OpenHarmony

3.2 Release 开发者文档介绍

Neen Yang

OpenHarmony SIG Docs Leader

目录 Contents

OpenHarmony文档概览

OpenHarmony

1000+开发者共建OpenHarmony文档内容

1078+	5k	2.8k	912
开发者贡献	开发者加星	开发者Fork	开发者关注

2.5w次	2.5倍	天级
Pulls Request	内容增长	官网文档更新

OpenHarmony应用开发者文档介绍

OpenHarmony

6本指南内容重构升级；新发布《学习ArkTS语言》、《错误码参考》、《文件管理》

OpenHarmony官网

说明：
New：新增配套文档
Enhanced：已有文档增强/优化

更易上手的关键特性开发指南

OpenHarmony

1 识别TOP指南

通过开发者访谈、原声、浏览量数据，识别出开发者关注的TOP特性开发指南

2 分析典型任务场景

针对开发者实际使用及系统推荐的典型开发场景/任务

补充场景化开发指导70+个

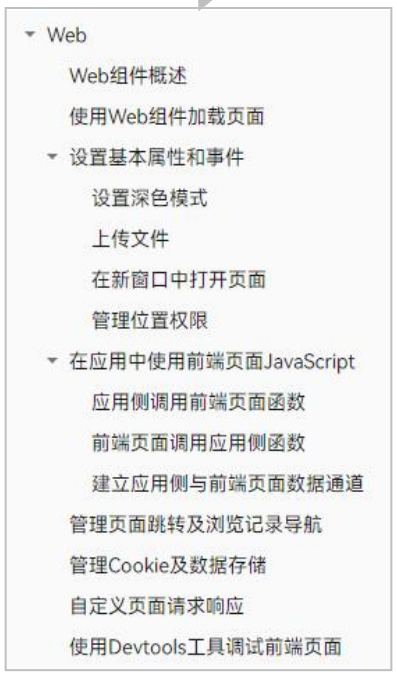
3 补充必备知识点

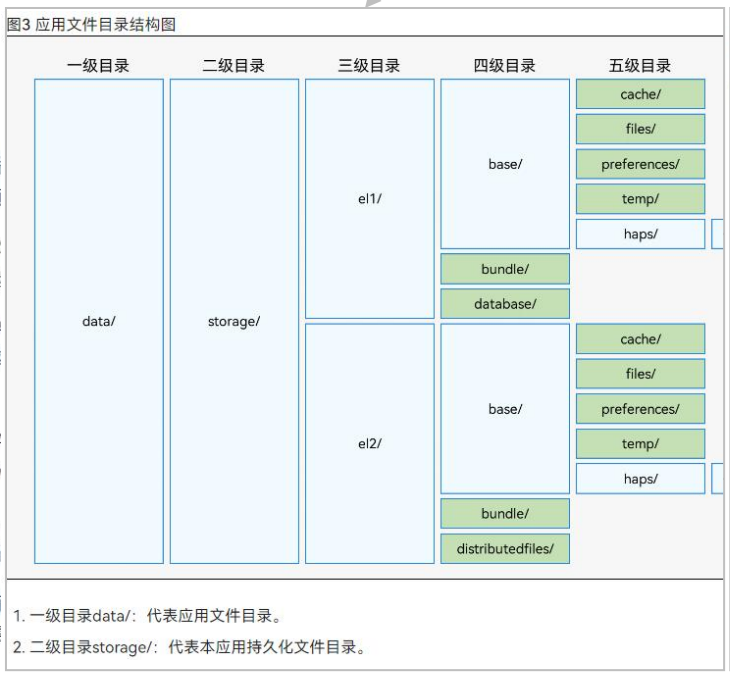
在完善开发场景的同时，提供支撑开发活动的必备知识点。

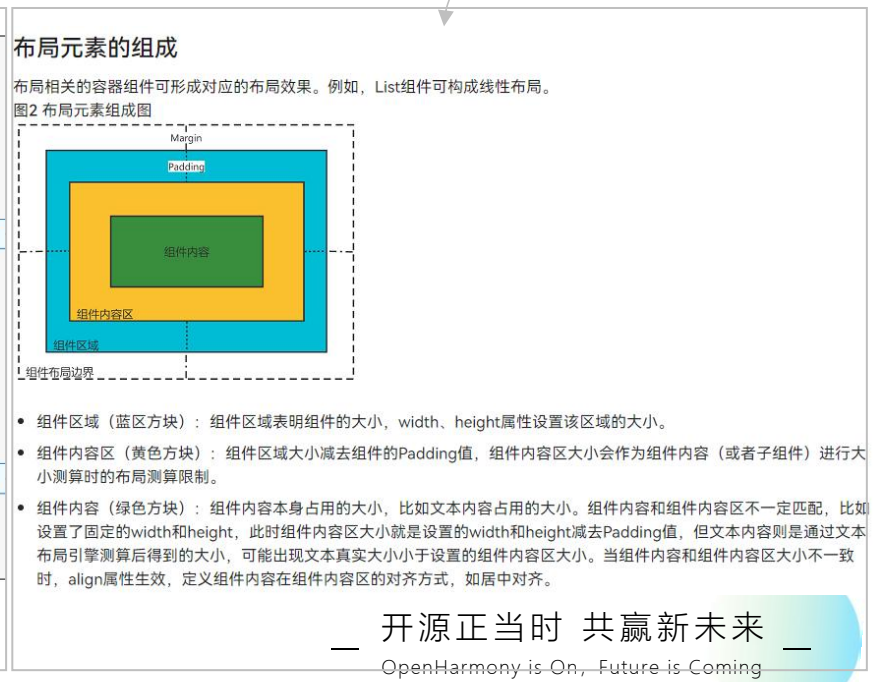
如相关概念、原理机制、约束限制，让开发者知其所以然，更快掌握相关能力

4 优化文档架构和内容逻辑

基于信息架构设计与技术写作相关专业能力，优化文档架构与内容逻辑，更符合开发者认知习惯；图文结合，力求内容清晰易懂





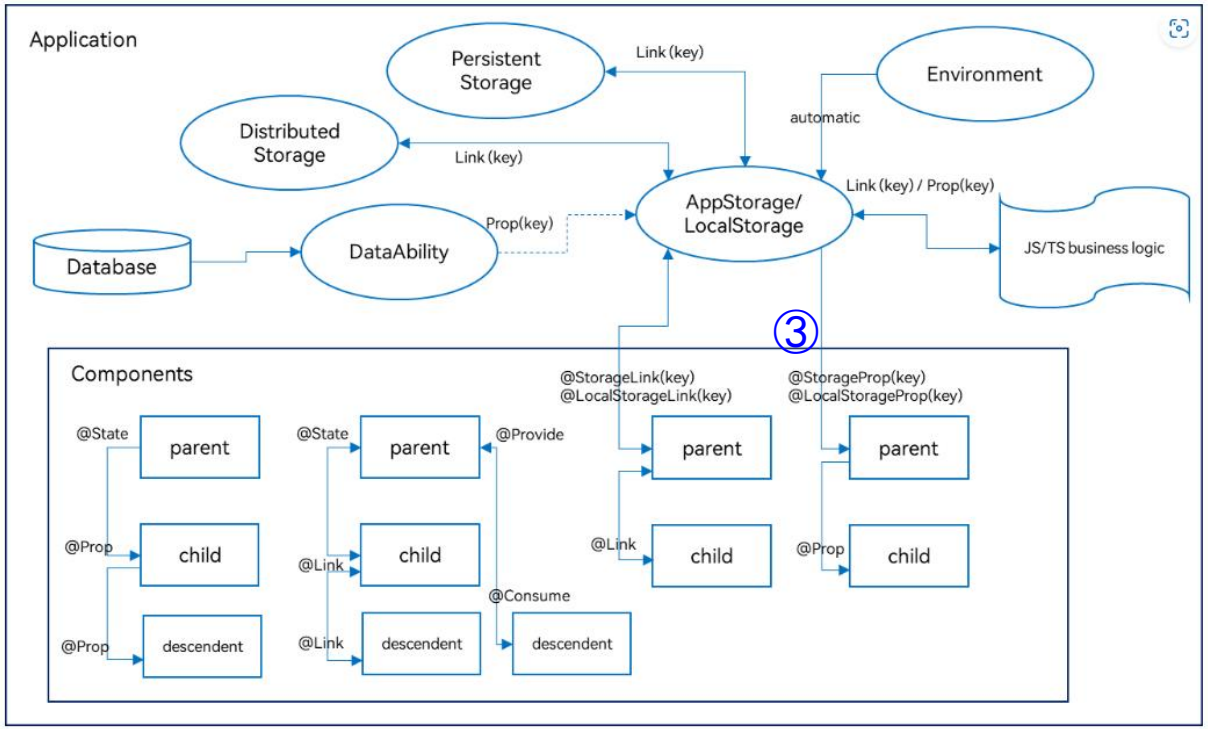


ArkTS语言开发指南：更易上手和入门

OpenHarmony

- 学习ArkTS语言
  - 初识ArkTS语言
  - 基本语法
    - 基本语法概述
    - 声明式UI描述
    - 自定义组件 ①
      - 创建自定义组件
      - 页面和自定义组件生命周期
    - @Builder：自定义构建函数
    - @BuilderParam：引用@Builder函数
    - @Styles：定义组件重用样式
    - @Extend：定义扩展组件样式
    - stateStyles：多态样式
  - 状态管理 ②
    - 状态管理概述
    - 管理组件拥有的状态
      - @State：组件内状态
      - @Prop：父子单向同步
      - @Link：父子双向同步
      - @Provide和@Consume：与后代组件双向同步
