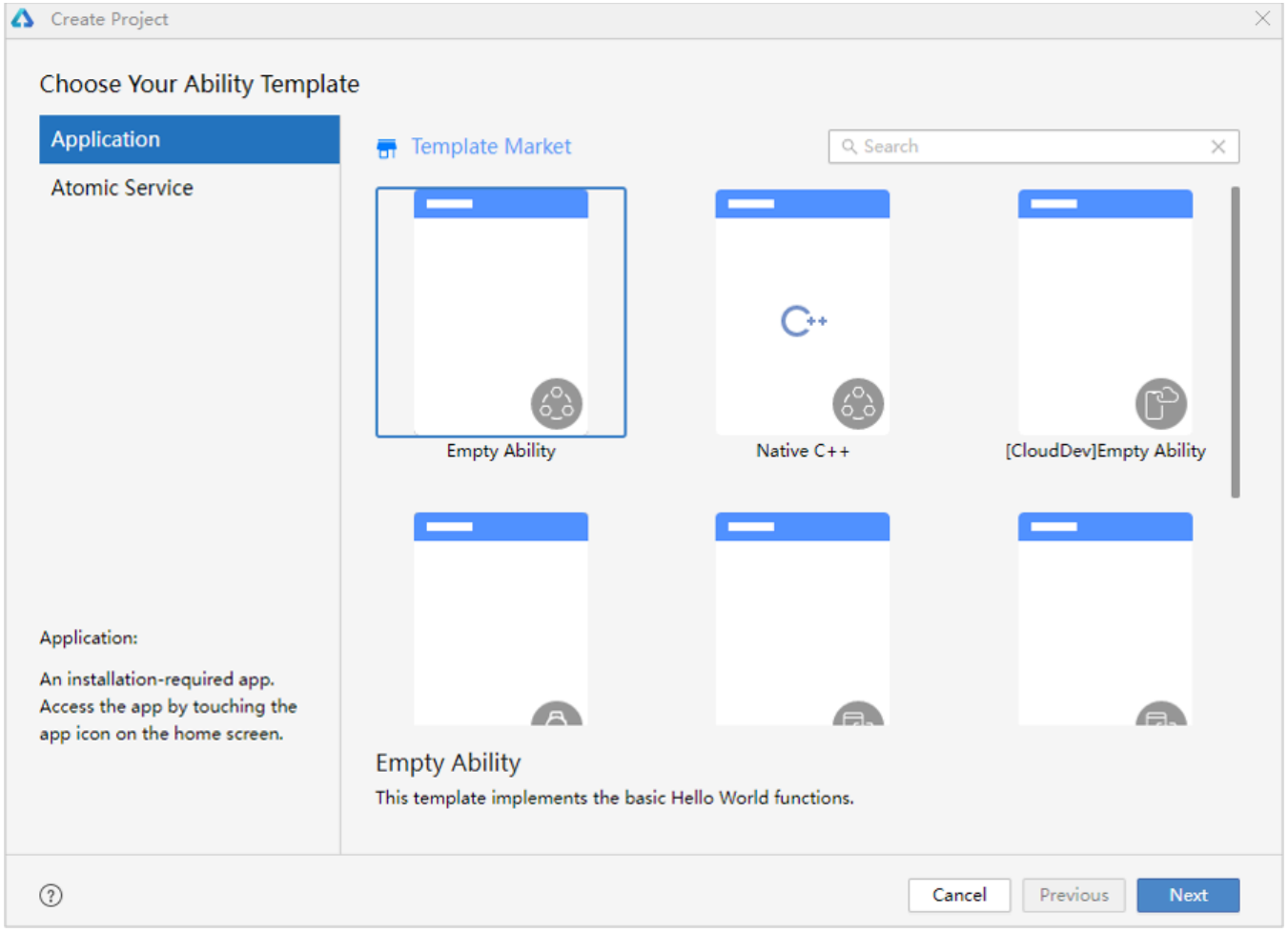
Windows (64-bit)
DevEco Studio for Windows 5.1.1.840(2.2GB)　SHA-256　|　PGP

