第 3 章

CHAPTER 3

Flutter UI 布局排版

组件核心基础

在 Flutter 中，应用于页面布局排版方式的组件 Widget 有线性排列（Column、Row）、层叠排列（Stack）、流式排列（Flow、Wrap）等。

3.1 Column 与 Row 实现线性排列

线性布局指页面布局中的控件摆放方式是以线性的方式摆放的，线性排列分为纵向（垂直方向）和横向（水平方向），如图 3-1 所示。

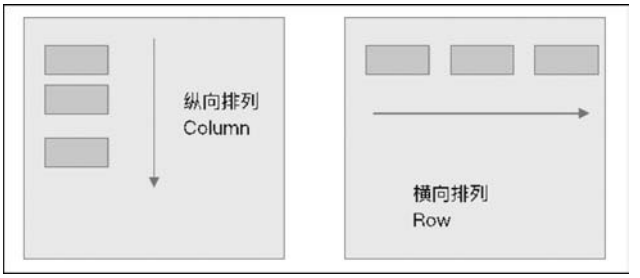

图 3-1 线性布局排列效果图

3.1.1 Column 用来实现竖直方向线性排列

Column 组件的主要功能是处理垂直方向的布局，Column 可以将子组件在竖直方向线性排列，如图 3-2 所示，代码如下：

```
//代码清单 3-1 Column 的基本使用
//代码路径 lib/code3/code301_Column.dart
class Exam220HomePage extends StatelessWidget {
  const Exam220HomePage({Key? key}) : super(key: key);
  @override
```

```
  Widget build(BuildContext context) {
    return Scaffold(
      //页面的头部
      appBar: AppBar(title: const Text("标题")),
      body: SizedBox(
        //宽度填充屏幕
        width: double.infinity,
        child: Column(
          children: const [
            Text("测试数据一"),
            Text("测试数据二"),
            Text("测试数据三"),
          ],
        ),
      ),
    );
  }
}
```

对于 Column 来讲，在默认情况下，将竖直方向称为主轴方向，将水平方向称为交叉轴方向，Column 默认在主轴方向填充，在交叉轴方向包裹，在上述代码中，通过组件 SizeBox 来设定填充宽度，Column 的宽度也会填充 SizeBox 的宽度。

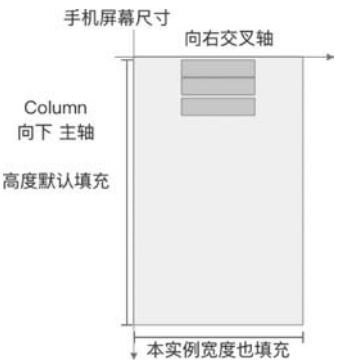

图 3-2　Column 排列效果图

将 Column 在主轴方向设置为包裹，以及将交叉轴对齐方式设置为左对齐，如图 3-3 所示，代码如下：

```
Column(
  //主轴方向包裹
  mainAxisSize: MainAxisSize.min,
  //主轴方向顶部对齐，默认方式
  mainAxisAlignment: MainAxisAlignment.center,
```

```
    //交叉轴方向左对齐,默认居中
    crossAxisAlignment: CrossAxisAlignment.start,
    children: const [
      Text("测试数据一"),
      Text("测试数据二"),
      Text("测试数据三"),
    ],
  )
```

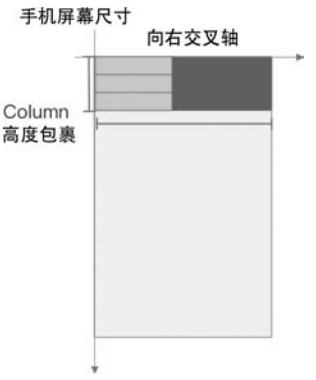

图 3-3　Column 左对齐效果图

3.1.2　Row 用来实现水平方向线性排列

Row 组件的主要功能是处理水平方向的布局,如图 3-4 所示,通过 Row 实现子组件的水平线性排列。

图 3-4　Row 基本使用

代码如下:

```
//代码清单 3-2 Row 的基本使用
//代码路径 lib/code3/code302_Row.dart
class Exam302HomePage extends StatelessWidget {
  const Exam302HomePage({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return Scaffold(
```

```
        //页面的头部
        appBar: AppBar(title: const Text("标题")),
        body: Row(
          //主轴方向包裹
          //mainAxisSize: MainAxisSize.min,
          //主轴方向左对齐,默认方式
          //mainAxisAlignment: MainAxisAlignment.center,
          //交叉轴方向顶部对齐,默认居中
          //crossAxisAlignment: CrossAxisAlignment.start,
          children: const [
            Text("测试数据一"),
            Text("测试数据二"),
            Text("测试数据三"),
          ],
        ),
      );
    }
  }
```