      - @Observed和@ObjectLink：嵌套类对象属性变化
    - 管理应用拥有的状态
      - 管理应用拥有的状态概述
      - LocalStorage：页面级UI状态存储
      - AppStorage：应用全局的UI状态存储
      - PersistentStorage：持久化存储UI状态
      - Environment：设备环境查询
    - 其他状态管理
      - 其他状态管理概述
      - @Watch：状态变量更改通知
      - $$语法：内置组件双向同步
  - 渲染控制
    - 渲染控制概述
    - if/else：条件渲染
    - ForEach：循环渲染
    - LazyForEach：数据懒加载

图示如下，具体装饰器的介绍，可详见管理组件拥有的状态和管理应用拥有的状态。开发者可以灵活地利用这些能力来实现数据和UI的联动。



上图中，Components部分的装饰器为组件级别的状态管理，Application部分为应用的状态管理。开发者可以通过@StorageLink/@LocalStorageLink和@StorageProp/@LocalStorageProp实现应用和组件状态的双向和单向同步。图中箭头方向为数据同步方向，单箭头为单向同步，双箭头为双向同步。

管理组件拥有的状态，即图中Components级别的状态管理：

- @State：@State装饰的变量拥有其所属组件的状态，可以作为其子组件单向和双向同步的数据源。当其数值改变时，会引起相关组件的渲染

页面和自定义组件生命周期

更新时间: 2023-04-11 09:09

在开始之前，我们先明确自定义组件和页面的关系：

- 自定义组件：@Component装饰的UI单元，可以组合多个系统组件实现UI的复用。
- 页面：即应用的UI页面。可以由一个或者多个自定义组件组成，@Entry装饰的自定义组件为页面的入口组件，即页面的根节点，一个页面有且仅能有一个@Entry。只有被@Entry装饰的组件才可以调用页面的生命周期。

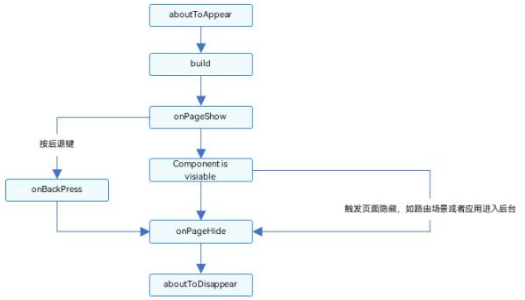
页面生命周期，即被@Entry装饰的组件生命周期，提供以下生命周期接口：

- onPageShow：页面每次显示时触发。
- onPageHide：页面每次隐藏时触发一次。
- onBackPress：当用户点击返回按钮时触发。

组件生命周期，即一般用@Component装饰的自定义组件，提供以下生命周期接口：

- aboutToAppear：组件即将出现时回调该接口，具体时机为在创建自定义组件的新实例后，在执行其build()函数之前执行。
- aboutToDisappear：在自定义组件即将析构销毁时执行。