Mac (X86)
DevEco Studio for Mac(x86) 5.1.1.840(2.9GB)　SHA-256　|　PGP

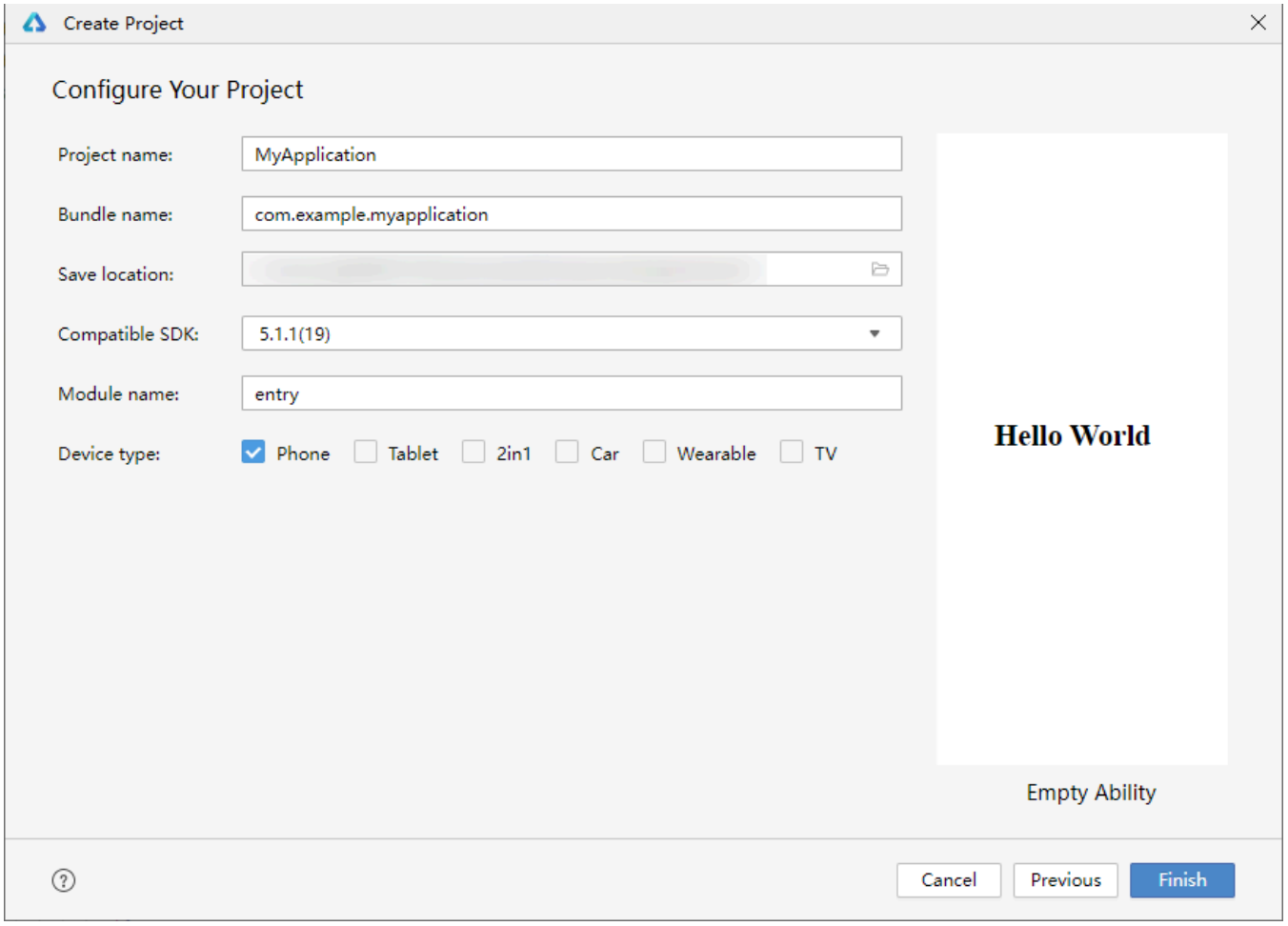
Mac (ARM)
DevEco Studio for Mac(ARM) 5.1.1.840(2.7GB)　SHA-256　|　PGP

创建ArkTS工程

1. 若首次打开**DevEco Studio**，请单击**Create Project**创建工程。如果已经打开了一个工程，请在菜单栏选择**File** > **New** > **Create Project**来创建一个新工程。
2. 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。

