第 5 章

UI 组 件

本章主要讲解常见 UI 组件的使用方法及其特性。第 4 章中介绍了 HarmonyOS 的常用 UI 布局，而一个界面除了 UI 布局，组件也是非常重要的组成部分。组件是构建页面的核心，每个组件通过对数据和方法的简单封装，实现独立的可视、可交互功能单元。组件之间相互独立，随取随用，也可以在需求相同的地方重复使用。HarmonyOS 的 UI 常见组件可以分为三大类：显示组件、交互组件和高级组件。组件的具体使用场景，需要根据业务需求来选择使用。

通过阅读本章，读者可以掌握：

- 如何创建使用各类组件。
- 了解各类组件所支持的属性。
- 如何设置组件样式。

5.1 展 示 组 件

5.1.1 文本组件

文本组件(Text)是最常用的组件之一。Text 是用来显示字符串的组件，在界面上显示为一块文本区域。Text 作为一个基本组件有很多扩展，常见的有按钮组件 Button，文本编辑组件 TextField。Text 是 Component 类的子类之一，所以它能够使用 Component 类的所有公开的属性和方法，Text 类自身也提供了一些特殊的属性、方法、内部类和接口。

1. 创建 Text

(1) 布局文件中创建 Text。

在 layout 文件下创建 text.xml，在 XML 文件中声明布局和组件。例如，创建一个高为 150vp，长为 300vp 的 text 组件，组件具体属性代码如下：

```
<Text
    ohos:id="$+id:text" //当前组件唯一标识,在单个布局文件中不允许 id 标识重复
    ohos:width="300vp"
    ohos:height="150vp"
    ohos:text="Text 组件"   //显示文本
    ohos:background_element="$graphic:background_text"  //设置 Text 背景
    ohos:text_size="66fp"
    ohos:text_color="white"
    ohos:italic="true"
```

```
    ohos:text_weight="700"
    ohos:text_font="serif"
/>
```

组件常用的背景可以采用 XML 格式放置在 graphic 目录下。在 graphic 目录下创建 background_text.xml，在 XML 文件中定义文本的背景，代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:ohos="http://schemas.huawei.com/res/ohos"
ohos:shape="rectangle">
    <corners ohos:radius="20"/>
    <solid ohos:color="#FF1992C3"/>
</shape>
```

最后，在 MainAbilitySlice.java 中，通过 setUIContent()加载该 XML 布局。代码如下：

```
public class MainAbilitySlice extends AbilitySlice {
    @Override
    public void onStart (Intent intent){
        Super.onStart(intent);
        Super.setUIContent(ResourceTable.Layout_text);
    }
}
```

运行效果如图 5-1 所示。

图 5-1　XML 创建 Text

(2) Java 代码中创建 Text。

打开 MainAbilitySlice.java 文件，找到 onStart()方法，创建 Text 组件，代码如下：

```
Text text = (Text) findComponentById
    (ResourceTable.Id_text);          //获取布局中的组件
text.setText("text 组件");            //设置文本
text.setTextSize(66);                 //设置文字大小
text.setTextColor(Color.WHITE);       //设置文字颜色
text.setWidth(300);
text.setHeight(150);
```

2. 设置 Text 组件样式

（1）设置字体大小和颜色。

通常设置文本大小 text_size 和文本颜色 text_color 两个属性。text_size 为 float 类型，可以是浮点数值，其默认单位为 px；还可以是带 px/vp/fp 单位的浮点数值；还可以引用 float 资源。text_color 为 color 类型，可以直接设置色值，也可以引用 color 资源。代码如下：

```
ohos:text_size="66fp"
ohos:text_color="#FF0000"
```

（2）设置字体风格和字重。

字体为 text_font 属性，可以设置的值包括 sans-serif、sans-serif-medium、HwChinese-medium、sans-serif-condensed、sans-serif-condensed-medium、monospace。italic 表示文本是否斜体字体，为 boolean 类型，可以直接设置 true/false，也可以引用 boolean 资源。text_weight 则表示字重。代码如下：

