第 3 章

万事俱备：基础知识

3.1 开发基础知识

本章简单讲解各模块的应用，如果已经掌握了这些开发基础知识，则可以直接进入第 4 章的实战演练。

3.1.1 程序

用户应用程序：用户应用程序泛指运行在设备的操作系统之上，为用户提供特定服务的程序，简称应用。

在 HarmonyOS 上运行的应用有两种形态：传统方式的需要安装的应用和提供特定功能免安装的应用(原子化服务)。

目前大部分应用程序包是 App 包，而 HarmonyOS 的用户应用程序包以 App Pack(Application Package)的形式发布，它由一个或多个 HAP(HarmonyOS Ability Package)及描述每个 HAP 属性的 pack.info 组成。HAP 是 Ability 的部署包，HarmonyOS 应用代码围绕 Ability 组件展开。

一个 HAP 是由代码、资源、第三方库及应用配置文件组成的模块包，可分为 Entry 和 Feature 两种模块类型，如图 3-1 所示。

Entry：应用的主模块。在一个 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，也可以不包含。只有包含 Ability 的 HAP 才能独立运行。

3.1.2 配置文件

应用的每个 HAP 的根目录下都存在一个 config.json 配置文件，文件内容主要涵盖以下 3 方面：

(1) 应用的全局配置信息，包含应用的包名、生产厂商、版本号等基本信息。

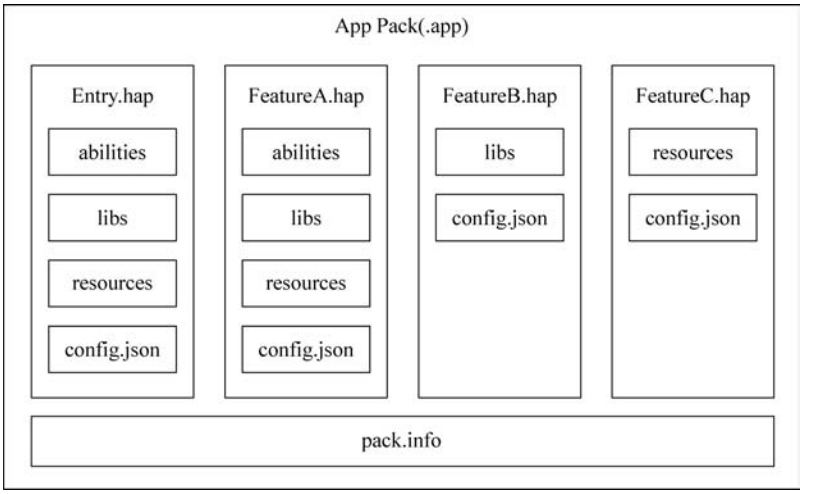

图 3-1　App 逻辑视图

(2) 应用在具体设备上的配置信息,包含应用的备份恢复、网络安全等能力。

(3) HAP 包的配置信息,包含每个 Ability 必须定义的基本属性(如包名、类名、类型及 Ability 提供的能力),以及应用访问系统或其他应用受保护部分所需的权限等。

其中配置文件 config.json 采用 JSON 文件格式,其中包含了一系列配置项,每个配置项由属性和值两部分构成。

(1) 属性:属性出现的顺序不分先后,并且每个属性最多只允许出现一次。

(2) 值:每个属性的值为 JSON 的基本数据类型(数值、字符串、布尔值、数组、对象或者 null 类型)。

3.1.3　资源文件

1. resources 目录

应用的资源文件(字符串、图片、声频等)统一存放于 resources 目录下,便于开发者使用和维护。resources 目录包括两大类目录,一类为 base 目录与限定词目录;另一类为 rawfile 目录,目录如下:

```
resources
|--- base                          //默认存在的目录
|     |--- element
|     |     |--- string.json
|     |--- media
|     |     |--- icon.png
|--- en_GB-vertical-car-mdpi  //限定词目录示例,需要开发者自行创建
|     |--- element
```

```
|     |     |--- string.json
|     |--- media
|     |     |--- icon.png
|--- rawfile                    //默认存在的目录
```

(1) 按组织形式分：base 目录与限定词目录按照两级目录形式来组织，目录命名必须符合规范，以便根据设备状态匹配相应目录下的资源文件。其中，一级子目录为 base 目录和限定词目录。base 目录是默认存在的目录。当应用的 resources 资源目录中没有与设备状态匹配的限定词目录时，会自动引用该目录中的资源文件。限定词目录需要开发者自行创建。目录名称由一个或多个表征应用场景或设备特征的限定词组合而成。二级子目录为资源目录，用于存放字符串、颜色、布尔值等基础元素，以及媒体、动画、布局等资源文件；rawfile 目录支持创建多层子目录，目录名称可以自定义，文件夹内可以自由放置各类资源文件。rawfile 目录的文件不会根据设备状态去匹配不同的资源。