生命周期流程如下图所示，下图展示的是被@Entry装饰的组件（首页）生命周期。



①提供自定义组件和组件生命周期丰富内容。

②完善各种装饰器概念、场景、约束限制。

③图文结合，清晰描述语法实现的原理机制。

应用模型开发指南：完善概念-原理-场景化开发指导

OpenHarmony

②

OpenHarmony应用模型概况

①

随着系统的演进发展，OpenHarmony先后提供了两种应用模型：

- FA（Feature Ability）模型：OpenHarmony API 7开始支持的模型，已经不再主推。
- Stage模型：OpenHarmony API 9开始新增的模型，是目前主推且会长期演进的模型。在该模型中，由于提供了AbilityStage、WindowStage等类作为应用组件和Window窗口的“舞台”，因此称这种应用模型为Stage模型。

Stage模型之所以成为主推模型，源于其设计思想。Stage模型的设计基于如下出发点。

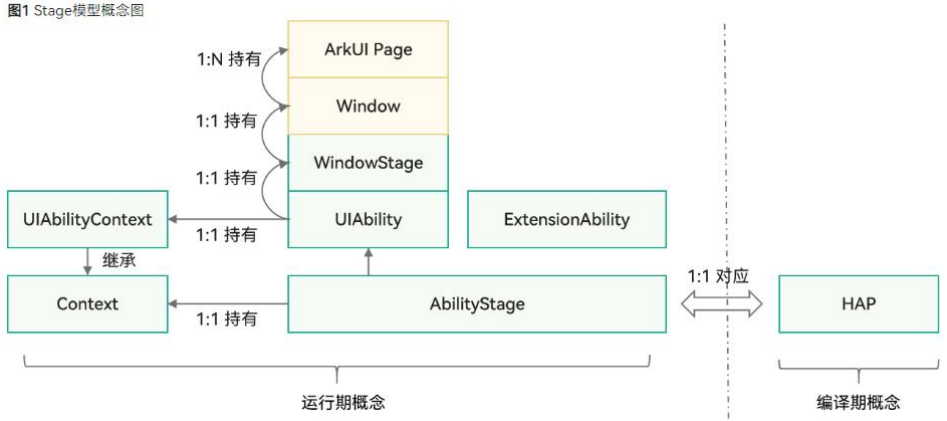
1. **为复杂应用而设计**
   - 多个应用组件共享同一个ArkTS引擎（运行ArkTS语言的虚拟机）实例，应用组件之间可以方便的共享对象和状态，同时减少复杂应用运行对内存的占用。
   - 采用面向对象的开发方式，使得复杂应用代码可读性高、易维护性好、可扩展性强。
2. **原生支持应用组件级的跨端迁移和多端协同**

   Stage模型实现了应用组件与UI解耦：
   - 在跨端迁移场景下，系统在多设备的应用组件之间迁移数据/状态后，UI便可利用ArkUI的声明式特点，通过应用组件中保存的数据/状态恢复用户界面，便捷实现跨端迁移。
   - 在多端协同场景下，应用组件具备组件间通信的RPC调用能力，天然支持跨设备应用组件的交互。
3. **支持多设备和多窗口形态**

   应用组件管理和窗口管理在架构层面解耦：
   - 便于系统对应用组件进行裁剪（无屏设备可裁剪窗口）。
   - 便于系统扩展窗口形态。
   - 在多设备（如桌面设备和移动设备）上，应用组件可使用同一套生命周期。
4. **平衡应用能力和系统管控成本**

   Stage模型重新定义应用能力的边界，平衡应用能力和系统管控成本。
   - 提供特定场景（如卡片、输入法）的应用组件，以便满足更多的使用场景。
   - 规范化后台进程管理：为保障用户体验，Stage模型对后台应用进程进行了有序治理，应用程序不能随意驻留在后台，同时应用后台行为受到严格管理，防止恶意应用行为。

图1 Stage模型概念图



③

UIAbility组件间交互（设备内）

UIAbility是系统调度的最小单元。在设备内的功能模块之间跳转时，会涉及到启动特定的UIAbility，该UIAbility可以是应用内的其他UIAbility，也可以是其他应用的UIAbility（例如启动三方支付UIAbility）。

本章节将从如下场景分别介绍设备内UIAbility间的交互方式。对于跨设备的应用组件交互，请参见应用组件跨设备交互（流转）。

- 启动应用内的UIAbility
- 启动应用内的UIAbility并获取返回结果
- 启动其他应用的UIAbility
- 启动其他应用的UIAbility并获取返回结果
- 启动UIAbility的指定页面
- 通过Call调用实现UIAbility交互（仅对系统应用开放）

启动应用内的UIAbility

当一个应用内包含多个UIAbility时，存在应用内启动UIAbility的场景。例如在支付应用中从入口UIAbility启动收付款UIAbility。

假设应用中有两个UIAbility：EntryAbility和FuncAbility（可以在应用的一个Module中，也可以在的不同Module中），需要从EntryAbility的页面中启动FuncAbility。

1. 在EntryAbility中，通过调用startAbility()方法启动UIAbility，want为UIAbility实例启动的入口参数，其中bundleName为待启动应用的Bundle名称，abilityName为待启动的UIAbility名称，moduleName在待启动的UIAbility属于不同的Module时添加，parameters为自定义信息参数。示例中的context的获取方式参见获取UIAbility的Context属性。

```ts
let wantInfo = {
  deviceId: '', // deviceId为空表示本设备
  bundleName: 'com.example.myapplication',
```

①各类基本概念解析。

②详实的原理机制解读。

③ 20+个开发场景指导高效开发。

ArkUI开发指南：场景历程完善、丰富

OpenHarmony

开发流程 ①

使用UI开发框架开发应用时，主要涉及如下开发过程。开发者可以先通过第一个入门实例了解整个应用的UI开发过程。

任务	简介	相关指导
学习ArkTS	介绍了ArkTS的基本语法、状态管理和渲染控制的场景。	• 基本语法 • 状态管理 • 渲染控制
开发布局	介绍了几种常用的布局方式以及如何提升布局性能。	• 常用布局 • 布局性能
添加组件	介绍了几种常用的内置组件、自定义组件以及通过API方式支持的界面元素。	• 常用组件 • 自定义组件 • 气泡和菜单
设置页面路由和组件导航	介绍了如何设置页面路由以及组件间的导航。	• 页面路由 • 组件导航
显示图形	介绍了如何显示图片、绘制自定义几何图形以及使用画布绘制自定义图形。	• 图片 • 几何图形 • 画布
使用动画	介绍了组件和页面使用动画的典型场景。	• 页面内的动画 • 页面间的动画
绑定事件	介绍了事件的基本概念和如何使用通用事件和手势事件。	• 通用事件 • 手势事件

