第 5 章

布　　局

【学习目标】

■ 理解 HarmonyOS 应用中的布局

■ 掌握布局创建的两种基本方法

■ 掌握常用的布局，如 DirectionalLayout、DependentLayout、StackLayout、TableLayout、PositionLayout 等

■ 了解自定义布局及组件继承体系

5.1　布局概述

在 HarmonyOS 的 UI 框架中，布局其实就是组件容器。组件是为了解决在界面上放什么的问题，而布局则是为了解决怎么放的问题。我们可以把用户看到的界面看成一个大容器，容器中可以放置组件，也可以放置小的容器。

放置在用户界面中的布局（组件容器）和组件结构可以用树表示，树根是一个布局，里面包含组件和子布局，子布局中又包含组件或布局。实际上，组件容器可以理解成特殊的组件，这样用户界面中的元素就形成了一棵组件树，如图 5-1 所示，根据这种规律，设计者可以设计出丰富的界面效果。

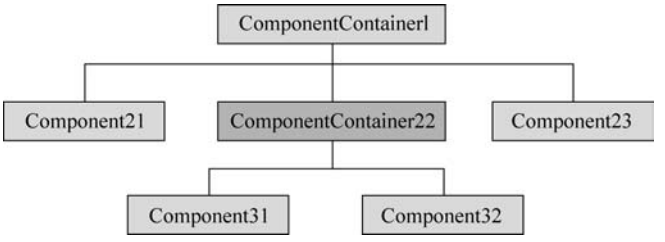

图 5-1　组件树

组件树只是说明了应用界面中组件容器和组件之间的逻辑关系，为了更好地理解，可以把设备屏幕理解成一个矩形框。用户界面中的组件树结构如图 5-2 所示，这样就可以更容

易地和设备中的应用界面对应。

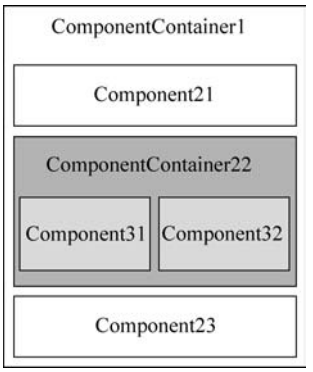

图 5-2　用户界面中的组件树结构

尽管根据图 5-2 更容易把组件树结构和应用界面联系在一起，但是，同一个组件树在屏幕中放置组件的位置还没有确定。如果不进行一定的设定或约束，则势必会造成界面凌乱。布局可以很好地组织界面中的元素，使应用界面组织更有序、更丰富多彩。

Java UI 框架关于组件和组件容器提供了 Component 和 ComponentContainer 两个类，前者称为组件，后者称为组件容器。实际上，ComponentContainer 类继承自 Component 类，因此，组件容器是特殊的组件。

正如 Text、Button、Image 等继承自 Componet 的具体组件一样，组件容器也有常用的具体组件容器，如 DirectionalLayout、DependentLayout、StackLayout 等，这些具体的组件容器一般以 Layout 结尾，故通常称为布局。

每种布局都有自己的特点，根据自身特点提供了布局配置，通过布局配置为布局中的组件设定布局属性和参数。布局通过设置属性对包含在其中的子组件进行布置和约束，进而使应用界面的效果达到设计要求。

在布局文件代码方面，组件树是以 XML 文件方式组织的，布局属性是和 XML 标签属性对应的，图 5-2 对应的 XML 文件的结构如下：

```
<ComponentContainer1
    属性 = 值
    … >
    <Component21 />
    <ComponentContainer22
        属性 = 值
        … >
        <Component31 />
        <Component32 />
    </ComponentContainer22>
    <Component23 />
</ComponentContainer1>
```

5.2　创建布局方式

创建布局有两种基本方式，一种方式是通过 XML 布局文件加载布局，另外一种方式是通过系统提供的布局类创建布局对象。

1. 通过 XML 文件加载布局

通过 XML 声明布局的方式简便直观。XML 文件本身就是一个树形结构的标签文档，系统支持的每种布局都可以用 XML 标签表示，布局的属性可以通过设置 XML 标签的属性表示。

定义方向布局(DirectionalLayout)，代码如下：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< DirectionalLayout
    xmlns:ohos = "http://schemas.huawei.com/res/ohos"
    ohos:width = "match_parent"
    ohos:height = "match_parent"
    ohos:orientation = "vertical"
    ohos:padding = "32">
    < Button
        ohos:id = " $ + id:button"
        ohos:margin = "50"
        ohos:width = "match_content"
        ohos:height = "match_content"
        ohos:layout_alignment = "horizontal_center"
        ohos:text = "My name is Button."
        ohos:text_size = "50"/>
</DirectionalLayout >
```

布局中可以放置组件或布局，这样编写的布局代码可以根据需要定义得更加复杂，只要按照布局树的基本结构，就可以任意定义布局文件。

通过 setUIContent()方法可以使用定义好的布局文件资源，代码如下：

```
setUIContent(ResourceTable.Layout_ability_main);
```

在项目中，布局文件一般放置在 entry→src→main→resources→base→layout 目录下。

2. 使用布局类创建布局

布局在代码中也是一个对象，使用内置的布局类可以直接创建布局对象，通过设置对象的一些属性达到使用布局的目的。

下面以方向布局(DirectionalLayout)为例，说明使用该方式的过程。

首先，创建布局对象，代码如下：

```
DirectionalLayout dirLayout = new DirectionalLayout(getContext());
```

其次，设置布局属性，一般需要设置大小，即宽度和高度。调用方法设置布局对象的属性基本的代码如下：

```
dirLayout.setWidth(ComponentContainer.LayoutConfig.MATCH_PARENT);
dirLayout.setHeight(ComponentContainer.LayoutConfig.MATCH_PARENT);
```

方向布局一般需要设置布局中组件的排列方向，将排列方向设置为垂直排列，代码如下：

```
dirLayout.setOrientation(Component.VERTICAL);
```

通过调用布局方法，可以向布局中添加组件，也可以添加子布局。向布局中添加一个事先定义好的 button 组件，代码如下：

```
dirLayout.addComponent(button);
```

最后，将布局设置到应用界面中，代码如下：

```
setUIContent(directionalLayout);
```

17min

5.3 常用布局

5.3.1 DirectionalLayout

DirectionalLayout 布局，也可以称为方向布局或线性布局，该布局中的组件会按照某一方向线性排列在其中，方向布局是最常用的布局之一。

DirectionalLayout 的排列方向（orientation）分为水平（horizontal）和垂直（vertical）两种方式。可以使用 orientation 属性设置布局内组件的排列方式，默认为垂直方式。

如果希望在一个布局里以垂直方向依次放置 3 个按钮组件，则可以选择方向布局，设计效果如图 5-3 所示。

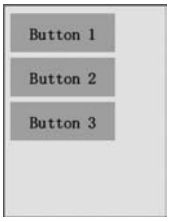

图 5-3 垂直方向线性排列

将 3 个按钮按照垂直方向依次放置到方向布局中，左边界和布局对齐，按钮之间有一定的间隔，代码如下：