(2) 按编译方式分：base 目录与限定词目录中的资源文件会被编译成二进制文件，并赋予资源文件 ID；rawfile 目录中的资源文件会被直接打包进应用，不经过编译，也不被赋予资源文件 ID。

(3) 按引用方式分：base 目录与限定词目录通过指定资源类型(type)和资源名称(name)来引用；rawfile 目录通过指定文件路径和文件名来引用。

2. 限定词目录

限定词目录可以由一个或多个表征应用场景或设备特征的限定词组合而成，包括移动国家码和移动网络码、语言、文字、国家或地区、横竖屏、设备类型、颜色模式和屏幕密度等维度，限定词之间通过下画线“_”或者半字线“-”连接。开发者在创建限定词目录时，需要掌握限定词目录的命名要求，以及限定词目录与设备状态的匹配规则。

其中限定词目录的命名要求有以下三点。

(1) 限定词的组合顺序：移动国家码_移动网络码-语言_文字_国家或地区-横竖屏-设备类型-深色模式-屏幕密度。开发者可以根据应用的使用场景和设备特征，选择其中的一类或几类限定词组成目录名称。

(2) 限定词的连接方式：语言、文字、国家或地区之间采用下画线“_”连接，移动国家码和移动网络码之间也采用下画线“_”连接，除此之外的其他限定词之间均采用半字线“-”连接。例如，zh_Hant_CN、zh_CN-car-ldpi。

(3) 限定词的取值范围：每类限定词的取值必须符合限定词类型的条件，否则将无法匹配目录中的资源文件。

限定词目录与设备状态的匹配规则有以下两点：

(1) 在为设备匹配对应的资源文件时，限定词目录匹配的优先级从高到低依次为移动国家码和移动网络码＞区域(可选组合：语言、语言_文字、语言_国家或地区、语言_文字_国家或地区)＞横竖屏＞设备类型＞颜色模式＞屏幕密度。

(2) 如果限定词目录中包含移动国家码和移动网络码、语言、文字、横竖屏、设备类型、

颜色模式限定词，则对应限定词的取值必须与当前的设备状态完全一致，这样该目录才能参与设备的资源匹配。例如，限定词目录 zh_CN-car-ldpi 不能参与 en_US 设备的资源匹配。

3. 资源组目录

在 base 目录与限定词目录下面可以创建资源组目录，包括 element、media、animation、layout、graphic、profile，用于存放特定类型的资源文件。

其中，element 目录表示元素资源，以下每一类数据都采用相应的 JSON 文件来表征。media 目录表示媒体资源，包括图片、声频、视频等非文本格式的文件。animation 目录表示动画资源，采用 XML 文件格式。layout 目录表示布局资源，采用 XML 文件格式。graphic 目录表示可绘制资源，采用 XML 文件格式。profile 目录表示其他类型文件，以原始文件的形式保存。

3.1.4 其他

(1) Ability：Ability 是应用所具备的能力的抽象，一个应用可以包含一个或多个 Ability。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。

(2) 库文件：库文件是应用依赖的第三方代码(例如 so、jar、bin、har 等二进制文件)，存放在 libs 目录。

(3) pack.info：描述应用软件包中每个 HAP 的属性，由 IDE 编译生成，应用市场根据该文件进行拆包和 HAP 的分类存储。

(4) delivery-with-install：表示该 HAP 是否支持随应用安装。true 表示支持随应用安装，false 表示不支持随应用安装。

(5) name：HAP 文件名。

(6) module-type：模块类型，即 Entry 或 Feature。

(7) device-type：表示支持该 HAP 运行的设备类型。

(8) HAR：HAR(HarmonyOS Ability Resources)可以提供构建应用所需的所有内容，包括源代码、资源文件和 config.json 文件。HAR 不同于 HAP，HAR 不能独立安装运行在设备上，只能作为应用模块的依赖项被引用。

3.2 Page Ability

Ability 是应用所具备能力的抽象，也是应用程序的重要组成部分。一个应用可以具备多种能力(可包含多个 Ability)，HarmonyOS 支持应用以 Ability 为单位进行部署。每种类型为开发者提供了不同的模板，以便实现不同的业务功能。

(1) FA 支持 Page Ability：Page 模板是 FA 唯一支持的模板，用于提供与用户交互的能力。一个 Page 实例可以包含一组相关页面，每个页面用一个 AbilitySlice 实例表示。

