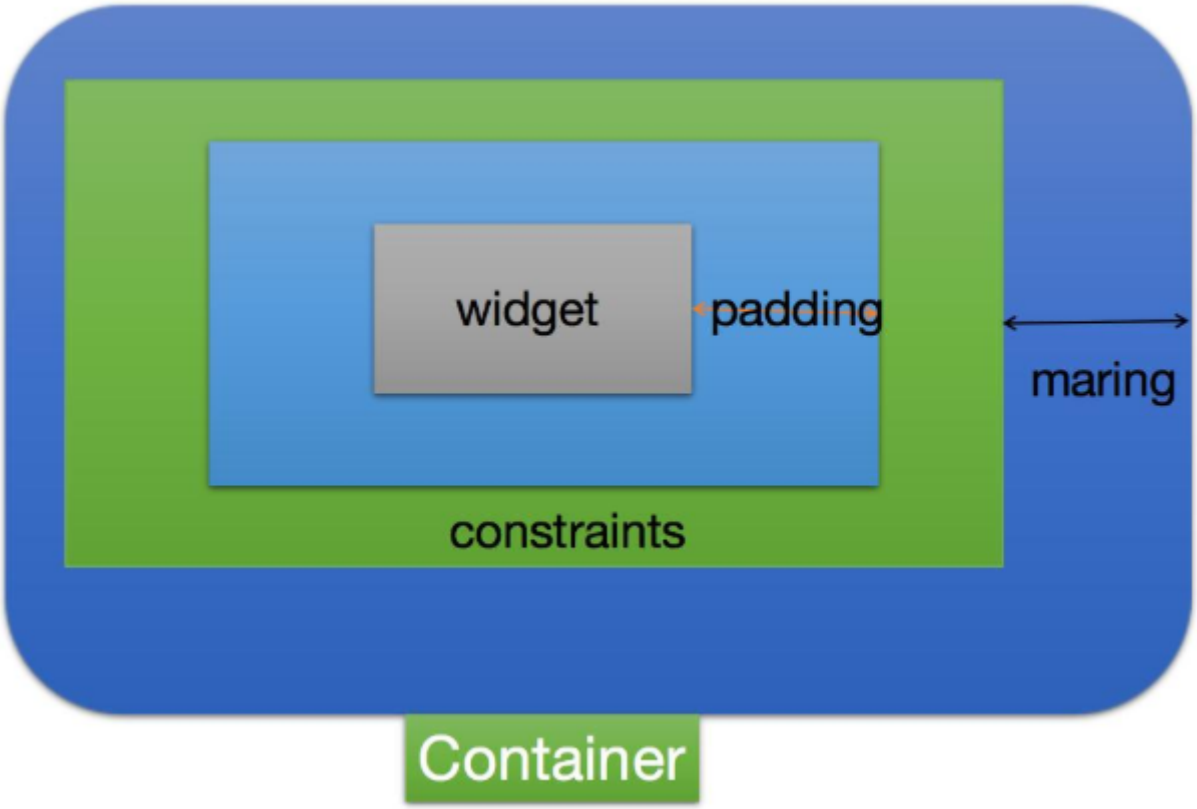
版本号	描述	日期(更新日期)/更新人
1.0	Android Studio Flutter UI布局	2019/06/13 **刘志保**
	https://api.flutter.dev/flutter/widgets/Container-class.html	

Flutter 布局的种类非常的丰富,有多达10种大型布局方式,比如Row, Column, Container, Center, Grid, ListView等等.这些布局开发定制化程度最高的是Container布局,也是后面开发使用最多的布局方式.这种布局方式主要是在小单元布局细节调整上有巨大的优势.这里也主要介绍Container,其他的布局方式可以自行补齐.

Container布局参考:https://api.flutter.dev/flutter/widgets/Container-class.html,Container结构图如下:



从结构图可以看出,基础控件在整体布局的细节位置可以如上设置进行.根据Container的构造函数如下:

Container({Key key, AlignmentGeometry alignment, EdgeInsetsGeometry padding, Color color, Decoration decoration, Decoration foregroundDecoration, double

width, double height, BoxConstraints constraints, EdgeInsetsGeometry margin, Matrix4 transform, Widget child }),构造函数传入的参数

