第 3 章

窗　口

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，是 Harmony 应用程序的一大核心。一个应用可以包含多个 Ability，而 Ability 又可以分为 FA（Feature Ability）和 PA（Particle Ability）两种类型，FA 仅支持 Page Ability，它对用户是可见的，承载了一个业务可视化界面，即用户可通过 FA 与应用程序进行交互，PA 支持 Data Ability（第 9 章）和 Service Ability（第 10 章），它在后台运行，对用户是不可见的，PA 无法提供与用户交互的能力。本章主要对 Page Ability 进行讨论。

通过阅读本章，读者可以掌握：

- 如何使用 Page Ability。
- 如何在 Page Ability 之间进行交互。
- 如何进行跨设备迁移 Page Ability。
- 如何使用 AbilitySlice。
- Page Ability 与 AbilitySlice 的生命周期。

3.1　Page Ability 概述

Page Ability 是 FA 唯一支持的模板，它本质上是一个窗口，用于提供与用户交互的能力，类似于 Android 中的 Activity。另外，HarmonyOS 提供了 AbilitySlice，AbilitySlice 是指应用的单个页面控制逻辑的总和，相当于页面内的子窗口，类似于导航窗口，其功能与 Page Ability 相同，在切换时可以在同一个 Page Ability 内完成，也可以跳转至不同的 Page Ability。Page Ability 之间的切换相当于 Web 网页之间的切换，而 AbilitySlice 之间的切换相当于在一个 Web 页面下不同导航窗口之间的切换。

Page Ability 可以使用一个或多个 AbilitySlice，也可以不使用。在创建 HarmonyOS 工程时，包含了一个默认的 AbilitySlice（MainAbilitySlice.java）。当在一个 Page Ability 中使用多个 AbilitySlice 时，这些 AbilitySlice 所提供的功能之间应该具有高度的相关性，换言之，页面提供的功能之间有高度相关性时，应该在一个 Page Ability 下使用两个 AbilitySlice，而不必使用两个 Page Ability，以减少冗余。Page Ability 和 AbilitySlice 的关系如图 3-1 所示。

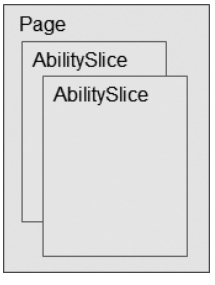

图 3-1　Page Ability 与 AbilitySlice 的关系

为了提高开发者的开发效率，使用 DevEco Studio 创建 HarmonyOS 工程时，IDE 提供了一些 Ability 模

板供开发者使用，如图 3-2 所示，读者可以使用这些 Ability 模板快速生成一个 HarmonyOS 工程的框架，相当于一个简单的 Hello World 工程。

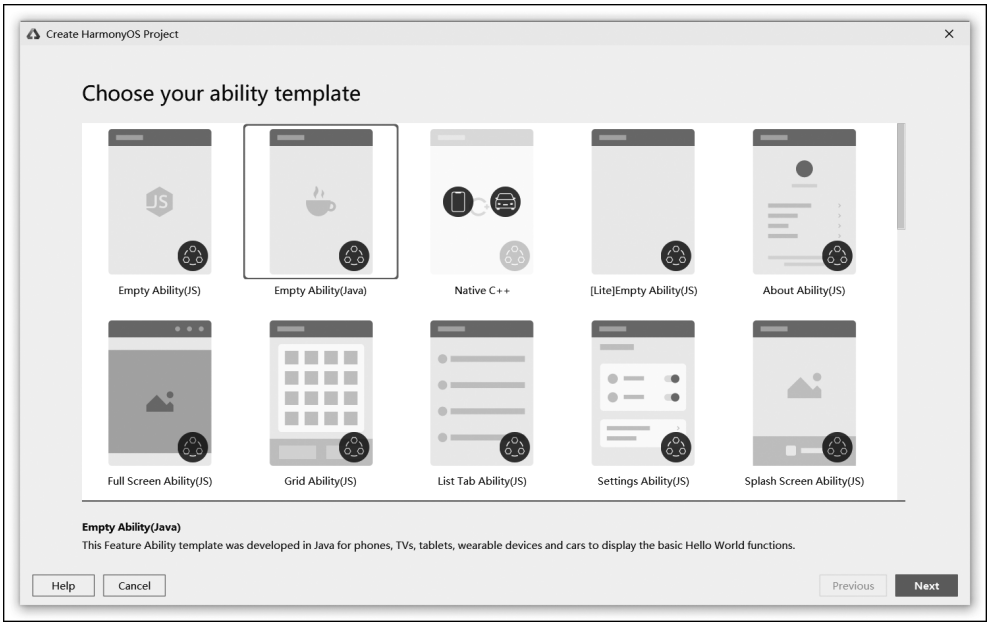

图 3-2　Ability 模板

3.2　Page Ability 的基本用法

为了提高开发效率，DevEco Studio 也为开发人员提供了自动创建 Page Ability 的功能，在创建过程中会自动创建 Page 类型的 Ability 类，同时创建一个 AbilitySlice 类以及布局文件，并自动向 config.json 文件中添加 Page Ability 的配置信息，这些都是开发工具自动完成的。为了使读者能够更好地理解和使用 Page Ability，本节将介绍如何手动创建 Page Ability、布局文件，以及如何装载布局文件，并在 config.json 中配置一个 Page Ability。