```
//ch05\DirectionalLayout1 项目中的 ability_main.xml
<?xml version = "1.0" encoding = "utf - 8"?>
< DirectionalLayout
    xmlns:ohos = "http://schemas.huawei.com/res/ohos"
    ohos:width = "match_parent"
    ohos:height = "match_content"
    ohos:orientation = "vertical">
< Button
```

```
            ohos:width = "90vp"
            ohos:height = "50vp"
            ohos:top_margin = "10vp"
            ohos:left_margin = "10vp"
            ohos:background_element = " $ graphic:color_button_element"
            ohos:text_size = "20vp"
            ohos:text = "Button 1"/>
        < Button
            ohos:width = "90vp"
            ohos:height = "50vp"
            ohos:top_margin = "10vp"
            ohos:left_margin = "10vp"
            ohos:background_element = " $ graphic:color_button_element"
            ohos:text_size = "20vp"
            ohos:text = "Button 2"/>
        < Button
            ohos:width = "90vp"
            ohos:height = "50vp"
            ohos:top_margin = "10vp"
            ohos:left_margin = "10vp"
            ohos:background_element = " $ graphic:color_button_element"
            ohos:text_size = "20vp"
            ohos:text = "Button 3"/>
    </DirectionalLayout >
```

以上布局代码运行的效果如图 5-4 所示，该布局中的组件排列方向是通过布局的属性 ohos:orientation 进行设置的，当其值为 vertical 时，布局中的组件将按照垂直方向排列，当其值为 horizontal 时，布局中的组件将按照水平方向排列。

图 5-4　垂直方向线性布局的运行效果

在布局代码中，方向布局属性 ohos:width 的值是 match_parent，表示其宽度匹配父组件容器，由于该方向布局的外层没有别的布局了，因此其父组件容器为设备屏幕，属性 ohos:height 的值为 match_content，表示适配其内容，即自动匹配其中的组件的高度。

代码中按钮组件属性 ohos:background_element 的值为 $ graphic:color_button_element，引用了别的资源文件，所引用的资源必须存在，因此对应的资源文件 color_button_element.xml 必须存在，代码如下：

```
//ch05\DirectionalLayout1 项目中的 color_button_element.xml
<?xml version = "1.0" encoding = "utf - 8"?>
```

```
<shape xmlns:ohos = "http://schemas.huawei.com/res/ohos"
        ohos:shape = "rectangle">
    <solid
        ohos:color = "#00FFFD"/>
</shape>
```

在方向布局中,通过 orientation 属性可以设定其中的组件应遵循的排列方向,但并不是说其中的组件只能放置在布局的一侧,可以使用组件的属性 layout_alignment 控制组件本身在布局中的对齐方式,当方向布局为垂直方向时,其中的组件 layout_alignment 的属性值可以为 left、horizontal_center、right。

如图 5-5 所示,3 个按钮按照垂直方向排列,第 1 个按钮居左,第 2 个按钮居中,第 3 个按钮居右。

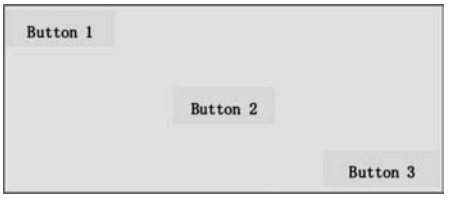

图 5-5　不同的对齐方式

设置布局中组件的对齐方式属性 ohos:layout_alignment 的值,代码如下:

```
//ch05\DirectionalLayout2 项目中的 ability_main.xml
<?xml version = "1.0" encoding = "utf-8"?>
<DirectionalLayout
    xmlns:ohos = "http://schemas.huawei.com/res/ohos"
    ohos:width = "match_parent"
    ohos:height = "match_content"
    ohos:orientation = "vertical">
    <Button
        ohos:width = "90vp"
        ohos:height = "50vp"
        ohos:margin = "10vp"
        ohos:background_element = "$graphic:color_button_element"
        ohos:text_size = "20vp"
        ohos:layout_alignment = "left"
        ohos:text = "Button 1"/>
    <Button
        ohos:width = "90vp"
        ohos:height = "50vp"
        ohos:margin = "10vp"
        ohos:background_element = "$graphic:color_button_element"
        ohos:text_size = "20vp"
```

```
            ohos:layout_alignment = "center"
            ohos:text = "Button 2"/>
        < Button
            ohos:width = "90vp"
            ohos:height = "50vp"
            ohos:margin = "10vp"
            ohos:background_element = " $ graphic:color_button_element"
            ohos:text_size = "20vp"
            ohos:layout_alignment = "right"
            ohos:text = "Button 3"/>
    </DirectionalLayout >
```

运行效果如图 5-6 所示。

实际上既可以理解成 3 个按钮组件按照垂直排列，也可以理解成按照水平排列，只不过按照水平方向排列时，它们在父组件中的对齐方式依次为上、中、下而已。

在应用中，很多时候需要平均或按照一定比例分配组件所在的布局的空间大小，如图 5-7(a)下方的导航菜单所示，4 个菜单选项“消息”“好友”“动态”“我的”被平均分布在一个布局中，这样当界面从竖屏调整到横批时可以动态地调整，如图 5-7(b)所示。

图 5-6 不同的对齐方式的运行效果图

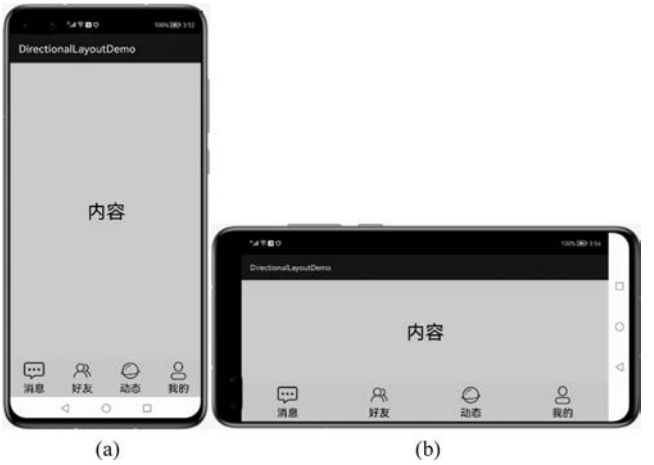

图 5-7 导航菜单效果图

为了达到组件在线性布局中按照一定比例显示的目的，可以为线性布局中的组件设置权重，即 weight 属性，代码如下：

```
//ch05\DirectionalLayoutDemo 项目中的 ability_main.xml
<?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 = "0"
        ohos:width = "match_parent"
        ohos:layout_alignment = "center"
        ohos:text_alignment = "center"
        ohos:background_element = " # CCCCCC"
        ohos:text = "内容"
        ohos:text_size = "36vp"
        ohos:weight = "1"
    >
    </Text >

    < DirectionalLayout
        xmlns:ohos = "http://schemas.huawei.com/res/ohos"
        ohos:height = "match_content"
        ohos:width = "match_parent"
        ohos:background_element = " # 330000FF"
        ohos:orientation = "horizontal">

        < Button
            ohos:height = "match_content"
            ohos:width = "0vp"
            ohos:element_top = " $ media:icon"
            ohos:margin = "5vp"
            ohos:text = "消息"
            ohos:text_size = "20vp"
            ohos:layout_alignment = "center"
            ohos:weight = "1"/>

