第5章 Page Ability

本章学习目标

- 熟练掌握 Ability 的架构。
- 了解 Page Ability 基础知识。
- 熟练掌握 Page Ability 生命周期。
- 掌握三种页面跳转实现方法。

5.1 Ability 概述

Ability 是 HarmonyOS 应用程序的核心组成部分，分为 FA(Feature Ability)和 PA(Particle Ability)两种类型，Page Ability 是 FA 唯一支持的 Ability，用于提供与用户交互的能力。本章先介绍 Ability 的基础知识，再重点学习 Page Ability 的生命周期以及各种页面跳转的使用方法。

在 HarmonyOS 应用中，有一个非常核心的概念，那就是 Ability(能力)。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 Ability(可以包含 Page Ability、Service Ability、Data Ability)，HarmonyOS 支持应用以 Ability 为单位进行部署。

Ability 可以分为 FA(Feature Ability)和 PA(Particle Ability)两种类型，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。其中，FA 有 UI 界面，提供与用户交互的能力；而 PA 无 UI 界面，提供后台运行任务以及访问文件或数据库的能力。

FA 唯一支持模板是 Page Ability，PA 则包括 Service Ability 和 Data Ability。Ability 之间的关系如图 5-1 所示。

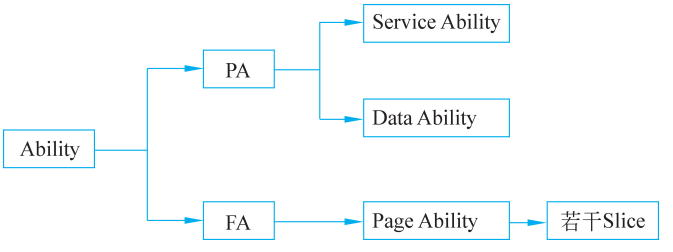

图 5-1 Ability 之间的关系

基于 FA 或 PA 开发的应用能够实现特定的业务功能，支持跨设备调度与分发，为用户

提供一致、高效的应用体验。例如，在 DevEco Studio 中创建第一个应用时，默认选择的是一个空的 FA 模板，即 Empty Ability，如图 5-2 所示。

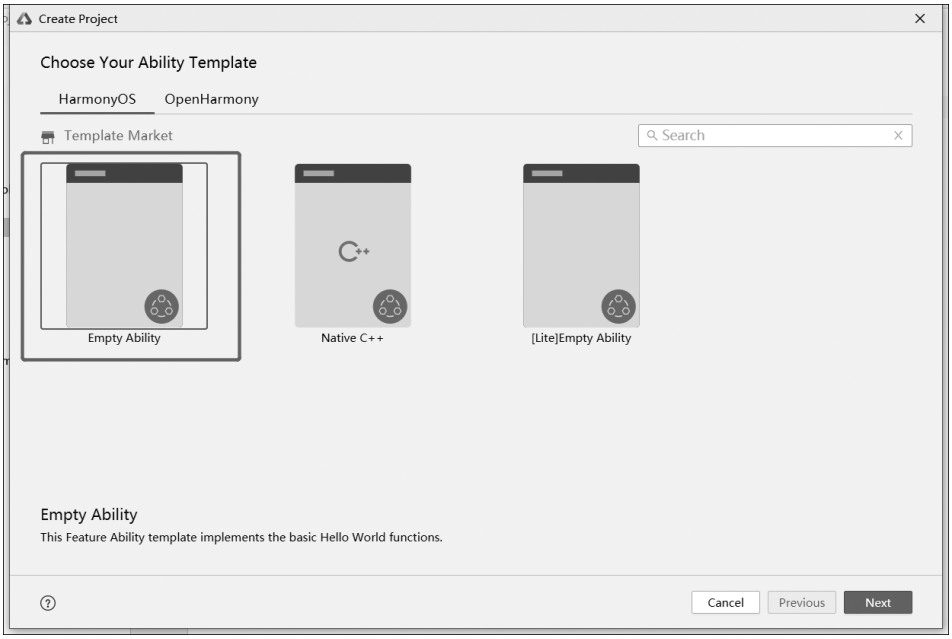

图 5-2　选择空的 FA 模板

5.2　Ability 的配置

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 模板类型。

其中，type 的取值可以为 page、service 或 data，它们分别代表 Page 模板、Service 模板、Data 模板。为了便于表述，后文中将基于 Page 模板、Service 模板、Data 模板实现的 Ability 分别简称为 Page、Service、Data。

