Material组件

2021年6月30日 11:22

Material组件提供实现了Material Design准则的视觉、行为和动作的Widget
大致有以下几个类型：

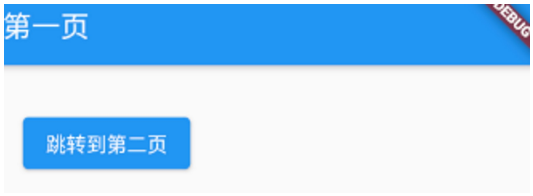
1. 应用程序结构和导航
2. Button
3. 输入和选择
4. 对话框、警告框和面板
5. 信息显示
6. 布局

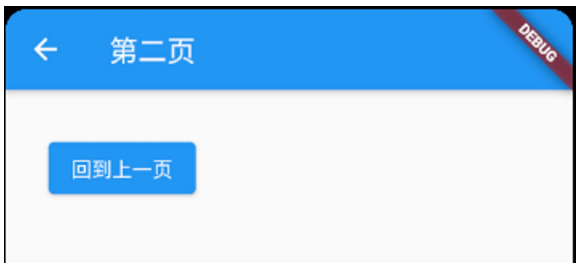
一、路由
flutter路由的使用方法主要有两种，一种是新建路由，一种是注册路由。

1.1 新建路由

```
classFirstPageextendsStatelessWidget{
constFirstPage({Key?key}):super(key:key);

@override
Widgetbuild(BuildContextcontext){
returnScaffold(
appBar:AppBar(title:Text('第一页'),),
body:Padding(padding:EdgeInsets.all(30.0),child:ElevatedButton(child:Text
('跳转到第二页'),onPressed:()
{Navigator.push(context,MaterialPageRoute(builder:(context)=>
SecondPage()));},),),
);
}
}

classSecondPageextendsStatelessWidget{
constSecondPage({Key?key}):super(key:key);

@override
Widgetbuild(BuildContextcontext){
returnScaffold(
appBar:AppBar(title:Text('第二页'),),
body:Padding(
padding:EdgeInsets.all(30.0),
child:ElevatedButton(
child:Text('回到上一页'),
onPressed:(){
Navigator.pop(context);
},
),
),
);
```

```
}
}
```





MaterialPageRoute是一个模态路由，可以根据各个平台进行页面替换，在Android平台进入是向上滑动并淡出，IOS平台是从右侧滑入

1.2 注册路由

```
classMyAppextendsStatelessWidget{
@override
Widgetbuild(BuildContextcontext){
returnMaterialApp(
title:'FlutterDemo',
theme:ThemeData(
primarySwatch:Colors.blue,
),
home:FirstPage(),
routes:<String,WidgetBuilder>{
'/first':(BuildContextcontext)=>FirstPage(),
'/second':(BuildContextcontext)=>SecondPage(),
},
initialRoute:'/first',
);
}
}

classFirstPageextendsStatelessWidget{
constFirstPage({Key?key}):super(key:key);

@override
Widgetbuild(BuildContextcontext){
returnScaffold(
appBar:AppBar(title:Text('第一页'),),
body:Padding(padding:EdgeInsets.all(30.0),child:ElevatedButton(child:Text
('跳转到第二页'),onPressed:()
{Navigator.pushNamed(context,'/second');},),),
);
}
}

classSecondPageextendsStatelessWidget{
constSecondPage({Key?key}):super(key:key);
```

```
@override
Widgetbuild(BuildContextcontext){
returnScaffold(
appBar:AppBar(title:Text('第二页'),),
body:Padding(
padding:EdgeInsets.all(30.0),
child:ElevatedButton(
child:Text('回到上一页'),
onPressed:(){
Navigator.pop(context);
},
),
),
);
}
}
```

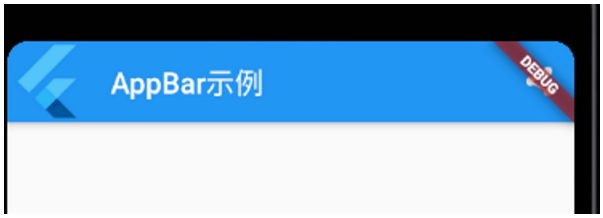
routes可以初始化一个路由列表，当推送路由时，将在routes中查找路径名称，如果名称存在，则关联的WidgetBuilder用于构造MaterialPageRoute

二、 AppBar

AppBar由toolBar和其他可选的Widget组成，比如TabBar和FlexibleSpaceBar。

AppBar会在顶部显示leading、title、actions等内容，底部bottom通常显示TabBar。

```
classMyScaffldextendsStatelessWidget{
constMyScaffld({Key?key}):super(key:key);

@override
Widgetbuild(BuildContextcontext){
returnScaffold(
appBar:AppBar(
title:Text('AppBar示例'),
leading:FlutterLogo(),
actions:<Widget>[
IconButton(onPressed:(){
print('pressed');
},icon:Icon(Icons.share))
],
),
);
}
}
```