        < Button
            ohos:height = "match_content"
            ohos:width = "0vp"
            ohos:element_top = " $ media:icon"
            ohos:margin = "5vp"
            ohos:text = "好友"
            ohos:text_size = "20vp"
            ohos:layout_alignment = "bottom"
            ohos:weight = "1"/>

        < Button
            ohos:height = "match_content"
            ohos:width = "0vp"
            ohos:element_top = " $ media:icon"
```

```
            ohos:margin = "5vp"
            ohos:text = "动态"
            ohos:text_size = "20vp"
            ohos:layout_alignment = "bottom"
            ohos:weight = "1"/>

        < Button
            ohos:height = "match_content"
            ohos:width = "0vp"
            ohos:element_top = " $ media:icon"
            ohos:margin = "5vp"
            ohos:text = "我的"
            ohos:text_size = "20vp"
            ohos:layout_alignment = "bottom"
            ohos:weight = "1"/>
    </DirectionalLayout >
</DirectionalLayout >
```

该布局文件，外层是一个方向布局，布局的方向是垂直方向，里面包括一个 Text 和一个内层方向布局。内层方向布局的方向是水平方向，里面包含了 4 个按钮，分别是“消息”“好友”“动态”和“我的”，这 4 个按钮的 ohos:weight 属性值都被设置成了 1，由于它们的权重一样，所以平均分配内层方向布局的水平空间。

一般情况下，组件在分配父布局的水平宽度空间的计算公式如下：

$$\text{组件宽度} = \frac{\text{组件权重}}{\text{同级所有组件权重之和}} \times \text{父布局可分配宽度} \tag{5-1}$$

其中：

$$\text{父布局可分配宽度} = \text{父布局宽度} - \text{所有同级组件宽度之和} \tag{5-2}$$

在实际使用过程中，为了使组件大小和权重成比例，一般会将同级的组件宽度属性 width 的值均设置为 0，这样组件宽度就会按设置的权重比例分配父布局的宽度。对于未设置权重的组件，默认其权重是 0。

在垂直方向的线性布局中，组件权重设置和组件高度分配与水平方向的线性布局类似，这里不再赘述。

5.3.2 DependentLayout

15min

DependentLayout 称为依赖布局，与方向布局相比，依赖布局拥有更多的排布方式。在依赖布局里，组件可以相对于其他组件设置位置，可以相对于其他同级元素组件的位置设置位置，也可以相对于父组件的位置设置位置。

依赖布局自身相对于父组件的对齐方式的属性取值如表 5-1 所示。

表 5-1 依赖布局的对齐方式的属性取值

属性取值	取值说明	使用说明
ohos:alignment="left"	左对齐,和其父组件左侧对齐	可以设置单个值,也可以使用"|"进行多值组合 如:表示居上和居左 ohos: alignment = "top|left"
ohos:alignment="top"	顶部对齐	
ohos:alignment="right"	右对齐	
ohos:alignment="bottom"	底部对齐	
ohos:alignment="horizontal_center"	水平居中,放置到其父组件的水平中心	
ohos:alignment="vertical_center"	垂直居中,放置到其父组件的垂直中心	
ohos:alignment="center"	居中,水平和垂直同时居中	

在依赖布局中,组件可以通过一些属性设置其相对于其他组件或依赖布局的相对位置,组件的主要属性及含义如表 5-2 所示。

表 5-2 依赖布局中组件的属性值说明

属性名称	值类型	说明
left_of	引用类型,仅可引用依赖布局中包含的其他组件的 ID	将右边缘与另一个子组件的左边缘对齐
right_of	同上	将左边缘与另一个子组件的右边缘对齐
above	同上	将下边缘与另一个子组件的上边缘对齐
below	同上	将上边缘与另一个子组件的下边缘对齐
align_baseline	同上	将子组件的基线与另一个子组件的基线对齐
align_left	同上	将左边缘与另一个子组件的左边缘对齐
align_top	同上	将上边缘与另一个子组件的上边缘对齐
align_right	同上	将右边缘与另一个子组件的右边缘对齐
align_bottom	同上	将底边与另一个子组件的底边对齐
start_of	同上	将结束边与另一个子组件的起始边对齐
end_of	同上	将起始边与另一个子组件的结束边对齐
align_start	同上	将起始边与另一个子组件的起始边对齐
align_end	同上	将结束边与另一个子组件的结束边对齐
align_parent_left	布尔类型值	将左边缘与父组件的左边缘对齐,true 表示对齐
align_parent_top	布尔类型值	将上边缘与父组件的上边缘对齐,true 表示对齐
align_parent_right	布尔类型值	将右边缘与父组件的右边缘对齐,true 表示对齐
align_parent_bottom	布尔类型值	将底边与父组件的底边对齐,true 表示对齐
center_in_parent	布尔类型值	将子组件保持在父组件的中心,true 表示对齐
horizontal_center	布尔类型值	将子组件保持在父组件水平方向的中心,true 表示对齐
vertical_center	布尔类型值	将子组件保持在父组件垂直方向的中心,true 表示对齐
align_parent_start	布尔类型值	将起始边与父组件的起始边对齐,true 表示对齐
align_parent_end	布尔类型值	将结束边与父组件的结束边对齐,true 表示对齐

表 5-2 中属性的名字和它的含义是基本对应的。下面以图 5-8 说明组件间的相对位置

依赖关系，假定依赖布局中组件 A 的位置已经确定，组件 B～组件 F 和组件 A 位于同一个依赖布局内，它们可以放置在 A 的周边，这时我们就可以设置组件 B～组件 F 的属性值来确定相对于组件 A 的位置。

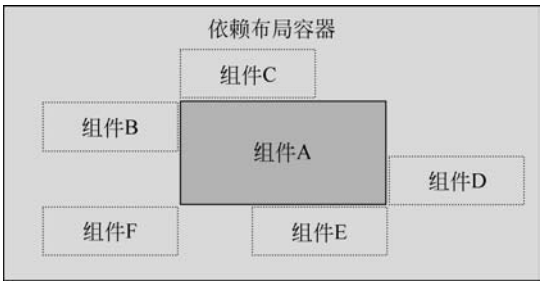

图 5-8 依赖组件的位置关系

假设组件 A 的 ID 是 component_a，如果希望达到图 5-8 所示的放置效果，则组件 B～组件 F 的属性设置如表 5-3 所示。

表 5-3 组件 B～F 相对于组件 A 的位置属性设置

组件	属性设置	说明
组件 B	ohos:left_of="$id:component_a" ohos:align_top="$id:component_a"	放到 A 的左侧 上边缘和 A 对齐
组件 C	ohos:above="$id:component_a" ohos:align_left="$id:component_a"	放到 A 的上方 左边缘和 A 对齐
组件 D	ohos:right_of="$id:component_a" ohos:align_bottom="$id:component_a"	放到 A 的右侧 下边缘和 A 对齐
组件 E	ohos:below="$id:component_a" ohos:align_right="$id:component_a"	放到 A 的下边 右边缘和 A 对齐
组件 F	ohos:left_of="$id:component_a" ohos:below="$id:component_a"	放到 A 的左侧 下边缘和 A 的上边缘对齐
	ohos:below="$id:component_a" ohos:start_of="$id:component_a"	放到 A 的下面 结束边(右边)和 A 的起始边(左边)对齐

具体实现的布局，代码如下：

