西南交通大學
Southwest Jiaotong University

移动商务应用开发

第5章 Fragment与Navigation

学习内容

5.1 认识Fragment

5.2 Fragment的生命周期

5.3 认识Navigation

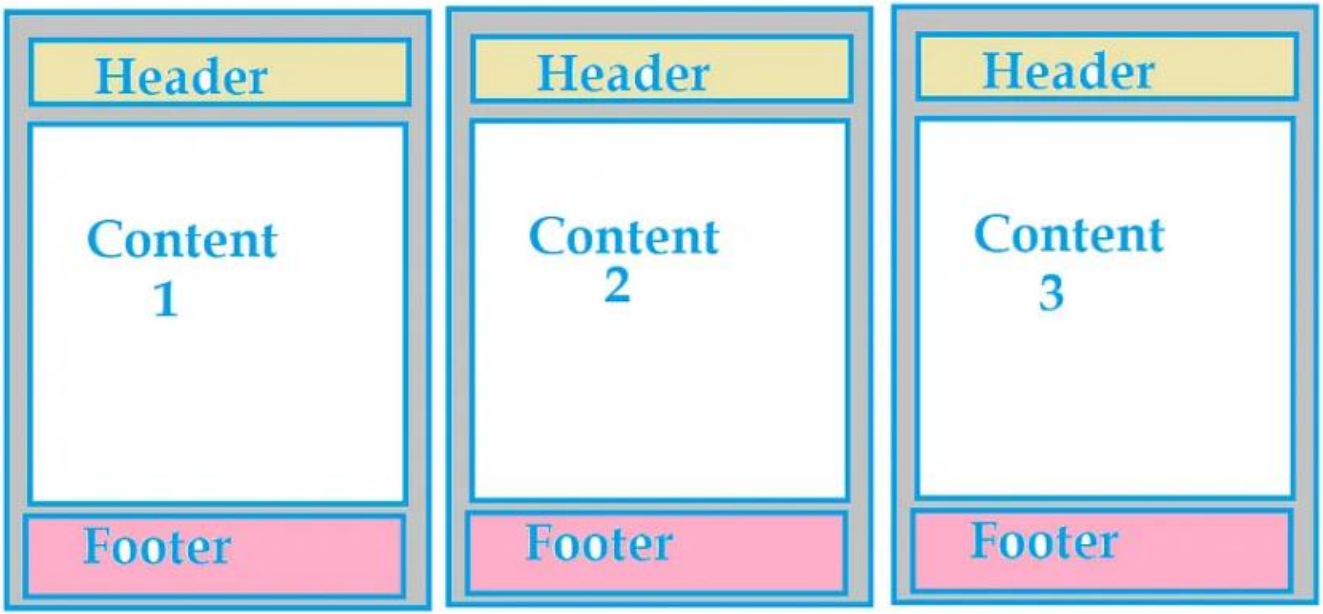
5.4 Navigation实现页面切换

5.5 Navigation实现底部导航

主要内容

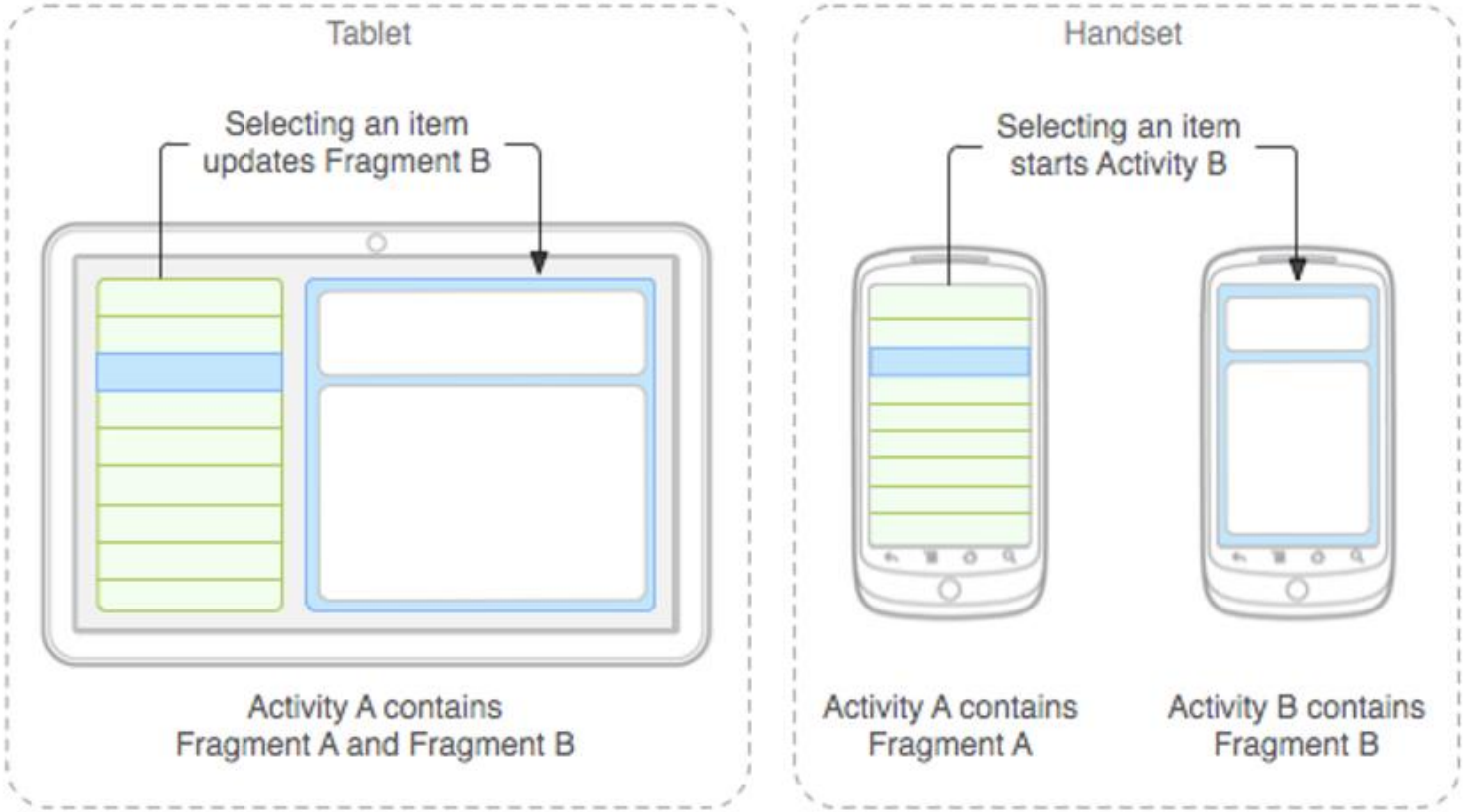
认识Fragment

- 场景一：多个页面具有相同的内容



认识Fragment