3.2.1　手动创建 Page Ability 类

Page Ability 是一个普通的 Java 类，其必须继承 Ability 类，该类属于 ohos.aafwk.ability 包，下面给出 MyFirstAbility 类的创建代码：

```
package com.example.createpageability;
import ohos.aafwk.ability.Ability;
public class MyFirstAbility extends Ability {
}
```

3.2.2　在 config.json 文件中注册 Page Ability

在 HarmonyOS App 中，所有 Ability 在使用前必须在 config.json 文件中的 abilities 配

置项配置。abilities是一个对象数组，每一个对象表示一个Ability，包括Page Ability、Data Ability和Service Ability。MyFirstAbility的配置代码如下：

```
{
    "skills": [
        {
            "actions": [
                "com.example.myfirstability"
            ]
        }
    ],
    "orientation": "unspecified",
    "formsEnabled": false,
    "name": "com.example.createpageability.MyFirstAbility",
    "icon": "$media:icon",
    "description": "$string:myfirstability_description",
    "label": "$string:myfirstability_label",
    "type": "page",
    "launchType": "standard"
}
```

下面对abilities配置信息中的常用属性进行介绍：

(1) skills表示该Ability能够接收Intent的特征，是一个对象数组。

skills中对象又有3种属性：

actions：表示能够接收的Intent的action值，可以包含一个或者多个action；

entities：表示能够接收的Intent的Ability的类别，可以包含一个或者多个entity；

uris：表示能够接收的Intent的uri，可以包含一个或者多个uri。

(2) orientation表示该Ability的显示模式，该标签仅适用于page类型的Ability。

其只有4种取值：

unspecified：由系统自动判断显示方向；

landscape：横屏模式；

portrait：竖屏模式；

followRecent：跟随栈中最近的应用。

(3) formsEnabled表示是否支持卡片功能，仅适用于page类型的Ability。

(4) name表示该Ability的名称，取值可采用反向域名方式表示，由包名加类名组成，如“com.example.createpageability.MyFirstAbility”，也可以使用“.”开头的类名方式表示，如“.MyFirstAbility”，一个应用中的每个Ability的name属性应该是唯一的。

(5) icon表示Ability图标资源文件的索引，用于配置该Ability的图标，这里使用“$media:icon”来表示对resources/base/media下的icon.png文件的引用。

(6) description表示对Ability的描述，这里使用“$string:myfirstability_description”来表示对resources/base/element下的string.json文件中定义的字符串的引用，以后所有的字符串都将在这个文件中定义，使用“$string:”来引用。

(7) label表示Ability对用户显示的名称。

（8）type 表示该 Ability 的类型。

其一共有四种取值：

page：表示基于 Page 模板开发的 FA，用于提供与用户交互的能力；

service：表示基于 Service 模板开发的 PA，用于提供后台运行任务的能力；

data：表示基于 Data 模板开发的 PA，用于对外部提供统一的数据访问抽象；

CA：表示支持其他应用以窗口方式调起该 Ability。

（9）launchType 表示 Ability 的启动模式。

它支持“standard”“singleMission”和“singleton”3 种模式：

standard：表示该 Ability 可以有多实例，“standard”模式适用于大多数应用场景；

singleMission：表示该 Ability 在每个任务栈中只能有一个实例；

singleton：表示该 Ability 在所有任务栈中仅可以有一个实例。例如，具有全局唯一性的呼叫来电界面即采用“singleton”模式。该标签仅适用于手机、平板、智慧屏、车机、智能穿戴。

（10）reqPermissions：表示此 Ability 需要申请的权限。

3.2.3 创建布局文件

HarmonyOS App 所有的布局文件将放在 entry/src/main/resources/base/layout 目录下面，首先，创建一个 Layout Resource File 并命名为 my_first_layout.xml，然后输入以下代码：

```
<?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="$+ id:text"
        ohos:height="200fp"
        ohos:width="match_parent"
        ohos:text_alignment="center"
        ohos:text="$string:myfirstability_description"
        ohos:text_size="30fp"></Text>
    <Button
        ohos:id="$+ id:button"
        ohos:height="50fp"
        ohos:width="200fp"
        ohos:background_element="#00ff00"
        ohos:text="销毁 Ability"
        ohos:text_size="30vp"></Button>
</DirectionalLayout>
```

关于布局和组件，之后的章节会详细进行介绍，读者在此模仿例子使用即可，这里用到

的是方向布局 DirectionalLayout，也是创建布局文件时，IDE 会创建的默认布局。这里用到了两个组件：Text 和 Button。

3.2.4 静态装载布局文件

创建完布局文件后，需要将布局文件加载到 Page Ability 上，才能显示布局中的组件。通常需要在 Page Ability 启动时装载布局文件，也就是需要在 Page Ability 的生命周期方法中的 onStart()方法里完成。关于生命周期方法，后面小节会详细介绍，读者只需要知道 onStart()方法在 Page Ability 启动时调用即可，通常会在这个方法里做一些初始化的工作，例如，加载布局文件，初始化组件，为组件添加事件监听器等。