```
ohos:italic="true"
ohos:text_weight="700"
ohos:text_font="sans-serif"
```

（3）设置文本对齐方式，换行和最大显示行数。

text_alignment 表示文本对齐方式，包含 left 文本靠左对齐，top 文本靠顶部对齐，right 文本靠右对齐，bottom 文本靠底部对齐，horizontal_center 文本水平居中对齐，vertical_center 文本垂直居中对齐，center 文本居中对齐，start 文本靠起始端对齐以及 end 文本靠结尾端对齐。multiple_lines 为多行模式设置，boolean 类型。max_text_lines 表示文本最大行数，为 integer 类型。代码如下：

```
ohos:text="Text 是一个文本组件"
ohos:text_alignment="horizontal_center "
ohos:multiple_lines="true"
ohos:max_text_lines="2"
```

运行效果如图 5-2 所示。

（4）自动调节字体大小。

Text 对象可以实现根据文本长度来自动调整文本的字体大小和换行。通过设置 auto_font_size 来控制文本是否自动调整文本字体大小，该属性为 boolean 类型。效果如图 5-3

所示，代码如下：

图 5-2　文本效果

图 5-3　自动调节字体大小

```
ohos:multiple_lines="true"      //自动换行
ohos:max_text_lines="1"         //最大显示行数
ohos:auto_font_size="true"      //自动调节字体大小
```

(5) 跑马灯效果。

在文本过长的情况下，可以通过设置跑马灯效果来实现文本滚动显示。但在这里要注意的是，首先要关闭文本换行并且设置最大显示行数为 1，一般默认即可。代码如下：

```
text.setTruncationMode(Text.TruncationMode.AUTO_SCROLLING);//自动滚动状态
text.setAutoScrollingCount(Text.AUTO_SCROLLING_FOREVER);
text.startAutoScrolling();                                  //启动跑马灯效果
```

5.1.2　图像组件

图像组件(Image)是用来在屏幕上显示图片的组件，各种图片的展示都需要该组件的使用。例如，在做网站首页时需要显示的背景图，动态加载显示用户头像以及朋友圈发布图片等场景，都可以使用 Image 图像组件进行显示。

1. 创建 Image

首先将所需图片资源 image.jpg 添加至 media 文件夹下，这里以“image.jpg”为例。在 image.xml 文件中声明组件，然后在 Java 文件中加载该 XML 布局。运行效果如图 5-4 所示，代码如下：

```
<Image
ohos:id="$+id:imageComponent"
ohos:height="match_parent"
ohos:width="match_parent"
ohos:image_src="$media:image"/>
```

或者用 Java 代码创建 Image 组件，在 MainAbilitySlice.java 文件中找到 onStart()方法，创建 Image 组件，代码如下：

```
Image image = new Image(getContext());                    //创建 Image
image.setPixelMap(ResourceTable.Media_image);    //设置要显示的图片
DirectionalLayout layout = new DirectionalLayout(getContext());//创建布局
layout.addComponent(image); //Image 组件添加到布局中
super.setUIContent(layout);
```

2. 设置 Image 样式

（1）设置透明度。

通过设置 alpha 透明度属性来修改图片的透明度，效果如图 5-5 所示。代码如下：

```
ohos:alpha="0.5"       //设置透明度为 0.5
```

（2）设置缩放系数。

通过设置 X 轴和 Y 轴缩放系数来控制图片的缩放，效果如图 5-6 所示。代码如下：

图 5-4　创建 Image 组件

图 5-5　透明度效果

图 5-6　坐标轴缩放

```
ohos:scale_x="0.5"
ohos:scale_y="0.5"
```

(3) 设置缩放方式。

当图片尺寸与 Image 尺寸不同时,可以根据不同的缩放方式对图片进行缩放。缩放方式 scale_mode 有下面几种:

- center 表示不缩放,按 Image 大小显示原图中间部分。
- zoom_center 表示原图按照比例缩放到与 Image 最窄边一致,并居中显示。
- zoom_start 表示原图按照比例缩放到与 Image 最窄边一致,并靠起始端显示。
- zoom_end 表示原图按照比例缩放到与 Image 最窄边一致,并靠结束端显示。
- stretch 表示将原图缩放到与 Image 大小一致。
- inside 表示将原图按比例缩放到与 Image 相同或更小的尺寸,并居中显示。
- clip_center 表示将原图按比例缩放到与 Image 相同或更大的尺寸,并居中显示。

图 5-7　缩放效果

例如,设置 Image 的宽为 300vp,高为 150vp,分别设置缩放方式为 zoom_start、zoom_center 和 zoom_end,为了突出效果,将 image 的背景色设为灰色,运行效果如图 5-7 所示。

代码如下:

