● Fragment

▶ API level 11 (Android 3.0)에서 추가된 뷰이며, 표준 라이브러리에서 제공하지만 API 하위 호환성 문제로 androidx의 클래스 (androidx.fragment.app.Fragment)를 이용한다. Fragment는 "액티비티처럼 이용할 수 있는 뷰"로, 액티비티의 생명주기를 그대로 따르는 뷰이다. 그러므로 액티비티에서 작성했던 코드를 Fragment 클래스 내에서 작성할 수 있고, 이런 이유로 Fragment를 "액티비티처럼 이용할 수 있는 뷰"라고 표현한다. 액티비티의 하나의 클래스가 너무 길고 복잡하게 작성되는 것을 막아준다. 액티비티에서 구현해야 하는 화면과 업무 로직 등으로 Fragment 클래스로 추상화하여 작성하고, 액티비티에서는 이렇게 만들어진 Fragment를 다른 뷰처럼 화면에 출력하는 구조이다.

→ 예를 들면 스마트폰에서의 화면 전환은 액티비티 A에서 데이터 목록을 보여주고 사용자가 항목을 선택하여 이벤트가 발생할 시에 엑티비티B를 실행해 상세보기 내용을 제공한다는 식이면, 이 앱은 액티비티 2개를 만들어 구현할 수 있다. 각각의 화면과 화면에서 발생하는 사용자 이벤트, 그리고 이벤트에 따든 업무 로직을 하나의 액티비티 클래스로 추상화하여 두 개의 클래스로 작성할 수 있는 것이다. 하지만 이 앱을 태블릿 PC에서 진행한다면 화면이 넓은 기기이므로 화면 전환 없이 한 화면에서 두 가지 내용을 동시에 보여줄 수도 있다. 하나의 액티비티 화면에서 왼쪽에 목록을 출력하고, 항목 선택 시 오든쪽에 상세 내용이 출력되도록 구현할 수 있다. 하지만 모든 내용을 하나의 액티비티 클래스 내에 작성해야 하므로 액티비티 클래스의 코드가 길어지며 로직도 복잡해지는데, 이는 개발 생성과 유지보수 문제와 관련이 있다. 이를 방지하려면 액티비티 내용 일부를 뷰 클래스로 개발하면 된다. 화면 출력과 업무 로직을 각각 하나의 뷰 클래스로 추상화하면 액티비티가 너무 길게 작성되는 것을 피할 수 있다. 하지만 액티비티 내의 다양한 로직을 제대로 처리하려면 액티비티의 생명주기 등을 이용해야 하는데, 뷰에서 액티비티의 생명 주기를 이용할 수는 없다. 정리하자면 액티비티 클래스가 길고 복잡하게 작성되는 것을 피하려면 일부분은 개발자 클래스로 추상화해야 하며, 그 클래스는 액티비티에서 활용되어야 하므로 뷰 클래스여야 한다. 하지만 뷰 클래스들의 생명주기는 액티비티 생명주기와 달라서 액티비티에 작성된 코드를 추상화할 수 없기 때문에, 뷰 중에 액티비티와 생명 주기가 같은 클래스가 필요하면 이런 목적으로 제공되는 클래스가 Fragment인 것이다. 즉, 태블릿 PC용의 앱의 화면을 구현할 때 액티비티가 복잡하게 작성되는 문제를 해결하기 위해 Fragment 클래스 제공된다고 이해하면 된다. 하지만 꼭 태블릿처럼 넓은 화면을 위한 액티비티를 개발할 때 외에도, 탭 화면이나 ViewPager 같은 화면으로 구성할 때 액티비티가 너무 길고 복잡하게 작성되는 경우가 많기 때문에, 하나의 액티비티가 길어지는 상황에서는 각각의 내용을 Fragment로 추상화하여 개발하고, 액티비티에서 Fragment를 조합해서 화면을 구성하는 게 구조상 더 좋기 때문에, Fragment는 자주 쓰인다.

▶ Fragment 작성법

1. 액티비티의 레이아웃 XML 파일에 <fragment> 태그로 Fragment를 등록하는 경우

1) 레이아웃 XML 파일을 만든다 (액티비티의 레이아웃 구성과 차이가 없다)
2) XML 파일을 화면에 출력해 주는 Fragment 클래스를 자바 소스 영역에 만들어 준다: 간단하게 화면만 출력하고자 한다면 onCreateView() 함수를 제정의한다. onCreateView() 함수에서 리턴한 뷰 객체가 Fragment 화면에 출력된다.

```
public class OneFragment extends Fragment {
    FragmentOneBinding binding;

    public View onCreateView(LayoutInflator inflator, ViewGroup container, Bundle savedI
        binding = FragmentOneBinding.inflate(inflator, container, false);
        return binding.getRoot();
    }
}
```

