第一部分

基础知识

本书的第一部分将全面介绍 HarmonyOS NEXT（也称为 HarmonyOS 5.0）及其 ArkTS 声明式 UI 开发规范的核心内容与应用，旨在为开发者提供坚实的基础知识和实用的开发技能。本部分共包含 7 章，分别是：

- 第 1 章　ArkTS 声明式 UI 开发规范：深入剖析 HarmonyOS NEXT 的创新特点和整体架构，详细讲解 ArkTS 声明式 UI 开发流程和通用规则。同时，介绍开发环境的搭建和入门程序的编写，帮助读者掌握 ArkTS 开发的基本技巧和流程。
- 第 2 章　ArkUI 常用开发布局：详细介绍 ArkUI 中的多种布局方式，如线性布局、层叠布局、弹性布局等，通过示例代码和图示展示每种布局的特点和使用方法，帮助读者提高页面开发的效率和质量。
- 第 3 章　ArkUI 中的常用组件：详细讲解 ArkUI 中常用的组件，如按钮、单选框、进度条等，包括其创建方法、属性设置、事件绑定和应用场景，帮助读者提升界面设计和交互实现的能力。
- 第 4 章　组件导航和页面路由：探讨 Navigation 组件的使用方法和页面路由的实现，包括模块内和跨模块的路由切换、页面显示模式设置、导航转场动画效果和页面跳转数据传递等内容，助力读者实现复杂的页面交互与数据传递。
- 第 5 章　交互事件：深入探讨 HarmonyOS 中的交互事件处理机制，包括触屏事件、焦点事件、拖曳事件和手势事件等，帮助读者掌握各种交互事件的处理技巧，从而提升应用的交互性和用户体验。
- 第 6 章　窗口管理：详细讲解 HarmonyOS NEXT 中窗口管理的相关知识，包括窗口模块的基本概念、实现原理、Stage 模型下的应用窗口管理，以及窗口属性设置和事件监听等操作。
- 第 7 章　ArkWeb：详细介绍 ArkWeb 组件及其在 HarmonyOS NEXT 中的应用，包括多进程模型、生命周期管理、性能指标、UserAgent 开发和与前端页面的 JavaScript 交互等内容，帮助读者提升处理 Web 内容的能力。

通过学习以上内容，读者将全面掌握 HarmonyOS NEXT 及其 ArkTS 声明式 UI 开发的基础知识和核心技能。这将为开发高质量的应用程序奠定坚实的基础。同时，读者将提升自身的开发能力，编写出更高效、更可维护的代码。希望读者能够灵活运用所学知识，不断优化开发流程，迎接未来的挑战，推动智能设备应用的创新与发展。

第 1 章

ArkTS 声明式 UI 开发规范

本章将全面介绍 HarmonyOS NEXT 及其 ArkTS 声明式 UI 开发规范的核心内容与应用。

首先，深入剖析 HarmonyOS NEXT 的创新特点，包括微内核架构、全场景覆盖、分布式创新、流畅性能、安全至上和开放生态等，展示其在智能设备领域的深远影响以及为开发者带来的巨大机遇。

接着，详细阐述 HarmonyOS NEXT 的整体架构，涵盖声明式 UI 前端、语言运行时、声明式 UI 后端引擎、渲染引擎和平台适配层等关键组成部分，帮助开发者更好地理解系统的工作原理和资源管理方式。

然后，系统讲解 ArkTS 声明式 UI 开发流程，为开发者提供全面的指导。同时，介绍 HarmonyOS NEXT 应用开发中的通用规则，如默认单位 vp 的使用和异常值处理，助力开发者提高编码效率和代码质量。

接下来，详细描述开发环境搭建的全过程，包括 HUAWEI DevEco Studio 的下载、安装、配置和开发环境诊断等，确保开发者能够顺利搭建高效、稳定的开发平台。

最后，通过编写“Hello HarmonyOS”入门程序，带领读者逐步创建 ArkTS 工程，构建页面，实现页面间跳转，从而深入理解 ArkUI 框架的 UI 开发范式和应用模型，掌握 ArkTS 开发的基本技巧和流程。

通过本章的学习，读者将全面掌握 HarmonyOS NEXT 及其 ArkTS 声明式 UI 开发规范的关键知识，为后续的深入学习和开发实践打下坚实的基础。

1.1 HarmonyOS NEXT 的介绍及其特点

在数字化浪潮中，操作系统不仅是硬件与用户之间的桥梁，更是驱动设备智能化的核心所在。华为推出的 HarmonyOS NEXT，不仅展现了对未来智能生活的憧憬，更是华为技术创新的经典之作。

1.1.1　HarmonyOS NEXT 概览

HarmonyOS NEXT 是华为精心研发的一款操作系统，秉承“全场景、全连接、全智能”的设计理念，为全球用户描绘了全新的数字生活图景。基于微内核架构，这一系统以卓越的性能、安全性和可扩展性，出色地适配多种设备和应用场景。

1.1.2　核心亮点

HarmonyOS NEXT 的核心亮点如下。

1. 微内核架构

HarmonyOS NEXT 采用微内核设计，与传统宏内核相比，具备更高的安全性和更快的响应速度，使系统既轻盈又强大。

2. 全场景覆盖

支持从智能手机到平板电脑、智能电视及穿戴设备的多种场景，HarmonyOS NEXT 实现了跨设备的无缝体验，确保用户在不同设备上的操作感受始终一致。

3. 分布式创新

分布式能力是 HarmonyOS NEXT 的核心特色之一，支持设备间资源共享和任务协同，无论是数据传输还是应用执行，均可在设备间顺畅衔接，为用户带来便捷的使用体验。

4. 性能流畅

通过先进的内存管理和任务调度技术，HarmonyOS NEXT 确保用户获得流畅的系统体验。此外，系统支持多语言编程，为开发者提供更大的“创作”空间和自由度。

5. 安全至上

从设计阶段起，HarmonyOS NEXT 便将安全置于核心位置，采用数据加密、安全启动等多层次防护措施，全面保障用户数据安全。

6. 开放生态

华为致力于构建开放的生态系统。HarmonyOS NEXT 提供丰富的 API 和开发工具，激励全球开发者与合作伙伴共同探索新的可能性，为生态注入更多活力。

1.1.3　深远影响

HarmonyOS NEXT 的问世标志着华为产品生态的一次重大飞跃，也将在全球操作系统市场掀起波澜。它不仅有望重塑用户对智能设备的使用习惯，还将推动行业技术不断向前发展。