```
//ch05\DependentLayout1 项目中的 ability_main.xml
<?xml version = "1.0" encoding = "utf - 8"?>
< DependentLayout
        xmlns:ohos = "http://schemas.huawei.com/res/ohos"
        ohos:width = "match_parent"
        ohos:height = "match_parent">
    < Button
        ohos:id = "$ + id:component_a"
```

```
        ohos:width = "100vp"
        ohos:height = "100vp"
        ohos:margin = "1vp"
        ohos:horizontal_center = "true"
        ohos:vertical_center = "true"
        ohos:background_element = " $ graphic:color_button_element"
        ohos:text_size = "20vp"
        ohos:text = "组件 A"/>
    < Button
        ohos:id = " $ + id:component_b"
        ohos:width = "60vp"
        ohos:height = "50vp"
        ohos:left_of = " $ id:component_a"
        ohos:align_top = " $ id:component_a"
        ohos:margin = "1vp"
        ohos:background_element = " $ graphic:color_button_element"
        ohos:text_size = "20vp"
        ohos:text = "组件 B"/>
    < Button
        ohos:id = " $ + id:component_c"
        ohos:width = "60vp"
        ohos:height = "50vp"
        ohos:above = " $ id:component_a"
        ohos:align_left = " $ id:component_a"
        ohos:margin = "1vp"
        ohos:background_element = " $ graphic:color_button_element"
        ohos:text_size = "20vp"
        ohos:text = "组件 C"/>
    < Button
        ohos:id = " $ + id:component_d"
        ohos:width = "60vp"
        ohos:height = "50vp"
        ohos:right_of = " $ id:component_a"
        ohos:align_bottom = " $ id:component_a"
        ohos:margin = "1vp"
        ohos:background_element = " $ graphic:color_button_element"
        ohos:text_size = "20vp"
        ohos:text = "组件 D"/>
    < Button
        ohos:id = " $ + id:component_e"
        ohos:width = "60vp"
        ohos:height = "50vp"
        ohos:below = " $ id:component_a"
```

```
            ohos:align_right = " $ id:component_a"
            ohos:margin = "1vp"
            ohos:background_element = " $ graphic:color_button_element"
            ohos:text_size = "20vp"
            ohos:text = "组件 E"/>
        < Button
            ohos:id = " $ + id:component_f"
            ohos:width = "60vp"
            ohos:height = "50vp"
            ohos:below = " $ id:component_a"
            ohos:start_of = " $ id:component_a"
            ohos:margin = "1vp"
            ohos:background_element = " $ graphic:color_button_element"
            ohos:text_size = "20vp"
            ohos:text = "组件 F"/>
    </DependentLayout >
```

以上布局代码的运行效果如图 5-9 所示。

当依赖布局作为组件容器时，其包含的组件位置也可以相对于依赖布局进行设置。容器 A 是一个依赖布局，里面包含了组件 B～组件 F，它们可以相对于容器 A 放置到指定的位置，如图 5-10 所示。

图 5-9　依赖布局相对组件的运行效果

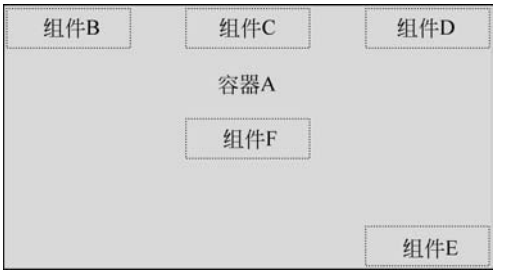

图 5-10　依赖布局的位置关系

如果希望达到该放置效果，则组件 B～组件 F 的属性设置如表 5-4 所示。

有时组件的位置可以通过不同的属性设置方式实现，如组件 F 位于父容器 A 的中心，可以直接通过 center_in_parent 属性设置，也可以通过 horizontal_center（水平居中）和 vertical_center（垂直居中）两个属性共同设置，代码如下：

表 5-4　组件 B～F 相对于容器 A 的位置属性设置

组件	属性设置	说　明
组件 B	ohos:align_parent_left="true" ohos:align_parent_top="true"	左边缘和 A 对齐 上边缘和 A 对齐
组件 C	ohos:align_parent_top="true" ohos:horizontal_center="true"	上边缘和 A 对齐 水平方向在父组件中心
组件 D	ohos:align_parent_right="true" ohos:align_parent_top="true"	右边缘和 A 对齐 上边缘和 A 对齐
组件 E	ohos:align_parent_right="true" ohos:align_parent_bottom="true"	右边缘和 A 对齐 下边缘和 A 对齐
组件 F	ohos:center_in_parent="true"	放到父组件 A 中心
	ohos:horizontal_center="true" ohos:vertical_center="true"	放到父组件 A 水平中心 放到父组件 A 垂直中心

```
//ch05\DependentLayout2 项目中的 ability_main.xml
<?xml version = "1.0" encoding = "utf - 8"?>
< DependentLayout
    xmlns:ohos = "http://schemas.huawei.com/res/ohos"
    ohos:width = "match_parent"
    ohos:height = "match_parent">
    < DependentLayout
        ohos:id = " $ + id:component_a"
        ohos:width = "300vp"
        ohos:height = "300vp"
        ohos:margin = "1vp"
        ohos:horizontal_center = "true"
        ohos:vertical_center = "true"
        ohos:background_element = " # FFBBBBBB"
    >
    < Button
            ohos:id = " $ + id:component_b"
            ohos:width = "60vp"
            ohos:height = "50vp"
            ohos:align_parent_left = "true"
            ohos:align_parent_top = "true"
            ohos:margin = "1vp"
            ohos:background_element = " $ graphic:color_button_element"
            ohos:text_size = "20vp"
            ohos:text = "组件 B"/>
        < Button
            ohos:id = " $ + id:component_c"
            ohos:width = "60vp"
            ohos:height = "50vp"
            ohos:align_parent_top = "true"
```