现在需要调用父类 Ability 的 onStart()方法，并使用 super.setUIContent()方法加载之前创建的布局文件 my_first_layout.xml，注意在写 Java 代码时，用到了其他类或其他类的静态方法时，需要 import 特定的类，这一点在之后的代码中不做体现，代码如下：

```
import ohos.aafwk.ability.Ability;
import ohos.aafwk.content.Intent;
public class MyFirstAbility extends Ability {
    //调用父类 onStart()方法
    public void onStart(Intent intent)
    {
        super.onStart(intent);
        super.setUIContent(ResourceTable.Layout_my_first_layout);
    }
}
```

在 HarmonyOS App 中，系统会将所有静态资源与一个 int 类型的值进行绑定，并将这些值以常量的形式定义在 static 类型的类 ResourceTable 中，以便通过这个静态类调用这些值，通过该值引用相关资源。这些值是自动生成的，以资源文件的名称加上资源类型（作为前缀）作为变量名。例如，布局文件生成的 ID 需要加上前缀 Layout，本例的布局文件是 my_first_layout.xml，因此在 ResourceTable 类中会自动生成 ID：Layout_my_first_layout。根据这个生成规则，还要求资源文件的命名必须符合 Java 标识符的命名规则，否则无法在 ResourceTable 自动生成 ID。

3.2.5 显示 Page Ability

目前为止，一个小型但完整的 Page Ability 已经创建完成，最后一步就是显示这个创建好的 Page Ability。如果想让 MyFirstAbility 作为应用的主 Ability（即程序运行后的显示的第一个页面）显示，可以修改 MyFirstAbility 的配置信息的 skills 部分，将其修改为如下形式：

```
"skills": [
    {
        "entities": [
            "entity.system.home"
```

```
            ],
            "actions": [
                "action.system.home"
            ]
        }
    ]
```

需要注意的是，一个应用只能有一个主 Ability，但在 config.json 文件中还有其他 Ability 的 actions 也设为“action.system.home”，而 HarmonyOS 只会显示在 config.json 文件中遇到的第一个主 Ability。因此如果需要将您的 Ability 设为主 Ability，就需要将您的 Ability 配置信息作为 abilities 中的第一个元素，或者删除其他的 action 属性值为“action.system.home”的配置项。将 MyFirstAbility 作为主 Ability 显示时，打开应用会看到如图 3-3(左)所示的页面，而如果从其他页面显示 MyFirstAbility，如从图 3-3（右）显示 MyFirstAbility，显示效果和图 3-3(左)相同。

图 3-3　显示 MyFirstAbility

3.2.6　销毁 Page Ability

在 Page Ability 使用完后，需要关闭（或销毁）Page Ability，调用如下代码即可销毁 Page Ability：

```
terminateAbility();
```

该方法属于 Ability 类，如果在 AbilitySlice(后面章节介绍)中需要调用该方法，需要获得包含该 AbilitySlice 的 Ability 对象。

3.3 Page Ability 之间的交互

本小节将介绍两个不同的 Page Ability 之间如何进行交互，例如，在两个不同 Page Ability 之间传递数据、通过显式和隐式的方式在一个 Page Ability 中显示另一个 Page Ability。

3.3.1 Intent 的基本概念

Intent 是对象之间传递信息的载体。例如，当一个 Ability 需要启动另一个 Ability 时，或者一个 AbilitySlice 需要导航到另一个 AbilitySlice 时，可以通过 Intent 指定启动的目标同时携带相关数据。Intent 的构成元素包括 Operation 与 Parameters。

Operation 又包含以下 7 项属性：

(1) Action：表示动作，可以使用系统内置的 Action，也可以使用用户在 config.json 文件中自定义的 Action。

(2) Entity：表示类别，可以使用系统内置的 Action，也可以使用用户在 config.json 文件中自定义的 Action。

(3) URI：表示 URI 描述。如果在 Intent 中指定了 URI，则 Intent 将匹配指定的 URI 信息。

(4) Flags：表示处理 Intent 的方式。如 Intent.FLAG_ABILITY_CONTINUATION 标记在本地的一个 Ability 是否可以迁移到远端设备继续运行。

(5) BundleName：表示包描述。

(6) AbilityName：表示待启动的 Ability 名称。如果在 Intent 中同时指定了 BundleName 和 AbilityName，则 Intent 可以直接匹配到指定的 Ability。

(7) DeviceId：表示运行指定 Ability 的设备 ID。

Parameters 是一种支持自定义的数据结构，开发者可以通过 Parameters 传递某些请求所需的额外信息。

3.3.2 显式使用 Intent

所谓显式使用就是同时指定 Operation 属性的 BundleName 和 AbilityName 两项子属性，即根据 Ability 的全称启动 Ability，隐式使用就是指未同时指定 Operation 属性的 BundleName 和 AbilityName，根据 Operation 的其他属性启动 Ability，通常指定 Ability 指定的 action 属性启动 Page Ability。