- 场景二：手机和平板有不同的排版要求



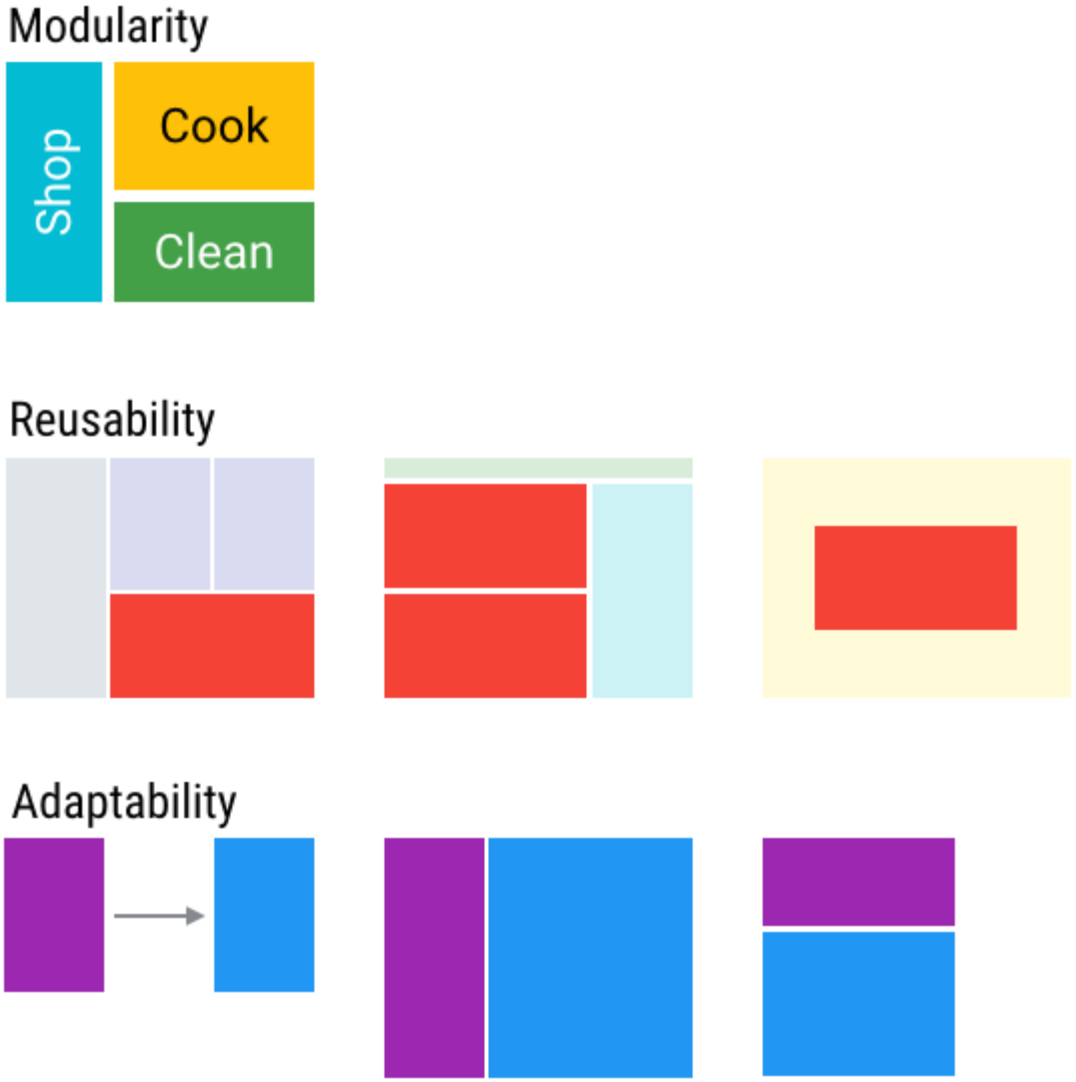
认识Fragment

- Fragment是一种嵌入到Activity中的UI片段。
- 主要作用：
  - Fragment允许将Activity提供的用户界面划分为多个模块。
- 特点：
  - Fragment依附于Activity，不能独立存在。
  - 一个Activity中可以包含并展示多个Fragment。
  - 一个Fragment可以被多个Activity重复使用。
  - Fragment有自己的布局和生命周期，并能和用户交互。
  - 可以在Activity运行时动态地添加或删除Fragment。

