

城市选择页面实现

点击热映页面的左上角的城市，会跳转到城市选择页面：





本节就实现这个功能，这里涉及到两个知识点：

1. 首先就是实现城市选择页面
2. 另一个就是页面跳转，Flutter 实现页面跳转需要用到 Route，也就是路由。

新建文件 CitysWidget.dart

首先实现城市选择页面，新建一个文件 CitysWidget.dart，在里面创建类 CitysWidget，注意，新的页面必须使用 Scaffold 作为 根 Widget：

```
import 'package:flutter/material.dart';

class CitysWidget extends StatefulWidget {
  @override
  State<StatefulWidget> createState() {
    // TODO: implement createState
    return CityWidgetState();
  }
}

class CityWidgetState extends State<CitysWidget> {
  @override
  Widget build(BuildContext context) {
    // TODO: implement build
    return Scaffold(
      body: Text('城市选择'),
    );
  }
}
```

这里返回的是一个 Text，先不急着想实现页面，而是先实现页面跳转功能，当页面跳转功能实现后，再去实现页面，因为这样就可以用 HotReload 实时看到页面的效果。

Route

接下来实现 Route，给 main.dart 的 MaterialApp 增加一个 routes 属性：

```
routes: {
  '/Citys': (context) => CitysWidget(),
},
```

MaterialApp 就变成:

```
return MaterialApp(  
  title: 'Flutter Demo',  
  theme: ThemeData(  
    primarySwatch: Colors.blue,  
  ),  
  home: MyHomePage(title: '豆瓣电影'),  
  routes: {  
    '/Citys': (context) => CitysWidget(),  
  },  
);
```

Navigator

然后给热映页面左上角的城市添加点击事件，调用 Navigator 跳转到城市选择页面：

```
GestureDetector(  
  child: Text(  
    '深圳',  
    style: TextStyle(fontSize: 16),  
  ),  
  onTap: () {  
    Navigator.pushNamed(context, '/Citys');  
  },  
),
```

新建变量来存储当前城市的数据

新建一个 String 变量 curCity 来存储当前城市的数据，并且默认值设为 深圳：

```
class HotWidgetState extends State<HotWidget> {  
  String curCity = '深圳';//用变量来存储当前的城市  
  ...  
}
```

然后其余用 深圳 的地方都替换成 curCity:

```
GestureDetector(  
  child: Text(  
    curCity,  
    style: TextStyle(fontSize: 16),  
  ),  
  onTap: (){  
    Navigator.pushNamed(context, '/Citys');  
  },  
)
```

同时要对 HotMoviesListWidget 进行重构，因为 HotMoviesListWidget 里的城市是写死的，现在需要将城市的数据通过参数传递过去，给 HotMoviesListWidget 的构造函数添加城市的参数，HotMoviesListWidget.dart 会变为：

```

class HotMoviesListWidget extends StatefulWidget
{
  String curCity ;

  HotMoviesListWidget(String city){
    curCity = city;
  }

  ...
}

class HotMoviesListWidgetState extends
State<HotMoviesListWidget> {
  ....

  void _getData() async {
    List<HotMovieData> serverDataList = new
List();
    var response = await http.get(
'https://api.douban.com/v2/movie/in_theaters?
apikey=0b2bdeda43b5688921839c8ecb20399b&city='+wi
dget.curCity+'&start=0&count=10&client=&udid=');
    ...
  }

  ...
}

```

HotWidget.dart 里的

```
HotMoviesListWidget(),
```

会变为：

```
HotMoviesListWidget(curCity),
```

使用 Navigator 传递数据

从热映页面跳转到城市选择页面，需要把当前的城市传过去，使用 `Navigator.pushNamed` 方法的 `arguments` 来传递参数：

```
class HotWidgetState extends State<HotWidget> {
  String curCity = '深圳'; // 用变量来存储当前的城市

  @override
  Widget build(BuildContext context) {
    // TODO: implement build
    return Column(
      ...
      GestureDetector(
        child: Text(
          curCity, // 这里改为变量来表示城市
          style: TextStyle(fontSize: 16),
        ),
        onTap: () {
          Navigator.pushNamed(context,
            '/Citys', arguments: curCity);
        },
      ),
      ...
    );
  }
}
```