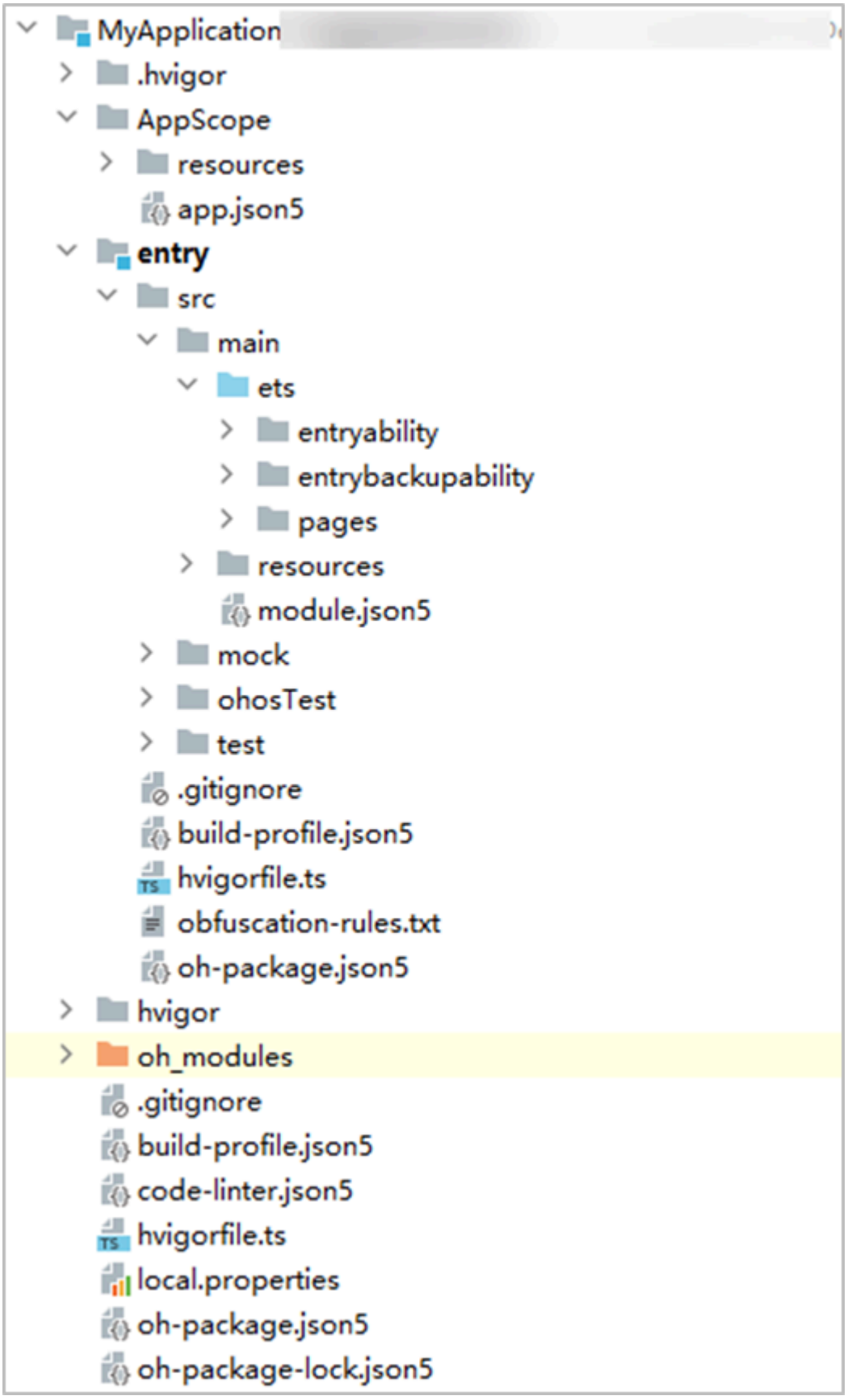


3. 进入配置工程界面，**Compatible SDK**表示兼容的最低API Version，此处以选择**5.1.1(19)**为例，其他参数保持默认设置即可。



4. 单击**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。

ArkTS工程目录结构（Stage模型）



- **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。

构建第一个页面

1. 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：

   ```
   // Index.ets
   @Entry
   @Component
   struct Index {
     @State message: string = 'Hello World';

     build() {
       Row() {
         Column() {
           Text(this.message)
             .fontSize(50)
             .fontWeight(FontWeight.Bold)
         }
   ```