认识Fragment

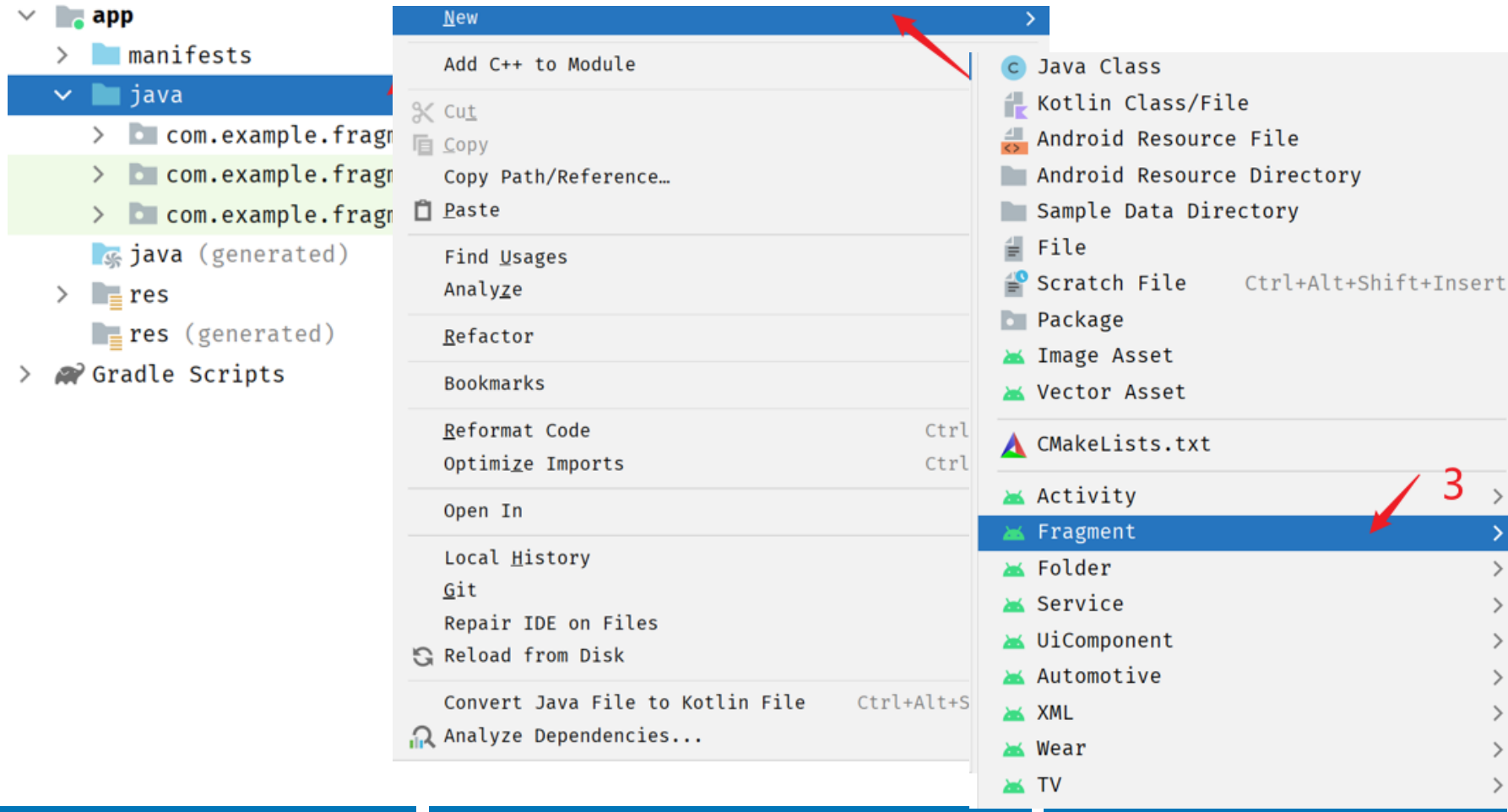
- Fragment的优势：
  - **模块化**。Fragment允许将用户界面划分为更小的模块，每个模块负责管理自己的布局和逻辑，使得代码更易于管理和维护。
  - **可重用性**。一个Fragment可以被不同的 Activity重用，减少了代码量。
  - **灵活性**。一方面，体现在可以在Activity运行时动态地添加或删除Fragment；另一方面，体现在可以根据屏幕尺寸和方向灵活布局。
  - **轻量化**。Fragment被认为是轻量级的Activity，功能类似，更省内存。

认识Fragment



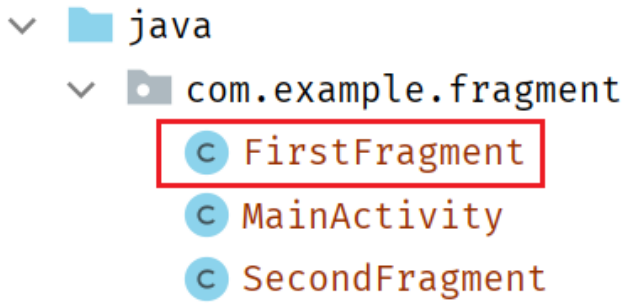
创建Fragment

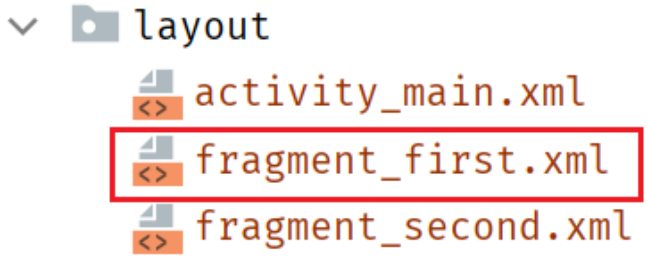
- 用户创建Fragment的步骤：



创建Fragment

- Android Studio创建Fragment的步骤（自动完成）：
  1. 生成了一个继承Fragment的**java文件**
  2. 生成了一个对应的**布局文件**