1. 设备智能化加速

HarmonyOS NEXT 的普及将加速设备智能化进程，更好地满足用户的个性化需求，提升使用体验。

2. 技术创新激发

系统的开放性和分布式架构将激发开发者的创新潜能，催生出新技术和新应用。

3. 市场竞争重塑

HarmonyOS NEXT 的加入可能重塑操作系统市场的竞争格局，为用户提供更加多元化的选择，推动行业健康发展。

1.1.4 开发者机遇

对于开发者而言，HarmonyOS NEXT 不仅是一个强大的平台，更是一个孕育创新与成长的广阔舞台。

1. 全栈自研能力

HarmonyOS NEXT 提供的全栈自研优势，能够助力开发者降低开发成本，提高开发效率，实现高效创新。

2. 原生应用开发

借助系统级 AI 能力，开发者可以轻松构建智能功能，为用户带来原生应用的体验。

3. 统一生态部署

通过“一次开发，多端适配”的生态能力，HarmonyOS NEXT 简化了跨设备开发流程，提升了应用开发效率与资源利用率。

4. 安全与隐私保障

在以安全为核心的平台上开发，有助于提升用户对应用的信任度。

5. 开放合作生态

华为秉持开放合作的理念，积极鼓励开发者加入，共同推动生态繁荣发展。

6. 技术支持与社区资源

华为提供全面的开发支持和活跃的社区资源，助力开发者快速成长。

HarmonyOS NEXT 是华为推出的新一代操作系统，其创新理念和强大功能充分展现了华为对未来智能生活的深刻洞察和技术实力。随着技术的不断完善和市场认可度的逐步提升，HarmonyOS NEXT 有望成为推动智能设备发展的重要引擎。

1.2 整体架构

了解操作系统架构设计对于学习应用开发至关重要，主要体现在以下几个方面：

- 资源管理与优化：操作系统是计算机系统的核心，负责管理硬件和软件资源。深入理解操作

系统架构设计有助于开发者掌握如何高效地利用这些资源（如内存管理、进程调度、文件系统等），从而编写出更加高效、稳定且资源占用更少的代码。

- 并发与并行处理：操作系统的并发控制机制和并行处理策略对多任务处理和数据并行处理的应用开发至关重要。掌握这些知识有助于开发者设计出响应更快、性能更优的应用程序。
- 系统调用与接口：操作系统为上层应用提供了丰富的系统调用接口（System Call Interface）和应用程序编程接口（Application Programming Interface，API），这些接口是应用与操作系统交互的桥梁。了解操作系统的架构设计有助于开发者深入理解系统调用的工作原理和性能特性，从而更准确地使用这些接口来满足应用的需求。
- 安全与稳定性：操作系统架构的设计直接决定了系统的安全性和稳定性。掌握操作系统的安全机制（如权限管理、访问控制等）以及稳定性保障措施（如错误处理、故障恢复等），有助于开发者在开发应用中有效地考虑这些因素，编写出更加安全可靠的应用。
- 深入理解技术栈：在软件开发领域，操作系统是整个技术栈的基础层。深入了解操作系统架构的设计有助于开发者全面理解技术栈的整体结构和核心逻辑，从而更好地把握应用开发的方向和关键技术点。

因此，了解操作系统架构设计对于学习应用开发意义重大。这不仅能显著提高开发效率和代码质量，还能培养开发者的系统思维能力和跨平台开发能力。

HarmonyOS NEXT 操作系统整体架构如图 1-1 所示。

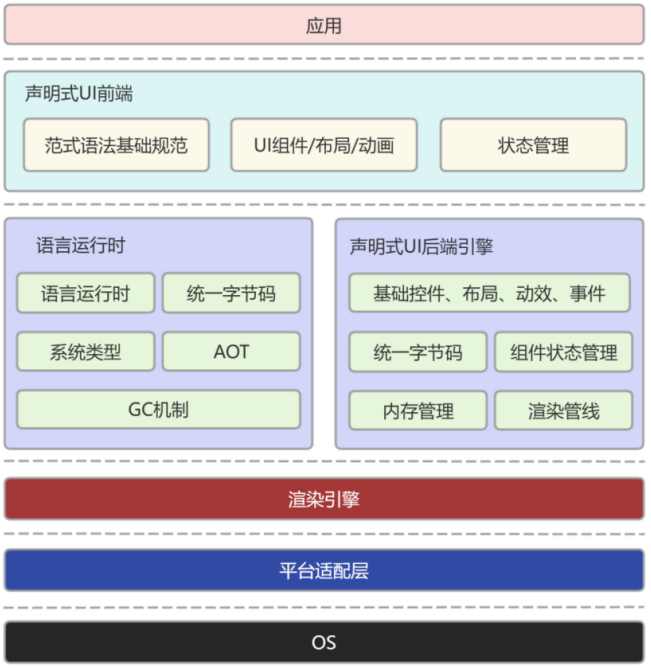

图 1-1　HarmonyOS NEXT 整体架构图

1. 声明式 UI 前端

声明式 UI 前端提供了 UI 开发范式的基础语言规范，内置丰富的 UI 组件、布局和动画，支持多种状态管理机制，为开发者提供完善的接口支持。

2. 语言运行时

基于 ArkTS 语言（即方舟语言）的运行时（runtime，也称为运行时库），具备对 UI 范式语法的解析能力，支持跨语言调用，并提供高性能的 TypeScript 语言运行环境。

3. 声明式 UI 后端引擎

后端引擎兼容多种开发范式，提供强大的 UI 渲染管线支持，包括基础组件、布局计算、动效处理、交互事件管理，以及状态管理和绘制能力。

4. 渲染引擎

渲染引擎提供高效的绘制能力，可将渲染管线生成的渲染指令快速呈现至屏幕。

5. 平台适配层

平台适配层提供系统平台的抽象接口，具备接入不同系统的能力，包括支持系统渲染管线的整合与生命周期调度管理。

1.3 开发流程

使用 UI 开发框架开发应用时，主要涉及以下开发流程：

（1）ArkTS：包括 ArkTS 的语法、状态管理和渲染控制的应用场景。
（2）开发布局：包括常用布局方式及其使用场景。
（3）添加组件：包括内置组件、自定义组件以及通过 API 支持的界面元素。
（4）设置组件导航和页面路由：配置组件间的导航及页面的路由功能。
（5）显示图形：显示图片、绘制自定义几何图形，以及使用画布绘制自定义图形。
（6）使用动画：包括组件和页面动画的典型应用场景及实现方法。
（7）绑定事件：包括事件的基本概念，以及使用通用事件和手势事件的方法。
（8）使用自定义能力：理解自定义能力的基本概念，并实现相关功能。
（9）主题设置：进行应用级和页面级的主题设置与管理。
（10）使用 NDK 接口构建 UI：通过 ArkUI 提供的 NDK 接口，创建和管理 UI 界面。
（11）使用镜像能力：了解镜像能力的基本概念，并掌握其使用方法。

1.4 通用规则

通用规则是指开发应用时，系统默认的处理方式。熟悉这些规则可以帮助开发者提高开发效率，并编写出更高质量的代码。在 HarmonyOS NEXT 应用开发中，通用规则主要包含以下两个方面。

