Navigation基本使用

版本号:2019-03-24(14:30)

简介

1、Navigation是什么?

> 1. 用于简化导航的架构组件
> 2. 能控制导航的流程

2、Navigation具有的优点

1. 自动处理fragment transaction
2. 正确的自动装卸
3. 动画和过渡
4. Deep linking
5. 实现导航UI模式(navigation drawer、bottom nav)
6. 导航时能正确传递信息
7. AS提供可视化编辑工具

依赖添加

3、Navigation的使用需要添加依赖

> build.gradle

```
/*=========================================
 * Navigation
 *=========================================*/
implementation 'android.arch.navigation:navigation-fragment:1.0.0'
implementation 'android.arch.navigation:navigation-ui:1.0.0'
```

核心组件

1、Navtion组件包含三个核心部分

1. Navigation Graph(新的XML资源)-包含所有navigation相关的信息
2. NavHostFragment(layout xml view)-特殊的小部件，需要添加到布局中。
3. NavController-记录navigation graph之中当前的位置，能交换NavHostFragment中目标内容

2、NavController

1. 指明你想要去哪里
2. 展示NavHostFragment中合适的目的地

Destinations

3、Navigation component提出了新的概念：目的地

1. destination目的地是指你想要去的地方，通常是activity or fragment
2. destination是可以直接使用的，也可以选择 自定义目的地类型

Navigation Graph

4、Navigation graph是一种新的资源类型

> 1. 定义了所有跳转页面可能的路径
> 2. 展示了一个给定destination，可以跳转的所有其他的destination(比如首页，可以跳转到我的页面、搜索页面、详情页面)

使用

3、Navigation graph的使用

> 1-新建资源目录navigation: `res/navigation`
> 2-新建一个Navigation Resource File: `res/navigation/home_navigation.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<navigation xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto" android:id="@+id/home_navigation">

</navigation>
```

> 3-进入Design模式，可以选择 `add a new destination` 创建新的 `Destination`
> 4-新建MainActivity、HomeFragment、SearchFragment等等
> 5-

4、新建MainActivity、HomeFragment、SearchFragment等等

> MainActivity

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.hao.architecture.MainActivity">

</android.support.constraint.ConstraintLayout>
```

> HomeFragment

```
public class HomeFragment extends Fragment {
    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @
        View view = inflater.inflate(R.layout.fragment_home_layout, container, false);
        return view;
    }
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/search_textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="搜索页面"
        android:textSize="20dp"
        app:layout_constraintBottom_toTopOf="@+id/my_textview"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"/>

    <TextView
        android:id="@+id/my_textview"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="我的页面"
        android:textSize="20dp"
        app:layout_constraintBottom_toTopOf="@+id/detail_txt"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/search_textView" />

    <TextView
        android:id="@+id/detail_txt"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="详情页面"
        android:textSize="20dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/my_textview"/>
</android.support.constraint.ConstraintLayout>
```

5、在MainActivity中配置Fragment的容器

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.hao.architecture.MainActivity">


    <fragment
        android:id="@+id/main_fragment_container"
        android:name="androidx.navigation.fragment.NavHostFragment"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:defaultNavHost="true"
        app:navGraph="@navigation/home_navigation"/>

</android.support.constraint.ConstraintLayout>
```

属性	作用
`android:name="xxx.NavHostFragment"`	作为容器必须要是NavHostFragment
`app:defaultNavHost="true"`	将系统返回键和该 `NavHostFragment` 连接起来
`app:navGraph="@navigation/home_navigation"`	指明用哪个 `navGraph` 进行导航处理

解析

1、Navigation Graph解析

```xml
<?xml version="1.0" encoding="utf-8"?>
<navigation xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/home_navigation"
    app:startDestination="@id/mainActivity">

    <activity
        android:id="@+id/mainActivity"
        android:name="com.hao.architecture.MainActivity"
        android:label="activity_main"
        tools:layout="@layout/activity_main" />
    <fragment
        android:id="@+id/homeFragment"
        android:name="com.hao.architecture.navigation.HomeFragment"
        android:label="fragment_home_layout"
        tools:layout="@layout/fragment_home_layout" >
        <action
            android:id="@+id/action_homeFragment_to_searchFragment"
            app:destination="@id/searchFragment" />
        <action
            android:id="@+id/action_homeFragment_to_searchFragment2"
            app:destination="@id/searchFragment" />
        <argument
            android:name="type"
            app:argType="integer"
            android:defaultValue="-1" />
    </fragment>
    <fragment
        android:id="@+id/searchFragment"
        android:name="com.hao.architecture.navigation.SearchFragment"
        android:label="fragment_search_layout"
        tools:layout="@layout/fragment_search_layout" >
        <argument
            android:name="type"
            app:argType="integer"
            android:defaultValue="-1" />
    </fragment>
</navigation>
```

> 1. 包含三个重要部分: activity、fragment、 `app:startDestination="@id/mainActivity"`
> 2. activity、fragment中有包含两个内容 `action`-指明跳转的目标，`argument`-指明参数
> 3. activity、fragment具有三个主要的属性 `id`、`name`-指明所属的类、`label`-指明布局、`deep link`

fragment

activity

action

1、页面的跳转可以通过 `actions` 实现

> 1. 可以设置 过渡动画
> 2. 可以设置 参数
> 3. 可以指定 回退栈的行为
> 4. 可以使用插件 `safe args` 进行导航

2、首页跳转到搜索页面的navigation