①完备的UI开发流程，提供各环节开发指导。

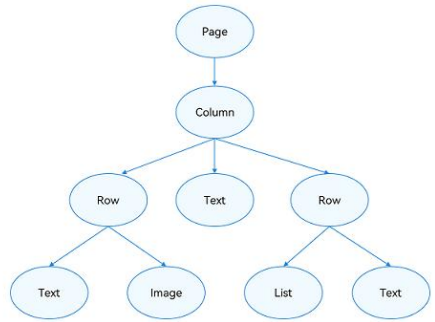
②清晰的布局概念、使用场景及约束，丰富UI基础知识。

③布局、常见组件、动效等开发场景丰富3倍。

布局结构 ②

布局的结构通常是分层级的，代表了用户界面中的整体架构。一个常见的页面结构如下所示：

图1 常见页面结构图



为实现上述效果，开发者需要在页面中声明对应的元素。其中，Page表示页面的根节点，Column/Row等元素为系统组件ArkUI提供了不同的布局组件来帮助开发者实现对应布局的效果，例如Row用于实现线性布局。

如何选择布局

声明式UI提供了以下8种常见布局，开发者可根据实际应用场景选择合适的布局进行页面开发。

布局	应用场景
线性布局（Row、Column）	如果布局内子元素为复数个，且能够以某种方式线性排列时优先考虑此布局。
层叠布局（Stack）	组件需要有堆叠效果时优先考虑此布局，层叠布局的堆叠效果不会占用或影响其他同容器内子组件的布局空间。例如Panel作为子组件弹出时将其他组件覆盖更为合理，则优先考虑在外层使用堆叠布局。
弹性布局（Flex）	弹性布局是与线性布局类似的布局方式。区别在于弹性布局默认能够使子组件压缩或拉伸。在子组件需要计算拉伸或压缩比例时优先使用此布局，可使得多个容器内子组件能有更好的视觉上的填充容器效果。
相对布局（RelativeContainer）	相对布局是在二维空间中的布局方式，不需要遵循线性布局的规则，布局方式更为自由。通过在子组件上设置锚点规则（AlignRules）使子组件能够将自己在横轴、纵轴中的位置与容器或容器内其他子组件的位置对齐。设置的锚点规则可以天然支持子元素压缩、拉伸，堆叠或形成多行效果。在页面元素分布复杂或通过线性布局会使容器嵌套层数过深时推荐使用。
栅格布局（GridRow、GridCol）	栅格是多设备场景下通用的辅助定位工具，通过将空间分割为有规律的栅格。栅格不同于网格布局固定的空间划分，可以实现不同设备下不同的布局，空间划分更随心所欲，从而显著降低适配不同屏幕尺寸的设计及开发成本，使得整体设计和开发流程更有秩序和节奏感，同时也保证多设备上应用显示的协调性和一致性，提升用户体验。推荐内容相同但布局不同时使用。
媒体查询（@ohos.mediaquery）	媒体查询可根据不同设备类型或同设备不同状态修改应用的样式。例如根据设备和应用的不同属性信息设计不同的布局，以及屏幕发生动态改变时更新应用的页面布局。
列表（List）	使用列表可以轻松高效地显示结构化、可滚动的信息。在ArkUI中，列表具有垂直和水平布局能力和自适应交叉轴方向上排列个数的布局能力，超出屏幕时可以滚动。列表适合用于呈现同类数据类型或数据类型集，例如图片和文本。
网格（Grid）	网格布局具有较强的页面均分能力，子组件占比控制能力，是一种重要自适应布局。网格布局可以控制元素所占的网格数量、设置子组件横跨几行或者几列，当网格容器尺寸发生变化时，所有子组件以及间距等比例调整。推荐在需要按照固定比例或者均匀分配空间的布局场景下使用，例如计算器、相册、日历等。
轮播（Swiper）	轮播组件通常用于实现广告轮播、图片预览、可滚动应用等。

媒体开发指南：丰富实现机制原理、场景指导

OpenHarmony

- 媒体
  - 媒体应用开发概述
  - 音视频
    - 音视频概述
    - AVPlayer和AVRecorder
    - 音频播放
      - 音频播放开发概述
      - 使用AVPlayer开发音频播放功能
      - 使用AudioRenderer开发音频播放功能
      - 使用OpenSL ES开发音频播放功能
      - 使用TonePlayer开发音频播放功能（仅对系统应用开放）
      - 多音频播放的并发策略
      - 播放音量管理
      - 音频播放流管理
      - 音频输出设备管理
      - 分布式音频播放（仅对系统应用开放）
    - 音频录制
      - 音频录制开发概述
      - 使用AVRecorder开发音频录制功能
      - 使用AudioCapturer开发音频录制功能
      - 使用OpenSLES开发音频录制功能
      - 管理麦克风
      - 音频录制流管理
      - 音频输入设备管理
    - 音频通话
      - 音频通话开发概述
      - 开发音频通话功能
    - 视频播放
    - 视频录制

音频播放开发概述

如何选择音频播放开发方式 ①

在OpenHarmony系统中，多种API都提供了音频播放开发的支持，不同的API适用于不同音频数据格式、音频资源来源、音频使用场景，甚至是不同开发语言。因此，选择合适的音频播放API，有助于降低开发工作量，实现更佳的音频播放效果。