字段名	描述	举例
AlignmentGeometry alignment	对齐方式,使用Alignment和AlignmentDirectional进行操作	alignment:Alignment.Center
EdgeInsetsGeometry padding	边框外沿边距,使用EdgeInsets EdgeInsetsDirectional进行操作	padding:EdgeInsets.all(8.0)
Color color	Container的布局背景颜色	color:Color(0x00ff00)
Decoration decoration	装饰类,Flutter大致有四种:BoxDecoration FlutterLogoDecoration ShapeDecoration UnderlineTabIndicator,可以实现一些边框颜色,图片,边框线宽度	decoration:BoxDecoration(color: const Color(0xff7c94b6), image: DecorationImage(image: ExactAssetImage('images/flowers.jpeg'), fit: BoxFit.cover,), border: Border.all(color: Colors.black, width: 8.0,),),
Decoration foregroundDecoration	背景装饰类,基本上同上	
double width	Container的宽度	
double height	Container的高度	
BoxConstraints constraints	Container宽高最大最小限制,当没有设置Container的width和height时,Container的宽高是自适应大小的,如果增加constraints限制,自适应大小就要受到限制.	constraints:BoxConstraints(double minWidth: 0.0, double maxWidth: double.infinity, double minHeight: 0.0, double maxHeight: double.infinity),其中double.infinity自适应
EdgeInsetsGeometry margin	内部边距设置,通过EdgeInsets EdgeInsetsDirectional进行操作	margin:const EdgeInsets.all(8.0)
Matrix4 transform	Container内部做向量变化,如旋转,位移等变形操作	
Widget child	这个即Container中的UI件	

注意一点,构造函数里面的参数类型绝大部分是以基类base class形式设置的,比如Decoration decoration,这个Decoration是base class,所以在使用的时候传入它的子类型,

比如BoxDecoration FlutterLogoDecoration ShapeDecoration UnderlineTabIndicator,这个很容易理解,这是面向对象编程常用的做法.Flutter 开发里面构造函数非常的重要,基本上Flutter里面在使用类的时候只使用类的构造函数出来一个对象就可以使用了,而这个类里面的Method很少涉及,比如BoxContraints,它在使用的时候如下:

```
constraints:BoxConstraints(double minWidth: 0.0, double maxWidth: double.infinity, double minHeight: 0.0, double maxHeight: double.infinity)
```

你就不能在布局外面如下操作:

```
BoxConstraints boxContraints=new BoxContraints();
boxContraints.setMinWidth(8.0);
...
```

然后调用:

```
contraints:boxContraints,
```

这种做法是不存在的.

当然可以这样

```
var boxContraints=new BoxContraints(8.0,8.0,8.0,8.0);
```

然后调用:

```
contraints:boxContraints,
```

也就是说,Flutter 开发中类所有的配置参数都在构造函数中完成,不会提供类方法设置参数.

这个地方可以举一反三.

下面举个例子如下:

开发一个这样的效果.

```
Center(
 child: Container(
 color: Colors.amber,
 child:
 Row(
 mainAxisAlignment: MainAxisAlignment.spaceBetween,
 children: <Widget>[
 new Container(
 margin: EdgeInsets.fromLTRB(16, 36, 0, 16),
  child: new Icon(Icons.access_alarm,color: Colors.green),
  ),
```

```
 new Container(
 margin: EdgeInsets.fromLTRB(16, 36, 0, 16),
 child: new Text("我的页面"),
 ),
 new Container(
 margin: EdgeInsets.fromLTRB(16, 36, 16, 16),
 child: new Icon(Icons.account_balance,color: Colors.green),
 ),
 ],
 ),
 ),
);
```

这个模拟自定义ActionBar,整部布局是Container,然后有三个widget元素排列在一行,所以使用了Row布局,然后三个元素需要设置更多的细节参数,然后再用Container包含它们,为他们做细节调整,比如margin等等,下面增加一些装饰:

```
Center(
 child: Container(
 color: Colors.amber,
 child:
 Row(
 mainAxisAlignment: MainAxisAlignment.spaceBetween,
 children: <Widget>[
 new Container(
 margin: EdgeInsets.fromLTRB(16, 36, 0, 16),
 child: new Icon(Icons.access_alarm,color: Colors.green),
 ),
 new Container(
 margin: EdgeInsets.fromLTRB(16, 36, 0, 16),
 child: new Text("我的页面"),
 decoration: BoxDecoration(
 image: new DecorationImage(
 image: new ExactAssetImage('assets/images/calendar.png'),
 fit: BoxFit.cover,
 ),
 color: Colors.amber,
 border: Border.all(
 color: Colors.green,
 width: 4.0),
```

```
  gradient:LinearGradient(
  begin: Alignment.topLeft,
  end: Alignment(0.8, 0.0), // 10% of the width, so there are ten blinds.
  colors: [const Color(0xFF0FFFEE), const Color(0xFF99ff99)], // whitish to gray
  tileMode: TileMode.repeated, // repeats the gradient over the canvas
  ),
  ),
  ),
  new Container(
  margin: EdgeInsets.fromLTRB(16, 36, 16, 16),
  child: new Icon(Icons.account_balance,color: Colors.green),
  ),
  ],
  ),
  ),
);
```

上面的效果用这一段代码实现,似乎有点长,看起来有点恐怖,但是代码越多细节调整就更准确!

