

# flutter 学习三

---

## 1. 课前复习

---

- App结构和导航组件
  - MaterialApp应用组件
  - Scaffold脚手架组件
  - Flutter主题
- 可滚动组件
  - ListView
  - GridView
  - SingleChildScrollView

## 2. 课堂目标

---

- 路由的使用
- 数据持久化
- 网络控件的使用
- webView的使用
- Provider介绍

## 3. 知识点

---

### 1.路由的使用

移动端通常是通过“屏幕”或者“页面”来显示内容，在Flutter中，页面之间的路由是通过导航器（Navigator）组件来管理的。导航器管理一组路由（Route）对象，并采用堆栈的方法管理。

- `Navigator.of(context).pushNamed('routeName')`:简单的将需要进入的页面push到栈顶，以此来显示当前页面,其参数是一个字符串类型，传入的是页面对应的路由名称；
- `Navigator.of(context).pushReplacementNamed('routeName')`:当前页面在栈中的位置替换成跳转的页面，当新的页面进入后，之前的页面将执行dispose方法。（eg：当前从页面1进入页面2，在页面2使用pushReplacementNamed('/screen3');进入页面3，当进入了页面3后，页面2将执行dispose方法，此时在页面3返回时，会回到页面1。一般使用场景启动页）；
- `Navigator.popAndPushNamed(context,'routeName')`:当前页面pop，然后跳转到置顶页面（将当前路由弹出导航器，并将命名路由推到它的位置。）；
- `Navigator.of(context).pushNamedAndRemoveUntil('routeName',ModalRoute.withName('routeName'))`:将指定的页面加入到路由中，然后将之前的路径移除直到制定的页面为止；
- `Navigator.pushAndRemoveUntil`:跟上述方法作用相同，不同之处在于，上述传递的是路由名称，这个名称需要你在入口处进行路由指定，而这种则无需指定，直接new 出来即可，而且可以传递参数；
- `Navigator.push`：导航到新的页面，该方法将添加Route到路由栈中；

- `Navigator.pop(context,data)`: 返回到前一个页面，返回按钮默认使用的就是这个方法
- `Navigator.popUntil(context,routeName)`: 返回到指定一个页面
- `Navigator.of(context).maybePop()`: 会自动进行判断，如果当前页面pop后，会显示其他页面，不会出现问题，则将执行当前页面的pop操作否则将不执行；

## 2.数据持久化

在开发应用的时候，有时候需要本地存储一个临时数据，这时候可以使用 Flutter 的 `shared_preferences` 插件，此插件在 iOS 上使用 `NSUserDefaults`，在 Android 上使用 `SharedPreferences`，为简单数据提供持久存储

使用方式：

- 添加依赖：

```
dependencies:
  shared_preferences: ^0.4.2
```

在使用出 `import 'package:shared_preferences/shared_preferences.dart';`即可

- 使用方式：
  - `get/setInt(key)`: 查询或设置整型键
  - `get/setBool(key)`: 查询或设置布尔键
  - `get/setDouble(key)`: 查询或设置浮点键
  - `get/setString(key)`: 查询或设置字符串键
  - `get/setStringList(key)`: 查询或设置字符串列表键
  - `getKeys()`: 获取所有键值名
  - `remove(key)`: 删除某个键内容
  - `clear()`: 清除全部内容

```
import 'package:shared_preferences/shared_preferences.dart';

getAsyncData() async {
  // 获取实例
  var prefs = await SharedPreferences.getInstance();
  // 获取存储数据
  var count = prefs.getInt('count') ?? 0 + 1;
  // 设置存储数据
  await prefs.setInt('count', count);
}
```

## 3.Provider

flutter的状态管理方式很多有 `Scoped Model`, `Redux`, `BLoC` 等, 相比之下[Provider](#)使用最为简单易用, 并且这也是官方推荐的使用方式

使用方法:

- Step1: 添加依赖

```
dependencies:  
  provider: ^4.0.4
```

- Step2: 创建model

```
import 'package:provider/provider.dart';  
class Counter with ChangeNotifier {  
  int _count;  
  Counter(this._count);  
  
  void add() {  
    _count++;  
    notifyListeners();  
  }  
  get count => _count;  
}
```

- Step3: 创建ChangeNotifierProvider

```
import 'package:flutter/material.dart';  
import 'package:provider/provider.dart';  
  
main() {  
  runApp(ChangeNotifierProvider<Counter>.value(  
    notifier: Counter(1),  
    child: MyApp(),  
  ));  
}  
  
class MyApp extends StatelessWidget {  
  @override  
  Widget build(BuildContext context) {  
    // TODO: implement build  
    return MaterialApp(  
      title: "Provider",  
      home: HomePage(),  
    );  
  }  
}
```

- Step4: 使用Provider获取数据

```
Provider.of<Counter>(context).count
```

<https://pub.dev/flutter/packages> 插件查询处

[https://javiercbk.github.io/json\\_to\\_dart/](https://javiercbk.github.io/json_to_dart/) map转model

## 4. 作业 && 答疑

---

完成如下界面：

暗号：原来还可以这么玩

- 下拉刷新和上拉加载更多
- 界面跳转（详情界面无需搭建）





## 5. 下节课内容

- 列表的刷新控件实现
- 登录界面开发
- flutter插件开发