对于 Row,其主轴指的是水平方向,交叉轴指的是竖直方向,在默认情况下,Row 在主轴方向上填充,主轴方向通过属性 mainAxisAlignment 来决定子 Widget 的对齐方式,配置的是 MainAxisAlignment 枚举类型,它的取值如表 3-1 所示。

表 3-1 主轴 alignment 对齐方式

类 别	描 述
MainAxisAlignment. start	沿着主轴方向开始位置对齐
MainAxisAlignment. end	沿着主轴方向结束位置对齐
MainAxisAlignment. center	沿着主轴方向居中对齐
MainAxisAlignment. spaceBetween	沿着主轴方向平分剩余空间
MainAxisAlignment. spaceAround	把剩余空间平分成 n 份,n 是子 widget 的数量,然后把其中一份空间分成两份,放在第 1 个 child 的前面和最后一个 child 的后面
MainAxisAlignment. spaceEvenly	把剩余空间平分成 $n+1$ 份,然后平分所有的空间

交叉轴方向通过属性 crossAxisAlignment 来决定子 Widget 的对齐方式,配置的是 CrossAxisAlignment 枚举类型,它的取值如表 3-2 所示。

表 3-2 交叉轴 alignment 对齐方式

类 别	描 述
CrossAxisAlignment. start	交叉轴方向开始位置对齐
CrossAxisAlignment. end	交叉轴方向结束位置对齐
CrossAxisAlignment. center	交叉轴方向居中位置对齐
CrossAxisAlignment. stretch	拉伸,使用子 Widget 填充交叉轴
CrossAxisAlignment. baseline	与基线相匹配(不常用)

3.1.3 Column 与 Row 中子 Widget 按比例权重布局

如图 3-5 所示，线性布局 Row 中两个按钮水平排列，默认按钮包裹子文本的大小，结合 Expanded 组件使用后，第 2 个 Button 填充了水平方向剩余的所有空白区域，代码如下：

```
//代码清单 3-3 Row 权重适配
//代码路径 lib/code3/code303_Row.dart
class Exam302HomePage extends StatelessWidget {
  const Exam302HomePage({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      //页面的头部
      appBar: AppBar(title: const Text("标题")),
      body: Row(
        //主轴方向居中对齐(对于 Row 来讲就是水平方向)
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          ElevatedButton(
            onPressed: () {},
            child: Text("A"),
          ),
          //填充
          Expanded(
            child: ElevatedButton(
              onPressed: () {},
              child: Text("B"),
            ),
          ),
        ],
      ),
    );
  }
}
```

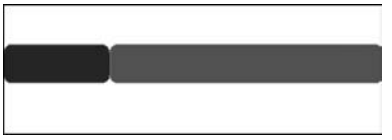

图 3-5 Row 中权重适配

如果期望 Row 中的子 Widget 平均分配宽度，如图 3-6 所示，则可以将 Row 中的子 Widget 分别使用 Expanded 来包裹，代码如下：

```
Row(
  //主轴方向居中对齐(对于 Row 来讲就是水平方向)
  mainAxisAlignment: MainAxisAlignment.center,
  children: [
Expanded(
flex: 1,
      child: ElevatedButton(
        onPressed: () {},
        child: Text("A"),
      ),
    ),
    //填充
Expanded(
flex: 1,
      child: ElevatedButton(
        onPressed: () {},
        child: Text("B"),
      ),
    ),
  ],
)
```

在这里通过 Expanded 中配置的 flex 值，来决定当前 Expanded 的子 Widget 占用的 height，如在这里的 3 个区域内容中，Expanded 分别将 flex 配置为 1，也就是将当前的 Column 的高度 height 平均分成 3 份，然后每个子 Widget 占用一份 height，也就实现了等比分布。同理应用在 Column 中，便可在竖直方向按比例分布。