```
<Image
    ohos:id="$+id:image"
    ohos:width="300vp"
    ohos:height="150vp"
    ohos:bottom_margin="20vp"
    ohos:image_src="$media:image"
    ohos:background_element="$graphic:background_image"
    ohos:scale_mode="zoom_start"/>
<Image
    ohos:id="$+id:image1"
    ohos:width="300vp"
    ohos:height="150vp"
    ohos:bottom_margin="20vp"
    ohos:image_src="$media:image"
    ohos:background_element="$graphic:background_image"
    ohos:scale_mode="zoom_center"/>
<Image
    ohos:id="$+id:image2"
    ohos:width="300vp"
    ohos:height="150vp"
    ohos:image_src="$media:image"
    ohos:background_element="$graphic:background_image"
    ohos:scale_mode="zoom_end"/>
```

（4）设置裁剪对齐方式。

当 Image 组件设定的尺寸小于图片实际尺寸时，可以对图片进行裁剪。通过设置 clip_alignment 样式来进行图片剪裁：left 表示左对齐裁剪；right 表示右对齐裁剪；top 表示顶部对齐裁剪；bottom 表示底部对齐裁剪；center 表示居中对齐裁剪。

例如，创建一个宽和高都为 200vp 的 Image 组件，设置左对齐 clip_alignment＝“left”，效果如图 5-8 所示。

图 5-8　裁剪效果

代码如下：

```
<Image
    ohos:id="$+id:image"
    ohos:width="200vp"
    ohos:height="200vp"
    ohos:layout_alignment="center"
    ohos:image_src="$media:image"
    ohos:clip_alignment="left"
/>
```

5.1.3　进度条组件

ProgressBar 用于显示内容或操作的进度。可以通过进度条查看一些功能操作的进度。使用场景：项目开发中通过设置数值改变进度条的样式，或者通过手动拖动改变进度条和进度值。

1. 创建 ProgressBar

在 layout 目录下的 XML 文件中创建一个 ProgressBar，同时在 Java 文件中加载该

XML 布局。运行效果如图 5-9 所示。代码如下：

```
<ProgressBar
    ohos:id="$+id:progressbar"
    ohos:orientation="horizontal"
    ohos:progress_width="40vp"
    ohos:height="120vp"
    ohos:width="match_parent"
    ohos:max="100"
    ohos:min="0"
    ohos:progress="60"
    ohos:progress_hint_text="60%"
    ohos:progress_hint_text_color="black"
/>
```

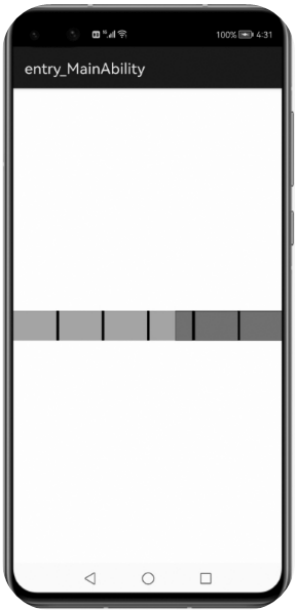

图 5-9　progressBar 效果

2. 设置 ProgressBar 样式

(1) 设置 ProgressBar 的方向。

属性 orientation 表示排列方向，包含 horizontal 水平显示和 vertical 垂直显示。设置 ProgressBar 方向为水平，代码如下：