（2）PA 支持 Service Ability 和 Data Ability：Service 模板用于提供后台运行任务的能力；Data 模板用于对外部提供统一的数据访问抽象。

在配置文件（config.json）中注册 Ability 时，可以通过配置 Ability 元素中的 type 属性来指定 Ability 的模板类型，其中 type 的取值可以为 page、service 或 data，分别代表 Page 模板、Service 模板、Data 模板，伪代码如下：

```
{
    "module": {
        ...
        "abilities": [
            {
                ...
                "type": "page"
                ...
            }
        ]
        ...
    }
    ...
}
```

Page 模板是 FA 唯一支持的模板，用于提供与用户交互的能力。一个 Page 可以由一个或多个 AbilitySlice 构成，AbilitySlice 是指应用的单个页面及其控制逻辑的总和。

当一个 Page 由多个 AbilitySlice 共同构成时，这些 AbilitySlice 页面提供的业务能力应具有高度相关性。例如，新闻浏览功能可以通过一个 Page 实现，其中包含了两个 AbilitySlice：一个 AbilitySlice 用于展示新闻列表；另一个 AbilitySlice 用于展示新闻详情。Page 和 AbilitySlice 的关系如图 3-2 所示。

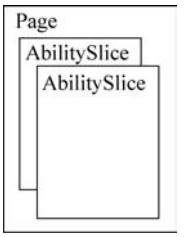

图 3-2　Page 和 AbilitySlice 的关系

3.2.1　Page 的生命周期

对于每个 Page 都拥有相同的生命周期函数：onStart()、onActive()、onInactive()、onBackground()、onForeground()和 onStop()。

（1）onStart()：当系统首次创建 Page 实例时，触发该回调。对于一个 Page 实例，该回调在其生命周期过程中仅触发一次，Page 在该逻辑后将进入 INACTIVE 状态。开发者必须重写该方法，并在此配置默认展示的 AbilitySlice。

（2）onActive()：Page 会在进入 INACTIVE 状态后来到前台，然后由系统调用此回调。Page 在此之后进入 ACTIVE 状态，该状态是应用与用户进行交互的状态。Page 将保持在此状态，除非某类事件导致 Page 失去焦点，例如用户单击返回键或导航到其他 Page。

当此类事件发生时，会触发 Page 回到 INACTIVE 状态，系统将调用 onInactive()回调函数。此后，Page 可能重新回到 ACTIVE 状态，系统将再次调用 onActive()回调函数，因此，开发者通常需要成对实现 onActive()和 onInactive()函数，并在 onActive()函数中获取在 onInactive()函数中被释放的资源。

(3) onInactive()：当 Page 失去焦点时，系统将调用此回调函数，此后 Page 进入 INACTIVE 状态。开发者可以在此回调函数中实现 Page 失去焦点时应表现的恰当行为。

(4) onBackground()：如果 Page 不再对用户可见，则系统将调用此回调函数通知开发者对用户进行相应的资源释放，此后 Page 进入 BACKGROUND 状态。开发者应该在此回调函数中释放 Page 不可见时无用的资源，或在此回调函数中执行较为耗时的状态保存操作。

(5) onForeground()：处于 BACKGROUND 状态的 Page 仍然驻留在内存中，当重新回到前台时(例如用户重新导航到此 Page)，系统将先调用 onForeground()回调函数通知开发者，而后 Page 的生命周期状态回到 INACTIVE 状态。开发者应当在此回调函数中重新申请在 onBackground()函数中释放的资源，最后 Page 的生命周期状态进一步回到 ACTIVE 状态，系统将通过 onActive()回调函数通知开发者用户。

(6) onStop()：系统将要销毁 Page 时，将会触发此回调函数，通知用户进行系统资源的释放。销毁 Page 的可能原因包括以下几方面：用户通过系统管理能力关闭指定 Page，例如使用任务管理器关闭 Page；用户行为触发 Page 的 terminateAbility()方法调用，例如使用应用的退出功能；配置变更导致系统暂时销毁 Page 并重建；系统出于资源管理的目的，自动触发对处于 BACKGROUND 状态的 Page 进行销毁。

Page 的生命周期的验证将在 5.6 节进行。

3.2.2 AbilitySlice 的生命周期

AbilitySlice 作为 Page 的组成单元，其生命周期依托于其所属 Page 的生命周期。AbilitySlice 和 Page 具有相同的生命周期状态和同名的回调函数，当 Page 的生命周期发生变化时，它的 AbilitySlice 也会发生相同的生命周期变化。此外，AbilitySlice 还具有独立于 Page 的生命周期变化，这发生在同一 Page 中的 AbilitySlice 之间进行导航时，此时 Page 的生命周期状态不会改变。AbilitySlice 的生命周期回调函数与 Page 的相应回调函数类似，因此不再赘述。