1. 默认单位

表示长度的输入参数的单位默认为 vp，即 number 类型的参数。对于以 number 类型值表示的 Length 类型的参数及 Dimension 类型的参数，其数值单位默认为 vp。

vp 是虚拟像素（Virtual Pixel）的缩写，指设备相对于应用的虚拟尺寸（区别于屏幕硬件像素单位）。vp 是一种灵活的单位，能根据不同屏幕的像素密度进行缩放，从而在各种屏幕上保持统一的尺寸显示效果，提供一致的视觉体验。

2. 异常值处理

当输入参数为异常值（undefined、null 或无效值）时，系统处理规则如下：

（1）若对应参数有默认值，则按默认值处理。
（2）若对应参数无默认值，则该参数对应的属性或接口不生效。

1.5　开发环境搭建

本节介绍 HarmonyOS NEXT 开发环境的搭建过程，包括安装和配置 HUAWEI DevEco Studio。

1.5.1　概述

HUAWEI DevEco Studio 基于 IntelliJ IDEA Community 开源版本构建，为在 HarmonyOS NEXT 系统上的应用和服务提供一站式开发平台。

作为一款专业的开发工具，DevEco Studio 除具备代码开发、编译构建、调测等功能外，还具备如下特点：

（1）高效智能代码编辑：支持 ArkTS、JavaScript、C/C++等多种语言，提供代码高亮、智能补齐、错误检查、自动跳转、格式化、查找等功能，大幅提升代码编写效率。

（2）多端双向实时预览：支持 UI 界面代码的双向预览、实时预览、动态预览、组件预览及多端设备预览，便于开发者快速查看代码运行效果。

（3）多端设备模拟仿真：提供 HarmonyOS NEXT 本地模拟器，并支持 iPhone 等设备的模拟仿真，便捷获取调试环境。

（4）DevEco Profiler 性能调优：提供实时监控能力和场景化调优模版，支持全方位的设备资源监测和多维度数据采集，帮助开发者实现高效的性能调优和快速定位问题代码行。

1.5.2　工具准备

从 HarmonyOS Developer 官网下载最新版的 DevEco Studio 安装包。

1.5.3　安装 DevEco Studio

DevEco Studio 支持 Windows 和 macOS 系统，下面分别介绍在这两种操作系统下安装 DevEco Studio 软件的步骤。

1. Windows 环境下安装 DevEco Studio

1）运行环境要求

为确保 DevEco Studio 正常运行，建议计算机配置满足以下要求：

- 操作系统：Windows 10 64 位、Windows 11 64 位。
- 内存：推荐 16GB 或以上，最低 8GB。
- 硬盘：100GB 或以上。
- 分辨率：1280 × 800 像素或更高。

2）安装 DevEco Studio

Windows 环境下安装 DevEco Studio 的操作步骤如下：

步骤 01 双击下载的 deveco-studio-xxxx.exe 文件，启动 DevEco Studio 安装向导，如图 1-2 所示。

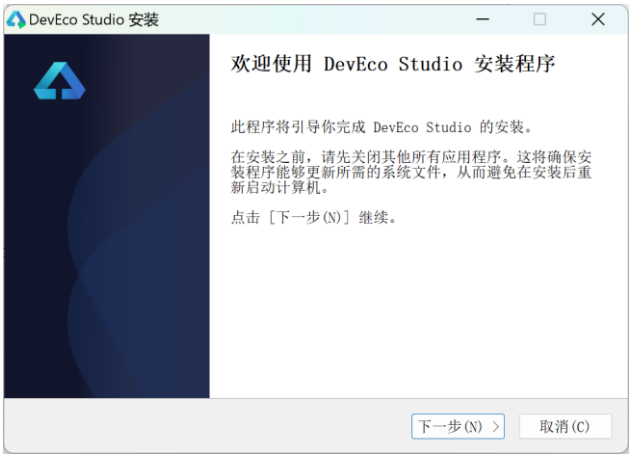

图 1-2　安装向导界面

步骤 02 单击“下一步”按钮，进入“选择安装位置”界面。默认安装路径在 C:\Program Files 下，也可单击“浏览”按钮指定其他安装路径，然后单击“下一步”按钮，如图 1-3 所示。

图 1-3　Windows 环境下 DevEco Studio 的“选择安装位置”界面

步骤 03 在如图 1-4 所示的“安装选项”界面中勾选 DevEco Studio 后，单击“下一步”按钮。

图 1-4　Windows 环境下 DevEco Studio 的“安装选项”界面

步骤 04 按照提示依次单击“下一步”按钮，直至安装完成，最后单击“完成”按钮，如图 1-5 所示。

图 1-5　Windows 环境下 DevEco Studio 安装完成时的界面

2. macOS 环境下安装 DevEco Studio

1）运行环境要求

为确保 DevEco Studio 正常运行，建议计算机配置满足以下要求：

- 操作系统：macOS(X86) 10.15/11/12/13/14；macOS(ARM) 11/12/13/14。
- 内存：推荐 16GB 或以上，最低 8GB。
- 硬盘：100GB 或以上。
- 分辨率：1280 × 800 像素或更高。

2）安装 DevEco Studio

macOS 环境下安装 DevEco Studio 的操作步骤如下：

步骤 01 在安装界面中，将 DevEco-Studio.app 拖曳到 Applications 文件夹中，等待安装完成，如

图 1-6 所示。

图 1-6　macOS 环境下 DevEco Studio 的安装界面

步骤 02 安装完成后，按照提示配置代理并检查开发环境。

1.5.4　诊断开发环境

为确保开发者在应用或服务开发过程中获得良好体验，DevEco Studio 提供了开发环境诊断功能，帮助开发者检查开发环境是否配置完备。开发者可以在欢迎页面单击 Diagnose 按钮进行诊断，如图 1-7 所示。如果已打开工程开发界面，也可以在菜单栏中依次单击 Help→Diagnostic Tools→Diagnose Development Environment 选项进行诊断。