【例 3.1】 首先需要创建一个名为 ability_main.xml 的布局文件，同时添加两个 Button 组件并绑定到 MainAbility，主 Ability 用于演示显式使用 Intent 和隐式使用 Intent 的区别。布局文件代码如下：

```
<?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">
        <Button
            ohos:id="$+ id:explicit"
            ohos:height="40fp"
            ohos:width="200fp"
            ohos:text="Explicit"
            ohos:text_size="25fp"
            ohos:background_element="$graphic:background_btn"></Button>
        <Button
            ohos:id="$+id:implicit"
            ohos:height="40fp"
            ohos:width="200fp"
            ohos:top_margin="20fp"
            ohos:text="Implicit"
            ohos:text_size="25fp"
            ohos:background_element="$graphic:background_btn"></Button>
    </DirectionalLayout>
```

然后创建一个名为 ExplicitAbility 的 Page Ability，同时绑定一个布局文件，并在配置文件中进行注册，由于代码简单，因此不在此赘述。显式地显示 Page Ability 的步骤如下：

（1）创建 Intent 对象；

（2）使用 Intent. OperationBuilder 类构造包含 BundleName 与 AbilityName 的 Operation 对象；

（3）通过 Intent 的 setOperation()方法指定 Intent 的 Operation 对象；

（4）调用 startAbility()方法显示 Page Ability。

按照以上步骤编写显示 ExplicitAbility 的代码如下：

```
    Intent intent = new Intent();
    Operation operation = new Intent.OperationBuilder()
        //指定设备标识,空串表示当前设备
        .withDeviceId("")
        //指定包名
        .withBundleName("com.example.explicitintent")
        //指定 Page Activity 的 name 属性值
        .withAbilityName("com.example.explicitintent.ExplicitIntentAbility")
        .build();
    intent.setOperation(operation);
    startAbility(intent);
```

其中，withDeviceID 用于指定设备 ID，在 config.json 文件中的 deviceConfig 中进行配置，空串表示当前设备。withBundleName 指定的是 HarmonyOS App 的包名，在 config.json 文件中 bundlename 属性指定。withAbilityName 指定的是 Page Ability 的全名（包名＋类名）。最后需要调用 build()方法返回一个 Operation 对象。执行程序，单击“Explicit”按钮将会显示 ExplicitAbility，效果如图 3-4 所示。

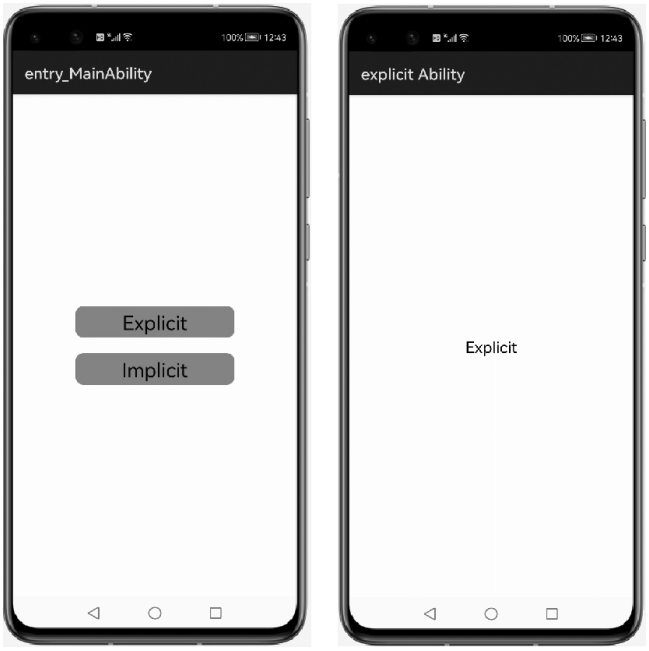

图 3-4　显式使用 Intent

3.3.3　隐式使用 Intent

首先创建一个名为 Implicit1Ability 的 Page Ability 类，同时绑定一个布局文件，然后在 config.json 文件中添加如下配置信息：

```
{
    "skills": [
        {
            "actions": [
                "action.implicit"
            ]
        }
    ],
    "orientation": "unspecified",
    "name": "com.example.intentuse.Implicit1Ability",
    "icon": "$media:icon",
    "description": "$string:implicit1",
    "label": "$string:implicit1",
    "type": "page",
    "launchType": "standard"
}
```

由于在隐式使用 Intent 的方式中需要用到 action 属性，因此需要配置 action 为“action.implicit”，并通过如下代码显示 Implicit1Ability：

```
Intent intent = new Intent();
Operation operation = new Intent.OperationBuilder()
                                .withAction("action. implicit ")
                                .build();
intent.setOperation(operation);
startAbility(intent);
```

隐式使用 Intent 时，不需要指定 BundleName 与 AbilityName，只需要指定 Action 即可。注意，由于在实际开发过程中，有可能遇到多个 Page Ability 指定同一个 action，因此，使用隐式显示 Page Ability 时，HarmonyOS 会弹出一个列表，列表中是所有绑定了同一个 action 的 Page Ability，供用户选择到底使用哪一个 Page Ability。这也是和显式地显示 Page Ability 的一个重要区别，由于显式方法指定确定的 BundleName 与 AbilityName，因此会显示指定 Page Ability。下面通过一个例子介绍多个 Page Ability 指定同一个 action 时会出现的情况。