3.2.3 Page 与 AbilitySlice 的生命周期关联

当 AbilitySlice 处于前台且具有焦点时，其生命周期状态随着所属 Page 的生命周期状态的变化而变化。当一个 Page 拥有多个 AbilitySlice 时，例如，MainAbility 下有 FooAbilitySlice 和 BarAbilitySlice，当前 FooAbilitySlice 处于前台并获得焦点，并且即将导航到 BarAbilitySlice，在此期间的生命周期状态的变化顺序为

(1) FooAbilitySlice 从 ACTIVE 状态变为 INACTIVE 状态。

（2）BarAbilitySlice 则从 INITIAL 状态首先变为 INACTIVE 状态，然后变为 ACTIVE 状态（假定此前 BarAbilitySlice 未曾启动）。

（3）FooAbilitySlice 从 INACTIVE 状态变为 BACKGROUND 状态。

对应两个 Slice 的生命周期方法的回调顺序为 FooAbilitySlice. onInactive()→BarAbilitySlice. onStart()→BarAbilitySlice. onActive()→FooAbilitySlice. onBackground()。

在整个流程中，MainAbility 始终处于 ACTIVE 状态，但是，当 Page 被系统销毁时，其所有已实例化的 AbilitySlice 将联动销毁，而不仅销毁处于前台的 AbilitySlice。

3.3　Service Ability

基于 Service 模板的 Ability 主要用于后台运行任务（如执行音乐播放、文件下载等），但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。

Service 是单实例的。在一个设备上，相同的 Service 只会存在一个实例。如果多个 Ability 共用这个实例，则只有当与 Service 绑定的所有 Ability 都退出后，Service 才能退出。由于 Service 是在主线程里执行的，因此，如果在 Service 里面的操作时间过长，则开发者必须在 Service 里创建新的线程来处理（详见线程间通信），防止造成主线程阻塞，从而防止应用程序无响应。

与 Page 类似，Service Ability 也拥有生命周期，如图 3-3 所示。

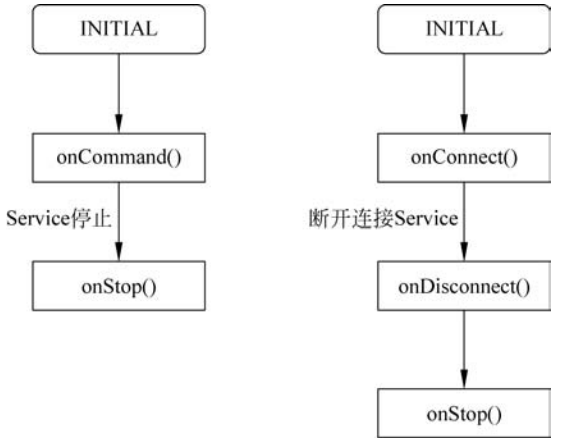

图 3-3　Service Ability 生命周期

根据调用方法的不同，其生命周期有以下两种路径。

（1）启动 Service：该 Service 在其他 Ability 调用 startAbility()函数时创建，然后保持运行。其他 Ability 通过调用 stopAbility()函数来停止 Service，Service 停止后，系统会将其销毁。

(2) 连接 Service：该 Service 在其他 Ability 调用 connectAbility()函数时创建，客户端可通过调用 disconnectAbility()函数断开连接。多个客户端可以绑定到相同的 Service，而且当所有绑定全部取消后，系统即会销毁该 Service。connectAbility()函数也可以连接通过 startAbility()函数创建的 Service。

3.4 Data Ability

使用 Data 模板的 Ability(以下简称 Data)有助于应用管理其自身和其他应用存储数据的访问，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，也支持跨设备不同应用的数据共享。

数据的存放形式多样，可以是数据库，也可以是磁盘上的文件。Data 对外提供对数据的增、删、改、查，以及打开文件等接口，这些接口的具体实现由开发者提供。

Data 的提供方和使用方都通过 URI(Uniform Resource Identifier)来标识一个具体的数据，例如数据库中的某个表或磁盘上的某个文件。HarmonyOS 的 URI 仍基于 URI 通用标准，格式如图 3-4 所示。