传递数据，要用到 `Navigator.pushNamed` 的 `arguments` 参数：

```
Navigator.pushNamed(context, '/Citys', arguments:
  curCity);
```

获取数据

在城市页面获取传递过来的数据需要使用 ModalRoute, ModalRoute 的使用需要 context, 所以这么使用:

```
class CityWidgetState extends State<CitysWidget> {  
  String curCity;  
  
  @override  
  Widget build(BuildContext context) {  
    // TODO: implement build  
    curCity =  
ModalRoute.of(context).settings.arguments;  
  
    return Scaffold...  
  }  
}
```

完善城市选择页面

AppBar

首先要实现 AppBar , 但要自定义一下, 首先背景是白色的, title 是黑色的, 而且 title 不是居中的, 而是居左, 并且返回按钮是绿色的, 所以 AppBar 这么实现:


```

class CityWidgtState extends State<CitysWidget> {
  @override
  Widget build(BuildContext context) {
    // TODO: implement build
    return Scaffold(
      appBar: AppBar(
        iconTheme: IconThemeData(color:
Colors.green),
        title: Text('选择城市', style:
TextStyle(color: Colors.black, fontSize: 16),),
        backgroundColor: Colors.white,
        elevation: 1,
        centerTitle: false,
      ),
      body : ...
    )
  }
}

```

TabBar 和 TabBarView

TabBar 和 TabBarView 我们之前已经很熟悉了，还是使用 DefaultTabController，不一样的是这次选中的是绿色的。TabBarView 里我们只实现国内页面的部分功能，需要用到 GridView，因为只实现部分页面，这次就不写在单独的文件里了。

最后完整的代码如下：

```

import 'package:flutter/material.dart';

class CitysWidget extends StatefulWidget {
  @override
  State<StatefulWidget> createState() {
    // TODO: implement createState
    return CityWidgtState();
  }
}

```

```
}  
}  
  
class CityWidgetState extends State<CitysWidget> {  
  String curCity;  
  
  @override  
  Widget build(BuildContext context) {  
    // TODO: implement build  
    curCity =  
ModalRoute.of(context).settings.arguments;  
    List<String> hotCitys = [  
      '北京',  
      '上海',  
      '广州',  
      '深圳',  
      '成都',  
      '武汉',  
      '杭州',  
      '重庆',  
      '郑州',  
      '南京',  
      '西安',  
      '苏州',  
      '长沙',  
      '天津',  
      '福州'  
    ];  
  
    return Scaffold(  
      backgroundColor: Color(0xE3FFFFFF),  
      appBar: AppBar(  
        iconTheme: IconThemeData(color:
```

```
Colors.green),
    title: Text(
      '选择城市',
      style: TextStyle(color: Colors.black,
fontSize: 16),
    ),
    backgroundColor: Colors.white,
    elevation: 1,
    centerTitle: false,
  ),
  body: DefaultTabController(
    length: 2,
    child: Column(
      children: <Widget>[
        Container(
          color: Colors.white,
          constraints:
BoxConstraints.expand(height: 50),
          child: TabBar(
            unselectedLabelColor:
Colors.black12,
            labelColor: Colors.black87,
            indicatorColor: Colors.green,
            tabs: <Widget>[Tab(text: '国内'), Tab(text: '国外')]),
          ),
        Expanded(
          child: Container(
            child: TabBarView(
              physics:
ClampingScrollPhysics(),
              children: <Widget>[
                Column(
```

```

crossAxisAlignment:
CrossAxisAlignment.start,
children: <Widget>[
  Container(
    height: 50,
    child: TextField(
      cursorColor:
Colors.green,
      decoration:
InputDecoration(
      hintText: '输入城市
名查询',
      hintStyle:
TextStyle(fontSize: 14),
      border:
InputBorder.none,
      prefixIcon: Icon(
        Icons.search,
        color:
Colors.black38,
      ),
      filled: true,
      fillColor:
Colors.white),
    ),
  ),
  Padding(
    padding:
EdgeInsets.only(top: 10, left: 20),
    child: Text(
      'GPS定位城市',
      style:
TextStyle(fontSize: 12, color: Colors.grey),