3) 액티비티의 레이아웃 XML 파일에서 Fragment를 <fragment> 태그로 등록하여 사용한다. Fragment도 액티비티 관점에서는 뷰이므로 다른 뷰처럼 화면 구성이 가능하다. <fragment> 태그 내에서 class 이름으로 등록하려는 Fragment 클래스를 지정해 주면 된다.

```
<fragment
    android:id="@+id/fragment_one"
    class="com.example.test3_8.OneFragment"
    android=layout_width="match_parent"
    android=layout_height="wrap_content"
    android:layout_weight="1" />
```

2. 자바 코드에서 동적으로 액티비티 화면에서 Fragment을 출력하는 경우

1) 액티비티의 레이아웃 XML에서 자바 코드로 출력하는 Fragment가 나올 위치를 준비한다. 다음은 Fragment를 포함할 LinearLayout을 준비하는 코드이다.

```
<LinearLayout
    android:id="@+id/main_container"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
</LinearLayout>
```

2) 액티비티 코드 내에서 FragmentManager 클래스로 FragmentTransaction 클래스를 획득하여 Fragment를 위의 LinearLayout에 포함한다. 필요한 Fragment 클래스를 직접 생성해서 FragmentTransaction의 add() 함수를 이용하여 화면에 추가하는 자바 코드이다. add()함수를 호출할 때 Fragment가 추가 될 위치를 R.id.mai_container로 지정해 주었다.

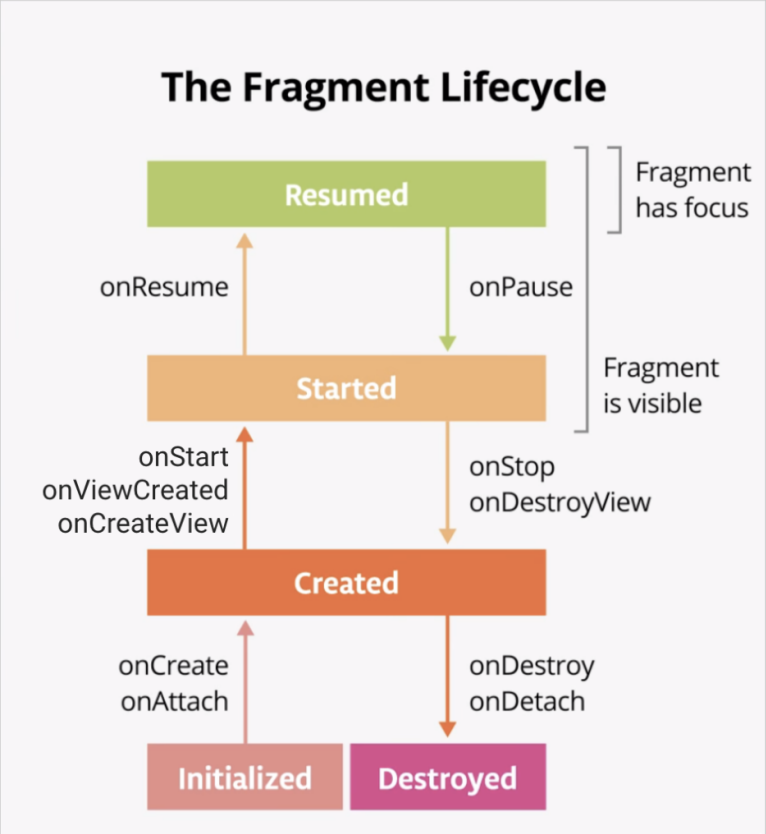
```
OneFragment oneFragment = new OneFragment();
//androidx.fragment.app.FragmentManager
FragmentManager manager = getSupportFragmentManager();
//androidx.fragment.app.FragmentTransaction
FragmentTransaction ft = manager.beginTransaction();
ft.add(R.id.main_container, oneFragment);
ft.commit;
```

이렇게 FragementTransaction을 이용하여 다양하게 액티비티 내에서 Fragment을 제어할 수 있다. 이때 사용하는 함수는 다음과 같다.

add(int containerViewId, Fragment fragment)	새로운 Fragment을 화면에 추가. id 영역에 추가.
add(int containerViewId, Fragment fragment,	id 영역에 Fragment 추가하여 추가한 Fragment의

String tag)	구분자를 태그명으로 설정.
replace(int containerViewId, Fragment fragment)	id 영역에 추가된 Fragment를 대체.
replace(int containerViewId, Fragment fragment, String tag)	id 영역에 추가된 Fragment를 대체하면서 tag 이름 설정.
remove(Fragment fragment)	추가된 Fragment 제거.
commit()	화면 적용.

● Fragment 생명주기



▶ Fragment는 액티비티의 생명주기를 그대로 따르는 뷰이고, Fragment의 생명주기는 액티비티 생명주기에 Fragment만을 위한 생명주기 함수가 추가된 구조이다. Fragment의 생명주기는 크게 5단계로 나뉜다.

