Flutter 路由插件 -- fluro

不不不紧张

The brightest, hippest, coolest router for Flutter.

fluro 是 flutter 开发中比较常见路由插件

插件地址：fluro | Flutter Package (pub.dev)

官方给出的例子

1

```
final router = FluroRouter();
```

2

```
var usersHandler = Handler(handlerFunc: (BuildContext
context, Map<String, dynamic> params) {
  return UsersScreen(params["id"][0]);
});

void defineRoutes(FluroRouter router) {
  router.define("/users/:id", handler: usersHandler);

  // it is also possible to define the route transition to
use
  // router.define("users/:id", handler: usersHandler,
transitionType: TransitionType.inFromLeft);
}
```

```
router.navigateTo(context, "/users/1234", transition:
TransitionType.fadeIn);
```

```
/// Push a route with custom RouteSettings if you don't
want to use path params
FluroRouter.appRouter.navigateTo(
  context,
  'home',
  routeSettings: RouteSettings(
    arguments: MyArgumentsDataClass('foo!'),
  ),
);

/// Extract the arguments using
[BuildContext.settings.arguments] or
[BuildContext.arguments] for short
var homeHandler = Handler(
  handlerFunc: (context, params) {
    final args = context.settings.arguments as
MyArgumentsDataClass;

    return HomeComponent(args);
  },
);
```

从例子中可以看出使用方式并不复杂，先创建`router`对象，之后为每个需要弹出的页面声明Handler和标识并在`router`对象中注册，在需要弹出页面时使用`router`对象调用navigateTo方法即可。

实际开发中的用法

虽然上面说到fluro使用方法不复杂，但是在实际开发中，出于提高代码可扩展性，降低耦合度等目的,一般不会以这么直白的方式使用fluro。

我这边使用的方式是将注册页面和推出页面分割，创建不同的工具类来实现对应功能的代码。

举例：

假设目前项目结构为两个模块:模块A,模块B,两个模块分别包含两个页面，既A1,A2, B1,B2。现在我们使用fluro实现这个项目的页面跳转功能

注册页面实现方式：
为了方便维护代码，每个模块都单独实现一个用来注册页面的工具类，既`RouterA`,`RouterB`。
`RouterA`：

```
class RouterA implements IRouterProvider {
 // 声明标识
  static String A1 = "/A1";
  static String A2 = "/A2";

// initRouter方法中实现注册
```

```
  @override
  void initRouter(FluroRouter router) {
    //给外部传入的fluro的`router`对象注册页面
    router.define(A1,
        handler: Handler(handlerFunc: (_, params) => const
A1()));
    router.define(A2,
        handler: Handler(handlerFunc: (_, params) => const
A2()));
  }
}
```

`RouterB`同理：

```
class RouterB implements IRouterProvider {
 // 声明标识
  static String B1 = "/B1";
  static String B2 = "/B2";

// initRouter方法中实现注册
  @override
  void initRouter(FluroRouter router) {
 //给外部传入的fluro的`router`对象注册页面
    router.define(B1,
        handler: Handler(handlerFunc: (_, params) => const
B1()));
    router.define(B2,
        handler: Handler(handlerFunc: (_, params) => const
B2()));
  }
}
```

之后创建工具类`Routes`,它的作用是将各个模块的注册工具类

与 fluro 的 `router` 对象关联

```
class Routes {
  static final List<IRouterProvider> _routerList = [];

  static void configureRoutes(FluroRouter router) {
    _routerList.clear();
    _routerList.add(RouterA());
    _routerList.add(RouterB());

    /// 使用各个模块的工具类初始化路由
    for (IRouterProvider routerProvider in _routerList) {
      routerProvider.initRouter(router);
    }
  }
}
```

在项目入口声明 `router` 对象，并将 `router` 传入工具类 `Routes` 进行页面注册

```
final router = FluroRouter();
//注册
Routes.configureRoutes(router);
```

注册页面工具完成

页面跳转工具类实现方式：

页面跳转工具类相对于注册功能内容较少，实现全局`router`对象的获取，并封装页面弹出收回的方法。

`router`对象获取工具类`Application`

```
class Application {
  static FluroRouter router;
}
```

在创建`router`对象时给 Application.router 赋值

```
final router = FluroRouter();
//注册
Routes.configureRoutes(router);
//赋值
Application.router = router;
```

跳转工具类`NavigatorTool`

```
/// fluro的路由跳转工具类
class NavigatorTool {
  static push(BuildContext context, String path,
   ) {
    Application.router.navigateTo(context, path);
  }

  /// 返回
  static void goBack(BuildContext context) {
```

```
    Navigator.pop(context);
  }
}
```

页面跳转工具完成

业务代码：

1. 跳转至 A1 页面

```
NavigatorTool.push(context, RouterA.A1);
```

2. 跳转至 A2 页面

```
NavigatorTool.push(context, RouterA.A2);
```

3.跳 转至 B1 页面

```
NavigatorTool.push(context, RouterB.B1);
```

4. 跳转至 B1 页面

```
NavigatorTool.push(context, RouterB.B2);
```

欢迎指正