图 3-4 URI

(1) Scheme：协议方案名，固定为 dataability，代表 Data Ability 所使用的协议类型。

(2) authority：设备 ID。如果为跨设备场景，则为目标设备的 ID；如果为本地设备场景，则不需要填写。

(3) path：资源的路径信息，代表特定资源的位置信息。

(4) query：查询参数。

(5) fragment：可以用于指示要访问的子资源。

3.5 JS 生命周期

生命周期分为应用生命周期和页面生命周期。

1. 应用生命周期

在 app.js 文件中可以定义如下应用生命周期函数。

(1) onCreate()：用于应用创建，当应用创建时调用。

(2) onShow()：当应用处于前台时触发。

(3) onHide()：当应用处于后台时触发。

(4) onDestroy()：当应用退出时触发。

2. 页面生命周期

在页面 JS 文件中可以定义如下页面生命周期函数。

(1) onInit()：页面数据初始化完成时触发，只触发一次。

(2) onReady()：页面创建完成时触发，只触发一次。

(3) onShow()：页面显示时触发。

(4) onHide()：页面消失时触发。

(5) onDestroy()：页面销毁时触发。

(6) onBackPress()：当用户单击返回按钮时触发。当返回值为 true 时表示页面自己处理返回逻辑；当返回值为 false 时表示使用默认的返回逻辑；当不返回值时会作为 false 处理。

(7) onActive()：页面激活时触发。

(8) onInactive()：页面暂停时触发。

常见的页面 A 的生命周期接口的调用顺序如下。

(1) 打开页面 A：onInit()→onReady()→onShow()。

(2) 在页面 A 打开页面 B：onHide()。

(3) 从页面 B 返回页面 A：onShow()。

(4) 退出页面 A：onBackPress()→onHide()→onDestroy()。

(5) 页面隐藏到后台运行：onInactive()→onHide()。

(6) 页面从后台运行恢复到前台：onShow()→onActive()。

3.6　Java UI 框架

应用的 Ability 在屏幕上将显示一个用户界面，该界面用来显示所有可被用户查看和交互的内容。

应用中所有的用户界面元素都由 Component 和 ComponentContainer 对象构成。Component 是绘制在屏幕上的一个对象，用户能与之交互。ComponentContainer 是一个用于容纳其他 Component 和 ComponentContainer 对象的容器。

Java UI 框架提供了一部分 Component 和 ComponentContainer 的具体子类，即创建用户界面(UI)的各类组件，包括一些常用的组件(例如：文本、按钮、图片、列表等)和常用的布局(例如：DirectionalLayout 和 DependentLayout)。用户可通过组件进行交互操作，并获得响应。

需要注意的是，所有的 UI 操作都应该在主线程进行设置。

用户界面元素统称为组件，组件根据一定的层级结构进行组合，从而形成布局。组件在未被添加到布局中时，既无法显示也无法交互，因此一个用户界面至少包含一个布局。在 UI 框架中，具体的布局类通常以 XXLayout 命名，完整的用户界面是一个布局，用户界面中的一部分也可以是一个布局。布局中容纳的是 Component 与 ComponentContainer 对象。

Component 与 ComponentContainer 的关系如图 3-5 所示。

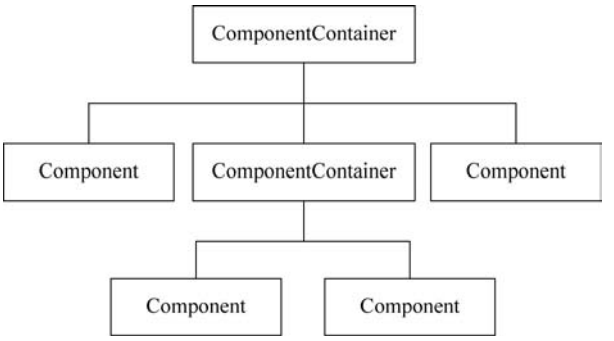

图 3-5　Component 与 ComponentContainer 的关系

（1）Component：提供内容显示，是界面中所有组件的基类，开发者可以给 Component 设置事件处理回调来创建一个可交互的组件。Java UI 框架提供了一些常用的界面元素，也可称为组件，组件一般直接继承自 Component 或它的子类，如 Text、Image 等。

（2）ComponentContainer：作为容器容纳 Component 或 ComponentContainer 对象，并对它们进行布局。Java UI 框架提供了一些标准布局功能的容器，它们继承自 ComponentContainer，一般以 Layout 结尾，如 DirectionalLayout、DependentLayout 等。

主要模块的开发基础知识就到此讲解完了。第 4 章将在 2.2 节 Hello World 项目的基础上不断地进行修改和完善，最终开发出一个完整的经典游戏 App——“数字华容道”。