3.2 非线性布局综合概述

3.2.1 Stack 用来实现层叠布局

Stack 是将子 Widget 重叠在一起，如图 3-6 所示，小点浮在图片上层。

图 3-6 Stack 的基本使用

图 3-6 中的小点可以使用 Container 结合裁剪组件 ClipOval 实现，Stack 中结合 Positioned 组合实现子 Widget 的上、下、左、右、居中排列，代码如下：

```
//代码清单 3-4 Stack 层叠布局
//代码路径 lib/code3/code304_Stack.dart
buildStack() {
  return SizedBox(
    width: 40, height: 40,//限定大小
    child: Stack(
      alignment: Alignment.center,//子组合居中
      children: [
        Positioned(
          width: 33, height: 33,//设定子组件大小
          child: Image.asset(
            "assets/images/warning_icon.png",
          ),
        ),
        Positioned(
          top: 0,right: 0,//设定子组件右上角对齐
          child: ClipOval(//裁剪
            child: Container(
              width: 10, height: 10,
              color: Colors.red,
            ),
          ),
        ),
      ],
    ),
  );
}
```

对于 alignment 属性配置的值，只对没有配置定位或部分定位的子组件起作用，如本实例中的图片没有设置对齐方式，默认使用 Stack 的 alignment 属性配置的对齐方式。

3.2.2 Wrap 用来实现层叠布局

在使用线性布局 Row 和 Colum 时，如果子 Widget 的宽度或者高度超出屏幕范围，则会报溢出错误，通过 Wrap 来支持流式布局，溢出部分则会自动折行，如图 3-7 所示。

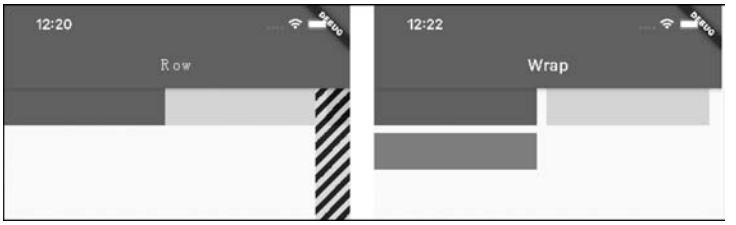

图 3-7 Row 与 Wrap 排列对比图

Wrap 流式布局的代码如下：

```
//代码清单 3-5 Wrap 流式布局
//代码路径 lib/code3/code305_Wrap.dart
Widget buildWrap(){
  return Wrap(
    //包裹的子 view
    children: [
      Container(color: Colors.blue,width: 200,height: 45,),
      Container(color: Colors.yellow,width: 200,height: 45,),
      Container(color: Colors.grey,width: 200,height: 45,),
    ],
    direction: Axis.horizontal,        //水平排列,默认此方式
    spacing: 12,                       //主轴方向上的两个 Widget 之间的间距
    runSpacing: 10,                    //行与行之前的间隔
    alignment: WrapAlignment.start,    //主轴方向的 Widget 的对齐方式
    runAlignment: WrapAlignment.start,//次轴方向上的对齐方式
  );
}
```

Wrap 的 alignment 属性用来配置主轴方向上子 Widget 的对齐方式，如这里配置为水平方向，它的取值如表 3-3 所示。

表 3-3 Wrap 的 alignment 取值概述

类　　别	描　　述
WrapAlignment.start	子组件沿开始方向对齐
WrapAlignment.end	子组件沿结束方向对齐
WrapAlignment.center	子组件居中对齐
WrapAlignment.spaceBetween	使子组件在主轴方向平均分配未占用的空间，两端对齐
WrapAlignment.spaceAround	在主轴方向，把剩余空间平分成 n 份，n 是子 widget 的数量，然后把其中一份空间分成两份，放在第 1 个 child 的前面和最后一个 child 的后面
WrapAlignment.spaceEvenly	在主轴方向，把剩余空间平分成 $n+1$ 份，然后平分所有的空间

3.2.3 实现登录页面

24min

使用层叠布局与线性布局，如图 3-8 所示，结合第 1 章中的基础组件实现登录页面，读者可观看视频【3.2.3 登录页面 Demo】。

页面主结构是通过层叠布局 Stack 将背景层、模糊层、信息输入层展示在一起，代码如下：

图 3-8　登录页面显示效果图

```
//代码清单 3-6 登录页面 Demo
//代码路径 lib/code3/code305_Wrap.dart
class Exam306HomePage extends StatefulWidget {
  const Exam306HomePage({Key? key}) : super(key: key);
  @override
  State<Exam306HomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<Exam306HomePage> {

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: SizedBox(
        width: double.infinity,
        height: double.infinity,
        child: Stack(
          children: [
            //第一层背景图片
            buildFunction1(),
            //第二层高斯模糊
            buildFunction2(),
            //第三层登录输入层
            buildFunction3(),
```

```
            ],
          ),
        ),
      );
    }
  …
  }
```

