Flutter入门笔记

1. Flutter目录结构

|  |  |
| --- | --- |
| 文件夹 | 作用 |
| Anoroid | Anoroid平台相关代码 |
| ios | Ios平台相关代码 |
| lib | Flutter相关代码，我们主要编写的代码就在这个文件夹 |
| test | 用于存放测试代码 |
| Pubspec.yaml | 配置文件，一般用来存放依赖 |

1. Flutter把内容单独抽离成一个组件

在flutter中自定义组件其实就是一个类，这个类需要继承StatelessWidget/StatefulWidget，前期我们都继承StatelessWidget。

StatelessWidget是无状态组件，状态不可改变的widget

StatefulWidget是有状态组件，持有的状态可能在widget生命周期改变。

1. MaterialApp和Scaffold两个组件装饰App
   1. MaterialApp

MaterialApp是一个方便的Widget，它封装了应用程序实现Material Design所需要的一些Widget。一般作为顶层widget使用。

* + 1. 常用属性：
* home（主页）
* title (标题)
* color (颜色)
* theme（主题）
* routes（路由）
* ……
  1. Scaffold

Scaffold是Material Design布局结构的基本实现。此类提供了用于显示drawer、snackbar和底部sheet的API

* + 1. 常用属性
* appBar-显示在界面顶部的一个AppBar。
* body-当前界面所显示的主要内容Widget
* drawer-抽屉菜单控件
* ……
  1. Flutter组件
     1. Container组件

|  |  |
| --- | --- |
| 常用属性 | 说明 |
| width | 宽度 |
| height | 高度 |
| color | 背景颜色，值为一个Color对象，不能与decoration属性同时设置 |
| margin | 外边距，值为一个EdgeInsets对象。EdgeInsets对象即可调用EdgeInsets.all()方法统一设置左上右下四条边距，也可以调用EdgeInsets.FromLTRB()分别设置左上右下四条边的边距。 |
| padding | 内间距，值同margin |
| alignment | 对齐方式，可选值 |
| decoration | 装饰，背景边框等不能与color属性同时设置，会绘制在child之下，也就是会被child覆盖 |
| foregroundDecoration | 也是装饰，但是会绘制在 child 之上，也就是会覆盖 child。 |
| constraints | 约束 |
| transform | 形状变换， |
| child | 子组件 |

BoxDecoration只是decoration的一个实现类，其他实现类还有FlutterLogoDecoration、ShapeDecoration、UnderlineTabIndicator，这里只是介绍一下常用的 BoxDecoration。

color：背景填充颜色，值为一个 Color 对象。

border：值为一个 BoxBorder 对象，该对象可以设置边框颜色、边框宽度、边框样式等。

borderRadius：边框圆角，可以调用 BorderRadius.all() 统一设置四个角的圆角，也可以调用 BorderRadius.only() 分别设置四个角的圆角。

gradient：设置成渐变效果的背景，会覆盖 color。

boxShadow：阴影效果，值为一个 BoxShadow 集合。

backgroundBlendMode：应该是背景混合模式，这个应该比较复杂，后面再研究。

image：使用图片作为装饰

* + 1. Text组件

|  |  |
| --- | --- |
| 名称 | 功能 |
| textAlign | 文本对齐方式：  Center居中，left左对齐，right右对齐，justify两端对齐 |
| textDirection | 文本方向:  ltr从左至右  rtl从右到左 |
| overflow | 文本超出屏幕之后的处理方式：  Clip裁剪，fade渐隐，ellipsis省略号 |
| textScaleFactor | 字体显示倍率 |
| maxLines | 文字显示最大行 |
| Style | 字体的样式设置 |

下面是style的参数：

|  |  |
| --- | --- |
| decoration | 文字装饰线：  none没有线，lineThrough删除线，overline上划线，underline下划线 |
| decorationColor | 文字装饰线颜色 |
| decorationStyle | 文字装饰线风格（[dashed，dotted]虚线），double两根线，solid一根实线，wavy波浪线] |
| wordSpacing | 单词间隙（如果是负数会让单词变得更经凑） |
| letterSpacing | 字母间隙（如果是负值，会让字母变得经凑） |
| fontStyle | 文字样式（italic斜体，normal正常体） |
| fontSize | 文字大小 |
| color | 文字颜色 |
| fontWeight | 字体粗细（bold粗体，normal正常体） |