```
ohos:orientation="horizontal"
```

(2) 设置 ProgressBar 的进度和提示文字。

设置当前进度 progress 为 integer 类型，可以直接设置整型数值，也可以引用 integer 资源。属性 progress_hint_text 表示进度提示文本，为 string 类型。progress_hint_text_color 表示进度提示文本颜色，为 color 类型。例如，设置进度为 60%，颜色为 black，代码如下：

```
ohos:progress="60"
ohos:progress_hint_text="60%"                    //设置进度条数值
ohos:progress_hint_text_color="black"            //设置进度条数值颜色
```

（3）设置最大值和最小值。

通过设置最大值 max 和最小值 min 来控制进度条，两个属性都为 integer 类型，可以直接设置整型数值，也可以引用 integer 资源。例如，设置最大值为 100，最小值为 0，在 XML 文件中设置：

```
ohos:max="100"
ohos:min="0"
```

或者在 Java 中设置：

```
progressBar.setMaxValue(100);
progressBar.setMinValue(0);
```

（4）设置 ProgressBar 的颜色和分割线。

属性 background_instruct_element 表示背景色，progress_element 表示进度条颜色。两者都为 element 类型，可直接配置色值，也可引用 color 资源或引用 media/graphic 下的图片资源。代码如下：

```
ohos:progress_element="#FF9900"
ohos:background_instruct_element="#FF0000"
```

属性 divider_lines_enabled 表示分割线，为 boolean 类型，可以直接设置 true/false，也可以引用 boolean 资源。divider_lines_number 表示分割线数量，为 integer 类型，可以直接设置整型数值，也可以引用 integer 资源。运行效果如图 5-10 所示，如在 XML 文件中配置：

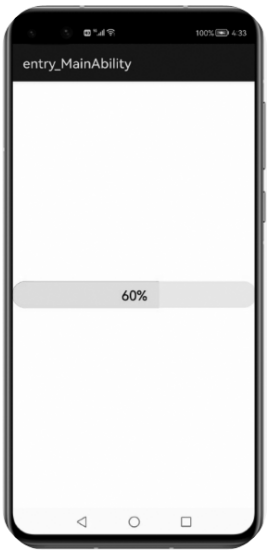

图 5-10　进度颜色和分割线效果

```
ohos:divider_lines_enabled="true"       //设置是否显示分割线
ohos:divider_lines_number="5"           //设置分割线个数
```

通过在 Java 文件中设置分割线颜色，代码如下：

```
progressBar.setDividerLineColor(Color.black);
```

5.1.4　圆形进度条

RoundProgressBar 继承自 ProgressBar，拥有 ProgressBar 的属性，在设置同样的属性时用法和 ProgressBar 一致，用于显示环形进度。

1. 创建 RoundProgressBar

在 layout 目录下的 XML 文件中创建一个 RoundProgressBar，同时在 Java 文件中加载该 XML 布局。运行效果如图 5-11 所示，代码如下：

```
<RoundProgressBar
    ohos:id="$+id:round_progress_bar"
    ohos:height="250vp"
    ohos:width="250vp"
    ohos:progress_width="15vp"
    ohos:progress="60"
    ohos:progress_color="blue"
    ohos:progress_hint_text="60%"
    ohos:progress_hint_text_color="#007DFF"
/>
```

2. 设置样式

组件 RoundProgressBar 的基础属性与组件 ProgressBar 保持一致，如设置进度条的大小、颜色、进度和提示文字等，都使用同样的属性进行控制，具体属性描述可以参考 5.1.3 节。这里主要讲解如何设置 RoundProgressBar 的开始和结束角度。

RoundProgressBar 的自有 XML 属性为 start_angle 和 max_angle。start_angle 表示圆形进度条的起始角度，max_angle 表示圆形进度条的最大角度。通过设置这两个属性值来控制进度条的开始和结束角度。例如，设置 start_angle 为 20，max_angle 为 340，运行效果如图 5-12 所示。代码如下：

```
ohos:start_angle="20"
ohos:max_angle="340"
```

5.1.5　时钟组件

时钟组件是 Text 的子类，所以可以使用 Text 的一些属性。

1. 创建 Clock

在 layout 目录下的 XML 文件中创建一个 Clock，同时在 Java 文件中加载该 XML 布