下面创建一个名为 PageAbilityLifeCycle 的应用，来说明 Page 中的各个配置选项以及 Page 的生命周期。在其 Project 窗口中打开 entry/src/main/config.json 文件，在 abilities 配置选项中包含了各个 Ability 的配置选项，内容如下：

```
⋮
   "abilities": [
     {
       "skills": [
         {
           "entities": [
             "entity.system.home"
           ],
```

```
            "actions": [
              "action.system.home"
            ]
          }
        ],
        "name": "com.example.pageabilitylifecycle.MainAbility",
        "description": "$string:mainability_description",
        "icon": "$media:icon",
        "label": "$string:entry_MainAbility",
        "launchType": "standard",
        "orientation": "unspecified",
        "visible": true,
        "type": "page"
      }
    ]
  ⋮
```

在新创建 PageAbilityLifeCycle 应用中，abilities 配置选项仅有一个 Ability 对象，类型是 Page。

Ability 对象中各个属性的含义如表 5-1 所示。

表 5-1 Ability 对象中各属性的含义

属性名称	属 性 含 义	数据类型	是否可缺省
name	Ability 的名称，采用“包名＋类名”的方式定义	字符串	否
description	Ability 的描述	字符串	可缺省
icon	Ability 图标资源文件的索引	字符串	可缺省
label	Ability 对用户显示的名称，默认显示在手机应用程序的标题栏	字符串	可缺省
launchType	Ability 的启动模式，支持 standard、singleMission 和 singleton 三种模式	字符串	可缺省
orientation	Ability 的显示模式，仅适用于 pageAbility。取值范围：unspecified，表示系统自动判断显示方向；landscape，表示横屏；portrait，表示竖屏；followRecent，表示跟随栈中最近的应用	字符串	可缺省
visible	Ability 是否可以被其他应用调用。true 表示可以，false 表示不可以	布尔类型	可缺省
type	Ability 的类型，取值可为 page、service、data	字符串	否

5.3 应 用 分 层

5.3.1 应用的三层架构

目前，比较常用的、典型的应用软件倾向使用三层架构（Three-Tier Architecture），包

括以下 3 层。

（1）表示层（Presentation Layer）：提供与用户交互的界面。GUI（Graphical User Interface，图形用户界面）和 Web 页面是表示层的两个典型的例子。

（2）业务层（Business Layer）：也称为业务逻辑层，用于实现各种业务逻辑，例如处理数据验证根据特定的业务规则和任务来响应特定的行为。

（3）数据访问层（DataAccess Layer）：也称为数据持久层，负责存放和管理应用的持久性业务数据。

三层架构如图 5-3 所示。

图 5-3　三层架构图

5.3.2　Ability 的三层架构

从应用的三层架构中，很容易识别出，Ability 同样遵循图 5-3 所示的三层架构。其中 PageAbility 代表表示层，ServiceAbility 代表业务层，DataAbility 代表数据访问层。

因此，读者在设计 Ability 时，应先考虑这个 Ability 需要完成什么样的功能，代表了哪个层次的业务。

5.4　Page Ability 简介

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，AbilitySlice 指应用的单个页面及其控制逻辑的总和。简而言之，FA 承担的就是前端与用户之间的交互工作。

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，聊天软件可以通过一个 Page 来实现，其中包含了两个 AbilitySlice：一个 AbilitySlice 用于构成联系人列表，另一个 AbilitySlice 用于和具体联系人聊天。Page 和 AbilitySlice 的关系如图 5-4 所示。

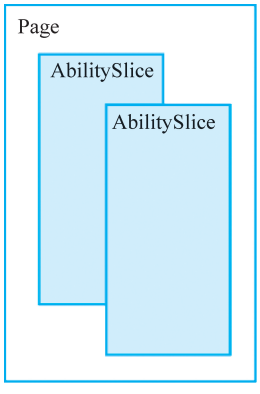

图 5-4　Page 和 AbilitySlice 的关系

相比于桌面场景，移动场景下应用之间的交互更为频繁。通常，单个应用专注某个方面的能力开发，当它需要其他能力辅助时，会调用其他应用提供的能力。例如，在聊天软件中提供了打开网络链接功能的入口，当用户点击链接时，会打开浏览器，访问网络链接。与此类似，HarmonyOS 支持不同 Page 之间的跳转，并可以指定跳转到目标 Page 中某个具体的 AbilitySlice。

5.5 生命周期

系统管理或用户操作等行为均会引起 Page 实例在其生命周期的不同状态之间进行转换。Ability 类提供的回调机制能够让 Page 及时感知外界变化，从而正确地应对状态变化（如释放资源），这有助于提升应用的性能和稳健性。