- AVPlayer：功能较完善的音频、视频播放ArkTS/JS API，集成了流媒体和本地资源解析、媒体资源解封装、音频解码和音频输出功能。可以用于直接播放mp3、m4a等格式的音频文件，不支持直接播放PCM格式文件。
- AudioRenderer：用于音频输出的的ArkTS/JS API，仅支持PCM格式，需要应用需要持续写入音频数据进行工作。应用可以在输入前添加数据预处理，如设定音频文件的采样率、位宽等，要求开发者具备音频处理的基础知识，适用于更专业、更多样化的媒体播放应用开发。
- OpenSLES：一套跨平台标准化的音频Native API，目前阶段唯一的音频类Native API，同样提供音频输出能力，仅支持PCM格式，适用于从其他嵌入式平台移植，或依赖在Native层实现音频输出功能的播放应用使用。
- TonePlayer：拨号和回铃音播放ArkTS/JS API，只能在固定的类型范围内选择播放内容，无需输入媒体资源或音频数据，适用于拨号盘按键和通话回铃音的特定场景。该功能当前仅对系统应用开放。
- 在音频播放中，应用时常需要用到一些急促简短的音效，如相机快门音效、按键音效、游戏射击音效等，当前只能使用AVPlayer播放音频文件替代实现，在OpenHarmony后续版本将会推出相关接口来支持该场景。

音频录制开发概述

如何选择音频录制开发方式 ①

在OpenHarmony系统中，多种API都提供了音频录制开发的支持，不同的API适用于不同录音输出格式、音频使用场景或不同开发语言。因此，选择合适的音频录制API，有助于降低开发工作量，实现更佳的音频录制效果。

- AVRecorder：功能较完善的音频、视频录制ArkTS/JS API，集成了音频输入录制、音频编码和媒体封装的功能。开发者可以直接调用设备硬件如麦克风录音，并生成m4a音频文件。
- AudioCapturer：用于音频输入的的ArkTS/JS API，仅支持PCM格式，需要应用持续读取音频数据进行工作。应用可以在音频输出后添加数据处理，要求开发者具备音频处理的基础知识，适用于更专业、更多样化的媒体播放应用开发。
- OpenSLES：一套跨平台标准化的音频Native API，目前阶段唯一的音频类Native API，同样提供音频输入原子能力，仅支持PCM格式，适用于从其他嵌入式平台移植，或依赖在Native层实现音频输入功能的录音应用使用。

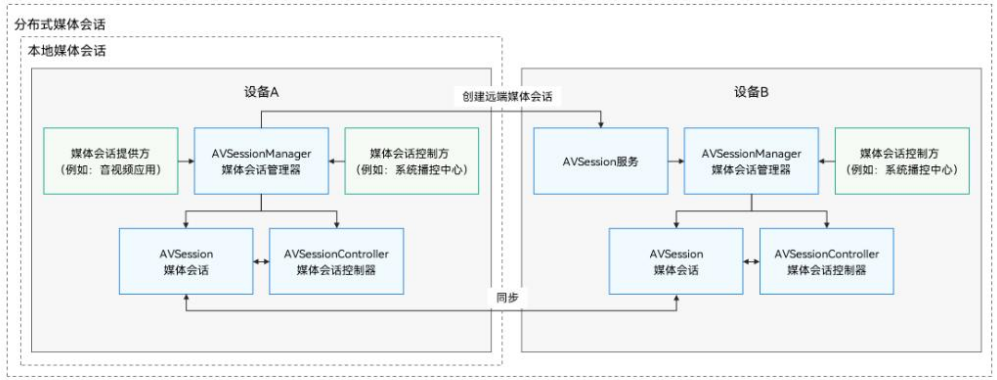
①音频播放和录制不同能力提供开发方式选择场景

②场景翻倍，新增TonePlayer、多音频播放、分布式媒体会话等

③图文结合、层层拆分，拆解复杂机制与原理

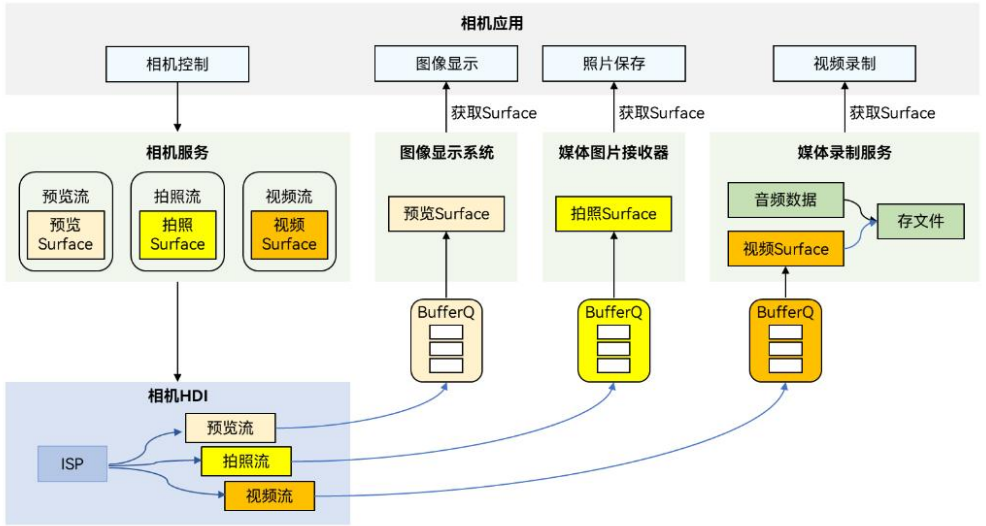
媒体会话交互过程 ③

媒体会话分为本地和分布式两种场景。



- 本地媒体会话 本地媒体会话在本地设备中的媒体会话提供方和媒体会话控制方之间建立连接，实现系统中音视频应用统一的媒体播放控制和媒体信息显示。
- 分布式媒体会话 分布式媒体会话在跨设备场景中的媒体会话提供方和媒体会话控制方之间建立连接，实现音视频应用跨设备的媒体播放控制和媒体信息显示。例如，将设备A中播放的内容投播到设备B，并在设备B中进行播放控制。

图2 相机开发模型



相机应用通过控制相机，实现图像显示（预览）、照片保存（拍照）、视频录制（录像）等基础操作。在实现基本操作过程中，相机服务会控制相

文件管理开发指南：概念更明晰，场景更丰富

OpenHarmony

- 文件管理
  - 文件管理概述 ①
  - 应用文件
    - 应用文件概述
    - 应用沙箱目录 ②
    - 应用文件访问与管理
      - 应用文件访问
      - 应用文件上传下载
      - 应用及文件系统空间统计
    - 向应用沙箱推送文件
    - 应用文件分享
  - 用户文件
    - 用户文件概述
    - 选择与保存用户文件（FilePicker）
      - 选择用户文件
      - 保存用户文件
      - 开发用户文件管理器（仅对系统应用开放）
      - 管理外置存储设备（仅对系统应用开放）
  - 分布式文件系统
    - 分布式文件系统概述
    - 设置分布式文件数据等级
    - 跨设备文件访问