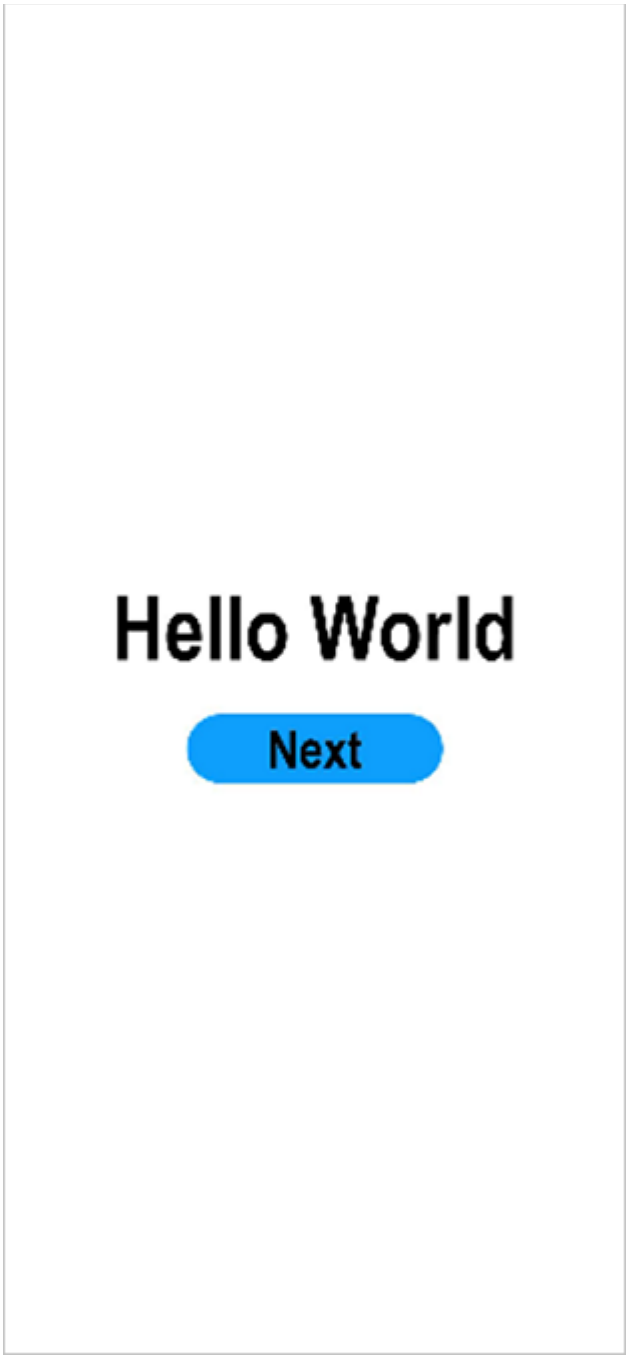
```
      .width('100%')
    }
    .height('100%')
  }
}
```

2. 添加按钮。

   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：

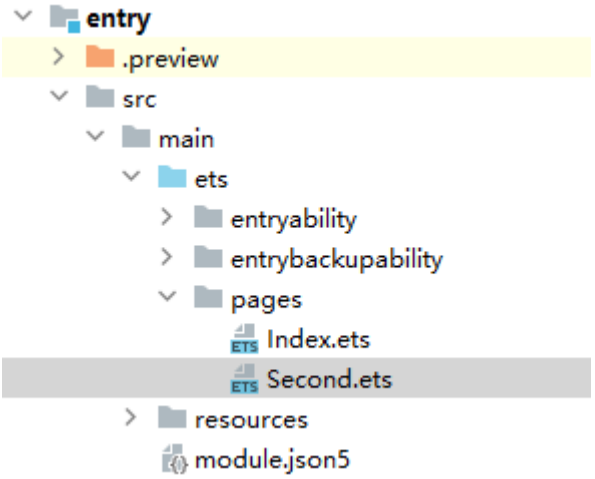
```
// Index.ets
@Entry
@Component
struct Index {
  @State message: string = 'Hello World';

  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%')
  }
}
```

