Flutter入门之（fluro路由跳转框架)

練碩

最近在学习flutter，看到一个fluro的路由框架不错，但在网上搜了下 很少对fluro的教程文章，所以闲暇之余写了此文章给需要的人懒，顺便再自己巩固下对此框架的运用，语言组织能力不好 勿喷（后续会在详细的对每个方法参数进行讲解）

Fluro

flutter的跳转框架，开发者称之为 **“Flutter最亮，最时髦，最酷的路由器。”**

官方地址 https://pub.dev/packages/fluro

入门

在 **pubspec.yaml**中加入下面代码

```
dependencies:
 fluro: "^1.5.1"
```

开始使用

首先我们定义一个全局的router对象，用于后续的全局调用

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

class Application {
  static Router router;

}
```

然后创建一个抽象类,用于后续的各模块router实现此类进行路由注册初始化

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

abstract class IRouterProvider{

  void initRouter(Router router);
}
```

示例

这是一个登录模块的路由管理示例，我们要实现刚才创建的 **IRouterProvider** 抽象类，实现其 **initRouter()** 方法进行每个页面的路由注册

```
import 'package:fluro/fluro.dart';
import 'package:flutter_deer/routers/router_init.dart';
```

```
import 'login_page.dart';
import 'register_page.dart';


class LoginRouter implements IRouterProvider{

  static String loginPage = "/login";
  static String registerPage = "/login/register";

  @override
  void initRouter(Router router) {
    router.define(loginPage, handler: Handler(handlerFunc:
(_, params) => Login()));
    router.define(registerPage, handler:
Handler(handlerFunc: (_, params) => Register()));
  }

}
```

接下来我们需要创建一个总的router管理类

```
    router.define(home, handler: Handler(
      handlerFunc: (BuildContext context, Map<String,
List<String>> params) => Home()));
```

此方法中第一个参数为注册地址，第二个handler用于界面的路由注册

```

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

```dart
import 'package:flutter_deer/home/404.dart';
import 'package:flutter_deer/login/login_router.dart';
import 'package:flutter_deer/routers/router_init.dart';

import 'package:flutter_deer/home/home_page.dart';
import 'package:flutter_deer/home/webview_page.dart';

class Routes {

  static String home = "/home";
  static String webViewPage = "/webview";

  //子router管理集合
  static List<IRouterProvider> _listRouter = [];

  static void configureRoutes(Router router) {

    /// 指定路由跳转错误返回页
    router.notFoundHandler = Handler(
      handlerFunc: (BuildContext context, Map<String,
List<String>> params) {
        debugPrint("未找到目标页");
        return WidgetNotFound();
      });

      //主界面可以在此类中进行注册（可定义传参）
    router.define(home, handler: Handler(
      handlerFunc: (BuildContext context, Map<String,
List<String>> params) => Home()));

    //一些共用的界面也可以在此处注册
    router.define(webViewPage, handler:
Handler(handlerFunc: (_, params){
      String title = params['title']?.first;
      String url = params['url']?.first;
```

```
      return WebViewPage(title: title, url: url);
    }));

    //每次初始化前 先清除集合 以免重复添加
    _listRouter.clear();
    /// 各自路由由各自模块管理，统一在此添加初始化
    _listRouter.add(LoginRouter());

    /// 初始化路由 循环遍历取出每个子router进行初始化操作
    _listRouter.forEach((routerProvider){
      routerProvider.initRouter(router);
    });
  }
}
```

接下来封装跳转工具类

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

import 'application.dart';
import 'routers.dart';

/// fluro的路由跳转工具类
class NavigatorUtils {

  //不需要页面返回值的跳转
  static push(BuildContext context, String path,
      {bool replace = false, bool clearStack = false}) {
    FocusScope.of(context).requestFocus(new FocusNode());
    Application.router.navigateTo(context, path, replace:
replace, clearStack: clearStack, transition:
TransitionType.native);
```

```
  }

  //需要页面返回值的跳转
  static pushResult(BuildContext context, String path,
Function(Object) function,
      {bool replace = false, bool clearStack = false}) {
    FocusScope.of(context).requestFocus(new FocusNode());
    Application.router.navigateTo(context, path, replace:
replace, clearStack: clearStack, transition:
TransitionType.native).then((result){
      // 页面返回result为null
      if (result == null){
        return;
      }
      function(result);
    }).catchError((error) {
      print("$error");
    });
  }

  /// 返回
  static void goBack(BuildContext context) {
    FocusScope.of(context).requestFocus(new FocusNode());
    Navigator.pop(context);
  }

  /// 带参数返回
  static void goBackWithParams(BuildContext context,
result) {
    FocusScope.of(context).requestFocus(new FocusNode());
    Navigator.pop(context, result);
  }

}
```

使用

首先我们需要在 main.dart 中进行初始化

```
class MyApp extends StatelessWidget {

  MyApp()  {
    //创建一个Router对象
    final router = Router();
    //配置Routes注册管理
    Routes.configureRoutes(router);
    //将生成的router给全局化
    Application.router = router;
  }

  @override
  Widget build(BuildContext context) {
    return OKToast(
      child: MaterialApp(
        title: 'fluro',
        theme: ThemeData(
          primaryColor: Colours.app_main,
        ),
        home: HomePage(),
        //生成路由
        onGenerateRoute: Application.router.generator,
      ),
    );
  }
}
```

无返回值跳转 第二个参数是目标地址，第三个是是否关闭当前界面 （非必传 默认为 false true的话是关闭 从内存中释放）
需要传参时 第二个参数用以下方式传（此为固定的格式）

```
'${Routes.webViewPage}?param1=$
{Uri.encodeComponent(content1)}&param2=$
{Uri.encodeComponent(content2)}'
```

问号前的是目标地址 **${Routes.webViewPage}**

问号后的是需要传的参数 **param=$ {Uri.encodeComponent(content)}**（fluro 不支持传中文,需转换）
param为参数名称，content为需要传递的参数

```
  //不需要穿参数的
  _goLogin(){
    NavigatorUtils.push(context, LoginRouter.loginPage,
replace: true);
  }

  //需要传参数的
  _goLogin(){
    NavigatorUtils.push(context,
  '${Routes.webViewPage}?param1=$
{Uri.encodeComponent(content1)}&param2=$
{Uri.encodeComponent(content2)}'
```

```
    , replace: true);
  }
```

有返回值跳转

```
NavigatorUtils.pushResult(context,
AccountRouter.citySelectPage, (result){
                                  setState(() {
                                    //result是返回的结果
                                    TestModel model = result;
                                    _name = model.name;
                                  });
                                });
```

返回上一级

```
NavigatorUtils.goBack(context);
```

带参数返回上一级 (result是返回结果)

```
NavigatorUtils.goBackWithParams(context, result);
```

参考自 https://github.com/simplezhli/flutter_deer

有兴趣学习 flutter的可以看下这个项目