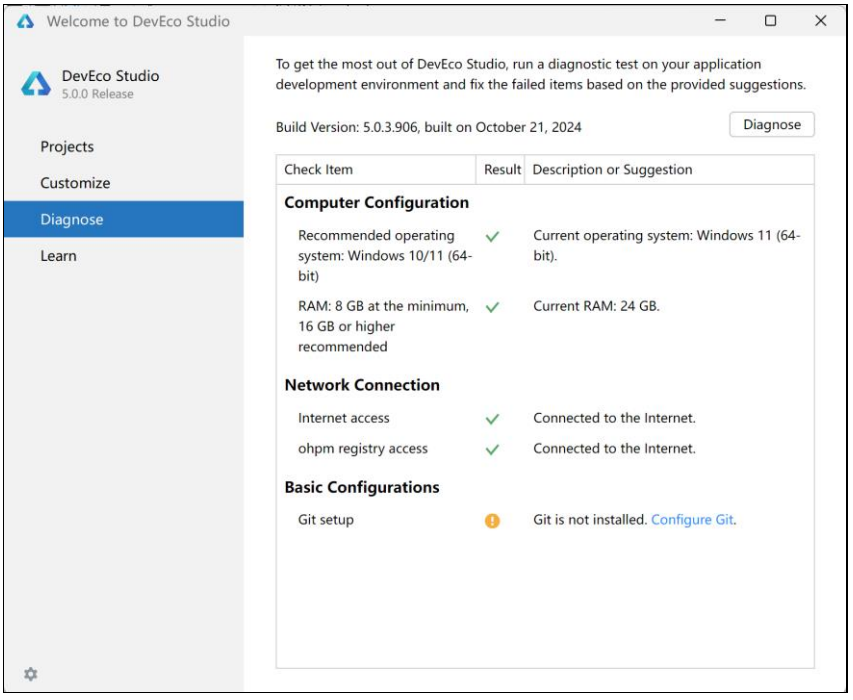

图 1-7　Windows 环境下 DevEco Studio 进入开发环境诊断时的界面

DevEco Studio 开发环境诊断项包括计算机的配置、网络的连通情况以及依赖工具的安装情况等。如果检测结果为未通过，可根据检查项的描述和修复建议进行处理，具体界面如图 1-8 所示。

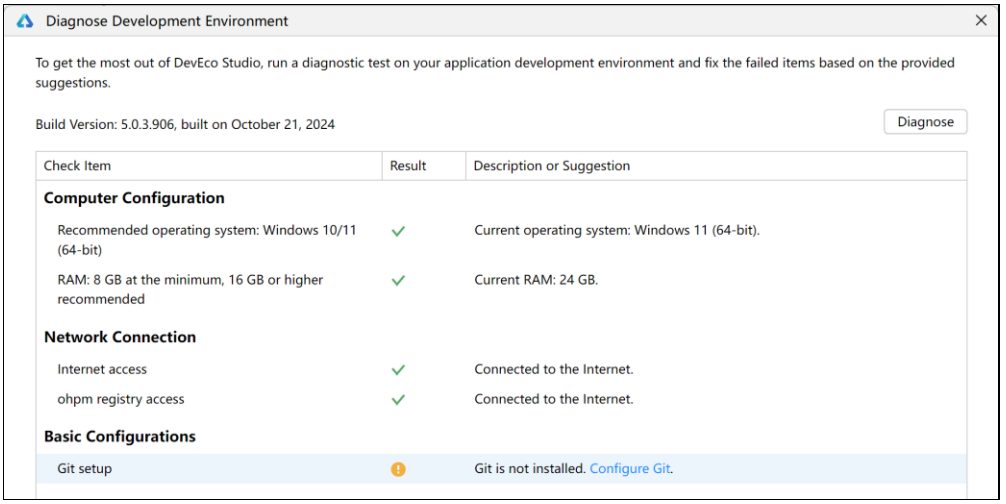

图 1-8　Windows 环境下 DevEco Studio 开发环境显示诊断结果时的界面

1.5.5　启用中文化插件

步骤 01 在工程开发界面的主菜单栏中，依次单击 File→Settings→Plugins 选项，选择 Installed 标签，在搜索框中输入 Chinese，搜索结果里将出现 Chinese（Simplified），在右侧单击 Enable 按钮，再单击右下角的 OK 按钮，如图 1-9 所示。

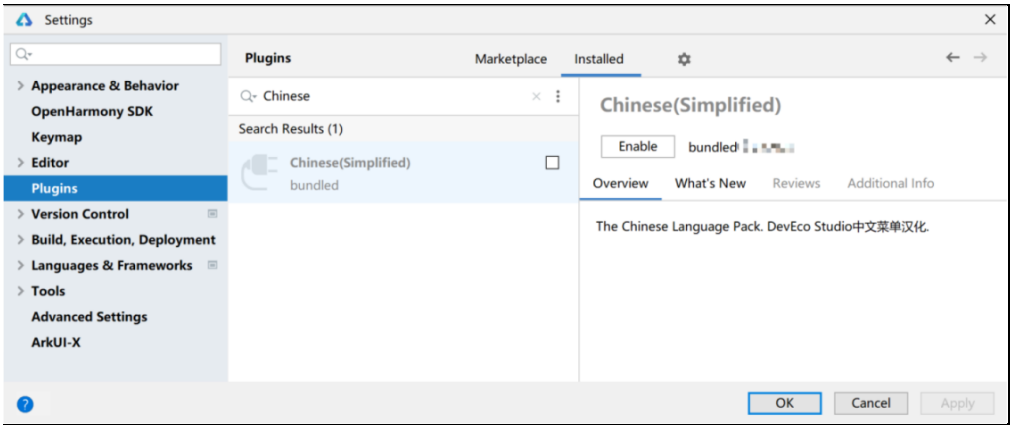

图 1-9　Windows 环境下 DevEco Studio 启用简体中文设置时的界面

步骤 02 在弹出的对话框中单击 Restart 按钮（见图 1-10），重启 DevEco Studio 后，中文设置即可生效。

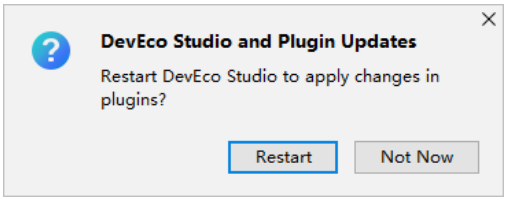

图 1-10　插件设置完成后提示是否立刻重启 DevEco Studio 的对话框

1.6 编写 HarmonyOS NEXT 入门程序

本节将带领读者编写一个简单的 HarmonyOS NEXT 入门程序——Hello World，让读者初步体验 HarmonyOS NEXT 的开发流程。

1.6.1 案例说明

1. UI 框架

HarmonyOS NEXT 提供了一套 UI 开发框架，即方舟开发框架（ArkUI 框架）。该框架为开发者提供了丰富的 UI 开发能力，包括多种组件、布局计算、动画能力、UI 交互、绘制功能等。

方舟开发框架针对不同开发需求和技术背景，提供了两种开发范式：基于 ArkTS 的声明式开发范式和兼容 JavaScript 的类 Web 开发范式。这两种开发范式的对比如表 1-1 所示。

表1-1 两种开发范式的对比

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	JavaScript 语言	数据驱动更新	界面较为简单的中小型应用和卡片	Web 前端开发人员

2. 应用模型

应用模型是 HarmonyOS NEXT 为开发者提供的应用程序抽象提炼和运行机制，包括应用程序必备的组件和框架。通过应用模型，开发者可以基于统一框架高效开发应用。