* + 1. Flutter 图片组件

图片组件是显示图像的组件，image组件有很多构造函数，这里只列出了两个

Image.asset 本地图片

Image.nerwork 远程图片

* + - 1. Image组件的常用属性

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 说明 |
| alignment | Alignment | 图片对齐的方式 |
| color  colorBlendMode |  | 设置图片的背景颜色通常和colorBlendMode配合一起使用，这样可以是图片颜色和背景色混合。上面的图片就是进行了颜色混合，绿色背景和图片红色的混合 |
| fit | BoxFit | fit属性用来控制图片的拉伸和挤压，这都是根据父容器来的  BoxFit.fill:全图显示，图片会被拉伸，并且充满父容器  BoxFit.Contain:全图显示，显示原比例，可能会有空隙。  BoxFit.cover显示可能拉伸，可能裁切，充满（图片要充满整个容器，还不变形）  BoxFit.fitWidth：宽度充满（横向充满），显示可能拉伸，可能裁切。  Box.fitHeight：高度充满（竖向充满），显示可能拉伸，可能裁切。  BoxFit.scaleDown:效果和contain差不多，但是此属性不允许显示超过原图片大小，即可小不可大。 |
| repeat | 平铺 | ImageRepeat.repeat：横向和纵向都进行重复，直到铺满整个画布。  ImageRepeat.repeatX：横向重复，纵向不重复  ImageRepeat.repeatY:纵向重复，横向不重复 |
| width |  | 宽度 一般结合ClipOval才能看到效果 |
| height |  | 高度 一般结合ClipOval才能看到效果 |

备注：clipOval可以将图片处理为圆形

clipOval(String url,{height:x,width:x,fit:BoxFit.xxx})，

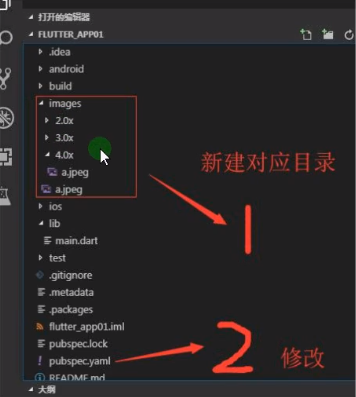
CircleAvatar(

backgroundImage:NetworkImage(“xxxxxx”)

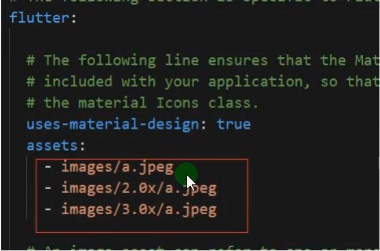
)

flutter导入本地图片

新建目录：在每个目录下都放入图片



然后打开pubspec.yaml声明一下添加的文件图片，注意要配对



最后在代码中就可以用了。这里很坑的是目前只支持jpeg格式（2020.02.05）

更多属性参考：www.https://api.flutter.dev/flutter/widgets/Image-class.html

* + 1. Flutter列表组件概述

列表布局是我们项目开发中最常用的一种布局方式。Flutter中我们可以通过ListView来定义列表项，支持垂直和水平方向展示。通过一个属性就可以控制列表的显示方向。列表有以下分类：

1、垂直列表

2、垂直图文列表

3、水平列表

4、动态列表

5、矩阵式列表

Flutter列表参数

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 说明 |
| scrollDirection | Axis | Axis.horizontal水平列表  Axis.vertical垂直列表 |
| padding | EdgeInsetsGeometry | 内边距 |
| resolve | bool | 组件反向排序 |
| children | List<Widget> | 列表元素 |

**注意：**在ListView中，如果是垂直布局，那么宽度是无效的，是父容器的100%。如果是水平布局，高度是无效的，默认是父容器的100%。因此，在使用ListView的时候最好用一层父容器给包裹。

ListView.builder(

itemCount:”此处输入list的长度，即数据的长度”,

itemBuilder: function(context,index) {

return ListTitle(

leading:xxxx,

title: xxxxxx

)

}

)