学习内容

5.1 认识Fragment

5.2 Fragment的生命周期

5.3 认识Navigation

5.4 Navigation实现页面切换

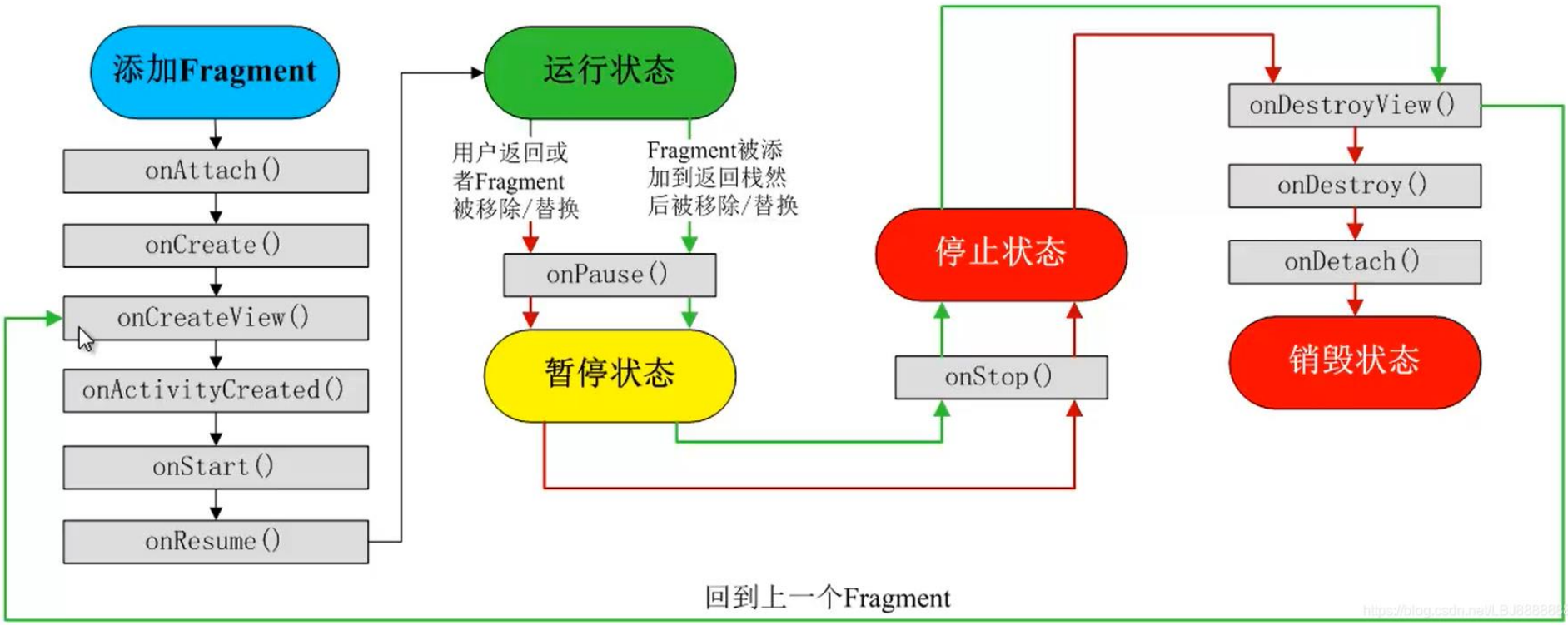
5.5 Navigation实现底部导航

主要内容

Fragment的生命周期

- Fragment具有自己的生命周期。
- Fragment的生命周期受到宿主Activity生命周期的影响。
  - 当Activity暂停时，它所包含的Fragment也会暂停；当Activity停止时，它所包含的Fragment也会停止；当Activity被销毁时，它所包含的Fragment也会被销毁。
- Fragment也具有自己的生命周期的回调方法。

Fragment的生命周期回调方法



Fragment的生命周期回调方法

- 通过重写回调方法和输出日志观察回调方法的调用时机
  - 当Activity加载Fragment时：

    onAttach -> onCreate -> onCreateView -> onActivityCreated -> onStart ->onResume
  - 当宿主Activity进入后台时（例如：点击Home键，切换到另一个Activity）：