5.5.1 Page 生命周期回调

Page 生命周期的不同状态转换及其对应的回调如图 5-5 所示。

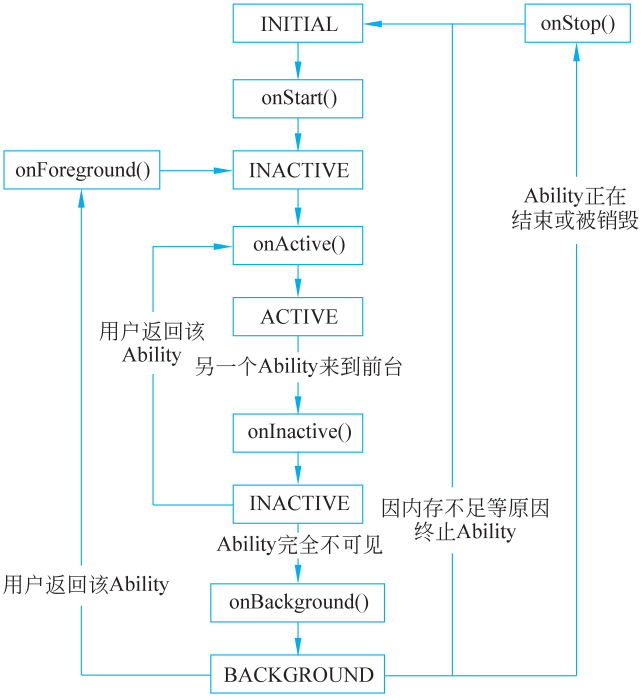

图 5-5 Page 生命周期

说明：INACTIVE 状态是一种短暂存在的状态，可理解为“激活中”。

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。示例代码如下：

```
@Override
public void onStart(Intent intent) {
    super.onStart(intent);
    super.setMainRoute(MainAbilitySlice.class.getName());
```

```
}
```

上述代码中的 setMainRoute()方法用来设置该 Page 的主路由，意思是启动该 Page 后默认打开其参数设置的 AbilitySlice。这里默认打开 MainAbilitySlice。

2. onActive()

Page 会在进入 INACTIVE 状态后来到前台，然后系统调用此回调。Page 在此之后进入 ACTIVE 状态，该状态是应用与用户交互的状态。Page 将保持在此状态，除非某类事件发生导致 Page 失去焦点，例如用户点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。此后，Page 可能重新回到 ACTIVE 状态，系统将再次调用 onActive()回调。因此，开发者通常需要成对实现 onActive()和 onInactive()，并在 onActive()中获取在 onInactive()中被释放的资源。示例代码如下：

```
@Override
public void onActive() {
    super.onActive();
}
```

3. onInactive()

当 Page 失去焦点时，系统将调用此回调，此后 Page 进入 INACTIVE 状态。开发者可以在此回调中实现 Page 失去焦点时应表现的恰当行为。示例代码如下：

```
@Override
protected void onInactive() {
    super.onInactive();
}
```

4. onBackground()

如果 Page 不再对用户可见，系统将调用此回调通知开发者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中释放 Page 不可见时无用的资源，或在此回调中执行较为耗时的状态保存操作。示例代码如下：

```
@Override
protected void onBackground() {
    super.onBackground();
}
```

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（如用户重

新导航到此 Page),系统将先调用 onForeground()回调通知开发者,而后 Page 的生命周期状态回到 INACTIVE 状态。开发者应当在此回调中重新申请在 onBackground()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()回调通知开发者用户。示例代码如下:

```
@Override
public void onForeground(Intent intent) {
    super.onForeground(intent);
}
```

6. onStop()

系统要销毁 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个方面。

(1) 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。

(2) 用户行为触发 Page 的 terminateAbility()方法调用,例如使用应用的退出功能。

(3) 配置变更导致系统暂时销毁 Page 并重建。

(4) 系统出于资源管理目的,自动触发对处于 BACKGROUND 状态 Page 的销毁。

示例代码如下:

```
@Override
protected void onStop() {
    super.onStop();
}
```

5.5.2 AbilitySlice 生命周期

AbilitySlice 作为 Page 的组成单元,其生命周期是依托其所属 Page 生命周期的。AbilitySlice 和 Page 具有相同的生命周期状态和同名的回调,当 Page 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 之间导航时,此时 Page 的生命周期状态不会改变。

AbilitySlice 生命周期回调与 Page 的相应回调类似,因此不再赘述。由于 AbilitySlice 承载具体的页面,开发者必须重写 AbilitySlice 的 onStart()回调方法,并在此方法中通过 setUIContent()方法设置页面,如下所示:

```
@Override
public void onStart(Intent intent) {
    super.onStart(intent);
    super.setUIContent(ResourceTable.Layout_ability_main);
}
```