1) Initialized 단계: 뷰가 준비되지 않은 상태로, Fragment를 초기화한다. onAttach() 함수와 onCreate() 함수가 호출되며, 이 두 함수는 듀 객체와 상관없이 Fragment의 초기화 로직만을 담기 위해 사용된다.
2) Created 단계: Fragment의 화면을 구성하기 위한 뷰를 준비한다. onCreateView(), onViewCreated(), onStart() 함수가 자동으로 호출되는데, 특히 onCreateView 함수의 매개변수로 LayoutInflator 객체가 전달되므로, 대부분 onCreateView 함수에서 뷰 객체를 생성한다.
3) Started 단계: Fragment 화면이 사용자에게 출력된다.
4) Resumed 단계: onResume() 함수가 자동으로 호출되는데, 포커스를 가지고 사용자의 이벤트 처리를 진행한다.

요약하자면, 최초로 Fragment가 준비되어 화면에 출력되면 onAttach() → onCreate() → onCreateView() → onViewCreated() → onStart() → onResume() 함수가 순서대로 호출된다. 이후 다든 Fragment로 교체되면 BackStack을 사용하는지에 따라 생명주기가 달라진다. BackStack은 화면에 보이지 않게 된 Fragment를 제거하지 않고 저장했다가, 다시 이용할 수 있게 하는 기능이다. 개발자가 코드로 뒤로가기 버든을 제어하지 않아도, 스마트

폰의 뒤로가기 버든에 의해 BackStack에 저장된 Fragment가 다시 나오게 된다. 이 기능을 사용할 것인지는 개발자가 결정할 수 있으며, BackStack을 이용할 때와 이용하지 않을 때 Fragment의 생명주기가 달라진다.

Fragment를 BackStack에 추가하려면, addToBackStack() 함수를 이용한다.

```
ft.addToBackStack(null);
```

5-1) BackStack을 사용하지 않는 경우: Fragment가 교체될 때 onDestroy 함수까지 호출되고, 기존의 Fragment는 제거된다. onPause() → onStop() → onDestoryView() → onDestroy() → onDetach()

5-2) BackStack을 사용하는 경우, onDestoryView 함수까지만 호출되어 기존의 Fragment가 제거되지 않는다. onPause() → onStop() → onDestoryView()까지만 자동 호출되고, onDestroyView() 함수까지 호출된 Fragment가 다시 준비되어 화면에 출력되면, onCreateView() → onViewCreated() → onStart() → onResume() 함수가 순차적으로 호출된다.

```
// activity_main.xml - 자바 코드로 출력하는 Fragment가 나올 위치를 준비

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/main_content"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
</LinearLayout>
```

```java
// MainActivity.java - 액티비티 코드 내에서 FragmentManager 클래스로 FragmentTransaction
// 클래스를 획득하여 Fragment를 위의 LinearLayout에 포함한다. 필요한 Fragment 클래스를
// 직접 생성해서 FragmentTransaction의 add() 함수를 이용하여 화면에 추가하는 자바 코드

package com.android.practicefragment;

import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentTransaction;

import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

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

        OneFragment oneFragment = new OneFragment();
        // androidx.fragment.app.FragmentManager
        FragmentManager manager = getSupportFragmentManager();
        // androidx.fragment.app.FragmentTransaction
        FragmentTransaction ft = manager.beginTransaction();
        ft.add(R.id.main_content, oneFragment);
        ft.commit();

    }
}
```

```
// fragment_one.xml -  레이아웃 XML 파일을 만든다

<androidx.constraintlayout.widget.ConstraintLayout
    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">
    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="100dp"
        android:text="OneFragment"
        android:textSize="20sp"
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"/>
    <Button android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:text="Go TwoFragment"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

```java
// OneFragment.java - XML 파일을 화면에 출력해 주는 Fragment 클래스를 자바 소스 영역에 만들어
// 준다: 간단하게 화면만 출력하고자 한다면 onCreateView() 함수를 제정의한다. onCreateView()
// 함수에서 리턴한 뷰 객체가 Fragment 화면에 출력된다

package com.android.practicefragment;

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentTransaction;

import com.android.practicefragment.databinding.FragmentOneBinding;

public class OneFragment extends Fragment {

    @Override
    public View onCreateView(LayoutInflater inflator, ViewGroup container, Bundle savedI
        FragmentOneBinding binding = FragmentOneBinding.inflate(inflator, container, fal

        binding.button.setOnClickListener(view -> {
            // Fragment 내에서 getActivity() 함수로 Activity 객체 획득
            FragmentManager manager = getActivity().getSupportFragmentManager();
            FragmentTransaction ft = manager.beginTransaction();
            TwoFragment twoFragment = new TwoFragment();
            ft.replace(R.id.main_content, twoFragment);

            ft.addToBackStack(null);
            ft.commit();
        });

        return binding.getRoot();
    }

}
```

```xml
// fragment_two.xml
<androidx.constraintlayout.widget.ConstraintLayout
    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">
    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="100dp"
        android:text="TwoFragment"
        android:textSize="20sp"
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

```java
// TwoFragment.java

package com.android.practicefragment;

import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.Fragment;

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class TwoFragment extends Fragment {

    @Override
    public View onCreateView(LayoutInflater inflator, ViewGroup container, Bundle savedI
        return inflator.inflate(R.layout.fragment_two, container, false);
    }
}
```

Resources: 깡쌤의 안드로이드 프로그래밍