再创建一个名为 Implicit2Ability 的 Page Ability，同时绑定布局文件，在 config.json 文件中的配置信息如下所示：

```
{
    "skills": [
        {
            "actions": [
                "action.implicit"
            ]
        }
    ],
    "orientation": "unspecified",
    "name": "com.example.intentuse.Implicit2Ability",
    "icon": "$media:icon",
    "description": "$string:implicit2",
    "label": "$string:implicit2",
    "type": "page",
    "launchType": "standard"
}
```

显然，这两个 Page Ability 指定了同一个 action，使用如下代码显示 Page Ability：

```
Intent intent = new Intent();
Operation operation = new Intent.OperationBuilder()
    .withAction("action. implicit ")
    .build();
    intent.setOperation(operation);
startAbility(intent);
```

执行程序，单击“Implicit”按钮执行以上代码，系统将会弹出一个列表供用户选择进入哪个 Page Ability，如图 3-5 所示。

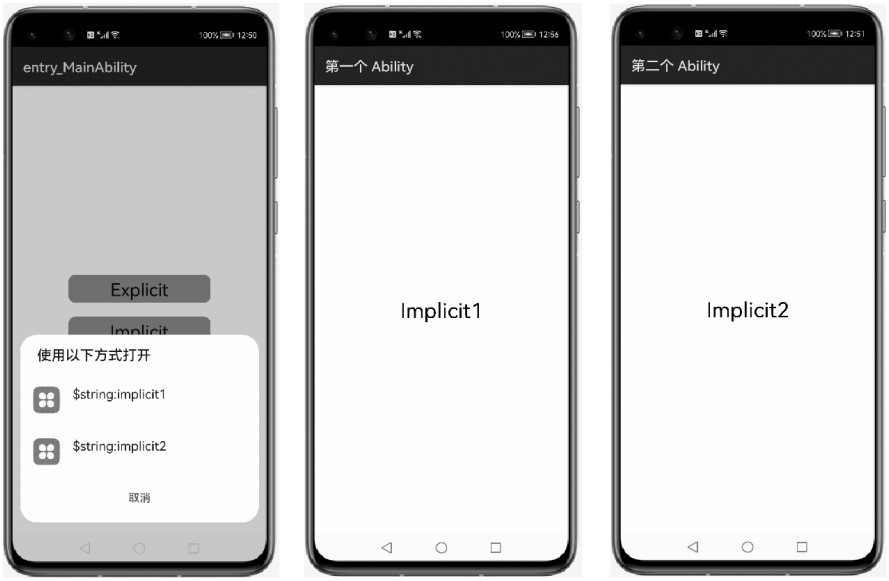

图 3-5　隐式使用 Intent

3.3.4　Page Ability 之间的通信

一般来说，Page Ability 之间是需要进行通信的，而通信方式可以分为两种，一种是 Page Ability 向下一个 Page Ability 传递数据；另一种是 Page Ability 向上一个 Page Ability 返回数据。

第一种通信方式是单向的，因此只用通过 Intent 对象的 setParam()函数携带相关数据并调用 startAbility()方法传递到下一个 Page Ability 即可，代码如下：

```
Intent intent = new Intent();
intent.setParam("data","mydata");
startAbility(intent);
```

执行这段代码，就能将数据传到下一个 Page Ability，在下一个 Page Ability 中，通过调用 Intent.getStringParam()方法就能获得传递过来的数据，代码如下：

```
Intent myintent = getIntent();
String data = myintent.getStringParam("data");
```

第二种通信方式是双向的，从 Page Ability1 传递数据并跳转到 Page Ability2 时，需要 Page Ability2 返回数据给 Page Ability1。此时，需要使用 startAbilityForResult()方法。当使用该方法跳转到 Page Ability2 时，由 Page Ability2 返回到 Page Ability1 后，Page Ability1 会自动调用 onAbilityResult()方法，因此，需要提前在 Page Ability1 中重写该方法用于接收从 Page Ability2 传递过来的数据，该方法的原型：

```
protected void onAbilityResult (int requestCode, int resultCode,
                                Intent resultData);
```

onAbilityResult()方法的参数含义如下。

（1）requestCode：请求码，通过在启动 Ability 时设置，即 startAbilityResult()方法的第 2 个参数。

（2）resultCode：响应码，在 Page Ability2 中调用 setResult()方法时设置。

（3）resultData：由 Page Ability2 返回的 Intent 对象形式的数据。

这里重点介绍一下 requestCode 和 resultCode。因为在 Page Ability 中可能会通过 startAbilityForResult()方法显示多个 Page Ability，所以 onAbilityResult()方法可能是多个 Page Ability 共享的，这就要求在 onAbilityResult()方法中区别是哪一个 Page Ability 返回的结果。可以通过 requestCode 或 resultCode 单独区分不同的 Page Ability，也可以使用 requestCode 和 resultCode 共同区分不同的 Page Ability。如果在 Page Abilityl 中通过 startAbilityForResult()方法显示 Page Ability2，那么 requestCode 应该在 Page Ability1 中指定，而 resultCode 应该在 Page Ability2 中指定。