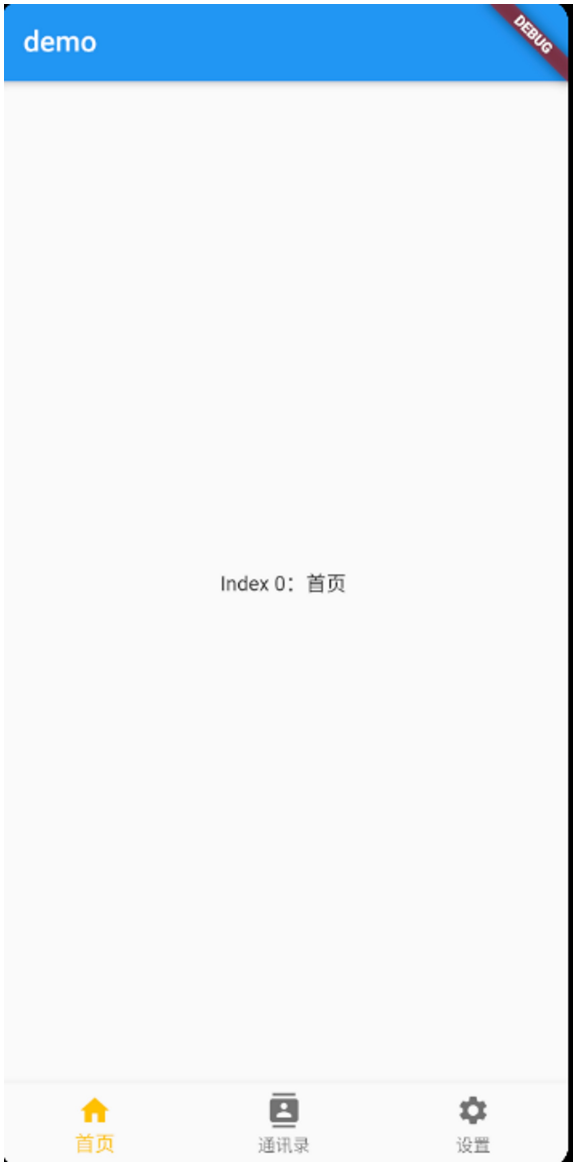


三、BottomNavigationBar

BottomNavigationBar是底部的导航栏，用于在3到5个的少量视图中进行选择。导航栏的选项卡由文本标签、图标或两者结合的形式组成。

通常与javaScaffold结合使用，它会作为Scaff.bottomNavigationBar参数。

```
class_MyStatefulWidgetStateextendsState<MyStatefulWidget>{
int_selectedIndex=0;
staticconstList<Widget>_widget=<Widget>[
Text(
'Index0：首页'
),
Text(
'Index1：通讯录'
),
Text(
'Index2：设置'
),
];
@override
Widgetbuild(BuildContextcontext){
returnScaffold(
appBar:AppBar(
title:Text('demo'),
),
body:Center(
child:_widget.elementAt(_selectedIndex),
),
bottomNavigationBar:BottomNavigationBar(
items:const<BottomNavigationBarItem>[
BottomNavigationBarItem(icon:Icon(Icons.home),label:"首页"),
BottomNavigationBarItem(icon:Icon(Icons.contacts),label:"通讯录"),
BottomNavigationBarItem(icon:Icon(Icons.settings),label:"设置"),
],
currentIndex:_selectedIndex,
selectedItemColor:Colors.amber,
onTap:_onItemTapped,
),
);
}

void_onItemTapped(intindex){
setState((){
_selectedIndex=index;
});
}
}
```



由于使用的Widget需要在Widget的生命周期中改变状态，因此需要使用StatefulWidget。createState方法会为此Widget创建可变状态。

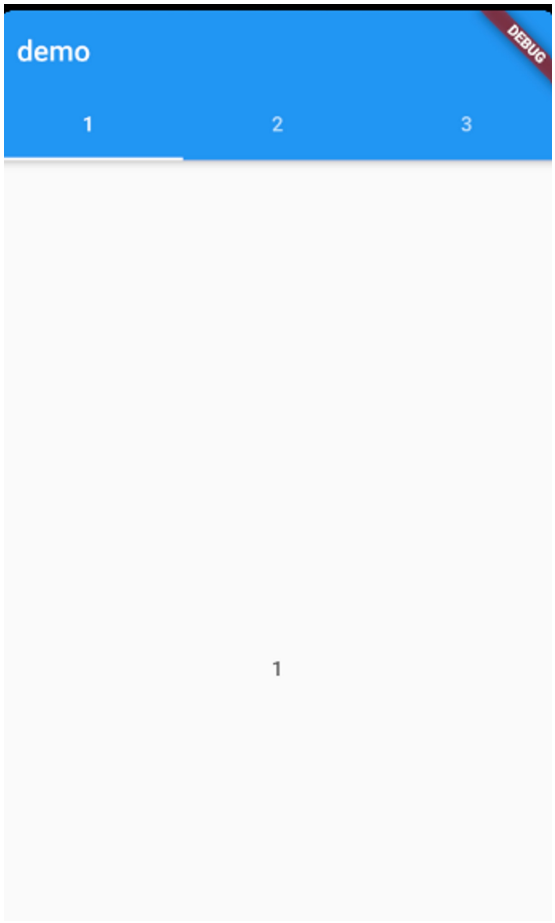
四、 TabBar

TabBar用于显示水平的选项卡，通常需要配合TabBarView和TabBarController。其中TabBarView用于显示与当前所选的选项卡对应的Widget视图；TabBarController是控制器，是TabBarView和TabBar的桥梁。