上述代码在 onStart()方法中调用 setUIContent()方法设置页面效果，这里通过 XML 文件构建页面内容，其参数是一个 XML 文件对象，对应的 XML 文件是 ability_main.xml。

AbilitySlice 实例创建和管理通常由应用负责，系统仅在特定情况下会创建 AbilitySlice 实例。例如，通过导航启动某个 AbilitySlice 时，是由系统负责实例化；但是在同一个 Page 中不同的 AbilitySlice 间导航时则由应用负责实例化。

5.5.3　案例：Page 的生命周期

下面创建一个名为 PageAbilityLifeCycle 的应用，来演示 Page 和 AbilitySlice 的生命周期。修改 ability_main.xml，其内容如下：

```
<?xml version="1.0" encoding="utf-8"?>
<DirectionalLayout
    xmlns:ohos="http://schemas.huawei.com/res/ohos"
    ohos:height="match_parent"
    ohos:width="match_parent"
    ohos:alignment="center"
    ohos:orientation="vertical">

    <Text
        ohos:id="$+id:Text_mainAbilitySlice"
        ohos:height="match_content"
        ohos:width="match_content"
        ohos:background_element="$graphic:background_ability_main"
        ohos:layout_alignment="horizontal_center"
        ohos:text="MainAbilitySlice"
        ohos:text_size="40fp"
        />

</DirectionalLayout>
```

修改 MainAbility 中的代码，添加 onActive()、onInactive()、onBackground()、onForeground()、onStop()生命周期方法，并定义日志标签，在每个生命周期方法中打印出对应方法名，以此演示 Page 的生命周期。其内容如下：

```
public class MainAbility extends Ability {
    //定义日志标签
    private static final HiLogLabel LABEL =new HiLogLabel(HiLog.LOG_APP
            , 0x00922
            , "MainAbility");

    @Override
    public void onStart(Intent intent) {
        super.onStart(intent);
```

```
        super.setMainRoute(MainAbilitySlice.class.getName());
        HiLog.info(LABEL, "onStart");
    }

    @Override
    public void onActive() {
        super.onActive();
        HiLog.info(LABEL, "onActive");
    }

    @Override
    protected void onInactive() {
        super.onInactive();
        HiLog.info(LABEL, "onInactive");
    }

    @Override
    protected void onBackground() {
        super.onBackground();
        HiLog.info(LABEL, "onBackground");
    }

    @Override
    public void onForeground(Intent intent) {
        super.onForeground(intent);
        HiLog.info(LABEL, "onForeground");
    }

    @Override
    protected void onStop() {
        super.onStop();
        HiLog.info(LABEL, "onStop");
    }
}
```

代码说明

使用 HiLogLabel(int type, int domain, String tag)定义日志标签,其参数的含义分别如下。

type:用于指定打印日志的类型,HiLog 中当前只提供了一种日志类型,即应用的日志类型 LOG_APP。

domain:用于指定打印日志所对应的业务领域,取值范围为 0x0～0xFFFFF,开发者可以根据需要进行自定义。

tag:用于指定日志标识,可以为任意字符串,建议标识调用所在的类,这里的日志标识为 MainAbility。

开发者可以根据自定义参数 domain 和 tag 来进行日志的筛选和查找。

在 onStart()方法中使用 HiLog.info(LABEL, "onStart");语句,意思是将此日志级别定义为 info。

在 MainAbilitySlice 中,重复上述操作,用日志打印方法名,演示 MainAbilitySlice 的生命周期。其内容如下:

```
public class MainAbilitySlice extends AbilitySlice {
    //定义日志标签
    private static final HiLogLabel LABEL =new HiLogLabel(HiLog.LOG_APP
            , 0x00922
            , "MainAbilitySlice");

    @Override
    public void onStart(Intent intent) {
        super.onStart(intent);
        super.setUIContent(ResourceTable.Layout_ability_main);
        HiLog.info(LABEL, "onStart");
    }

    @Override
    public void onActive() {
        super.onActive();
        HiLog.info(LABEL, "onActive");
    }

    @Override
    protected void onInactive() {
        super.onInactive();
        HiLog.info(LABEL, "onInactive");
    }

    @Override
    protected void onBackground() {
        super.onBackground();
        HiLog.info(LABEL, "onBackground");
    }

    @Override
    public void onForeground(Intent intent) {
        super.onForeground(intent);
        HiLog.info(LABEL, "onForeground");
    }

    @Override
```