    onPause->onStop
  - 当宿主Activity重新回到前台时：

    onStart -> onResume

Fragment的生命周期回调方法

- 当Fragment进入后台（切换到另一个Fragment）时：

  onPause -> onStop -> onDestoryView

- 当Fragment重新回到前台时：

  onCreateView -> onActivityCreated -> onStart -> onResume

- 当Fragment被销毁时：

  onPause -> onStop -> onDestoryView -> onDestory -> onDetach

学习内容

5.1 认识Fragment

5.2 Fragment的生命周期

5.3 认识Navigation

5.4 Navigation实现页面切换

5.5 Navigation实现底部导航

主要内容

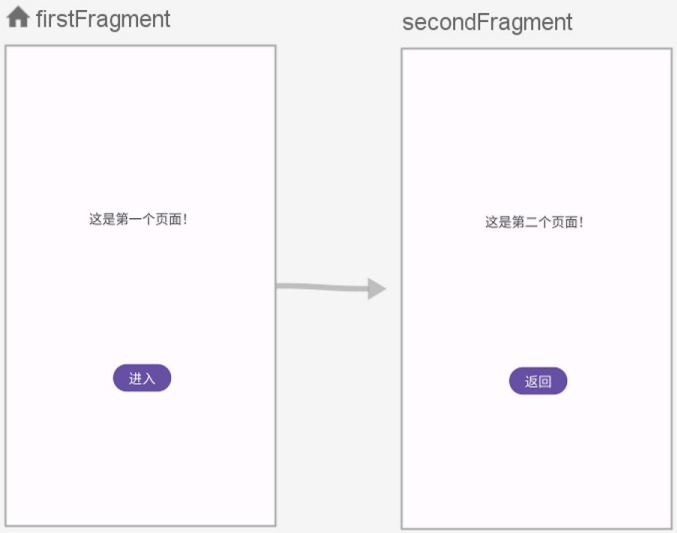
认识Navigation

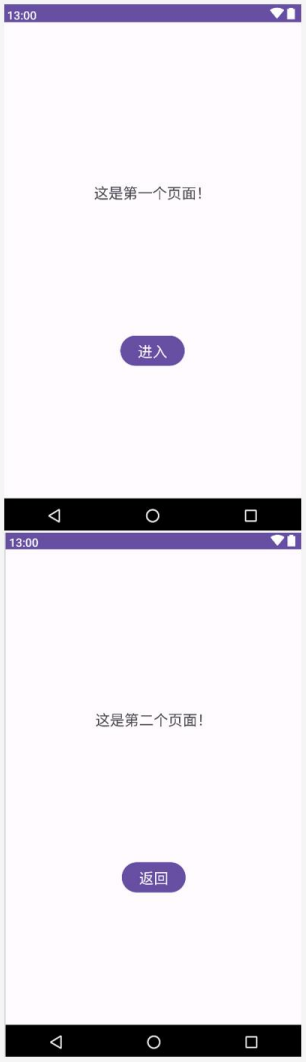
- Navigation是Android Jetpack的一部分，用于管理多个页面间的导航。
- Navigation把页面当作一个个目的地（Destination），各目的地之间形成一张导航图（NavGraph），由导航控制器（NavController）来统一管理页面间的跳转。
  - 此处的“页面”主要指Fragment。

认识Navigation

- Navigation中的三个关键组成部分：
  - **NavHost**：导航宿主，用来显示当前目的地的空白容器，NavHost将Activity和Fragment关联起来了。
  - **NavGraph**：导航图，包含了所有目的地及目的地之间的导航路径。
  - **NavController**：导航控制器，管理目的地的切换。
- 当你想要切换页面的时候，使用NavController对象，告诉它你想要去NavGraph中的哪个页面，NavController会将相关的页面展示在NavHostFragment中。