```
            ohos:horizontal_center = "true"
            ohos:margin = "1vp"
            ohos:background_element = " $ graphic:color_button_element"
            ohos:text_size = "20vp"
            ohos:text = "组件 C"/>
        < Button
            ohos:id = " $ + id:component_d"
            ohos:width = "60vp"
            ohos:height = "50vp"
            ohos:align_parent_right = "true"
            ohos:align_parent_top = "true"
            ohos:margin = "1vp"
            ohos:background_element = " $ graphic:color_button_element"
            ohos:text_size = "20vp"
            ohos:text = "组件 D"/>
        < Button
            ohos:id = " $ + id:component_e"
            ohos:width = "60vp"
            ohos:height = "50vp"
            ohos:align_parent_right = "true"
            ohos:align_parent_bottom = "true"
            ohos:margin = "1vp"
            ohos:background_element = " $ graphic:color_button_element"
            ohos:text_size = "20vp"
            ohos:text = "组件 E"/>
        < Button
            ohos:id = " $ + id:component_f"
            ohos:width = "60vp"
            ohos:height = "50vp"
            ohos:center_in_parent = "true"
            ohos:margin = "1vp"
            ohos:background_element = " $ graphic:color_button_element"
            ohos:text_size = "20vp"
            ohos:text = "组件 F"/>
    </DependentLayout >
</DependentLayout >
```

这里，整体是一个依赖布局，在这个依赖布局里又包含了一个依赖布局，即容器 A，其位置在父布局的中心，宽度和高度均为 300vp，id 为 container_a，其内部又包含了组件 B～F，运行效果如图 5-11 所示。

下面给出一个使用依赖布局的综合例子，该示例的运行效果如图 5-12 所示。

图 5-12 所示的界面效果与一些支付类移动应用界面类似，可以供开发者借鉴参考，界面中的一些图标及图片资源需要开发者自行准备，界面布局的代码如下：

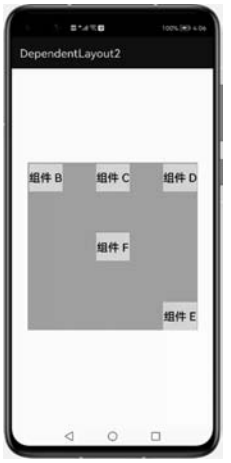

图 5-11　依赖布局相对布局的运行效果

图 5-12　依赖布局综合示例


7min

```
//ch05\DependentLayoutDemo 项目中的 ability_main.xml
<?xml version = "1.0" encoding = "utf - 8"?>
< DependentLayout
    xmlns:ohos = "http://schemas.huawei.com/res/ohos"
    ohos:width = "match_parent"
    ohos:height = "match_parent"
    ohos:padding = "5vp"
    ohos:background_element = "#25878787"
>
    < DirectionalLayout
        ohos:id = "$ + id:dirlayout1"
        ohos:width = "match_parent"
        ohos:height = "match_content"
        ohos:orientation = "horizontal"
        ohos:align_parent_top = "true"
        ohos:background_element = "#FF00FF00"
    >
        < Button
            ohos:width = "0"
            ohos:weight = "1"
            ohos:height = "match_content"
            ohos:padding = "30vp"
            ohos:element_top = "$ media:icon1"
            ohos:text = "收款"
            ohos:text_size = "20vp" />
        < Button
            ohos:width = "0"
            ohos:weight = "1"
            ohos:height = "match_content"
            ohos:padding = "30vp"
            ohos:text = "钱包"
```

```
            ohos:element_top = " $ media:icon2"
            ohos:text_size = "20vp" />
    </DirectionalLayout >
    < Text
        ohos:id = " $ + id:category01"
        ohos:height = "match_content"
        ohos:width = "match_parent"
        ohos:top_margin = "5vp"
        ohos:below = " $ id:dirlayout1"
        ohos:align_left = " $ id:dirlayout1"
        ohos:background_element = " # FFFFFF"
        ohos:text_size = "18vp"
        ohos:text = "理财类"
        />
    < Button
        ohos:id = " $ + id:btn01"
        ohos:width = "80vp"
        ohos:height = "match_content"
        ohos:below = " $ id:category01"
        ohos:align_left = " $ id:category01"
        ohos:padding = "10vp"
        ohos:text = "信用卡"
        ohos:element_top = " $ media:icon3"
        ohos:background_element = " # FFFFFF"
        ohos:text_size = "18vp"   />
    < Button
        ohos:id = " $ + id:btn02"
        ohos:width = "80vp"
        ohos:height = "match_content"
        ohos:right_of = " $ id:btn01"
        ohos:align_top = " $ id:btn01"
        ohos:padding = "10vp"
        ohos:text = "理财通"
        ohos:element_top = " $ media:icon4"
        ohos:background_element = " # FFFFFF"
        ohos:text_size = "18vp"   />

    < Text
        ohos:id = " $ + id:category02"
        ohos:height = "match_content"
        ohos:width = "match_parent"
        ohos:top_margin = "5vp"
        ohos:below = " $ id:btn01"
        ohos:align_left = " $ id:dirlayout1"
        ohos:background_element = " # FFFFFF"
        ohos:text_size = "18vp"
        ohos:text = "生活类"
        />
    < Button
```

```
            ohos:id = " $ + id:btn11"
            ohos:width = "80vp"
            ohos:height = "match_content"
            ohos:below = " $ id:category02"
            ohos:align_left = " $ id:category02"
            ohos:padding = "10vp"
            ohos:text = "水电费"
            ohos:element_top = " $ media:icon5"
            ohos:background_element = " # FFFFFF"
            ohos:text_size = "18vp"   />
    </DependentLayout >
```

这里,最外层是一个依赖布局,其内部的上方包含了一个方向布局,方向布局内又包含了收款和钱包功能。依赖布局的下方有多个文本和按钮,通过依赖的位置关系进行了布局。类似本例效果的界面在很多支付类应用中会出现,只不过会有一些差异,开发者可以根据实际需求,合理利用布局构建应用界面。

8min

5.3.3 StackLayout

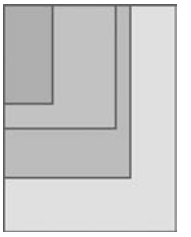

图 5-13 栈布局

StackLayout,即栈布局,栈布局直接在屏幕上开辟出一块空白的区域,位于栈布局中的组件以层叠的方式依次放置,栈布局中的元素类似一个栈结构,故称为栈布局。先放置的元素位于栈的下层,后放置的元素位于栈的上层,上一层的元素会遮挡下一层的元素视图,如图 5-13 所示。在栈布局中,所有的元素默认放置在栈布局的左上角。

栈布局好像没有布局,因为放置在其中的组件只是一层层堆放而已,因此栈布局自身没有定义更多的属性,一般仅需要设置栈布局的宽度、高度和背景。

栈布局中的元素默认被放置到左上角,也可以通过设置组件的布局对齐方式(layout_alignment)属性控制组件在栈布局中放置的位置,该属性的取值及说明如表 5-5 所示。

表 5-5 栈布局中组件对齐方式

组件布局方式取值	取 值 说 明
ohos:layout_alignment="left"	左对齐,组件左侧和所在布局左侧对齐
ohos:layout_alignment="top"	顶部对齐,组件上侧和所在布局上侧对齐
ohos:layout_alignment="right"	右对齐,组件右侧和所在布局右侧对齐
ohos:layout_alignment="bottom"	底部对齐,组件底边和所在布局底边对齐
ohos:layout_alignment="horizontal_center"	水平居中对齐,组件在水平方向上位于所在布局中心
ohos:layout_alignment="vertical_center"	垂直居中对齐,组件在垂直方向上位于所在布局中心
ohos:layout_alignment="center"	居中对齐,组件在水平和垂直方向上都位于所在布局中心

布局对齐方式 layout_alignment 属性的值可以使用“|”进行组合，如下代码表示顶部对齐且右对齐，即组件的右上角和所在的布局的右上角对齐，组件会被放置到所在布局的右下角，代码如下：

