主题	负责人	日期
Flutter 路由	杜德汉	2010/07/09

备注：

路由配置文件路径：“flutter_prometheum/lib/business/main/config/app_router.dart”

路由配置（三步骤）

```
///路由配置
///1、配置跳转不需传参的route
routes: AppRouter.routers,
///2、配置跳转需要参数的route
onGenerateRoute: AppRouter.onGenerateRoute,
///3、配置路由跳转出错页面
onUnknownRoute: AppRouter.onUnknownRoute,
///4、入口
home: _home(context, widget.routeName),
```

1、routers (配置跳转不需传参的路由)

- **routers定义**

```
/// 由定义可以看出是一个<string,WidgetBuilder>类型的Map
final Map<String, WidgetBuilder> routes;
///
typedef WidgetBuilder = Widget Function(BuildContext context);
```

- **实现routers**
  - 先给页面定义别名，统一在RoutePath中定义
  - 在AppRouter中实现routers（如下代码）

    注：此处定义的Map内的WidgetBuilder方法并未被执行，只有当页面跳转时才会根据key找到对应的方法进行调用

```
class AppRouter {
  ///不带参数跳转路由配置
  ///如需要自定义动画需在 onGenerateRoute 方法中实现
  static final Map<String, WidgetBuilder> routers = {
    RoutePath.marketPage: (context) => MarketPage(),
    RoutePath.accountBusiness: (context) => AccountBusiness(),
    RoutePath.accountPage: (context) => AccountPage(),
    RoutePath.detailContentPage: (context) => DetailContentPage(),
```

```
    RoutePath.orderPage: (context) => OrderPage(),
    RoutePath.orderLoadingPage: (context) => OrderLoadingPage(),
    RoutePath.orderResultPage: (context) => OrderPageResult(),
    RoutePath.orderDetailPage: (context) => OrderDetailPage(),
    RoutePath.positionPage: (context) => PositionPage(),
  };
  /// 带参数跳转路由配置
  static RouteFactory onGenerateRoute = (settings) {...}
  ///跳转出错页面
  static RouteFactory onUnknownRoute = (setting) {...}
}
/// 申明路由名称
/// 后续跳转都使用此别名，方便统一修改，同时防止写错
/// 建议以dart文件名命名
class RoutePath {
  static const String marketPage = '/market_page';
  static const String accountOrderType = '/account_order_type';
  static const String accountBusiness = '/account_page';
  static const String accountPage = '/account_order_detail';
  static const String detailContentPage = '/account_position_detail';
  static const String orderPage = '/order_page';
  static const String orderLoadingPage = '/order_page_loading';
  static const String orderResultPage = '/order_page_result';
  static const String orderDetailPage = '/order_detail_page';
  static const String positionPage = '/position_page';
  static const String tokenDetails = '/token_details';
}
```

2、onGenerateRoute（配置跳转需要参数的route）

- onGenerateRoute的定义

```
/// onGenerateRoute 是一个接受一个RouteSettings参数返回route的函数
final RouteFactory onGenerateRoute;
typedef RouteFactory = Route<dynamic> Function(RouteSettings settings);
```

- 实现onGenerateRoute
  - 在AppRouter内定义onGenerateRoute函数，此处只需要在函数内创建页面路由即可
  - settings.name：页面名称，可根据名称判断是要跳转哪一个页面
  - settings.arguments：页面跳转需要传递的参数
  - **注：不带参路由跳转时，如果需要自定义动画则无需在routers中定义，直接在onGenerateRoute函数内实现，如下示例**

```
/// 带参数跳转路由配置
static RouteFactory onGenerateRoute = (settings) {
/// token_details 页面
    if (settings.name == RoutePath.tokenDetails) {
```

```
      Map<String, dynamic> map = settings.arguments ?? Map<String,
dynamic>();
      return MaterialPageRoute(builder: (ctx) {
        return TokenDetails(map['value']);
      });
    }

  ///如需添加转场动画示例：
    if (settings.name == RoutePath.positionPage) {
      return PageRouteBuilder(
        transitionDuration: Duration(seconds: 1),
        pageBuilder: (ctx, animation1, animation2) {
          /// 此处只是简单的添加一个透明度动画
          return FadeTransition(
            opacity: animation1,
            child: PositionPage(),
          );
        },
      );
    }
}
```