认识Navigation

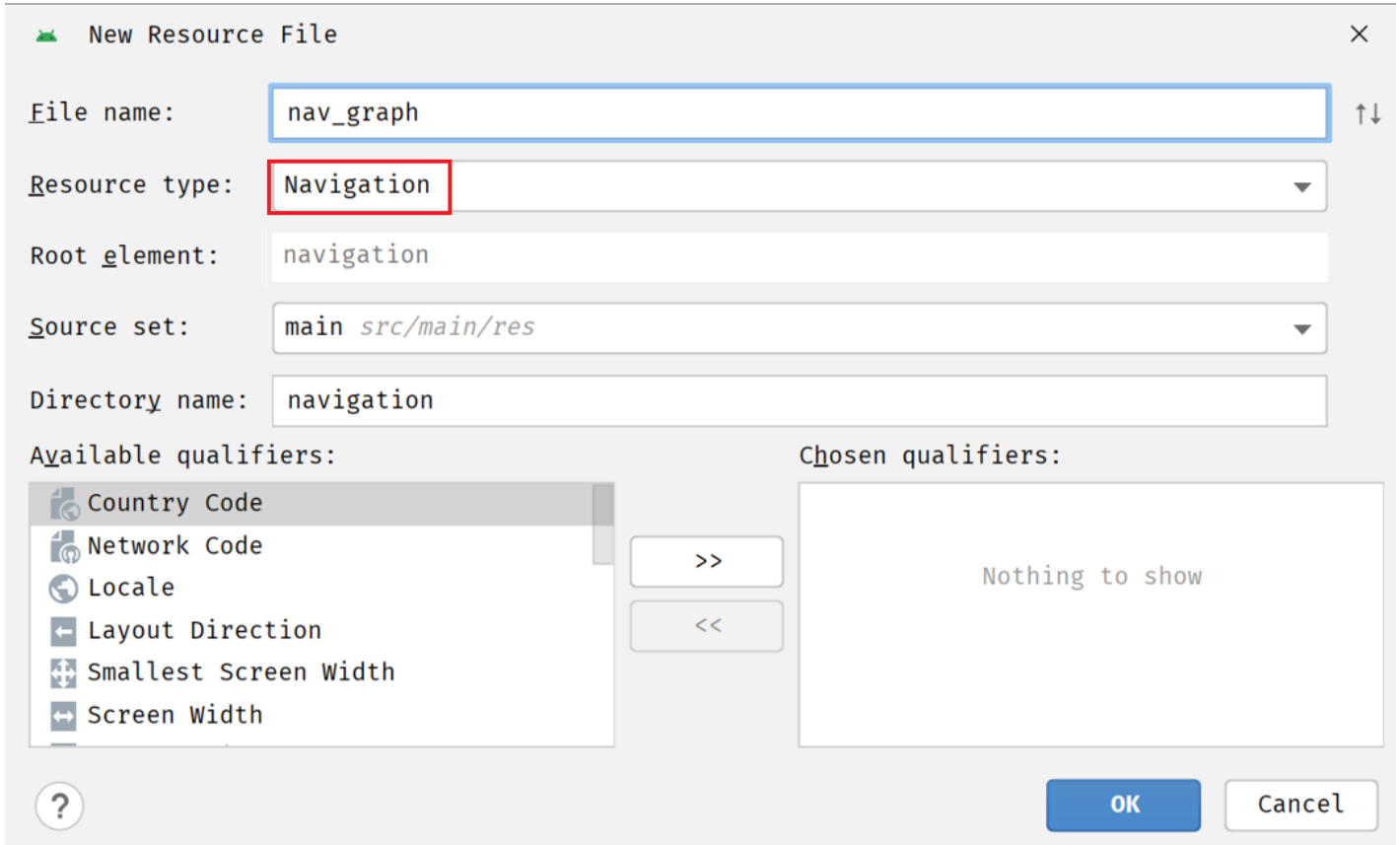




- NavHost
- NavGraph
- NavController
- Fragment

创建NavGraph

- 选中项目资源文件夹 res 右击 >> New >> Android Resource File，创建Navigation类型的资源



创建NavGraph

- 在NavGraph中添加fragments和action

NavGraph代码说明

```
<navigation xmlns:android="http://schemas.android.com/apk/res/androic
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/nav_graph"
    app:startDestination="@id/firstFragment">

    <fragment
        android:id="@+id/firstFragment"
        android:name="com.example.fragment.FirstFragment"
        android:label="fragment_first"
        tools:layout="@layout/fragment_first" >
        <action
            android:id="@+id/action_firstFragment_to_secondFragment"
            app:destination="@id/secondFragment" />
    </fragment>
    <fragment
        android:id="@+id/secondFragment"
        android:name="com.example.fragment.SecondFragment"
        android:label="fragment_second"
        tools:layout="@layout/fragment_second" />
</navigation>
```

NavGraph代码说明

```
<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/nav_graph"
    app:startDestination="@id/firstFragment">
```

navigation的属性:

- `android:id`：navigation的id，会被NavHost引用
- `app:startDestination`：加载的第一个页面

NavGraph代码说明

```
<fragment
    android:id="@+id/secondFragment"
    android:name="com.example.fragment.SecondFragment"
    android:label="fragment_second"
    tools:layout="@layout/fragment_second" />
```

fragment的属性:

- `android:id` : fragment的id，会被NavController引用
- `android:name` : 指明了navGraph中的目的地所对应的Fragment
- `android:label` : fragment的标签，可用于修改ActionBar上显示的文本

NavGraph代码说明

```
<action
    android:id="@+id/action_firstFragment_to_secondFragment"
    app:destination="@id/secondFragment" />
```

action的属性:

- `android:id`：action的id，会被NavController引用，用于执行fragment间的跳转
- `app:destination`：指明了跳转的目的地

创建NavHost

- 在 activity_main.xml布局中添加NavHostFragment
- 设置NavHostFragment的defaultNavHost 和navGraph属性。

```
<androidx.fragment.app.FragmentContainerView
    android:id="@+id/fragmentContainerView"
    android:name="androidx.navigation.fragment.NavHostFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:defaultNavHost = "true"
    app:navGraph = "@navigation/nav_graph"
```

NavHost代码说明

```
<androidx.fragment.app.FragmentContainerView
    android:id="@+id/fragmentContainerView"
    android:name="androidx.navigation.fragment.NavHostFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:defaultNavHost = "true"
    app:navGraph = "@navigation/nav_graph"
```

FragmentContainerView的属性:

- `android:id` : NavHostFragment的id,可用以获取NavController
- `android:name` : 固定使用
- `app:defaultNavHost` : 设置点击Back键时是返回上一个Fragment，还是退出Activity
- `app:navGraph` : 设置容器中使用的NavGraph

获取NavController

- 方法一（常见于Activity中）

```
NavHostFragment navHostFragment = (NavHostFragment)
getSupportFragmentManager().findFragmentById(R.id.fragmentContainerView);
NavController navController = navHostFragment.getNavController();
```

- 方法二（常见于Activity中）

```
NavController navController = Navigation.findNavController(this,
R.id.fragmentContainerView);
```

- 方法三（常见于Fragment中）

```
NavController navController = Navigation.findNavController(view);
```

使用NavController导航

- 页面导航

```
// 从firstFragment到secondFragment
navController.navigate(R.id.action_firstFragment_to_secondFragment);
```

- 退回

```
navController.navigateUp();
```

导航时传入数据

- 在传出数据的Fragment中：

```
String str = editText.getText().toString().trim();
// 通过Bundle传入数据
Bundle bundle = new Bundle();
bundle.putString("username", str);
// 获取NavController
NavController navController = Navigation.findNavController(v);
// 导航时传入数据
navController.navigate(R.id.action_firstFragment_to_secondFragment,
bundle);
```

导航时传入数据

- 在接收数据的Fragment中：

```
// 从参数中获取Bundle
Bundle bundle = getArguments();
// 解析数据
String str = bundle.getString("username");
textView.setText(str);
```

学习内容

5.1 认识Fragment

5.2 Fragment的生命周期

5.3 认识Navigation

5.4 Navigation实现页面切换

5.5 Navigation实现底部导航

主要内容

Navigation实现页面切换

Navigation实现页面切换

Navigation实现页面切换的步骤：

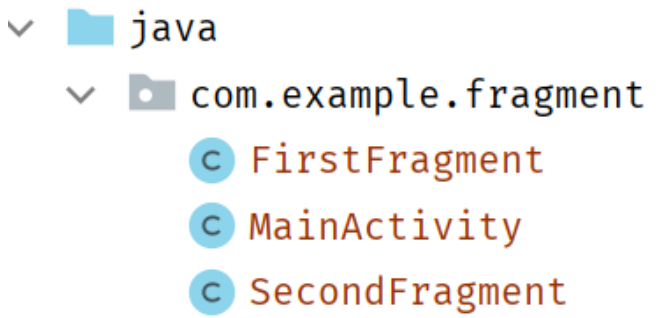
1. 添加Navigation相关依赖
2. 创建Fragment及其布局文件
3. 创建Navigation Graph
4. 在Activity的布局文件中创建NavHostFragment
5. 在Fragment中获取NavController实现页面切换