①按文件分类类型提供开发指导。

②完善应用沙箱概念、原理、详细介绍不同文件路径及场景。

在文件管理模块中，按文件所有者的不同，有如下文件分类模型，其示意图如下面文件分类模型示意图：

- 应用文件：文件所有者为应用，包括应用安装文件、应用资源文件、应用缓存文件等。
- 用户文件：文件所有者为登录到该终端设备的用户，包括用户私有的图片、视频、音频、文档等。
- 系统文件：与应用和用户无关的其他文件，包括公共库、设备文件、系统资源文件等。这类文件不需要开发者进行文件管理，本文不展开介绍。

按文件系统管理的文件存储位置（数据源位置）的不同，有如下文件系统分类模型： ①

- 本地文件系统：提供本地设备或外置存储设备（如U盘、移动硬盘）的文件访问能力。本地文件系统是最基本的文件系统，本文不展开介绍。
- 分布式文件系统：提供跨设备的文件访问能力。所谓跨设备，指文件不一定存储在本地设备或外置存储设备，而是通过计算机网络与其他分布式设备相连。

图1 文件分类模型示意图

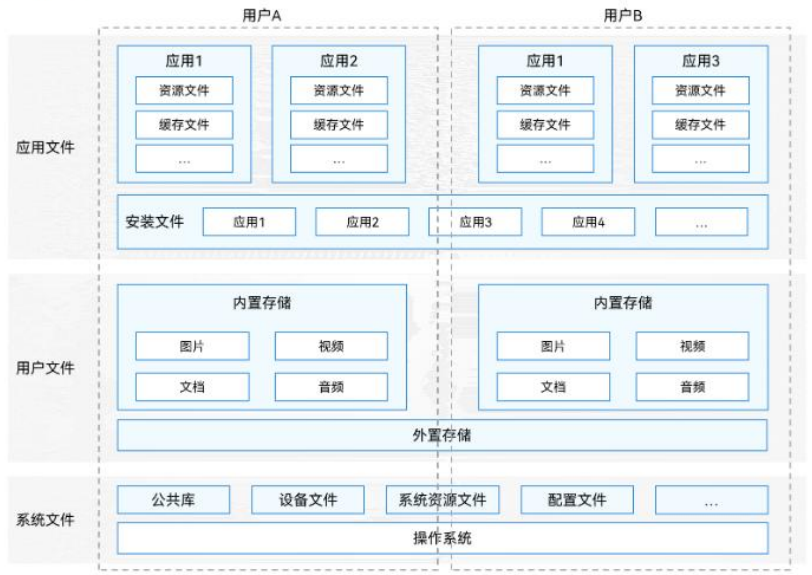


图3 应用文件目录结构图

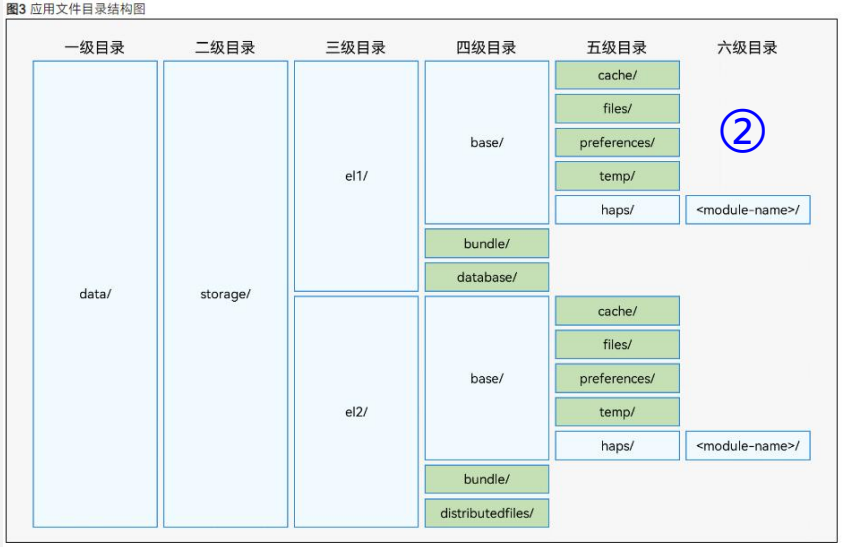


表1 应用文件路径详细说明 ②

目录名	Context属性名称	类型	说明
bundle	bundleCodeDir	安装文件路径	应用安装后的app的hap资源包所在目录；随应用卸载而清理。
base	NA	本设备文件路径	应用在本设备上存放持久化数据的目录，子目录包含files/、cache/、temp/和haps/；随应用卸载而清理。
database	databaseDir	数据库路径	应用在el1加密条件下存放通过分布式数据库服务操作的文件目录；随应用卸载而清理。
distributedfiles	distributedFilesDir	分布式文件路径	应用在el2加密条件下存放分布式文件的目录，应用将文件放入该目录可分布式跨设备直接访问；随应用卸载而清理。
files	filesDir	应用通用文件路径	应用在本设备内部存储上通用的存放默认长期保存的文件路径；随应用卸载而清理。
cache	cacheDir	应用缓存文件路径	应用在本设备内部存储上用于缓存下载的文件或可重新生成的缓存文件的路径，应用cache目录大小超过配额或者系统空间达到一定条件，自动触发清理该目录下文件；用户通过系统空间管理类应用也可能触发清理该目录。应用需判断文件是否仍存在，决策是否需重新缓存该文件。
preferences	preferencesDir	应用首选项文件路径	应用在本设备内部存储上通过数据库API存储配置类或首选项的目录；随应用卸载而清理。详见通过用户首选项实现数据持久化。
temp	tempDir	应用临时文件路径	应用在本设备内部存储上仅在应用运行期间产生和需要的文件，应用退出后即清理。

