Fragment的创建和使用

原创 xyzso1z 2020-07-26 15:07:20 58 收藏 编辑 版权

分类专栏: Android 文章标签: android fragment

一、Fragment添加至Activity

- 由于 `Fragment` 作为 `Activity` 一部分，所以 `Fragment` 的使用一般是添加到 `Activity` 中
- 将 `Fragment` 添加到 `Activity` 中一般有2种方法：

1. 在 `Activity` 的 `layout.xml` 布局文件中静态添加
2. 在 `Activity` 的 `.java` 文件中动态添加

方法 1：在 `Activity` 的 `layout.xml` 布局中静态添加

- `Fragment` 的布局文件
  `fragment_test.xml`

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical">


    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="测试Fragment" />

</LinearLayout>
```

- `Fragment` 的 `.java` 文件
  要创建 `Fragment`，应该继承 `Fragment` 类，然后替换关键生命周期方法插入应用逻辑，创建方式类似于 `Activity` 类。创建 `Fragment` 时一个区别是我们必须使用 `onCreateView()` 回调定义布局。
  `ExampleFragment.java`

```
import androidx.fragment.app.Fragment;
public class ExampleFragment extends Fragment{
   @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
       // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_test, container, false);
  }
}
```

- `Activity` 的布局文件
  `activity_statically_add_fragment.xml`

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

```
5
6     <fragment
7         android:id="@+id/fragment"
8         android:name="com.xyz.fragment.activity.ExampleFragment"
9         android:layout_width="match_parent"
10        android:layout_height="match_parent" />
11
12    </LinearLayout>
```

- Activity 的 .java 文件
  StaticallyAddFragmentActivity.java

```
1  public class StaticallyAddFragmentActivity extends AppCompatActivity{
2    @Override
3    protected void onCreate(Bundle savedInstanceState) {
4        super.onCreate(savedInstanceState);
5        setContentView(R.layout.activity_statically_add_fragment);
6    }
7  }
```

方法 2：在运行时为 Activity 添加 Fragment

要执行添加或移除 Fragment 等事务，必须使用 FragmentManager 创建一个 FragmentTransaction ，后者将提供添加、移除、替换 Fragment 以及执行其他 Fragment 事务所需的 API 。
如果 Activity 允许移除和替换 Fragment ，应在 Activity 的 onCreate() 方法执行期间为其添加初始 Fragment 。

- 设置 Fragment 的布局文件

```
1  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
2    android:layout_width="match_parent"
3    android:layout_height="match_parent"
4    android:gravity="center"
5    android:orientation="vertical">
6
7    <TextView
8        android:layout_width="wrap_content"
9        android:layout_height="wrap_content"
10       android:text="测试Fragment" />
11
12   </LinearLayout>
```

- Fragement 的代码

```
1  public class ExampleFragment extends Fragment{
2    @Override
3    public View onCreateView(LayoutInflater inflater, ViewGroup container,
4                             Bundle savedInstanceState) {
5        return inflater.inflate(R.layout.fragment_test, container, false);
6
7    }
8  }
```

- 在 Activity 的布局文件中定义1占位容器(FrameLayout)
  可以动态在 Activity 中添加不同的 Fragment

```
<?xml version="1.0" encoding="utf-8"?>
   <FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/fragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

- 在 Activity 的 .java 文件动态添加 Fragment

```
public class DynamicAddFragmentActivity extends BaseActivity {
  @Override
  protected void onCreate(@Nullable Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_dynamic_add_fragment);

      //1. 获取FragmentManager
      FragmentManager fragmentManager = getSupportFragmentManager();

      //2. 获取FragmentTransaction
      FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();

      //3. 创建需要添加的Fragment
      ExampleFragment fragment = new ExampleFragment();

      //4. 动态添加Fragment(即将创建的fragment添加到Activity布局文件中定义的FrameLayout)
      fragmentTransaction.add(R.id.fragment, fragment);

      fragmentTransaction.commit();
  }
}
```

注： FragmentActivity 是支持库中提供的特殊 Activity ， 用于处理早于 API 级别11的系统版本上的 Fragment 。 如果使用 V7 appompat 库，那么 Activity 应继承 AppCompatActivity ， 它是 FragmentActivity 的子类。

二、替换 Fragment

替换 Fragment 的过程与动态添加 Fragment 类似，但需要调用 replace() 方法，而非 add() 。
需要注意的是，当执行替换或移除 Fragment 等 Fragment 事务时，通常最好让用户能够回退并“撤销”更改。要让用户回退所执行的 Fragment 事务，我们必须先调用 addToBackStack() ,然后再提交 FragmentTransaction 。

> 注意：当我们移除或替换 Fragment 并向返回堆栈添加相应事务时，系统会停止(而非销毁)移除的 Fragment 。如果用户执行回退操作进行 Fragment 恢复，该 Fragment 将重新启动。如果不向返回堆栈添加相应事务，则系统会在移除或替换 Fragment 时将其销毁。

```
    // Create fragment and give it an argument specifying the article it should show
    ArticleFragment newFragment = new ArticleFragment();
    Bundle args = new Bundle();
    args.putInt(ArticleFragment.ARG_POSITION, position);
    newFragment.setArguments(args);

    FragmentTransaction transaction = getSupportFragmentManager().beginTransaction();

```

```
9
10      // Replace whatever is in the fragment_container view with this fragment,
11      // and add the transaction to the back stack so the user can navigate back
12      transaction.replace(R.id.fragment_container, newFragment);
13      transaction.addToBackStack(null);
14
15      // Commit the transaction
        transaction.commit();
```

`addToBackStack()` 方法采用一个可选的字符串参数，该参数会为事务指定一个唯一的名称。

三、资源

- 源码
- Fragment生命周期