3、onUnknownRoute（配置跳转需要参数的route）

- 设置与带参数路由跳转类似，只是在onUnknownRoute函数内实现

```
static RouteFactory onUnknownRoute = (setting) {
    return MaterialPageRoute(builder: (ctx) {
        return UnknownPage();
    });
    //return null;
  };
```

使用

- push跳转
  - 不带参：

    ```
    Navigator.of(context).pushNamed(RoutePath.PositionPage);
    ```

  - 带参数：

    ```
    Navigator.of(context).pushNamed(RoutePath.tokenDetails,arguments:
    {'value':model.instrumentId});
    ```

- pop
  - 返回到上一层

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

  - 跨层pop

    ```
    Navigator.of(context).popUntil(ModalRoute.withName(RoutePath.PositionPage));
    ```

代码总览

- Main.dart

```
///路由配置
///1、配置跳转不需传参的route
routes: AppRouter.routers,
///2、配置跳转需要参数的route
onGenerateRoute: AppRouter.onGenerateRoute,
///3、配置路由跳转出错页面
onUnknownRoute: AppRouter.onUnknownRoute,

home: _home(context, widget.routeName),
```

- app_router.dart

```
class AppRouter {
  ///不带参数跳转路由配置
  ///如需要自定义动画需在 onGenerateRoute 方法中实现
  static final Map<String, WidgetBuilder> routers = {
    RoutePath.marketPage: (context) => MarketPage(),
    RoutePath.accountBusiness: (context) => AccountBusiness(),
    RoutePath.accountPage: (context) => AccountPage(),
    RoutePath.detailContentPage: (context) => DetailContentPage(),
    RoutePath.orderPage: (context) => OrderPage(),
    RoutePath.orderLoadingPage: (context) => OrderLoadingPage(),
    RoutePath.orderResultPage: (context) => OrderPageResult(),
    RoutePath.orderDetailPage: (context) => OrderDetailPage(),
    RoutePath.positionPage: (context) => PositionPage(),
  };

  ///带参数跳转路由配置
  static RouteFactory onGenerateRoute = (settings) {

    ///如需添加转场动画示例:
```

```
    /*if (settings.name == RoutePath.positionPage) {
      return PageRouteBuilder(
        transitionDuration: Duration(seconds: 1),
        pageBuilder: (ctx, animation1, animation2) {
          return FadeTransition(
            opacity: animation1,
            child: PositionPage(),
          );
        },
      );
    }*/

    /// account_order_type页面
    if (settings.name == RoutePath.accountOrderType) {
      Map<String, dynamic> map = settings.arguments ?? Map<String,
dynamic>();
      return MaterialPageRoute(builder: (ctx) {
        CommonOrderTypeModel model = map['value'];
        return AccountOrderType(orderTypeModel: model);
      });
    }
    /// token_details 页面
    if (settings.name == RoutePath.tokenDetails) {
      Map<String, dynamic> map = settings.arguments ?? Map<String,
dynamic>();
      return MaterialPageRoute(builder: (ctx) {
        return TokenDetails(map['value']);
      });
    }
    return null;
  };

  ///跳转出错页面
  static RouteFactory onUnknownRoute = (setting) {
//      return MaterialPageRoute(builder: (ctx) {
//          return UnknownPage();
//      });
    return null;
  };
}

///申明路由名称
class RoutePath {
  static const String marketPage = '/market_page';
  static const String accountOrderType = '/account_order_type';
  static const String accountBusiness = '/account_page';
  static const String accountPage = '/account_order_detail';
  static const String detailContentPage = '/account_position_detail';
  static const String orderPage = '/order_page';
```

```
  static const String orderLoadingPage = '/order_page_loading';
  static const String orderResultPage = '/order_page_result';
  static const String orderDetailPage = '/order_detail_page';
  static const String positionPage = '/position_page';
  static const String tokenDetails = '/token_details';
}
```