数据管理开发指南：场景更清晰

OpenHarmony

- 数据管理 ①
  - 数据管理概述
  - 应用数据持久化
    - 应用数据持久化概述
    - 通过用户首选项实现数据持久化
    - 通过键值型数据库实现数据持久化
    - 通过关系型数据库实现数据持久化
  - 同应用跨设备数据同步（分布式）
    - 同应用跨设备数据同步概述
    - 键值型数据库跨设备数据同步
    - 关系型数据库跨设备数据同步
    - 分布式数据对象跨设备数据同步
  - 数据可靠性与安全性
    - 数据可靠性与安全性概述
    - 数据库备份与恢复
    - 数据库加密
    - 基于设备分类和数据分级的访问控制
  - 同设备跨应用数据共享（仅对系统应用开放）
    - 同设备跨应用数据共享概述
    - 通过DataShareExtensionAbility实现数据共享
    - 通过静默数据访问实现数据共享

文档 > 应用开发文档 > 开发 > 数据管理 > 应用数据持久化 ②

应用数据持久化概述

应用数据持久化，是指应用将内存中的数据通过文件或数据库的形式保存到设备上。内存中的数据形态通常是任意的数据结构或数据对象，存储介质上的数据形态可能是文本、数据库、二进制文件等。

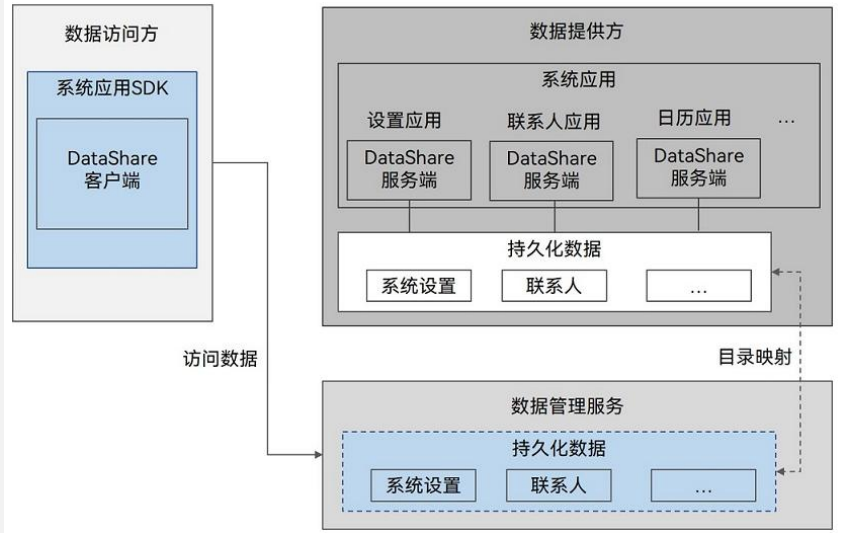
OpenHarmony标准系统支持典型的存储数据形态，包括用户首选项、键值型数据库、关系型数据库。

开发者可以根据如下功能介绍，选择合适的数据形态以满足自己应用数据的持久化需要。

- **用户首选项（Preferences）**：通常用于保存应用的配置信息。数据通过文本的形式保存在设备中，应用使用过程中会将文本中的数据全量加载到内存中，所以访问速度快、效率高，但不适合需要存储大量数据的场景。
- **键值型数据库（KV-Store）**：一种非关系型数据库，其数据以“键值”对的形式进行组织、索引和存储，其中“键”作为唯一标识符。适合很少数据关系和业务关系的业务数据存储，同时因其在分布式场景中降低了解决数据库版本兼容问题的复杂度，和数据同步过程中冲突解决的复杂度而被广泛使用。相比于关系型数据库，更容易做到跨设备跨版本兼容。
- **关系型数据库（RelationalStore）**：一种关系型数据库，以行和列的形式存储数据，广泛用于应用中的关系型数据的处理，包括一系列的增、删、改、查等接口，开发者也可以运行自己定义的SQL语句来满足复杂业务场景的需要。

运作机制 ③

图1 静默数据访问视图



①基于数据存储、数据管理、数据同步关键场景，提供清晰完备的开发指导。

②完善场景及概念介绍，明确场景选择及适用推荐。

③场景翻倍，新增数据可靠性与安全性、数据静默访问等内容。

错误码参考

OpenHarmony

错误现象/原因/处理步骤帮助应用开发者快速定位报错及时修复

错误码统一设计

上报机制：

- 参数检查错误
- 业务逻辑错误
- 其他类错误

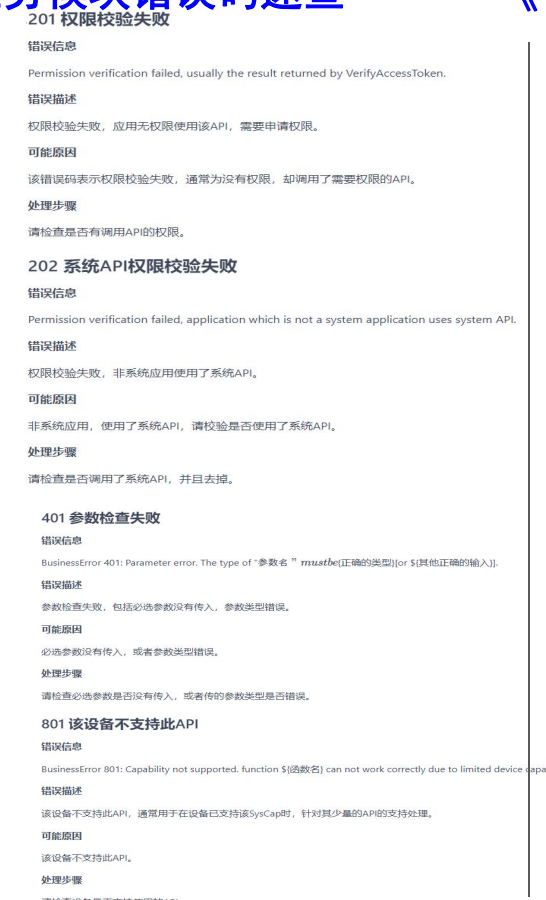
分类：

- **通用错误码**
- 201权限校验失败
- 202系统API权限校验失败
- 401参数检查失败
- 801该设备不支持此API
- **业务错误码**
- Ability框架
- 包管理
- 公共事件与通知
- UI界面
- ……