```
ohos:layout_alignment = "right|buttom"
```

下面是一个栈布局的例子，运行效果如图 5-14 所示。在屏幕中有 3 个组件，分别是 Layer1、Layer2、Layer3，Layer1 被放置于最底层，Layer2 被放置于第 2 层，Layer3 被放置于 Layer2 之上，3 个组件以中心对齐的方式被放置于同一个栈布局中。

图 5-14　栈布局示例的运行效果

如图 5-14 所示，运行效果对应的布局代码如下：

```
//ch05\StackLayout1 项目中的 ability_main.xml
<?xml version = "1.0" encoding = "utf - 8"?>
< StackLayout
    xmlns:ohos = "http://schemas.huawei.com/res/ohos"
    ohos:id = " $ + id:stack_layout"
    ohos:height = "match_parent"
    ohos:width = "match_parent">
    < Text
        ohos:id = " $ + id:text_blue"
        ohos:text_alignment = "bottom|horizontal_center"
        ohos:text_size = "24fp"
        ohos:text = "Layer 1"
        ohos:height = "400vp"
        ohos:width = "400vp"
        ohos:layout_alignment = "center"
        ohos:background_element = " # 3F56EA" />
```

```
    <Text
        ohos:id = " $ + id:text_light_purple"
        ohos:text_alignment = "bottom|horizontal_center"
        ohos:text_size = "24fp"
        ohos:text = "Layer 2"
        ohos:height = "300vp"
        ohos:width = "300vp"
        ohos:layout_alignment = "center"
        ohos:background_element = " # 00AAEE" />
    <Text
        ohos:id = " $ + id:text_orange"
        ohos:text_alignment = "bottom|horizontal_center"
        ohos:text_size = "24fp"
        ohos:text = "Layer 3"
        ohos:height = "200vp"
        ohos:width = "200vp"
        ohos:layout_alignment = "center"
        ohos:background_element = " # 00BFC9" />
</StackLayout >
```

栈布局中组件首次放置时有一定的层次,但是,栈布局中组件的显示层次是可以动态调整的,可以通过代码加以修改控制。栈布局提供了改变其子组件显示层次的方法,将栈布局中的一个 Text 组件移动到栈布局的最上层,代码如下:

```
//ch05\StackLayout1 项目中的 MainAbilitySlice.java
public void onStart(Intent intent) {
super.onStart(intent);
    super.setUIContent(ResourceTable.Layout_ability_main);
    ComponentContainer stackLayout = (ComponentContainer)
            findComponentById(ResourceTable.Id_stack_layout);
    Text textFirst = (Text)findComponentById(ResourceTable.Id_text_blue);
    //设置单击监听
    textFirst.setClickedListener(new Component.ClickedListener() {
        @Override
        public void onClick(Component component) {
            //将组件移动到最前面,即最上层
            stackLayout.moveChildToFront(component);
        }
    });
}
```

5.3.4 PositionLayout

PositionLayout,即坐标布局,也可称为位置布局或绝对布局。位于坐标布局中的子组

件通过指定准确的(x,y)坐标值确定在布局中显示的位置。坐标布局可以理解成一个坐标系,其左上角为(0,0),向右为 x 轴方向,向下为 y 轴方向,在坐标布局中,每个组件都可以设定一个坐标位置。

在坐标布局中,设置组件位置坐标的属性为 ohos:translation_x 和 ohos:translation_y,前者表示 x 坐标,后者表示 y 坐标。使用坐标布局进行界面布局的示例,代码如下:

```
//ch05\PositionLayout 项目中的 ability_main.xml
<?xml version = "1.0" encoding = "utf - 8"?>
< PositionLayout
    xmlns:ohos = "http://schemas.huawei.com/res/ohos"
    ohos:id = " $ + id:position"
    ohos:height = "match_parent"
    ohos:width = "300vp"
    ohos:background_element = " # 3387CEFA"
    >
    < Text
        ohos:id = " $ + id:position_text_1"
        ohos:height = "50vp"
        ohos:width = "200vp"
        ohos:background_element = " # 9987CEFA"
        ohos:translation_x = "100"
        ohos:translation_y = "20"
        ohos:text = "Title"
        ohos:text_alignment = "center"
        ohos:text_size = "20fp"/>
    < Text
        ohos:id = " $ + id:position_text_2"
        ohos:height = "200vp"
        ohos:width = "200vp"
        ohos:background_element = " # 9987CEFA"
        ohos:translation_x = "20"
        ohos:translation_y = "180"
        ohos:text = "Content2"
        ohos:text_alignment = "center"
        ohos:text_size = "20fp"/>
    < Text
        ohos:id = " $ + id:position_text_3"
        ohos:height = "200vp"
        ohos:width = "300vp"
        ohos:background_element = " # 9987CEFA"
        ohos:translation_x = "180"
        ohos:translation_y = "400"
        ohos:text = "Content3"
        ohos:text_alignment = "center"
        ohos:text_size = "20fp"/>
</PositionLayout >
```

以上代码的运行效果如图 5-15 所示。

组件坐标位置指的是组件左上角的位置。组件大小超出布局的部分将无法显示，如上例中 Content3 右侧超出了坐标布局的宽度，被自动裁剪了。当组件出现重叠时，处理方法和栈布局相同，即后放置的组件会被放置于先放置的组件的上方，如上例中 Content3 遮挡了 Content2 的一部分。

坐标布局中的组件位置也可以动态变化，组件提供了 3 个设置方法，帮助开发者在程序中控制组件在坐标布局的位置。setTranslation_x(float x)用于设置 x 坐标，setTranslation(float y)用于设置 y 坐标，setTranslation(float x, float y)可以同时设置 x 和 y 坐标。

坐标布局主要在同一个平面内通过坐标位置确定组件的位置关系，栈布局更适用于布置立体的层次关系。二者结合可以很好地解决应用中面与层之间的关系。下面以坦克大战游戏为例，综合应用一下栈布局和坐标布局。

在坦克大战游戏中有若干坦克、草地、奖品等，它们需要位于不同的层次，坦克位于底层，草地位于第 2 层，奖品位于最上层，因此可采用栈布局，这样不管何时出现坦克，坦克都能在草地下行驶，如图 5-16 所示。

图 5-15　坐标布局示例的运行效果

图 5-16　坦克游戏布局的应用示例

在同一层中，元素位置是可以变化的，如坦克是可以移动的，因此可以采用坐标布局，进而可以通过程序代码控制游戏，代码如下：

```
//ch05\TankDemo 项目中的 ability_main.xml
<?xml version = "1.0" encoding = "utf - 8"?>
< StackLayout
    xmlns:ohos = "http://schemas.huawei.com/res/ohos"
    ohos:height = "match_parent"
    ohos:width = "match_parent"
    ohos:background_element = "＃000000"
```

```
>
    <PositionLayout
        ohos:id = " $ + id:tanks"
        ohos:height = "match_parent"
        ohos:width = "match_parent"
    >
        <Image
            ohos:id = " $ + id:tank1"
            ohos:height = "40vp"
            ohos:width = "40vp"
            ohos:image_src = " $ media:pic"
            ohos:clip_alignment = "top|center"
            />