随着系统的演进，HarmonyOS NEXT 先后提供了两种应用模型：

- Stage 模型：自 HarmonyOS API 9 开始新增的模型，是目前主推并将长期演进的模型。该模型提供 AbilityStage 和 WindowStage 等类作为应用组件和窗口的“舞台”，因此而得名。本节的 Hello World 程序将基于 Stage 模型开发。
- FA（Feature Ability）模型：自 HarmonyOS API 7 开始支持的模型，目前已不再作为主要推荐的模型。

本案例将展示一个包含两个页面的开发实例，并基于 Stage 模型构建第一个 ArkTS 应用，帮助读者理解相关概念及应用开发流程。

1.6.2 创建 ArkTS 工程

创建 ArkTS 工程的步骤如下：

步骤 01 如果是首次打开 DevEco Studio，就单击 Create Project 按钮创建工程。如果已经打开了一个工程，则在菜单栏中依次单击 File→New→Create Project 新建一个工程。

步骤 02 在新建工程界面，选择 Application 类型的应用开发，选择 Empty Ability 模板，然后单击 Next

按钮进入下一步配置，如图 1-11 所示。

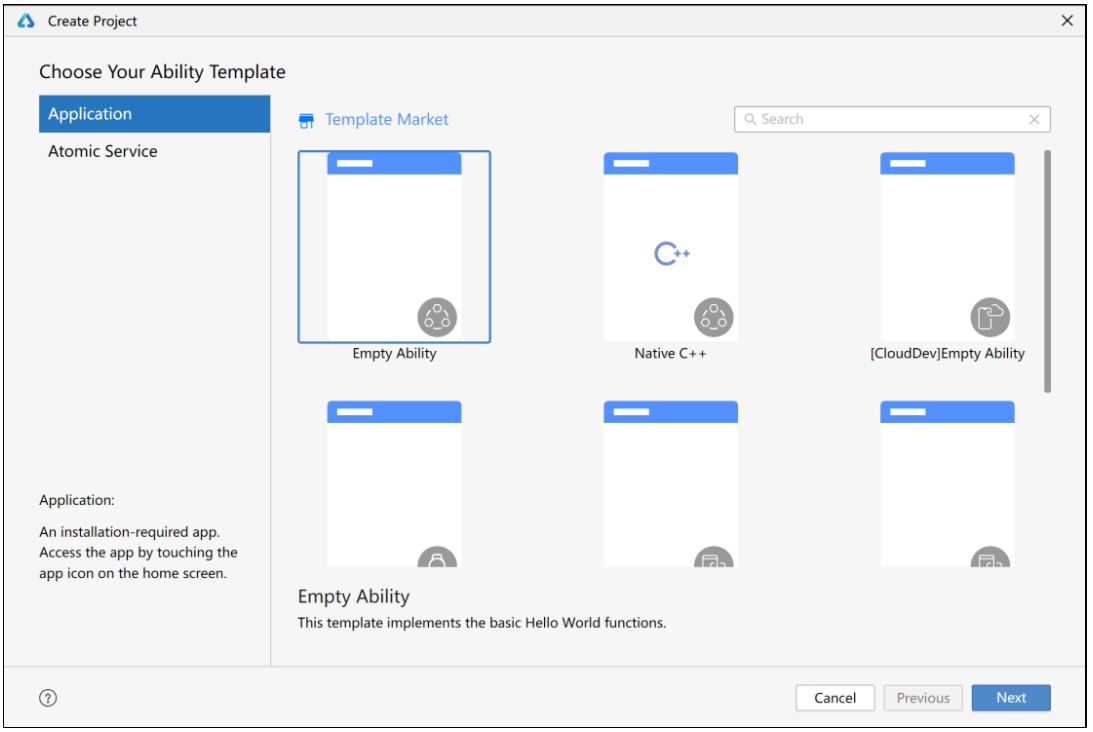

图 1-11　DevEco Studio 新建工程界面

工程模板支持的开发语言及模板说明如表 1-2 所示。

表1-2　工程模板支持的开发语言及模板说明

模板名称	说　明
Empty Ability	用于 Phone、Tablet、2in1、Car 设备的模板，展示基础的 Hello World 功能
Native C++	用于 Phone、Tablet、2in1、Car 设备的模板，作为应用调用 C++代码的示例功能
[CloudDev]Empty Ability	端云一体化开发通用模板
[Lite]Empty Ability	用于 Lite Wearable 设备的模板，展示基础的 Hello World 程序的功能。可基于此模板修改设备类型及 RuntimeOS，进行小型嵌入式设备开发
Flexible Layout Ability	用于创建跨设备应用开发的三层工程结构模板，包含 common（公共能力层）、features（基础特性层）、products（产品定制层）
Embeddable Ability	用于开发支持被其他应用嵌入运行的元服务的工程模板

步骤 03 进入工程配置界面，将 Compatible SDK 设置为 5.0.0(12)，将 Module name 设置为 entry，其他参数保持默认设置即可，如图 1-12 所示。