```

```

        ),
    ),
    Padding(
        padding:
EdgeInsets.only(top: 5, left: 20),
        child: MaterialButton(
            child: Container(
                width: 50,
                child: Row(
                    children:
<Widget>[
                        Icon(
                            Icons.location_on,
                            size: 14,
                            color:
Colors.green,
                        ),
                        Text(curCity)
                    ],
                ),
            ),
            color: Colors.white,
            elevation: 0,
            onPressed: () {},
        ),
    ),
    Padding(
        padding:
EdgeInsets.only(top: 5, left: 20),
        child: Text('热门城市',
            style:

```

```

TextStyle(fontSize: 12, color: Colors.grey)),
    ),
    Expanded(
      flex: 1,
      child: Padding(
        padding:
EdgeInsets.only(left: 20, right: 20, top: 10),
        child:
GridView.builder(
          gridDelegate:
SliverGridDelegateWithFixedCrossAxisCount(
  crossAxisCount: 3,
  childAspectRatio: 2.6,
  mainAxisSpacing: 20,
  crossAxisSpacing: 50),
  itemCount:
hotCitys.length,
  itemBuilder:
(context, index) {
    return
MaterialButton(
      child:
Text(hotCitys[index]),
      color:
Colors.white,
      elevation: 0,
      onPressed: ()

```

```

    {} ,
    );
    },
    ),
    ),
    ],
    ),
    Center(
      child: Text('国外'),
    ),
    ],
    ),
    ),
    ],
    ),
    ),
    );
  }
}

```

运行的效果为:





返回结果

选择城市后，返回选中的结果。

GridView 的 item 点击事件

将 GridView 的 itemBuilder 返回的是 MaterialButton, MaterialButton 有点击事件。

返回结果

在 MaterialButton 的 onPressed 里返回结果：

```
return MaterialButton(  
  child: Text(hotCitys[index]),  
  color: Colors.white,  
  elevation: 0,  
  onPressed: () {  
    Navigator.pop(context, hotCitys[index]);  
  },  
)
```

接受结果

接受结果需要用到 await, 因为 Navigator.pushNamed 的返回结果是 Future 对象, 所以跳转的操作要写成 async 异步函数：

```
void _jumpToCitysWidget() async{  
  var selectCity = await  
Navigator.pushNamed(context, '/Citys', arguments:  
curCity);  
  if(selectCity == null) return;  
  setState(() {  
    curCity = selectCity;  
  });  
}
```

跳转的方法改写为：

```
GestureDetector(  
  child: Text(  
    curCity,  
    style: TextStyle(fontSize: 16),  
  ),  
  onTap: (){  
    _jumpToCitysWidget();  
  },  
)
```

接下来详细讲一下 Flutter 页面跳转及参数传递的用法。

代码目录重构

截止到现在，已经实现了比较多的功能，我们在去看现在的代码，发现文件很多，而且比较杂乱：



```
lib  
├── CitysWidget.dart  
├── HotMovieData.dart  
├── HotMovieItemWidget.dart  
├── HotMoviesListWidget.dart  
├── HotWidget.dart  
├── main.dart  
├── MineWidget.dart  
└── MoviesWidget.dart
```

所以，对代码的目录进行重构，将功能相关的代码放在同一个目录下，改造后的代码为：

◀ lib

▶ citys

▶ hot

▶ mine

▶ movies

≡ main.dart