ListView不能嵌套ListView组件

* + 1. Flutter GridView组件（矩阵布局，网格布局）

GridView 创建网格列表有很多种方式，下面主要介绍两种：

* 可以通过GridView.count实现网格布局
* 通过GridView.builder实现网格布局

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 说明 |
| scrollDirection | Axis | 滚动方法 |
| padding | EdgeInsetsGeometry | 内边距 |
| resolve | bool | 组件反向排序 |
| crossAxisSpacing | double | 水平子Widget之间的间距 |
| mainAxisSpacing | double | 垂直子Widget之间间距 |
| crossAxisCount | int | 一行的Widget数量 |
| childAspectRatio | double | 子Widget宽高比例 |
| children |  | <Widget>[] |
| gridDelegate | SliverGridDelegateWithFixedCrossAxisCount  (常用)  SliverGridDelegateWithMaxCrossAxisExtent | 控制布局主要在GridView.builder里面 |

* + 1. Flutter Padding 组件

在html中长见的布局标签都有padding属性，但是Flutter中很多Widget是没有padding属性的。这个时候我们就可以使用Padding组件处理容器与子元素之间的间距。

|  |  |
| --- | --- |
| 属性 | 说明 |
| padding | padding值，EdgeInsets设置填充的值 |
| child | 子组件 |

* + 1. Flutter Row水平布局组件

|  |  |
| --- | --- |
| 属性 | 说明 |
| mainAxisAlignment | 主轴的排序方式 |
| crossAxisAlignment | 次轴的排序方式 |
| Children | 组件子元素 |

* + 1. Flutter Column垂直布局组件

|  |  |
| --- | --- |
| 属性 | 说明 |
| mainAxisAlignment | 主轴的排序方式 |
| crossAxisAlignment | 次轴的排序方式 |
| children | 组件子元素 |

* + 1. Flutter Expanded

类似Web中的flex布局 Expanded可以用在Row和Column布局中

|  |  |
| --- | --- |
| 属性 | 说明 |
| flex | 元素站整个父Row/Column的比例 |
| Child | 子元素 |

* + 1. Flutter Stack组件

Stack表示堆的意思，我们可以用Stack或者Stack结合Align或者Stack结合Position来实现页面的定位布局。

|  |  |
| --- | --- |
| 属性 | 说明 |
| alignment | 配置所有子元素的显示位置 |
| children | 子组件 |

* + 1. Flutter Stack Align

Stack组件中结合Align组件可以控制每个子元素的显示位置。

|  |  |
| --- | --- |
| 属性 | 说明 |
| alignment | 配置所有子元素的显示位置 |
| children | 子组件 |

* + 1. Flutter Stack Positioned

Stack组建中结合Positioned组建也可以控制每个子元素的显示位置

|  |  |
| --- | --- |
| 属性 | 说明 |
| top | 子元素距离顶部距离 |
| bottom | 子元素距离底部距离 |
| left | 子元素距离左边距离 |
| right | 子元素距离右边距离 |
| child | 子组件 |

* + 1. Flutter AspectRatio组建

AspectRatio的作用是根据设置调整子元素child的宽高比。

AspectRatio首先会在布局限制条件下允许的范围内尽可能的扩展，widget的高度是由宽度和比率决定的，类似于BoxFit中的contain，按照固定的比率去尽量占满区域。

如果在满足所有限制条件过后无法找到一个可行的尺寸，AspectRatio最终将会去优先适应布局限制条件，而忽略所设置的比率。

|  |  |
| --- | --- |
| 属性 | 说明 |
| aspectRatio | 宽高比，最终可能不会根据这个值去布局，具体要看综合因素，外层是否允许按照这个比率进行布局 |

一般用于图片平铺

* + 1. Flutter Card组件

Card是卡片组件块，内容可以由大多数类型的Widget构成，Card具有圆角和阴影，这样它看起来有立体感。

|  |  |
| --- | --- |
| 属性 | 说明 |
| margin | 外边距 |
| child | 子组件 |
| shape | Card的阴影效果，默认的阴影效果为圆角的长方形边。 |

* + 1. Flutter Wrap组件

Wrap可以实现流布局，单行的Wrap跟Row表现几乎一致，单列的Wrap则跟Row表现几乎一致。但Row与Column都是单行单列的，Wrap则突破了这个限制，mainAxis上空间不足的时候，则向crossAxis上去扩展展示。

|  |  |
| --- | --- |
| 属性 | 说明 |
| direction | 主轴的方向，默认水平 |
| alignment | 主轴的对齐方式 |
| spacing | 主轴方向上的间距 |
| textDirection | 文本方向 |
| verticalDirection | 定义了children摆放顺序，默认是down，间Flex相关属性 |
| runAlignment | run的对齐方式。run可以理解为新的行或者列，如果是水平方向布局的话，run可以理解为新的一行 |
| runSpacing | run的间距（次轴的间距） |