4.1 DefaultTabController

```
classMyTabControllerextendsStatelessWidget{
constMyTabController({Key?key}):super(key:key);

@override
Widgetbuild(BuildContextcontext){
returnDefaultTabController(length:3,child:Scaffold(
appBar:AppBar(title:Text('demo'),bottom:TabBar(tabs:<Widget>[
Tab(text:'1',),Tab(text:'2',),Tab(text:'3',),
],),),),
body:TabBarView(
children:<Widget>[
Center(child:Text("1"),),
```

```
Center(child:Text("2"),),
Center(child:Text("3"),)
],
),
));
}
}
```



4.2 自定义TabController

如果要切换动画或者监听切换的交互，可以自定义TabController

```
classMySelfTabControllerextendsStatefulWidget{
constMySelfTabController({Key?key}):super(key:key);

@override
_MySelfTabControllerStatecreateState()=>_MySelfTabControllerState();
}

class_MySelfTabControllerStateextendsState<MySelfTabController>
withSingleTickerProviderStateMixin{
lateTabController_tabController;

@override
voidinitState(){
super.initState();
_tabController=newTabController(length:3,vsync:this);
}


@override
voiddispose(){
```

```
_tabController.dispose();
super.dispose();
}

@override
Widgetbuild(BuildContextcontext){
returnScaffold(
appBar:AppBar(
title:Text('demo'),
bottom:TabBar(
tabs:<Widget>[
Tab(text:"1",),Tab(text:"2",),Tab(text:"3",),
],
controller:_tabController,
),
),
body:TabBarView(
controller:_tabController,
children:<Widget>[
Center(child:Text("1"),),Center(child:Text("2"),),Center(child:Text("3"),
),
],
),
);
}
}
```

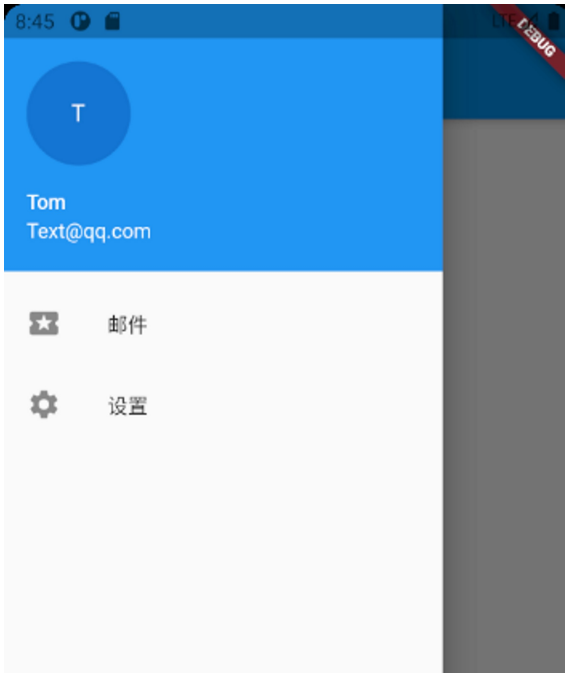
五、Drawer

可以实现拉出推入的效果，抽屉的子项通常是ListView，第一个子项是头部，头部主要有两个Widget可以实现：

1. DrawerHeader：展示基本的信息
2. UserAccountDrawerHeader：展示用户头像、用户名、邮件等信息

```
classMyDrawerextendsStatefulWidget{
constMyDrawer({Key?key}):super(key:key);

@override
_MyDrawerStatecreateState()=>_MyDrawerState();
}

class_MyDrawerStateextendsState<MyDrawer>{
@override
Widgetbuild(BuildContextcontext){
returnScaffold(
appBar:AppBar(title:Text("demo"),),
drawer:_drawer,
);
}
get_drawer=>Drawer(
child:ListView(
padding:EdgeInsets.zero,
children:<Widget>[
UserAccountsDrawerHeader(accountName:Text('Tom'),accountEmail:Text('Text@
qq.com'),currentAccountPicture:CircleAvatar(child:Text("T"),),),
ListTile(leading:Icon(Icons.local_activity),title:Text("邮件"),),
ListTile(leading:Icon(Icons.settings),title:Text("设置"),),
],
```

```
),
);
}
```


8:45
T
Tom
Text@qq.com
邮件
设置