步骤 04 配置完成后，单击 Finish 按钮，DevEco Studio 会自动生成示例代码和相关资源，用户只需等待工程创建完成即可。

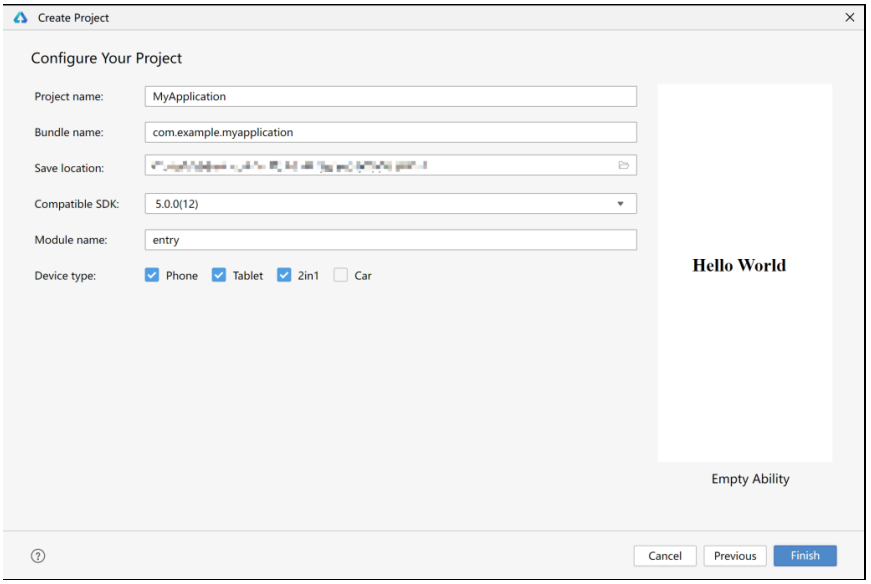

图 1-12　DevEco Studio 工程配置界面

1.6.3　ArkTS 工程目录结构（Stage 模型）

新建 ArkTS 工程后的目录结构如图 1-13 所示。

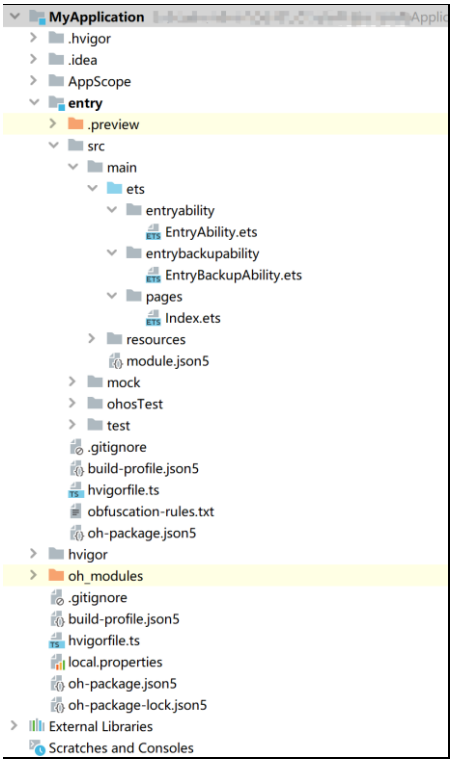

图 1-13　DevEco Studio 新建工程的目录结构

目录说明如下：

（1）AppScope > app.json5：应用的全局配置文件。

（2）entry：HarmonyOS 工程模块，编译构建生成一个 HAP 包。

（3）src > main > ets：用于存放 ArkTS 源代码。

（4）src > main > ets > entryability：应用或服务的入口。

（5）src > main > ets > pages：应用或服务包含的页面。

（6）src > main > resources：用于存放应用或服务使用到的资源文件，如图形、多媒体、字符串、布局文件等。

（7）src > main > module.json5：Stage 模型模块配置文件，主要包含 HAP 包的配置信息、应用或服务在具体设备上的配置信息以及全局配置信息。

（8）src > test > build-profile.json5：当前的模块信息，包括编译信息配置项（如 buildOption 和 targets 配置等）。其中，targets 可配置当前运行环境，默认为 HarmonyOS。

（9）src > test > hvigorfile.ts：模块级编译构建任务脚本，开发者可自定义相关任务和代码实现。

（10）oh_modules：用于存放第三方库依赖信息。

（11）oh_modules > build-profile.json5：应用级配置文件，包括签名和产品配置等信息。

（12）oh_modules > hvigorfile.ts：应用级编译构建任务脚本。

1.6.4　构建第一个页面

1. 使用文本组件

工程同步完成后，在 Project 窗口中依次单击 entry→src→main→ets→pages 选项，打开 Index.ets 文件。该文件定义的页面由 Text 组件构成，它的代码如文件 1-1 所示。

文件 1-1　Index.ets 文件中的代码

```
@Entry
@Component
struct Index {
  @State message: string = 'Hello World'

  build() {
    Row() {
      Column() {
        Text(this.message).fontSize(50).fontWeight(FontWeight.Bold)
      }.width('100%')
    }
  }
}
```

代码说明如下：

- `@Entry`：装饰器（decorator），在 ArkTS 中用于标记当前组件为应用程序的入口点，运行应用程序时，该组件会首先被加载和显示。

- `@Component`：这是另一个装饰器，用于声明后续的 struct 为一个组件，可包含状态、方法和 UI 布局。
- `struct Index { ... }`：定义名为 Index 的结构体，表示应用程序的主界面。
- `@State message: string = 'Hello World'`：声明状态变量 message，类型为 string，初始值为'Hello World'。@State 装饰器表示该变量的变化可以被追踪，当它变化时，会触发 UI 自动更新。
- `build() { ... }`：这是一个方法，用来构建组件的 UI 布局。ArkTS 使用声明式方式来构建 UI，开发者只需描述所需界面，不必关心具体实现。
- `Row() { ... }`：水平布局组件，用来水平排列子组件。
- `Column() { ... }`：垂直布局组件，嵌套在 Row 内，用于垂直排列子组件。
- `Text(this.message)`：文本组件，显示状态变量 message 的值。
- `fontSize(50)`：设置文本字体大小为 50。
- `fontWeight(FontWeight.Bold)`：设置文本字体为加粗样式。
- `width('100%')`：设置 Column 的宽度为父组件（Row）的 100%，意味着它将占据其父组件的全部宽度。

这段代码创建了一个应用程序的主界面，包含一个水平布局，该布局内嵌垂直排列的文本组件，显示“Hello World”，字体大小为 50，字体样式加粗，宽度占满父容器。

2. 添加按钮

在默认页面基础上，通过添加 Button 组件实现按钮的单击响应功能，用于跳转到另一个页面。修改后的 Index.ets 文件内的代码见文件 1-2。

文件 1-2　修改后的 Index.ets

```
@Entry
@Component
struct Index {
  @State message: string = 'Hello World'

  build() {
    Column() {
      Text(this.message).fontSize(50).fontWeight(FontWeight.Bold)
      // 添加按钮，以响应用户的单击
      Button() {
        Text('Next').fontSize(30).fontWeight(FontWeight.Bold)
      }
      .type(ButtonType.Capsule)
      .margin({ top: 20 })
      .backgroundColor('#0D9FFB')
      .width('40%')
      .height('5%')
    }.width('100%')
    .height('100%')
```

```
    }
}
```

下面来解释一下添加按钮的代码片段的含义：

- `Button() { ... }`：创建按钮组件，内部可包含文本或其他子组件。
- `Text('Next')`：按钮内部添加文本组件，显示文本“Next”。
- `fontSize(30)`：设置按钮内部文本字体大小为 30。
- `fontWeight(FontWeight.Bold)`：设置按钮内部文本的字体样式为加粗。
- `type(ButtonType.Capsule)`：设置按钮样式为胶囊形状。
- `margin({ top: 20 })`：设置按钮的上外边距为 20。
- `backgroundColor('#0D9FFB')`：设置按钮背景颜色为#0D9FFB。
- `width('40%')`：设置按钮的宽度为父组件宽度的 40%。
- `height('5%')`：设置按钮的高度为父组件高度的 5%。

ArkTS 的注释语法与 JavaScript 或 TypeScript 相同，因为它们都是基于 JavaScript 的语法。以下是几种常见的注释方式：

（1）单行注释：从两个斜杠（//）开始到行尾的所有内容均为注释内容，都会被编译器或解释器忽略，即不会被执行。例如：

```
// 这是一个单行注释
let value = 10; // 这行代码后面也有一个单行注释
```