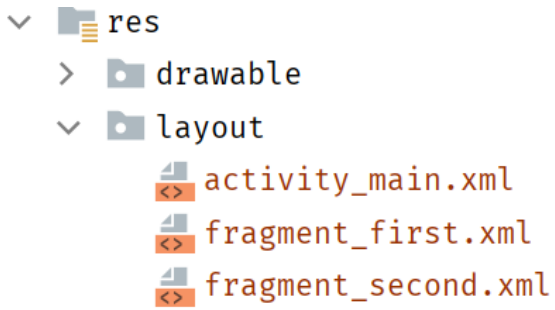
Navigation实现页面切换

1. 在build.gradle中添加Navigation相关依赖

```
implementation("androidx.navigation:navigation-fragment:2.7.6")
implementation("androidx.navigation:navigation-ui:2.7.6")
```

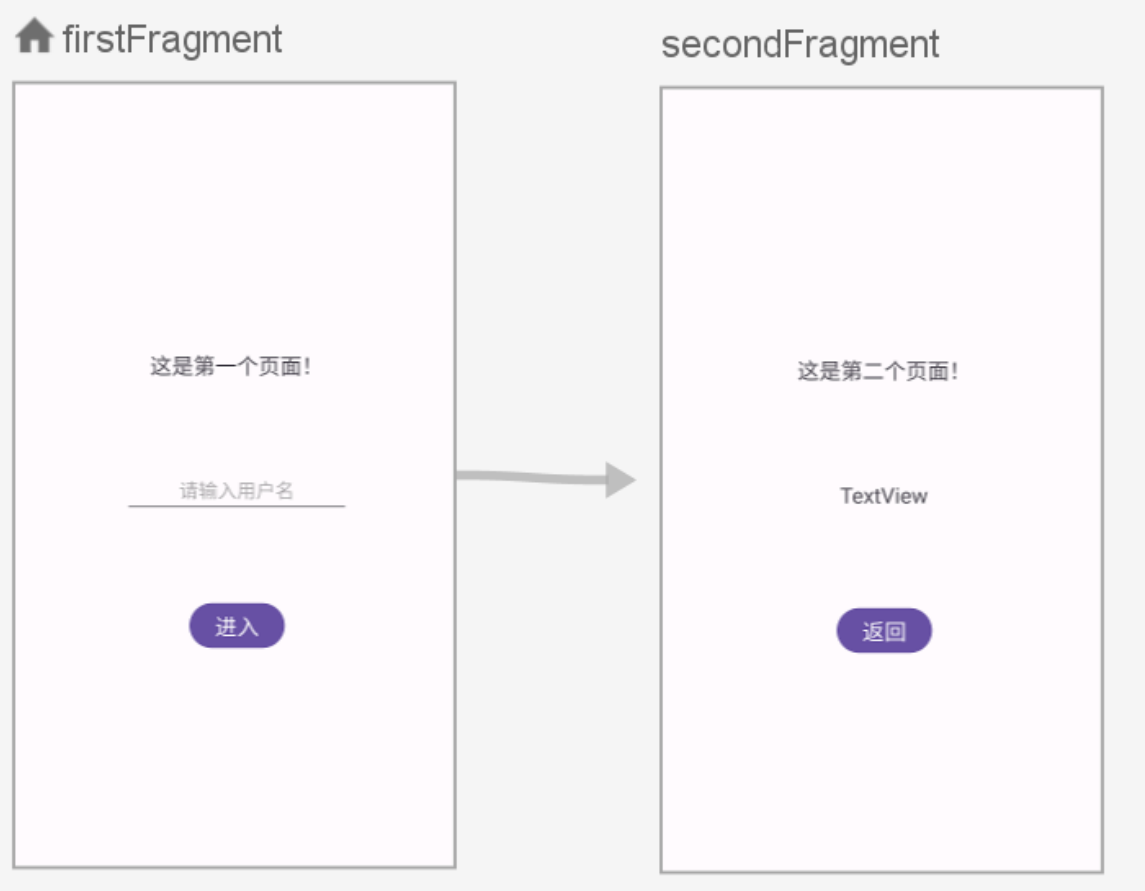
2. 创建Fragment及其布局文件





Navigation实现页面切换

3. 创建Navigation Graph



Navigation实现页面切换

4. 在activity_mail.xml中添加NavHostFragment，并设置其defaultNavHost和navGraph属性

```
<fragment
    android:id="@+id/fragmentContainerView"
    android:name="androidx.navigation.fragment.NavHostFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:defaultNavHost = "true"
    app:navGraph = "@navigation/nav_graph"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

Navigation实现页面切换

- 在FirstFragment中获取NavController实现页面切换

```
buttonEnter.setOnClickListener(v → {
  // 获取输入内容
  if (TextUtils.isEmpty(editText.getText())) {
    Toast.makeText(view.getContext(), "输入不能为空", Toast.LENGTH_SHORT).show();
  } else {
    String str = editText.getText().toString().trim();
    // 通过Bundle传入数据
    Bundle bundle = new Bundle();
    bundle.putString("username", str);
    // 获取NavController实现页面切换
    NavController navController = Navigation.findNavController(v);
    navController.navigate(R.id.action_firstFragment_to_secondFragment, bundle);
  }
});
```

Navigation实现页面切换

- 在SecondFragment中获取NavController实现页面切换