下面介绍如何阅读Flutter SDK API :

接着上面的案例来看看装饰Decoration(基类)如何进行操作.

<1>:首先查找Decoration的类说明:https://api.flutter.dev/flutter/painting/Decoration-class.html

由于Decoration是基类,那么:

```
decoration:[这里必须传Decoration的子类]
```

我们就看看Decoration有哪些子类:

Decoration class

A description of a box decoration (a decoration applied to a Rect).

This class presents the abstract interface for all decorations. See BoxDecoration for a concrete example.

To actually paint a Decoration, use the createBoxPainter method to obtain a BoxPainter. Decoration objects can be shared between boxes; BoxPainter objects can cache resources to make painting on a particular surface faster.

Inheritance

Object › Diagnosticable › Decoration

Implementers

BoxDecoration, FlutterLogoDecoration, ShapeDecoration, UnderlineTabIndicator

可以看出,Decoration的子类有四个,上面测试Demo中使用了BoxDecoration.然后再看看BoxDecoration类如何构造:

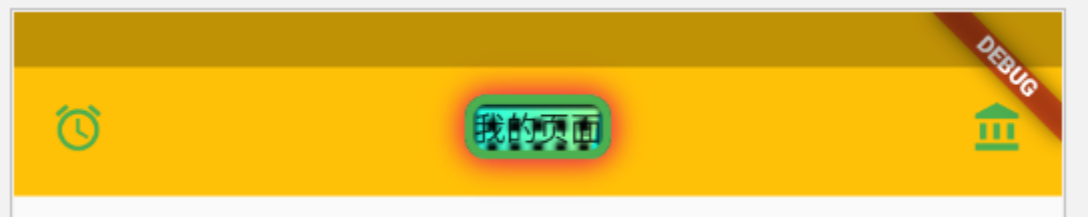
```
BoxDecoration({Color color, DecorationImage image, BoxBorder border, BorderRadiusGeometry borderRadius, List<BoxShadow> boxShadow, Gradient gradient, BlendMode backgroundBlendMode, BoxShape shape: BoxShape.rectangle })
```

参数	说明	举例
Color color	背景颜色	
DecorationImage image	背景图片	image: new ExactAssetImage('assets/images/calendar.png'),
BoxBorder border	边界宽度和颜色	
BorderRadiusGeometry borderRadius	边界倒角,BorderRadiusGeometry 也是基类,需要通过构造他们的子类 BorderRadius BorderRadiusDirectional 进行操作修饰	borderRadius: BorderRadius.all(Radius.circular(10.0)),
List<BoxShadow> boxShadow	阴影,可以设置多重	List<BoxShadow> list=new List(); list.add(BoxShadow(color: Colors.deepOrange,blurRadius: 10.0)); list.add(BoxShadow(color: Colors.pink,blurRadius: 10.0)); boxShadow: list,
Gradient gradient	背景渐变色,渐变有三种方式:LinearGradient RadialGradient SweepGradient	gradient:LinearGradient(begin: Alignment.topLeft, end: Alignment(0.8, 0.0), // 10% of the width, so there are ten blinds. colors: [const Color(0xFF0FFFEE), const Color(0xFF99ff99)], // whitish to gray tileMode: TileMode.repeated, // repeats the gradient over the canvas)
BlendMode backgroundBlendMode	背景适合混合,如果背景有多种颜色或者图片背景,如果设置混合即多层背景颜色值进行混合运算后的颜色值显示,如果不进行就直接	具体参考:https://api.flutter.dev/flutter/dart-ui/BlendMode-class.html 更详细

	覆盖	class.html#l 更详细
BoxShape shape: BoxShape.rectangle	默认是方形的	

其实还是非常的简单的,只需要照着SDK API的指导去开发即可！

上面大致调整的效果如下,阴影边框背景：



其他的可以根据SDK API去尝试,举一反三!

其他的布局可以参考githhub上面的资源!