

Fragment does **not** inherit from the **View** class, so it cannot be displayed like different widgets such as buttons. Like menus, you need to create a separate fragment.xml and then load it into the main page using the inflate method.

First, create two fragments containing the same elements.

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

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="fragment_1" />

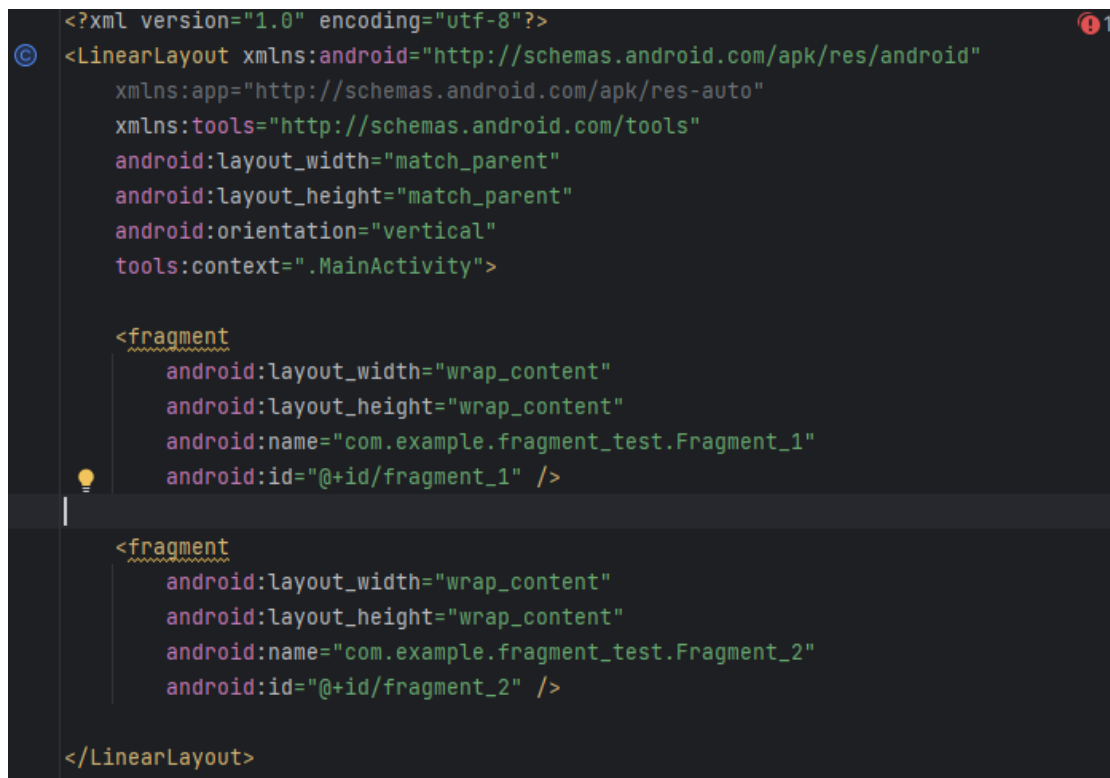
</LinearLayout>
```

In the Java code, establish two corresponding classes.

```
public class Fragment_1 extends Fragment {

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_1, container, attachToRoot: false);
    }
}
```

Add two fragments in the main\_activity.xml, making sure to declare their **class names** (fully qualified) and **IDs**, to fill the fragment into them.



```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <fragment
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:name="com.example.fragment_test.Fragment_1"
        android:id="@+id/fragment_1" />

    <fragment
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:name="com.example.fragment_test.Fragment_2"
        android:id="@+id/fragment_2" />

</LinearLayout>

```

When launching the emulator, you can see two fragments, which are no different from ordinary controls.

Note that this way of adding is **a syntactic sugar**, and the most common way is to dynamically add fragments through the **fragment manager**.

We delete the fragment in main\_activity.xml and add two buttons and a framelayout to contain the fragment.

```

<FrameLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/fragment_container" />

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Button_1"
    android:id="@+id/button_1"/>

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Button_2"
    android:id="@+id/button_2"/>

```

In Java, by clicking on the corresponding button, you can dynamically switch the fragment elements that appear in the framelayout.

```

Button button_1 = findViewById(R.id.button_1);
Button button_2 = findViewById(R.id.button_2);

button_1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        FragmentManager fragmentManager = getSupportFragmentManager();
        FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
        fragmentTransaction.replace(R.id.fragment_container, new Fragment_1());
        fragmentTransaction.commit();});

button_2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        FragmentManager fragmentManager = getSupportFragmentManager();
        FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
        fragmentTransaction.replace(R.id.fragment_container, new Fragment_2());
        fragmentTransaction.commit();}
});

```

Here, we first instantiate a **fragmentManager** to manage the lifecycle of the fragment, and then create a **fragmentTransaction** to add, delete, or replace fragments. Both the add and replace methods accept two parameters, one is the container ID that holds the

fragment, and the other is the instance of the fragment. Finally, use the commit method to submit the fragmentTransaction to the fragmentManager.