【例 3.2】 演示两个 Page Ability 之间是如何通信的。在 Page Ability1 调用 startAbilityForResult()方法显示 Page Ability2，并传递一个字符串类型数据，Page Ability2 接收这个数据，并显示在页面上，然后关闭 Page Ability2 并返回一些数据给 Page Ability1，Page Ability1 在 onAbilityResult()方法中接收来自 Page Ability2 的数据。

首先创建一个名为 pageability1_layout.xml 的布局文件，布局中包括一个 Text，用于显示 Page Ability2 返回的数据，一个 Button，用于显示 Page Ability2，代码如下：

```
<?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="$+id:text_page1"
        ohos:height="50fp"
        ohos:width="match_parent"
        ohos:text_alignment="center"
        ohos:text="待接收数据..."
        ohos:text_size="25fp"></Text>
    <Button
        ohos:id="$+id:button_page1"
        ohos:height="50fp"
        ohos:width="300fp"
        ohos:background_element="$graphic:bcakground_btn"
        ohos:text="To PageAbility2"
        ohos:text_size="30fp"></Button>
</DirectionalLayout>
```

然后创建一个名为 pageability2_layout.xml 的布局文件，布局中包括一个 Text，用于显示 Page Ability1 传递来的数据，一个 Button，用于关闭 Page Ability2，代码如下：

```
<?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="$+id:text_page2"
        ohos:height="50fp"
        ohos:width="match_parent"
        ohos:text_alignment="center"
        ohos:text="待接收数据..."
        ohos:text_size="25fp"></Text>
    <Button
        ohos:id="$+id:button_page2"
        ohos:height="50fp"
        ohos:width="300fp"
        ohos:background_element="$graphic:bcakground_btn"
        ohos:text="Back PageAbility1"
        ohos:text_size="30fp"></Button>
</DirectionalLayout>
```

接下来创建一个名为 PageAbility1 的 Page Ability 类，主要代码如下：

```
public class PageAbility1 extends Ability {
    Button button;
    Text text;
    public void onStart(Intent intent){
        super.onStart(intent);
        //加载布局文件
        super.setUIContent(ResourceTable.Layout_pageability1_layout);
        //通过组件 ID 获取组件
        button = (Button)findComponentById(ResourceTable.Id_button_page1);
        text = (Text)findComponentById(ResourceTable.Id_text_page1);
        //绑定按钮监听事件
        button.setClickedListener(new Component.ClickedListener() {
            @Override
            public void onClick(Component component) {
                Intent intent = new Intent();
                //传递一个字符串
                intent.setParam("pageability1_data",
                    "这是 Page Ability1 传递的数据");
                    Operation operation = new Intent.OperationBuilder()
                        .withBundleName("com.example.interactionofpage")
                        .withAbilityName("com.example.interactionofpage
                        .PageAbility2").build();
                    intent.setOperation(operation);
                    //显示 PageAbility2,设置请求码为 100
                    startAbilityForResult(intent,100);
```

```
                }
            });
        }
    //重写 onAbilityResult 方法,当 PageAbility2 关闭时自动调用
    protected void onAbilityResult(int requestCode, int resultCode,
                                   Intent resultData){
        switch (requestCode){
            case 100:
                switch(resultCode){
                    case 101:
                        //接收返回的数据
                        String data = resultData.getStringParam
                        ("pageability2_data");
                        //将接收到的数据显示在页面中 Text 组件上
                        if(data!=null)text.setText(data);
                        break;
                    }
                    break;
        }
    }
}
```

这段代码实现了使用 startAbilityForResult()方法从 PageAbility1 显示 PageAbility2，并传递一个字符串类型数据，重写了 onAbilityResult()，对 requestCode 和 resultCode 进行了判断，如果满足条件，则接收返回的数据，现在需要创建一个名为 PageAbility2 的 PageAbility 类，主要代码如下：

```
public class PageAbility2 extends Ability {
    Button button;
    Text text;
    public void onStart(Intent intent){
        super.onStart(intent);
        //加载布局文件
        super.setUIContent(ResourceTable.Layout_pageability2_layout);
        //获取 PageAbility1 传递的数据
        Intent myintent = getIntent();
        String data = myintent.getStringParam("pageability1_data");
        //通过组件 ID 获取组件
        button = (Button)findComponentById(ResourceTable.Id_button_page2);
        text = (Text)findComponentById(ResourceTable.Id_text_page2);
        //将传递来的数据显示在页面的 Text 组件上
        if(data != null)text.setText(data);
        //绑定按钮监听,返回 PageAbility1,并返回一个字符串类型数据
        button.setClickedListener(new Component.ClickedListener() {
            @Override
            public void onClick(Component component) {
                Intent resultintent = new Intent();
                //设置返回到 PageAbility1 的数据
                resultintent.setParam("pageability2_data",
                            "这是 PageAbility2 返回的数据");
```

```
                //设置响应码以及返回的 Intent 对象
                setResult(101, resultintent);
                //关闭 PageAbility2
                terminateAbility();
            }
        });
    }
}
```