3. 在编辑窗口**右上角**的侧边工具栏，单击**Previewer**，打开预览器。第一个页面效果如下图所示：



构建第二个页面

1. 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**，单击**回车键**。可以看到文件目录结构如下：



> 说明
>
> 开发者也可以在右键单击**pages**文件夹时，选择**New > Page > Empty Page**，命名为**Second**，单击**Finish**完成第二个页面的创建。使用此种方式则无需再进行下文中第二个页面路由的手动配置。

- 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"。示例如下：

```
{
  "src": [
    "pages/Index",
    "pages/Second"
  ]
}
```

2. 添加文本及按钮。

   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
@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。

1. 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%')
  }
}
```

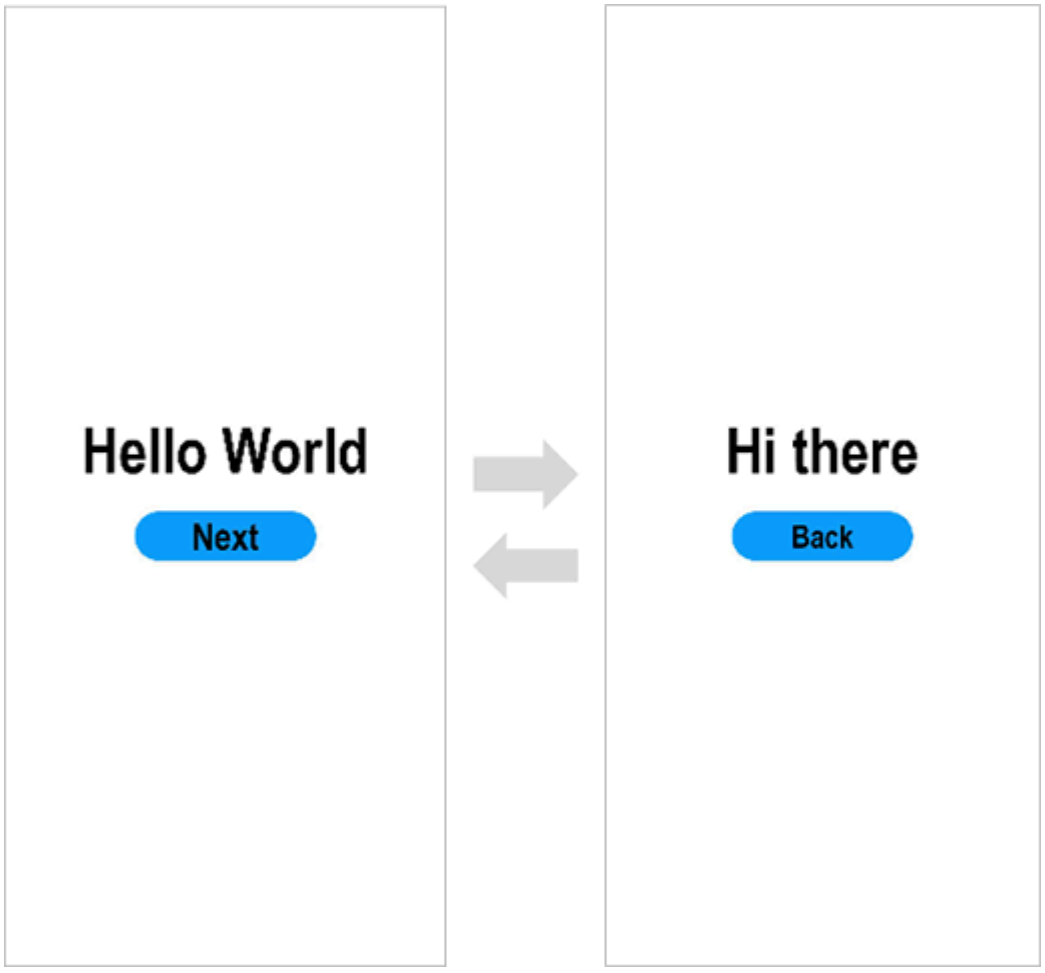
2. 第二个页面返回到第一个页面。

   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.`)
```