```
// 接收数据
Bundle bundle = getArguments();
String str = bundle.getString("username");
textView.setText(str);
buttonBack.setOnClickListener(v → {
    // 获取NavController实现页面导航
    NavController navController = Navigation.findNavController(v);
    navController.navigateUp();
});
```

学习内容

5.1 认识Fragment

5.2 Fragment的生命周期

5.3 认识Navigation

5.4 Navigation实现页面切换

5.5 Navigation实现底部导航

主要内容

Navigation实现底部导航

Navigation实现底部导航

Navigation实现底部导航的步骤：

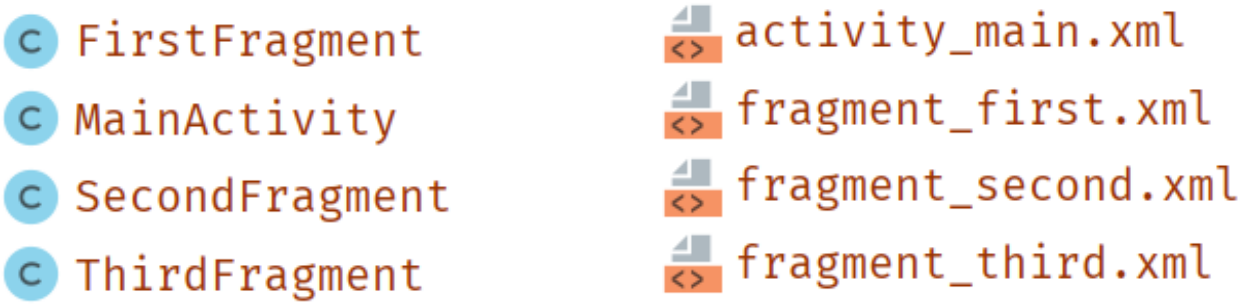
1. 添加Navigation相关依赖
2. 创建Fragment及其布局文件
3. 创建Navigation Graph
4. 创建底部菜单
5. 在Activity的布局文件中添加NavHostFragment和BottomNavigationView
6. 在Activity中获取NavController和BottomNavigationView实现页面切换

Navigation实现页面切换

1. 在build.gradle中添加Navigation相关依赖

```
implementation("androidx.navigation:navigation-fragment:2.7.6")
implementation("androidx.navigation:navigation-ui:2.7.6")
```

2. 创建Fragment及其布局文件



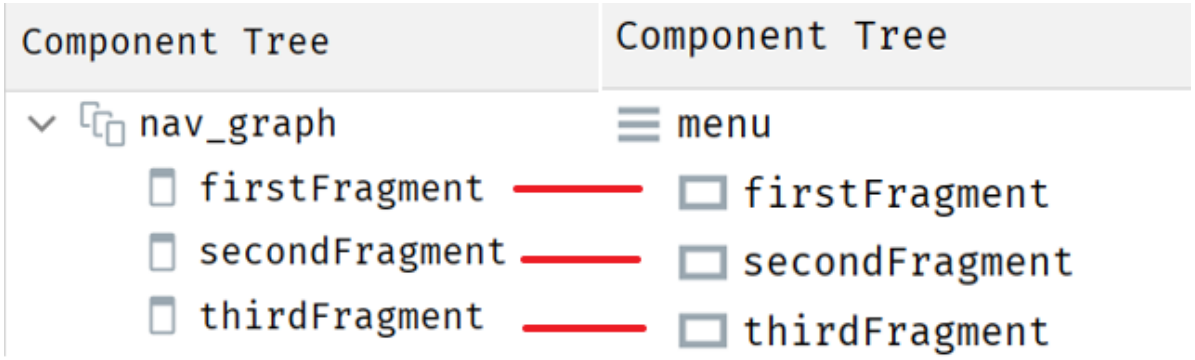
Navigation实现页面切换

3. 创建Navigation Graph

Navigation实现页面切换

4. 创建底部菜单

- 菜单项的id要和navGraph中fragment的id保持一致



Navigation实现页面切换

5. 在activity_mail.xml中添加NavHostFragment和BottomNavigationView

```
<androidx.fragment.app.FragmentContainerView
    android:id="@+id/fragmentContainerView"
    android:name="androidx.navigation.fragment.NavHostFragment"
    android:layout_width="0dp"
    android:layout_height="0dp"
    app:navGraph="@navigation/nav_graph"
```

Navigation实现页面切换

5. 在activity_mail.xml中添加NavHostFragment和BottomNavigationView

```
<com.google.android.material.bottomnavigation.BottomNavigationView
    android:id="@+id/bottomNavigationView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:menu="@menu/bottom_menu"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent" />
```

Navigation实现页面切换

6. 在Activity中获取NavController和BottomNavigationView实现底部导航

```
// 获取NavController和BottomNavigationView
NavHostFragment navHostFragment =
    (NavHostFragment)
getSupportFragmentManager().findFragmentById(R.id.fragmentContainerView
);
NavController navController = navHostFragment.getNavController();
BottomNavigationView bottomNavigationView =
findViewById(R.id.bottomNavigationView);
// 将NavController与bottomNavigationView绑定
NavigationUI.setupWithNavController(bottomNavigationView,
navController);
```

练习

- 掌握创建Fragment的方法
  - 观察Fragment的生命周期
- 掌握Navigation管理Fragment导航的方法
  - 实现页面跳转
  - 实现底部导航
  - 实现数据传递

课程总结

5.1 认识Fragment

5.2 Fragment的生命周期

5.3 认识Navigation

5.4 Navigation实现页面切换

5.5 Navigation实现底部导航

作业

本章作业

- 根据课堂练习要求，利用Navigation和Fragment完成页面导航 + 底部导航
  - 视频记录运行全过程
  - 第一个页面文本内容改成：学号 + 姓名

作业提交方式

- 视频，文件名：学号+姓名+第5章作业
- 邮件给助教，主题：学号+姓名+第5章作业