        <Image
                ohos:id = " $ + id:tank2"
                ohos:height = "40vp"
                ohos:width = "40vp"
                ohos:image_src = " $ media:pic"
                ohos:clip_alignment = "left|top"
                ohos:translation_x = "200"
                ohos:translation_y = "300"
                />
    </PositionLayout>
    <PositionLayout
        ohos:id = " $ + id:grassland"
        ohos:height = "match_parent"
        ohos:width = "match_parent"
    >
        <Image
                ohos:id = " $ + id:grassland1"
                ohos:height = "40vp"
                ohos:width = "40vp"
                ohos:image_src = " $ media:pic"
                ohos:clip_alignment = "right|bottom"
                ohos:translation_x = "260"
                ohos:translation_y = "220"
                />
        <Image
                ohos:id = " $ + id:grassland2"
                ohos:height = "40vp"
                ohos:width = "40vp"
                ohos:image_src = " $ media:pic"
                ohos:clip_alignment = "right|bottom"
                ohos:translation_x = "380"
                ohos:translation_y = "220"
```

```
            />
        </PositionLayout>
        <PositionLayout
            ohos:id = " $ + id:reward"
            ohos:height = "match_parent"
            ohos:width = "match_parent"
            >
            <Image
                    ohos:id = " $ + id:star"
                    ohos:height = "40vp"
                    ohos:width = "40vp"
                    ohos:image_src = " $ media:pic"
                    ohos:clip_alignment = "right|top"
                    ohos:translation_x = "320"
                    ohos:translation_y = "160"
                    />
        </PositionLayout>
    </StackLayout>
```

在以上代码中，多次引用了图片资源 $ media:pic，并进行了适当的裁剪。资源图片文件 pic.png 如图 5-17 所示，该图片是一个 120×80 像素的图片，每个基本元素是 40×40 像素，共 6 个基本元素，分别代表 4 个方向的坦克、五角星、草地，当按照适当的方式裁剪后刚好可以得到其中的一个基本元素。

5.3.5 TableLayout

TableLayout，即表格布局，表格布局使用表格的方式来组织其中的组件，一个表格可以分为若干行和若干列，表格布局中的组件依次被放置到各个单元格中。一个 3 行 2 列的表格布局示意图，如图 5-18 所示。

7min

图 5-17 资源图片文件 pic.png

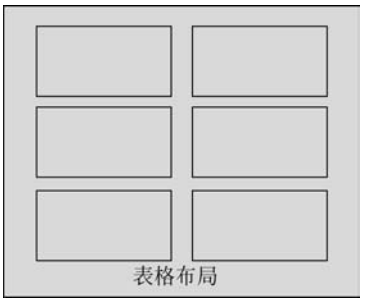

图 5-18 表格布局示意图

表格布局除了拥有布局统一具有的属性外，还有几个特有的属性，几个常用的属性及说明如表 5-6 所示。

表 5-6 表格布局的主要属性

属性名称	说明
ohos:row_count	值取整数,表示表格的行数
ohos:column_count	值取整数,表示表格的列数
ohos:orientation	表格中组件排列方向。当取值为 horizontal 时表示水平方向,即表格布局的组件按照行优先的方式依次放入表格布局中,水平方向是默认方向;当取值为 vertical 时表示垂直方向,即表格布局的组件按照列优先的方式依次放入表格布局中

表格布局为放置在其中的组件布置了一个隐形若干行和若干列的单元格,表格中的元素将依次被放置到各个单元格中。

对于一个 3 行 2 列的表格,如果按照行优先放置 4 个组件,则它们放置的顺序如图 5-19(a)所示;如果按照列优先放置 4 个组件,则放置的顺序如图 5-19(b)所示。

下面是一个使用表格布局显示图片的示例,效果如图 5-20 所示。界面中包含 6 张图片,采用 3 行 2 列的表格布局。

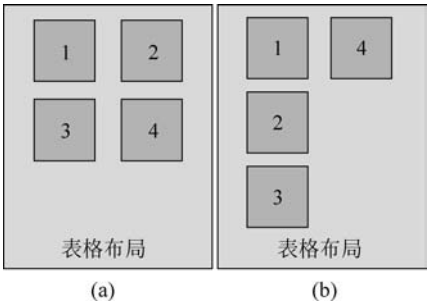

图 5-19 按照行或列排列组件

图 5-20 使用表格布局

对应的布局文件的代码如下:

```
//ch05\TableLayoutDemo 项目中的 ability_main.xml
<?xml version = "1.0" encoding = "utf - 8"?>
< TableLayout
    xmlns:ohos = "http://schemas.huawei.com/res/ohos"
    ohos:height = "match_parent"
    ohos:width = "match_parent"
    ohos:background_element = "#EEEEEE"
    ohos:layout_alignment = "horizontal_center"
    ohos:row_count = "3"
```

```
        ohos:column_count = "2"
        ohos:padding = "8vp">
        < Image
            ohos:height = "100vp"
            ohos:width = "130vp"
            ohos:image_src = " $ media:pic1"
            ohos:background_element = " # 2200FF00"
            ohos:scale_mode = "inside"
            ohos:margin = "8vp"
            ohos:text = "1"
            ohos:text_alignment = "center"
            ohos:text_size = "20fp"/>

        < Image
            ohos:height = "100vp"
            ohos:width = "130vp"
            ohos:image_src = " $ media:pic2"
            ohos:background_element = " # 2200FF00"
            ohos:scale_mode = "inside"
            ohos:margin = "8vp"
            ohos:text = "2"
            ohos:text_alignment = "center"
            ohos:text_size = "20fp"/>

        < Image
            ohos:height = "130vp"
            ohos:width = "130vp"
            ohos:image_src = " $ media:pic3"
            ohos:background_element = " # 2200FF00"
            ohos:scale_mode = "inside"
            ohos:margin = "8vp"
            ohos:text = "2"
            ohos:text_alignment = "center"
            ohos:text_size = "20fp"/>

        < Image
            ohos:height = "100vp"
            ohos:width = "130vp"
            ohos:image_src = " $ media:pic4"
            ohos:background_element = " # 2200FF00"
            ohos:scale_mode = "inside"
            ohos:margin = "8vp"
            ohos:text = "2"
            ohos:text_alignment = "center"
            ohos:text_size = "20fp"/>
```

```
    <Image
        ohos:height = "100vp"
        ohos:width = "130vp"
        ohos:image_src = " $ media:pic5"
        ohos:background_element = " # 2200FF00"
        ohos:scale_mode = "inside"
        ohos:margin = "8vp"
        ohos:text = "2"
        ohos:text_alignment = "center"
        ohos:text_size = "20fp"/>

    <Image
        ohos:height = "130vp"
        ohos:width = "130vp"
        ohos:image_src = " $ media:pic6"
        ohos:background_element = " # 2200FF00"
        ohos:scale_mode = "inside"
        ohos:margin = "8vp"
        ohos:text = "2"
        ohos:text_alignment = "center"
        ohos:text_size = "20fp"/>