现在，两个 Page Ability 已经准备就绪，最后在 config.json 文件中注册这两个 Page Ability 即可使用这两个 Page Ability。配置信息如下：

```
{
    "orientation": "unspecified",
    "name": "com.example.interactionofpage.PageAbility1",
    "icon": "$media:icon",
    "description": "$string:pageability1",
    "label": "$string:pageability1",
    "type": "page",
    "launchType": "standard"
},
{
    "orientation": "unspecified",
    "name": "com.example.interactionofpage.PageAbility2",
    "icon": "$media:icon",
    "description": "$string:pageability2",
    "label": "$string:pageability2",
    "type": "page",
    "launchType": "standard"
}
```

运行程序，效果如图 3-6 所示。

图 3-6　Page Ability 交互效果图

3.4 Page Ability 的启动类型

在之前的例子中，所有 Ability 的配置信息 launchType 的值都为 standard，这也是 launchType 属性的默认值。launchType 还有两个值为 singleMission 和 singleton，本节将对这 3 种属性值的作用进行介绍。

standard 表示此 Ability 可以创建多个实例，且在任何情况下，无论 Page Ability 被显示多少次，每次被显示都会创建一个新的 Page Ability 实例。

singleMission 表示此 Ability 在每个任务栈中只能有一个实例。如果要显示的 Page Ability 在栈顶，那么再次显示这个 Page Ability 时，不会再创建新的 Page Ability 实例，而是直接使用这个 Page Ability 实例。如果 Page Ability 上面有其他的 Page Ability，那么首先弹出这些 Page Ability，然后再复用这个 Page Ability。

singleton 表示该 Ability 在所有任务栈中只能有一个实例，例如，具有全局唯一性的呼叫来电界面即采用“singleton”模式。

其中涉及的栈是 HarmonyOS 管理 Page Ability 的模式。由于 HarmonyOS App 只能显示一个 Page Ability，HarmonyOS App 会使用栈来管理 App 中的所有 Page Ability，只有在栈顶的 Page Ability 才能显示，如果要让非栈顶的 Page Ability 显示，那么就需要将要显示的 Page Ability 之前的所有 Page Ability 销毁，使得要显示的 Page Ability 位于栈顶，要销毁 Page Ability，则需要调用 terminateAbility()方法。

【例 3.3】 由于 singleMission 模式和 singleton 模式都只允许 Ability 只有一个实例，本例仅演示 standard 和 singleMission 启动类型的区别。

首先创建一个名为 LaunchTypeAbility1 的 Page Ability 类，主要代码如下：

```
public class LaunchTypeAbility1 extends Ability {
    private static int count = 0;       //计数器
    public void onStart(Intent intent){
        super.onStart(intent);
        super.setUIContent(ResourceTable.Layout_launchtype1);
        //每创建一个 LaunchTypeAbility1 对象,计数器加一
        count++;
        Button button = (Button)findComponentById(ResourceTable.Id_button1);
        Text count1 = (Text)findComponentById(ResourceTable.Id_count1);
        if(count1 != null){
            //将计数器的值显示在页面中
            count1.setText(String.valueOf(count));
        }
        //设置按钮单击事件监听器
        button.setClickedListener(new Component.ClickedListener() {
            @Override
            public void onClick(Component component) {
                Intent intent = new Intent();
                Operation operation = new Intent.OperationBuilder()
                .withBundleName("com.example.launchtype").withAbilityName
```

```
                ("com.example.launchtype.LaunchTypeAbility1").build();
                intent.setOperation(operation);
                startAbility(intent);
            }
        });
    }
}
```

其次在 config.json 文件中添加这个类的配置信息，注意，先将这个类的 launchType 属性都设置为 standard，在 LaunchTypeAbility1 重复显示 LaunchTypeAbility1，这样可以很好地观察到在 standard 和 singleMission 这两种启动模式下，计数器的变化情况。

LaunchTypeAbility1 这个类中定义了一个变量 count 作为计数器，每次创建一个 LaunchTypeAbility1 实例对象，计数器都会加一，在 standard 启动模式下，显示三次 LaunchTypeAbility1，系统则会创建三个 LaunchTypeAbility1 实例对象，因此计数器的值变为 3，页面效果如图 3-7 所示。

现在将 LaunchTypeAbility1 的启动模式改为 singleMission，然后再显示三次 LaunchTypeAbility1。由于在 singleton 模式下，显示 LaunchTypeAbility1 并不会创建新的 LaunchTypeAbility1 实例对象，可以观察到计数器的值都为 1，页面效果如图 3-8 所示。

图 3-7 standard 启动模式的页面效果图

图 3-8 singleMission 启动模式的页面效果

3.5 Page Ability 的跨设备迁移

HarmonyOS 的一大技术特性是分布式任务调度，所谓分布式任务调度，就是构建统一的分布式服务管理（发现、同步、注册、调用）机制，支持对跨设备的应用进行远程启动、远程