编号规则：

${SysCap编号} ${错误编号}

《错误码参考》业务模块错误码速查



《 API参考》上报错误码速查，一键跳转详细指导

开发者常见问题

OpenHarmony

现网开发者常见问题，总结沉淀技术支持团队支撑经验，助力开发者快速解决问题

常见问题

- full-SDK编译指南
- full-SDK替换指南
- 开发语言常见问题
- Ability框架开发常见问题
- 应用程序包管理开发常见问题
- ArkUI组件（ArkTS）开发常见问题
- ArkUI Web组件（ArkTS）开发常见问题
- UI框架（JS）开发常见问题
- 公共事件与通知开发常见问题
- 图形图像开发常见问题
- 文件管理开发常见问题
- 媒体开发常见问题
- 网络与连接开发常见问题
- 数据管理开发常见问题
- 设备管理开发常见问题
- DFX开发常见问题
- 国际化开发常见问题
- Native API使用常见问题
- 三四方库使用常见问题
- IDE使用常见问题
- 开发板使用常见问题

160+条常见问题

如何解决must have required property 'startWindowIcon'报错

适用于：OpenHarmony SDK 3.2.3.5版本，API9 Stage模型

Ability配置中缺少startWindowIcon属性配置，需要在module.json5中abilities中配置startWindowIcon。

参考文档：Stage模型应用程序包结构

示例：

```
{
  "module": {
    // do something
    "abilities": [{
      // do something
      "startWindowIcon": "$media:space",
      "startWindowBackground": "$color:white",
    }]
  }
}
```

如何获取设备横竖屏的状态变化的通知

适用于：OpenHarmony SDK 3.2.3.5版本， API9 Stage模型

使用Ability的onConfigurationUpdated回调实现，系统语言、颜色模式以及Display相关的参数，比如方向、Density，发生变化时触发该回调。

参考文档：Ability开发指导

覆盖高频场景及问题

full-SDK替换指南

public-SDK是提供给应用开发的工具包，跟随DevEco Studio下载，不包含系统应用所需要的高权限API

full-SDK是提供给OEM厂商开发应用的工具包，不能随DevEco Studio下载，包含了系统应用所需要的高权限API

三方开发者通过DevEco Studio自动下载的**API8版本**SDK均为**public版本**。public-SDK**不支持**开发者使用**所有的系统API**，包括animator组件、xcomponent组件、@ohos.application.abilityManager.d.ts、@ohos.application.formInfo.d.ts、@ohos.bluetooth.d.ts等，如工程必须依赖于系统API，请按照以下步骤替换**full-SDK**。

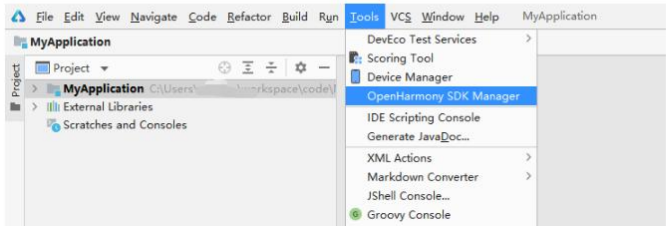
> **说明**：本文中的截图仅为参考，具体的显示界面请以实际使用的DevEco Studio和SDK的版本为准。

下载full-SDK

full-SDK需要手动下载。请参考版本说明书中的获取方式，从镜像站点获取所需的操作系统的full-SDK。

查看本地SDK路径(此处以ets工程为例，1.0工程请以相同方式替换js-SDK)

打开DevEco Studio——>Tools——>OpenHarmony SDK Manager，查看本地SDK安装路径。



获取资料：openharmony.cn官网获取文档

1. 选择“支持”—>“文档”，选择“总览”、“进行设备开发”、“进行应用开发”

2. 选择开发阶段及内容

3. 按需选择版本 “v3.2 Release”

4. 按需搜索需要的主题内容，用于导航内搜索

5. 使用网站右上角全局搜索快速获取需要内容

6. 浏览导航获取相关内容

入门
快速构建首个应用的快速入门，以及开发OpenHarmony应用所必备的基础概念、配置文件等内容。

开发
OpenHarmony应用开发指南，如应用模型、UI开发、媒体、数据管理、公共事件与通知、窗口管理、电话服务等。

工具
DevEco Studio工具是OpenHarmony应用开发的推荐IDE工具。 介绍工具的详细用法，包括使用该工具进行工程创建、应用签名、应用调试、应用安装运行的指导。

API参考
提供了OpenHarmony全量组件和接口的参考文档、错误码参考，可以帮助开发者快速查找到指定接口的详细描述和调用方法。

常见问题
提供了日常支持渠道积累的问题及解决办法

如何反馈资料问题并获取帮助

OpenHarmony

方式1：Gitee Docs仓直接反馈

1. 登录Gitee网站，打开OpenHarmony开源项目，选择Docs仓。
2. 单击“+ Issue”，在新建Issue界面中反馈问题。

方式2：官网问题反馈按钮拉起Gitee Issue直接反馈

1. 浏览OpenHarmony网站内容，单击“问题反馈”按钮，进入Gitee Docs仓Issue界面。
2. 单击“+ 新建Issue”，在新建Issue界面中反馈问题。
3. 清晰的描述问题所在文档链接，文档具体问题，如需开发定位的问题需要给出报错日志、工具/软件版本等。

Issue反馈模板参考

清晰的描述问题所在文档链接，文档具体问题，如需开发定位的问题需要给出报错日志、工具/软件版本等。

【原文链接】：必选。给出完整文档原文链接。

【原文描述】：必选。请明确问题原文描述，方便我们定位问题。

如：进入OpenHarmony代码根目录执行如下命令，从而进入Docker构建环境。

【文档问题】：必选。描述文档具体错误、与产品实现不一致、内容缺失等问题。

如：xxx示例代码运行错误，

【修改建议】：可选。建议给出优化意见、或解决方案如果可能请给出代码片段、写出期望解决方案。

总结

OpenHarmony

1. 6大特性开发指南场景化内容丰富

2. 全新错误码参考快速定位报错、常见问题快速解决问题

3. 我们期待您的反馈，您的意见帮助我们持续提升文档体验

OpenHarmony

THANK YOU

长按识别二维码 关注官方公众号

【官网网址】www.openharmony.cn