（2）多行注释：从斜杠加星号（/*）开始到星号加斜杠（*/）结束，包围的内容为注释内容。例如：

```
/*
  这是一个多行注释的例子
  所有这些行都不会被执行
*/
let value = 20;
```

（3）HTML 风格的注释：虽然 HTML 风格的注释不是 JavaScript 或 TypeScript 的官方注释方式，但在 HTML 中嵌入 JavaScript 代码时，有时会使用 HTML 注释（<![CDATA[]]>）来避免被 HTML 解析器错误解释。例如：

```
<!-- 这不是 JavaScript 的官方注释方式，但可以在 HTML 中使用 --><script>
<!--
let value = 30;
// -->
</script>
```

在 ArkUI 框架中，编写 UI 组件时，可以通过在组件代码中添加注释来提高代码的可读性和可维护性。

接下来，在编辑窗口右上角的侧边工具栏中单击 Previewer，打开预览器。第一个页面的效果如图 1-14 所示。

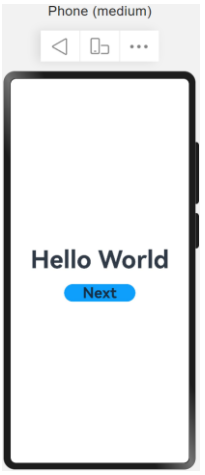

图 1-14　Hello World 页面在预览器中的显示效果

1.6.5　构建第二个页面

1. 新建第二个页面

在 Project 窗口中依次单击 entry→src→main→ets 选项，选择 pages 文件夹并右击，在弹出的快捷菜单中依次选择 New→ArkTS File 选项，新建一个 ArkTS 页面，并将它命名为 Second，最后单击 Finish 按钮。此时，文件目录结构如图 1-15 所示。

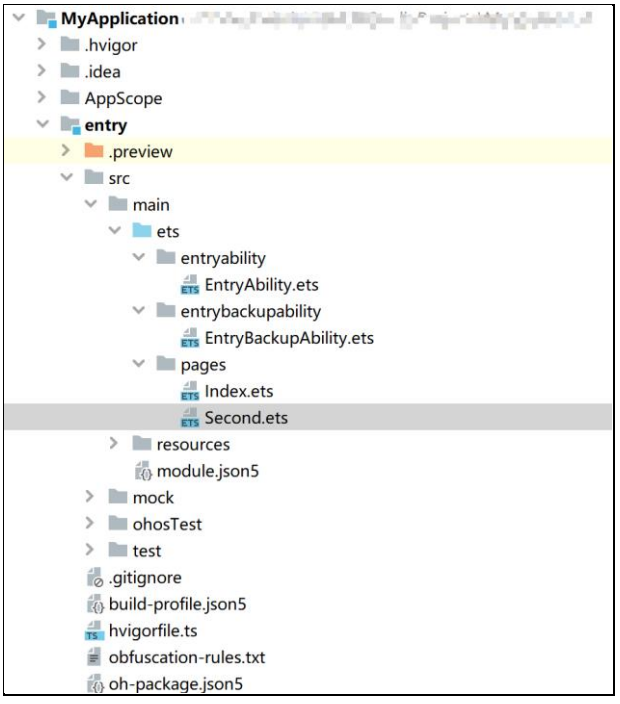

图 1-15　添加 Second.ets 后的文件目录结构

说　明
开发者也可以在右击 pages 文件夹后，在弹出的快捷菜单中依次选择 New→Page 选项，这样可以自动配置相关页面路由。

2. 配置第二个页面的路由

在 Project 窗口中依次单击 entry → src → main → resources → base → profile 选项，打开 main_pages.json 文件。在 src 字段中为第二个页面配置路由路径 pages/Second，代码见文件 1-3。

文件 1-3　main_pages.json 文件中的代码

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

这段代码是一个 JSON 对象，定义了一个包含两个元素的数组，每个元素都是指向页面路径的字符串。在 Web 开发或移动应用开发中，这类配置通常用于指定应用程序的页面路由。

- "pages/Index"：指向应用程序的主页，可能是一个名为 Index 的页面组件。在单页应用程序（SPA）或多页应用程序中，这通常是用户加载应用时看到的第一个页面。
- "pages/Second"：指向应用程序的第二个页面，可能是一个名为 Second 的页面组件。通常在用户导航到此页面时加载。

3. 添加文本及按钮

参照第一个页面，在第二个页面中添加 Text 组件、Button 组件等，并设置其样式。Second.ets 文件中的代码见文件 1-4。

文件 1-4　Second.ets 文件中的代码

```
@Entry
@Component
struct Second {
  @State message: string = 'Hi there'

  build() {
    Row() {
      Column() {
        Text(this.message).fontSize(50).fontWeight(FontWeight.Bold)
        Button() {
          Text('Back').fontSize(25).fontWeight(FontWeight.Bold)
        }
        .type(ButtonType.Capsule)
        .margin({ top: 20 })
        .backgroundColor('#0D9FFB')
        .width('40%')
        .height('5%')
```

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

代码说明如下：

- `struct Second { ... }`：定义了一个名为 Second 的结构体，包含组件的状态和 UI 布局。
- `@State message: string = 'Hi there'`：定义了一个状态变量 message，初始值为'Hi there'。状态变量用于存储动态数据，当该数据变化时 UI 会自动更新。
- `build() { ... }`：这是组件的构建函数，用于定义组件的 UI 布局。
- `Row() { ... }`：创建了 Row 组件，用于水平排列子组件。
- `Column() { ... }`：创建了 Column 组件，用于垂直排列子组件，它被放置在 Row 内部。
- `Text(this.message)`：在 Column 内部创建了 Text 组件，用于显示状态变量 message 的值。
- `fontSize(50)`：设置文本组件的字体大小为 50。
- `fontWeight(FontWeight.Bold)`：将文本组件的字体样式设置为加粗。
- `Button() { ... }`：在 Column 内部创建了 Button 组件，用于响应用户单击事件。
- `Text('Back')`：在按钮内部添加文本'Back'。
- `fontSize(25)`：设置按钮内部文本的字体大小为 25。
- `fontWeight(FontWeight.Bold)`：设置按钮内部文本的字体为加粗。
- `type(ButtonType.Capsule)`：设置按钮的样式为胶囊型。
- `margin({ top: 20 })`：设置按钮的上外边距为 20。
- `backgroundColor('#0D9FFB')`：设置按钮的背景颜色为#0D9FFB。
- `width('40%')`：设置按钮的宽度为父组件宽度的 40%。
- `height('5%')`：设置按钮的高度为父组件高度的 5%。
- `Column().width('100%')`：设置 Column 组件的宽度占满其父组件的宽度。
- `Column().height('100%')`：设置 Column 组件的高度占满其父组件的高度。
- `Row().height('100%')`：设置 Row 组件的高度占满其父组件的高度。