* + 1. Flutter中自定义有状态组件

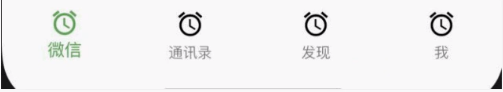
在flutter中自定义组件其实就是一个类，这个类需要继承StatelessWidget/StatefulWidget。

StatelessWidget是无状态组件，状态不可改变的widget。

StatefulWidget是有状态组件，持有的状态可能就在widget生命周期改变。通俗的讲：如果我们想改变页面中的数据的话，这个时候就需要用到StatefulWidget

* + 1. Flutter BottomNavigationBar组件

BottomNavigationBar是底部导航条，可以让我们定义底部Tab切换，bottomNavigationBar是Scaffold组件的参数



BottomNavigationBar常见的属性

|  |  |
| --- | --- |
| 属性名 | 说明 |
| items | List<BottomNavigationBarItem>底部导航条按钮集合 |
| iconSize | 图标的大小 |
| currentIndex | 默认选中第几个 |
| onTap | 按下事件，可以用来设置点击的效果 |
| fixedColor | 选中时候的颜色 |
| type | BottomNavigationBarType.fixed  BottomNavigationBarType.shifting |

* + 1. 1
  1. Flutter中的路由

Flutter中的路由通俗的讲，就是页面跳转。在flutter中通过Navigator组件管理路由导航。并且提供了管理堆栈的方法。如Navigatior.push（跳转到某个页面）和Navigator.pop（返回到某个页面）

Flutter中给我们提供了两种配置路由跳转的方式;

1. 基本路由
2. 命名路由
   * 1. Flutter中的基本路由使用

比如我们现在想从HomePage组件跳转到SearchPage组件。

1. 需要在HomePage中引入SearchPage.dart

import ‘../SearchPage.dart’

1. 在HomePage中通过下面方法跳转

RaisedButton(

child: Text(“跳转到搜索页面”),

onPressed: () {

Navigator.of(context).push(

MaterialPageRoute(

builder: (BuildContext context) {

return SerachPage();

}

)

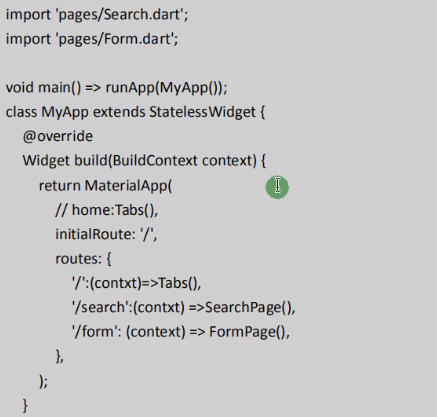
)

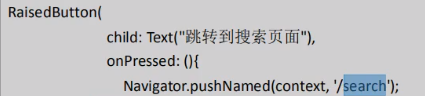
}

)

* + 1. 命名路由

首先需要在MaterialApp中配置路由，然后通过Navigator.pushNamed(context,’xxx’)方法进行跳转:





* + 1. Flutter中的替换路由

比如我们从用户中心页面跳转到RegisterFirst页面，然后从RegisterFirst页面通过pushReplacementNamed跳转到了registerSecond页面。这个时候当我们点击registerSecond的返回按钮的时候，它会直接返回到用户中心。

Navigator.of(context).pushReplacementNamed(‘/registerSecond’)

* + 1. Flutter返回到跟路由

比如我们从用户中心跳转到RegisterFirst页面，然后从RegisterFirst页面跳转到RegisterSecond页面，然后从RegisterSecond跳转到RegisterThird页面。这个时候我们想的是RegisterThird注册成功之后返回到用户中心。这个时候就用到了返回跟路由的方法。

Navigator.of(context).pushAndRemoveUntil(

new MaterialPageRoute(builder: (context)=> new Tabs(index:1)),

(route)=>route == null

)

* + 1. 1
  1. MaterialApp