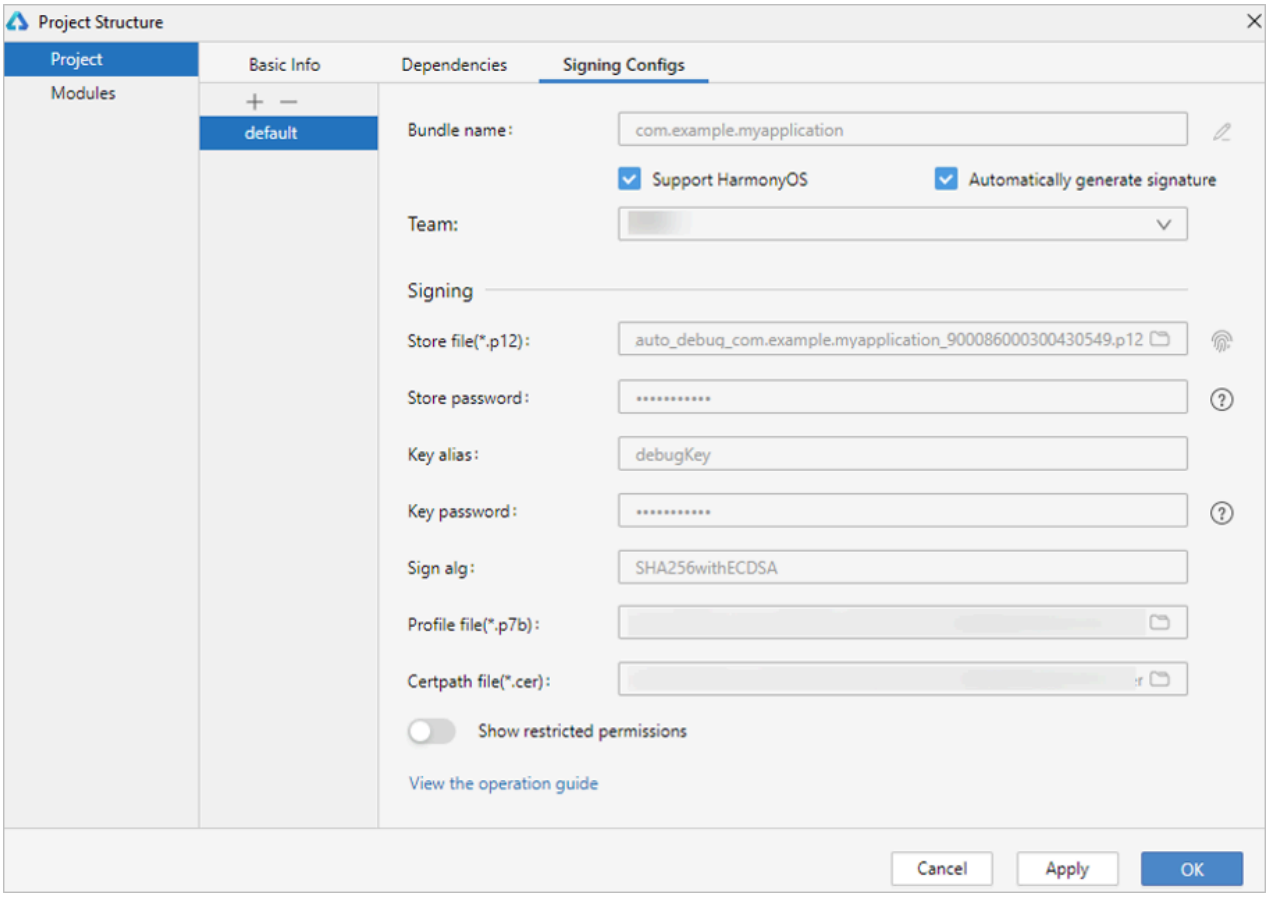
```
          // 获取UIContext
          let uiContext: UIContext = this.getUIContext();
          let router = uiContext.getRouter();
          try {
            // 返回第一页
            router.back()
            console.info('Succeeded in returning to the first page.')
          } catch (err) {
            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%')
  }
}
```