这段代码定义了一个包含欢迎信息和返回按钮的页面。当用户单击 Back 按钮时，通常会触发一个事件，允许用户返回到上一个页面。然而，这段代码中尚未实现按钮单击事件的处理逻辑。若需要支持返回功能，可通过添加事件监听器并定义相应的处理函数来完成。

1.6.6 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，从而完成跳转。使用页面路由需要导入 router 模块。

1. 第一个页面跳转到第二个页面

在第一个页面中，为跳转按钮绑定 onClick 事件，当用户单击该按钮时即可跳转到第二个页面。

修改后的 Index.ets 文件中的代码见文件 1-5。

文件 1-5　Index.ets 文件中的代码

```
// 导入页面路由模块
import { router } from '@kit.ArkUI'
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，用于响应用户的单击
        Button() {
          Text('Next')
            .fontSize(30)
            .fontWeight(FontWeight.Bold)
        }
        .type(ButtonType.Capsule)
        .margin({ top: 20 })
        .backgroundColor('#0D9FFB')
        .width('40%')
        .height('5%')
        // 为跳转按钮绑定 onClick 事件，单击该按钮时跳转到第二个页面
        .onClick(() => {
          console.info(`成功接收到 'Next' 按钮的单击事件。`)
          // 跳转到第二个页面
          router.pushUrl({ url: 'pages/Second' }).then(() => {
            console.info('跳转到第二个页面成功。')
          }).catch((err: BusinessError) => {
            console.error(`跳转到第二个页面失败，错误代码为：${err.code}，错误消息为：
${err.message}`)
          })
        })
      }
      .width('100%')
    }
    .height('100%')
  }
}
```

关键代码的解释如下：

- `import { router } from '@kit.ArkUI'`：导入 ArkUI 框架中的 router 模块，用于页面路由管理。
- `import { BusinessError } from '@kit.BasicServicesKit'`：导入 BusinessError 类，用于错误处理。
- `onClick(() => { ... })`：为按钮添加单击事件监听器。
- `console.info(...)`：在控制台输出信息，表示按钮单击事件被成功接收。
- `router.pushUrl({ url: 'pages/Second' })`：调用路由模块的 pushUrl 方法，尝试跳转到 pages/Second 页面。
- `then(() => { ... })`：路由跳转成功后执行的回调函数，输出跳转成功的信息。
- `catch((err: BusinessError) => { ... })`：路由跳转失败后执行的回调函数，捕获错误并输出错误代码和消息。

2. 第二个页面返回到第一个页面

在第二个页面中，为返回按钮绑定 onClick 事件，当用户单击该按钮时将返回到第一个页面。修改后的 Second.ets 文件中的代码见文件 1-6。

文件 1-6　Second.ets 文件中的代码

```
// 导入页面路由模块
import { router } from '@kit.ArkUI'
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(25).fontWeight(FontWeight.Bold)
        }
        .type(ButtonType.Capsule)
        .margin({ top: 20 })
        .backgroundColor('#0D9FFB')
        .width('40%')
        .height('5%')
        // 为返回按钮绑定 onClick 事件，单击该按钮时返回到第一个页面
        .onClick(() => {
          console.info(`成功接收到'Back'按钮的单击事件。`)
          try {
```

```
            // 返回第一个页面
            router.back()
            console.info('成功返回到第一个页面。')
          } catch (err) {
            let code = (err as BusinessError).code
            let message = (err as BusinessError).message
            console.error(`跳转到第一个页面失败。错误代码为：${code}，错误消息为：${message}`)
          }
        })
      }
      .width('100%')
    }
    .height('100%')
  }
}
```

关键代码的解释如下：

- `onClick(() => { ... })`：为按钮添加单击事件监听器。
- `console.info(...)`：在控制台输出信息，表示按钮单击事件被成功接收。
- `router.back()`：调用路由模块的 back 方法，尝试返回到前一个页面。
- `console.info('成功返回到第一个页面。')`：如果成功返回，则输出信息到控制台。
- `catch (err) { ... }`：如果返回时发生错误，则捕获错误并处理。
- `let code = (err as BusinessError).code`：从错误对象中获取错误代码。
- `let message = (err as BusinessError).message`：从错误对象中获取错误消息。
- `console.error(...)`：如果返回失败，则输出错误信息到控制台。

打开 Index.ets 文件，先单击预览器中的按钮进行刷新，然后在预览器中单击第一个页面中的 Next 按钮，即可跳转到第二个页面；单击第二个页面中的 Back 按钮，将返回到第一个页面，如图 1-16 所示。

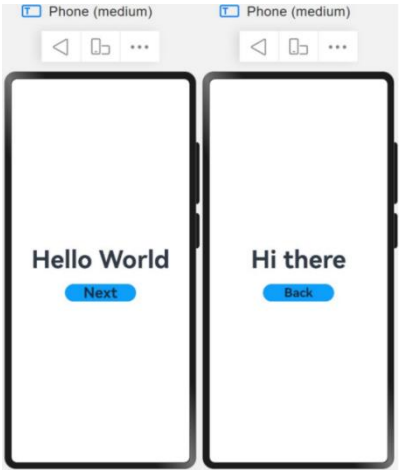

图 1-16　预览器中页面显示效果

1.7 本章小结

华为推出的 HarmonyOS NEXT 是一款面向未来的智能操作系统，具有微内核架构、全场景覆盖、分布式创新、性能流畅、安全至上、开放生态的特点。

HarmonyOS NEXT 的深远影响包括加速设备智能化进程，激发技术创新活力和重塑市场竞争格局。对于开发者而言，它不仅是一个平台，更是一个充满机遇的创新舞台，提供了全栈自研能力、原生应用开发、统一生态部署、安全与隐私保障、开放合作生态和丰富的技术支持与社区资源。

在开发环境搭建方面，本章介绍了如何从 HarmonyOS Developer 官网下载并安装 DevEco Studio，配置开发环境，并启用中文化插件。

通过入门程序示例，本章展示了创建 ArkTS 工程的基本流程，包括构建页面、添加组件和实现页面间跳转的具体方法。

HarmonyOS NEXT 的普及将推动智能设备的快速发展，为开发者提供更广阔的创新空间，进一步促进智能生态的繁荣。

1.8 本章习题

1. HarmonyOS NEXT 的架构设计对开发者在资源管理方面有哪些帮助？
2. ArkTS 语言在 HarmonyOS NEXT 中主要应用于哪些方面？
3. 在 HarmonyOS NEXT 中，声明式 UI 后端引擎提供了哪些功能？
4. 在 HarmonyOS NEXT 的应用开发中，默认的长度单位是什么？
5. DevEco Studio 提供了哪些主要功能？
6. 在 ArkTS 中，如何实现页面间的跳转？
7. 请简述在 HarmonyOS NEXT 中创建 ArkTS 工程的基本步骤。