第一层通过 Image 加载需要显示的图片，第二层通过 BackdropFilter 结合 ImageFilter 实现高斯模糊效果，代码如下：

```
buildFunction2() {
  return Positioned.fill(
    child: BackdropFilter(
      filter: ImageFilter.blur(sigmaX: 3, sigmaY: 3),
      child: Container(
        color: Colors.white.withOpacity(0.4),
      ),
    ),
  );
}
```

本实例的完整源码可查看本书配套源码 flutter_base_widget 中的代码清单 3-6。

3.3 弹框用于提示用户信息

弹框又可称为对话框，用于提示用户，如软件应用升级信息弹框、优惠券活动提示、提交修改删除操作给用户的确认弹框等，本节内容概述 Material 风格与苹果风格的弹框使用、弹框中内容刷新问题。

一个基本 Material 风格弹框 AlertDialog 的内容分布区域效果图，如图 3-9 所示。

图 3-9 Material 风格 AlertDialog 效果图

3.3.1 showDialog 显示基本弹框

showDialog()方法是 Material 组件库提供的一个用于弹出 Material 风格弹框的方法，代码如下：

```
//代码清单 3-7 Dialog 的基本使用 showDialog 运行效果如图 3-9 所示
//lib/code3/example_307_showDialog.dart
void showDialogFunction() async {
  bool? isSelect = await showDialog<bool>(
    context: context,
    builder: (context) {
      return AlertDialog(
        title: const Text("温馨提示"),
        //title 的内边距,默认 left: 24.0,top: 24.0, right 24.0
        //默认底部边距,如果 content 不为 null,则底部内边距为 0
        //如果 content 为 null,则底部内边距为 20
        titlePadding: EdgeInsets.all(10),
        //标题文本样式
        titleTextStyle: TextStyle(color: Colors.black87, fontSize: 16),
        //中间显示的内容
        content: const Text("你确定要删除吗?"),
        //中间显示的内容边距
        //默认 EdgeInsets.fromLTRB(24.0, 20.0, 24.0, 24.0)
        contentPadding: EdgeInsets.all(10),
        //中间显示内容的文本样式
        contentTextStyle: TextStyle(color: Colors.black54, fontSize: 14),
        //底部按钮区域
        actions: <Widget>[
          TextButton(
            child: const Text("再考虑一下"),
            onPressed: () {
              //关闭返回 false
              Navigator.of(context).pop(false);
            },
          ),
          FlatButton(
            child: const Text("考虑好了"),
            onPressed: () {
              //关闭返回值为 true
              Navigator.of(context).pop(true);
            },
          ),
        ],
      );
    },
  );

  print("弹框关闭 $isSelect");
}
```

3.3.2 showCupertinoDialog 显示苹果风格弹框

showCupertinoDialog 用于弹出苹果风格的弹框，如图 3-10 所示。

图 3-10 showCupertinoDialog 苹果风格弹框效果图

代码如下：

```
//代码清单 3-7-1 运行效果如图 3-10 所示
//lib/code3/example_307_showDialog.dart
void showCupertinoDialogFunction(BuildContext context) async {
  bool isSelect = await showCupertinoDialog(
    //单击背景弹框是否消失
    barrierDismissible: true,
    context: context,
    builder: (context) {
      return CupertinoAlertDialog(
        title: Text('温馨提示'),
        //中间显示的内容
        content: Text("你确定要删除吗?"),
        //底部按钮区域
        actions: [
          CupertinoDialogAction(
            child: Text('确认'),
            onPressed: () {
              Navigator.of(context).pop();
            },
          ),
          CupertinoDialogAction(
            child: Text('取消'),
            isDestructiveAction: true,
            onPressed: () {
              Navigator.of(context).pop();
            },
          ),
        ],
      );
    },
  );
}
```

3.3.3 showBottomSheet 底部显示弹框

showBottomSheet 用来在视图底部弹出一个 Material Design 风格对话框，如图 3-11 所示。这种业务应用场景也比较多，如页面中的分享面板等。