```
    <action
        android:id="@+id/action_homeFragment_to_searchFragment"
// 1. 目标页面
        app:destination="@id/searchFragment"
// 2. 动画
        app:enterAnim="@anim/nav_default_enter_anim"
        app:exitAnim="@anim/nav_default_exit_anim"            app:popEnterAnim="@anim/nav_d
        app:popExitAnim="@anim/nav_default_pop_exit_anim"
// 3. 是否采用singleTop模式
        app:launchSingleTop="true"
// 4. 如果弹出，返回到哪个页面
        app:popUpTo="@+id/homeFragment"
// 5. popUpToInclusive
        app:popUpToInclusive="true" />
```

SingleTop

1、Activity启动模式的singleTop是栈顶复用模式，距离讲解。

> 1. 栈中Activity: ABCD，采用singleTop模式。
> 2. 打开D: ABCD，不会是ABCDD
> 3. 打开B: ABCDB

argument

NavController(页面跳转)

1、NavController能帮助进行页面跳转

> 1. 如点击某个Button，触发navigate命令，然后在 `NavHostFragment` 中进行了页面跳转
> 2. 通过 `Navigation.findNavController()` 能获取到 `NavController`

2、对MainActivity的返回按键进行委托，根据栈中Fragment的数量进行处理

> MainActivity.java

```
// public class MainActivity extends AppCompatActivity
    /**====================================
     * 将返回按钮的点击事件，委托出去。
     *   1. 栈上有多个Fragment，会依次回退
     *==================================*/
    @Override
    public boolean onSupportNavigateUp() {
        return Navigation.findNavController(this, R.id.main_fragment_container).navigateUp();
    }
```

3、Fragment中进行点击跳转

```
        view.findViewById(R.id.search_textView).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Navigation.findNavController(view).navigate(R.id.action_homeFragment_to_searchF
            }
        });
```

4、点击跳转的另一种形式: createNavigateOnClickListener

> 1. `Navigation.createNavigateOnClickListener(@IdRes destId: int, bundle: Bundle)` 能直接构造出一个 `OnClickListener` ，并且传入参数

```
view.findViewById(R.id.search_textView)
    .setOnClickListener(Navigation.createNavigateOnClickListener(R.id.action_homeFragment_to_se
```

Transition过渡动画

1、Navigation支持使用过渡动画

> 1. 默认的过渡动画可以通过 `NavOptions` 进行覆盖
> 2. NavOptions使用建造者模式，允许进行各种设置

NavOptions自定义过渡动画

action中添加

2、自定义过渡动画

> 1- `res/anim` 中创建动画，如 `fade_in.xml` 等
> 2- `slide_left.xml`

```xml
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">

    <translate android:fromXDelta="100%" android:toXDelta="0%"
               android:fromYDelta="0%" android:toYDelta="0%"
               android:duration="700"/>
</set>
```

> 3- action标签 中配置动画

```xml
        <action
            android:id="@+id/action_homeFragment_to_searchFragment"
            app:destination="@id/searchFragment"
            app:enterAnim="@anim/slide_left"/>
```

代码中添加

3、navigation进行页面跳转时携带NavOptions参数

```
// 1. 构造自定义动画的NavOptions
NavOptions navOptions = new NavOptions.Builder()
                        .setEnterAnim(R.anim.slide_left)
                        .build();

// 2. 携带NavOptions进行页面跳转
Navigation.findNavController(view).navigate(R.id.action_homeFragment_to_searchFragment, null,
      navOptions);
```

View动画

1、 res/anim 中定义的动画属于View动画

参数传递

1、Navigation具有一个Gradle插件 safe args

> 1. 该插件能自动生成页面跳转的参数相对应的类
> 2. 通过该类能避免传统的参数传递可能会笔误的问题

```
String username = arguments.getString("usernameKey")


String username = args.getUsername();
```

2、依赖添加

```
dependencies {
    classpath "android.arch.navigation:navigation-safe-args-gradle-plugin:1.0.0"
}
```

```
dependencies {
    classpath 'com.android.tools.build:gradle:3.3.2'
    classpath "android.arch.navigation:navigation-safe-args-gradle-plugin:1.0.0"
}
```

3、安全参数插件的使用

> 插件会根据actions中设置好的参数，生成对应的参数类
> 传入参数：

```
// 1. 利用插件为SearchFragment产生的参数类，构造出Bundle
Bundle bundle = new SearchFragmentArgs.Builder()
                        .setType(1)
                        .build()
                        .toBundle();

// 2. 携带NavOptions进行页面跳转
Navigation.findNavController(view).navigate(R.id.action_homeFragment_to_searchFragment, bundle,
      navOptions);
```

> 目标获取参数：

```
        if(getArguments() != null){
            // 获取到参数
            SearchFragmentArgs args = SearchFragmentArgs.fromBundle(getArguments());
        }
```

Deep

通知栏、Widget跳转到App页面

Web跳转到App页面

参考资料

1. Google实验室: Navigation
2. Android官方架构组件Navigation：大巧不工的Fragment管理框架