# 5.3 装饰容器DecoratedBox

DecoratedBox可以在其子组件绘制前(或后)绘制一些装饰（Decoration），如背景、边框、渐变等。DecoratedBox定义如下：

const DecoratedBox({  
 Decoration decoration,  
 DecorationPosition position = DecorationPosition.background,  
 Widget child  
})

* decoration：代表将要绘制的装饰，它的类型为Decoration。Decoration是一个抽象类，它定义了一个接口 createBoxPainter()，子类的主要职责是需要通过实现它来创建一个画笔，该画笔用于绘制装饰。
* position：此属性决定在哪里绘制Decoration，它接收DecorationPosition的枚举类型，该枚举类有两个值：
  + background：在子组件之后绘制，即背景装饰。
  + foreground：在子组件之上绘制，即前景。

#### BoxDecoration

我们通常会直接使用BoxDecoration类，它是一个Decoration的子类，实现了常用的装饰元素的绘制。

BoxDecoration({  
 Color color, //颜色  
 DecorationImage image,//图片  
 BoxBorder border, //边框  
 BorderRadiusGeometry borderRadius, //圆角  
 List<BoxShadow> boxShadow, //阴影,可以指定多个  
 Gradient gradient, //渐变  
 BlendMode backgroundBlendMode, //背景混合模式  
 BoxShape shape = BoxShape.rectangle, //形状  
})

各个属性名都是自解释的，详情读者可以查看API文档。下面我们实现一个带阴影的背景色渐变的按钮：

DecoratedBox(  
 decoration: BoxDecoration(  
 gradient: LinearGradient(colors:[Colors.red,Colors.orange[700]]), //背景渐变  
 borderRadius: BorderRadius.circular(3.0), //3像素圆角  
 boxShadow: [ //阴影  
 BoxShadow(  
 color:Colors.black54,  
 offset: Offset(2.0,2.0),  
 blurRadius: 4.0  
 )  
 ]  
 ),  
 child: Padding(padding: EdgeInsets.symmetric(horizontal: 80.0, vertical: 18.0),  
 child: Text("Login", style: TextStyle(color: Colors.white),),  
 )  
)

运行后效果如图5-9所示：

图5-9

怎么样，通过BoxDecoration我们实现了一个渐变按钮的外观，但此示例还不是一个标准的按钮，因为它还不能响应点击事件，我们将在后面“自定义组件”一章中实现一个完整功能的GradientButton。另外，上面的例子中使用了LinearGradient类，它用于定义线性渐变的类，Flutter中还提供了其它渐变配置类，如RadialGradient、SweepGradient，读者若有需要可以自行查看API文档。