图 3-11 showBottomSheet 底部弹框效果图

代码如下：

```
//代码清单 3-7-2 showBottomSheet
//lib/code3/example_307_showDialog.dart
void showBottomSheetFunction(BuildContext context) async {
  showBottomSheet(
    context: context,
    builder: (BuildContext context) {
      return buildContainer(context);
    },
  );
}

Container buildContainer(BuildContext context) {
  return Container(
    color: Colors.white,
    height: 240,
    width: double.infinity,
    child: Column(
      children: <Widget>[
        Container(
          alignment: Alignment.center,
          height: 44,
          child: Text(
            "温馨提示",
            style: TextStyle(fontSize: 16, fontWeight: FontWeight.w500),
          ),
        ),
        Expanded(
```

```
            child: Text("这里是内容区域"),
          ),
          Container(
            height: 1,
            color: Colors.grey[200],
          ),
          Container(
            height: 64,
            child: Row(
              children: [
                Expanded(
                  child: TextButton(
                    child: Text("再考虑一下"),
                    onPressed: () {
                      //关闭后返回值为 false
                      Navigator.of(context).pop(false);
                    },
                  ),
                ),
                Container(
                  width: 1,
                  color: Colors.grey[200],
                ),
                Expanded(
                  child: FlatButton(
                    child: Text("考虑好了"),
                    onPressed: () {
                      //关闭后返回值为 true
                      Navigator.of(context).pop(true);
                    },
                  ),
                ),
              ],
            ),
          ),
        ],
      ),
    );
  }
```

在使用 showBottomSheet 时，可能会出现异常，异常信息如下：

```
[VERBOSE-2:ui_dart_state.cc(177)] Unhandled Exception: No Scaffold widget found.
Example701 widgets require a Scaffold widget ancestor.
The specific widget that could not find a Scaffold ancestor was:
…
```

这是因为在调用 showBottomSheet 时使用的 Context 不是 Scaffold 对应的 Context，可以考虑使用 Builder 组件来包裹，如在单击按钮时调用显示底部弹框，结合 Builder 的代码如下：

```
Builder(
  builder: (BuildContext context) {
    return ElevatedButton(
      child: Text("BottomSheet "),
      onPressed: () {
        showBottomSheetFunction(context);
      },
    );
  },
),
```

showBottomSheet()方法相当于调用了 Scaffold 的 showBottomSheet()方法，相当于在当前视图中插入显示的一个布局视图。

3.3.4 showModalBottomSheet 底部弹出对话框

showModalBottomSheet 用来在视图底部弹出一个 Modal Material Design 风格对话框，是菜单或对话框的替代品，可以防止用户与应用程序的其余部分进行交互，showBottomSheet 创建的底部弹窗视图，不阻止用户与应用程序交互基本使用，代码如下：

```
//代码清单 3-7-3 showModalBottomSheet
//lib/code3/example_307_showDialog.dart
void showModalBottomSheetFunction(BuildContext context) async {
  showModalBottomSheet(
    context: context,
    //背景颜色
    backgroundColor: Colors.grey,
    //阴影颜色
    barrierColor: Color(0x30000000),
    //单击背景消失
    isDismissible: true,
    //下滑消失
    enableDrag: true,
    builder: (BuildContext context) {
      //代码清单 3-7-2 中定义的视图布局
      return buildContainer(context);
```

```
    },
  );
}
```

showCupertinoModalPopup 用来快速构建并弹出 iOS 风格的底部弹框，代码如下：

```
//代码清单 3-7-4 showCupertinoModalPopup
  //lib/code3/example_307_showDialog.dart
  void showCupertinoModalFunction(BuildContext context) async {
    showCupertinoModalPopup<int>(
      context: context,
      builder: (cxt) {
        CupertinoActionSheet dialog = CupertinoActionSheet(
          title: Text("温馨提示"),
          message: Text('请选择分享的平台'),
          //取消按钮
          cancelButton: CupertinoActionSheetAction(
            onPressed: () {},
            child: Text("取消"),
          ),
          actions: <Widget>[
            CupertinoActionSheetAction(
                onPressed: () {
                  Navigator.pop(cxt, 1);
                },
                child: Text('QQ')),
            CupertinoActionSheetAction(
                onPressed: () {
                  Navigator.pop(cxt, 2);
                },
                child: Text('微信')),
            CupertinoActionSheetAction(
                onPressed: () {
                  Navigator.pop(cxt, 3);
                },
                child: Text('系统分享')),
          ],
        );
        return dialog;
      },
    );
  }
```

运行效果如图 3-12 所示。

图 3-12 showModalBottomSheet 运行效果图

3.4 小结

本章概述了线性布局 Column、Row、层叠布局 Stack、流式布局 Wrap 的基本排版 Widget，用这些排版 Widget 再结合第 1 章中的基础组件综合使用，就可以构建出基本的应用程序，这样便进入了 Flutter 开发的初级阶段。