3. 打开**Index.ets**文件，单击预览器中的 按钮进行刷新。效果如下图所示：

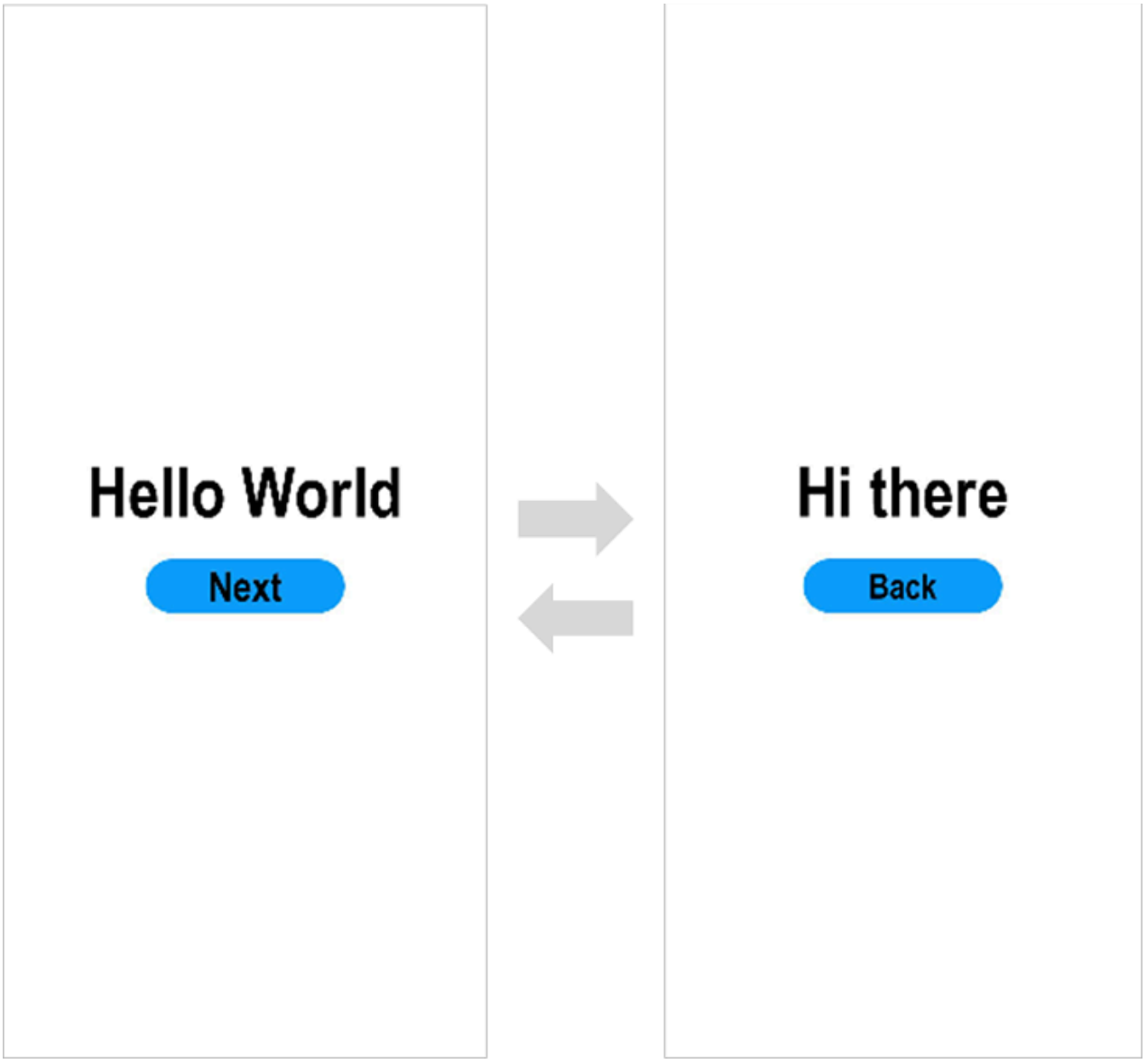


使用真机运行应用（选做）

1. 将搭载HarmonyOS系统的真机与电脑连接。
2. 单击**File** > **Project Structure...** > **Project** > **SigningConfigs**界面勾选**Support HarmonyOS**和**Automatically generate signature**，单击界面提示的**Sign In**，使用华为账号登录。等待自动签名完成后，单击**OK**即可。如下图所示：



3. 在编辑窗口右上角的工具栏，单击 ▶ 按钮运行。效果如下图所示：



恭喜您已经基于ArkTS语言构建完成第一个HarmonyOS应用，快来探索更多的HarmonyOS功能吧。