调用、远程连接以及迁移等操作，能够根据不同设备的能力、位置、业务运行状态、资源使用情况，以及用户的习惯和意图，选择合适的设备运行分布式任务。而 Page Ability 的跨设备迁移则依赖了分布式设备调度中的业务迁移能力，是分布式任务调度的一个具体实现。跨设备迁移支持将 Page 在同一用户的不同设备间迁移，以便支持用户无缝切换的诉求。实现这个操作的前提是参加跨设备迁移的设备在同一个网段内，或者登录了同一个 HUAWEI 账号，接下来将介绍跨设备前需要准备的工作。

3.5.1 跨设备迁移前的准备工作

在进行跨设备迁移之前（后面几章所讲的跨设备调用 Data Ability、Service Ability 也同样要进行这些准备），需要对 HarmonyOS 设备做一些准备工作：

（1）打开 HarmonyOS 设备的蓝牙，并把设备名称修改为可识别的名称，如图 3-9 所示；

（2）将 HarmonyOS 设备连入 Wi-Fi，并且所有参与跨设备迁移的 HarmonyOS 设备在同一个网段；

（3）所有参与跨迁移的 HarmonyOS 设备登录同一个 HUAWEI 账号；

（4）选择“设置”→“超级终端”查看附近设备如图 3-10 所示。

图 3-9 设置设备名称

图 3-10 附近设备

3.5.2 获取设备列表

跨设备迁移需要知道目标设备的 ID，因此需要提前获取所有可用的设备 ID。HarmonyOS 提供了一个用于获取所有设备信息的方法，即 DeviceManager.getDeviceList() 方法，该方法返回一个 List 列表，类型是 DeviceInfo。DeviceInfo 类型描述了设备的名称和

设备 ID 等相关信息，实现代码如下：

```
List<DeviceInfo> deviceInfoList =
        DeviceManager.getDeviceList(DeviceInfo.FLAG_GET_ONLINE_DEVICE)
```

其中，getDeviceList()方法的参数表示获取哪种状态下的设备的信息，一共有 3 种取值：

(1) DeviceInfo.FLAG_GET_ONLINE_DEVICE：所有在线设备；

(2) DeviceInfo.FLAG_GET_OFFLINE_DEVICE：所有离线设备；

(3) DeviceInfo.FLAG_GET_ALL_DEVICE：所有设备。

一般会使用第一个值，因为只有设备在线，才能进行迁移 Page Ability 操作。

【例 3.4】 实现一个获取在线设备列表的 Page Ability，单击一个设备会返回该设备的 ID。首先创建一个名为 device_ids.xml 的布局文件，代码如下：

```
<?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:orientation="vertical">
    <Text
        ohos:height="50vp"
        ohos:width="match_parent"
        ohos:text="可用设备 ID"
        ohos:text_size="35vp"></Text>
    <ListContainer
        ohos:id="$+id:device_id_List"
        ohos:height="match_parent"
        ohos:width="match_parent"></ListContainer>
</DirectionalLayout>
```

布局中包含了一个 ListContainer 组件，用于显示所有设备信息，再创建一个名为 device_id_item.xml 的布局文件，作为 ListContainer 组件的 Item 布局，代码如下：

```
<?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:orientation="vertical">
    <DirectionalLayout
        ohos:height="50vp"
        ohos:width="match_parent"
        ohos:orientation="horizontal">
    <Text
        ohos:height="match_parent"
        ohos:width="150vp"
        ohos:text="device_name:"
```

```
            ohos:text_size="20fp"></Text>
        <TextField
            ohos:id="$+id:device_name"
            ohos:height="match_parent"
            ohos:width="match_parent"
            ohos:text="huawei"
            ohos:text_size="20vp"></TextField>
        </DirectionalLayout>
        <DirectionalLayout
            ohos:height="200vp"
            ohos:width="match_parent"
            ohos:orientation="horizontal">
        <Text
            ohos:height="match_parent"
            ohos:width="150vp"
            ohos:text="device_id:"
            ohos:text_size="20fp"></Text>
        <TextField
            ohos:id="$+id:device_id"
            ohos:height="match_parent"
            ohos:width="match_parent"
            ohos:text="000"
            ohos:text_size="20vp"></TextField>
        </DirectionalLayout>
    </DirectionalLayout>
```

该布局文件中放置的两个 TextField 组件，分别显示设备名称和设备 ID。接下来是 Page Ability 的实现。创建一个名为 DeviceIDAbility 的 Page Ability 类，代码如下：

```
public class DeviceIDAbility extends Ability {
    //存放获取到的在线设备信息
    private List<DeviceInfo> deviceInfoList;
    //展示设备信息的容器
    private ListContainer listContainer;
    //获取所有在线设备信息
    private static List<DeviceInfo> getAllOnlineDeviceInfo(){
        List<DeviceInfo> deviceInfoList =
            DeviceManager.getDeviceList (DeviceInfo.FLAG_GET_ONLINE_DEVICE);
        if(deviceInfoList == null || deviceInfoList.size() == 0)
        {
            return new ArrayList<>();
        }
        else{
            return deviceInfoList;
        }
    }
    public void onStart(Intent intent) {
```