</TableLayout >
```

表格布局的单元格是隐形的，需放置的组件依次被放置到表格的每个格中，当表格中的元素大小不一致时，每一行的高度会取该行最高的元素的高度，每一列的宽度会取该列最宽的元素的宽度。如上例中，尽管每一行的顶部是对齐的，但是由于图片组件的高度不同，出现了下边界不齐的情况，因此，开发者在使用表格布局时，可以将其中的组件设置为相同的高度和宽度，以便表格中的元素显示及排列得更加整齐。

5.4 自定义布局

6min

系统提供了一些常用的布局，多数情况下可以满足开发者的需要，但是，有时现有的布局不能很好地满足特殊需求，这时可以考虑自定义布局。

在 Java UI 框架中，布局其实就是一个具体的组件容器类，Java UI 中组件类的继承关系如图 5-21 所示，箭头指向的方向为父类，前面介绍的常见布局 DirectionalLayout、DependentLayout、StackLayout、PositionLayout、TableLayout 都继承自 ComponentContainer 类。

在整个组件体系中，Component 类是所有类的父类，由它派生出来很多组件容器和组件类。一般情况下，开发者只需选择合适的组件或组件容器便可满足开发应用的需求，但是，如果开发者需要，则完全可以通过继承组件体系中的类，进而扩展自定义布局或组件，以满足用户的特定需求。

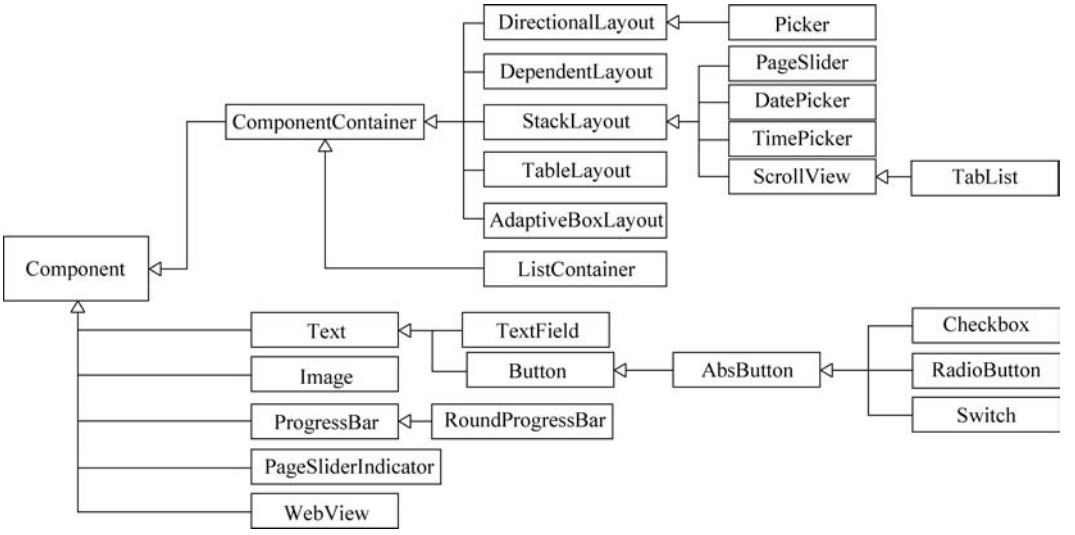

图 5-21　组件类继承关系图

自定义布局可以继承 ComponentContainer 类或其子类，自定义布局类一般需要重新定义布局里的监听，需要测量控制布局及其包含的组件的大小，还需要控制组件的排列方式和位置等。下面是自定义布局的基本步骤：

(1) 创建自定义布局的类，继承 ComponentContainer 类或其子类。

(2) 实现 ComponentContainer. EstimateSizeListener 接口，在 onEstimateSize()方法中进行测量计算。

(3) 测量计算布局中每个子组件的大小和位置数据，并保存。

(4) 实现 ComponentContainer. ArrangeListener 接口，并在 onArrange()方法中根据测量计算数据布置排列自定义布局中的子组件。

(5) 使用自定义布局创建布局对象，并在布局中添加若干子组件。

实际上，现有的常用布局也是通过继承 ComponentContainer 类实现的，它们只不过是事前定义好的自定义布局而已，开发者也可以学习其开源的源代码实现。总之，所有在已有布局中通过配置 XML 布局文件可以实现的，在自定义布局时都需要通过代码进行实现。

一般情况下，开发者不需要自定义布局或组件，可以通过现有的布局或组件嵌套、组合等多种方式完成复杂的界面设计效果。

小结

布局是组件容器，它可以更好地布置应用中的组件及组件容器，从而为应用创建丰富且灵活的符合用户需求的图形界面。Java UI 框架主要提供了两种创建布局的方式，一种是通过 XML 定义布局，另一种是通过系统提供的布局类创建布局。常用的布局包括方向布

局、依赖布局、栈布局、表格布局、坐标布局等，每种常用布局都有自身的特点。常用布局都继承自 ComponentContainer 类，开发者也可通过继承该类或其子类自定义布局。Java UI 框架以 Component 类为根，建立了丰富的界面布局和组件体系。

习题

1. 判断题

(1) 布局一般继承自 Container 类。(　　)

(2) 组件容器 ComponentContainer 类继承自 Component 类，因此布局其实是一种特殊的组件。(　　)

(3) 创建和使用布局有两种基本方式，一种方式是通过 XML 布局文件加载布局，另一种方式是通过系统提供的布局类创建布局对象。(　　)

(4) setUIContent()方法的作用是把布局设置到界面中。(　　)

(5) 系统已经提供给了丰富的布局，可以通过组合、嵌套等方式实现所有的布局方式，因此开发者只能使用系统提供的布局，不可以自定义布局。(　　)

2. 选择题

(1) 下面不是鸿蒙应用开发常用的布局的是(　　)。

A. StackLayout　　B. TableLayout

C. PositionLayout　　D. GridLayout

(2) 关于布局说法不正确的是(　　)。

A. 布局也是组件　　B. 布局可以嵌套

C. 布局也有大小　　D. 布局属性一旦确定不能改变

(3) 关于 DirectionLayout 说法不正确的是(　　)。

A. 该布局中元素可以按水平方向线性排列

B. 该布局中元素可以按垂直方向线性排列

C. 该布局中元素默认为居中对齐

D. 该布局中可以嵌套 DirectionLayout

(4) 下面的属性设置，可以把当前组件放到其父容器且水平居中的是(　　)。

A. ohos:horizontal_center="true"　　B. ohos:align_parent_top="true"

C. ohos:align_parent="center"　　D. ohos:align_parent_in="center"

(5) 位置布局中组件通过指定(x,y)坐标设置位置，这里的坐标系的原点在(　　)。

A. 设备屏幕的左上角　　B. 组件所在位置布局的左上角

C. 屏幕的中心　　D. 组件所在位置布局的中心

3. 填空题

(1) DirectionalLayout 的排列方向属性 ohos:orientation 的值为_____时，表示其中的组件按垂直方向排列。

（2）_____称为依赖布局，在依赖布局里，每个组件可以指定相对于其他同级元素的位置，也可以指定相对于父组件的位置。

（3）_____布局中放置的组件是一层层的。

（4）TableLayout，即表格布局，在表格布局中，属性 ohos：_____可以设置表格的列数。

（5）_____布局是以组件坐标值确定在布局中位置的。

4. 上机题

（1）仿照微信，实现其聊天界面的布局。

（2）请实现一个九宫格界面布局，每个格中放置一张图片。