# FRAGMENTS

2020년도 1학기 앱 프로그래밍 5주차 수업

# 목 차

# I 교시

2020년도 1학기 앱 프로그래밍 – FRAGMENTS 소개

# 1. FRAGMENT 소개

- 정의
  - FragmentActivity 내의 동작 또는 사용자 인터페이스의 일부 표현
  - 여러 프래그먼트를 하나의 액티비티에 결합하여 창이 여러 개인 UI를 구성
  - 하나의 프래그먼트를 여러 액티비티에서 재사용
  - 액티비티의 모듈식 섹션이라고 정의할 수 있음

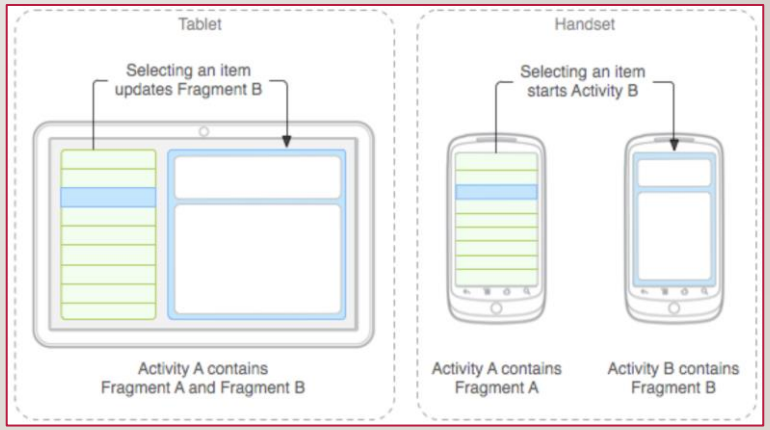
- 특징
  - 자체 생명 주기 사용
  - 자체 입력 이벤트 수신
  - 항상 액티비티 내에서 사용
    - 해당 프래그먼트의 생명 주기는 호스트 액티비티의 생명 주기에 직접적인 영향을 받음
    - 액티비티 실행 중 추가 및 삭제 가능 – 호스트 액티비티에 의해 관리

# 1. FRAGMENT 소개 (계속)

- 관리
  - 액티비티에 추가된 프래그먼트는 ViewGroup 하위에 위치
    - 자체  뷰 레이아웃으로 정의
  - <fragment> 태그
    - 레이아웃 파일에 <fragment> 요소 선언 방식
  - Layout 뷰
    - 기존 ViewGroup 에 추가 방식
      - Java 코드에 프래그먼트 선언하여 레이아웃에 프래그먼트 추가

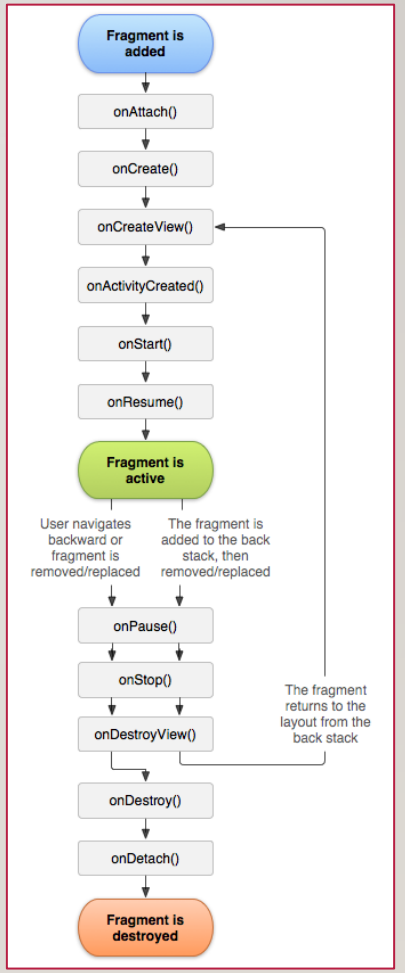
# 2. FRAGMENT 설계

- Design
  - Android 3.0 도입
  - 태블릿과 같은 큰 화면에서 보다 역동적이고 유연한 UI 디자인 지원 목적
  - 동일 액티비티에서 화면을 분할하는 기능
  - 모듈식, 재사용 가능한 액티비티 구성 요소
  - 하나의 프래그먼트를 여러 액티비티에서 사용 가능
  - 태블릿과 핸드폰 모두를 지원하는 App 제작
  - 다양한 화면 지원
  - 생명 주기 존재



# 3. FRAGMENT 생성

- Fragment 생명 주기 참조
- 생성 → Fragment 클래스 상속
- Fragment Class
  - 액티비티와 유사한 코드 존재
    - onCreate() → 초기화 코드 작성
    - onStart()
    - onPause() → 프래그먼트를 벗어나는 신호
    - onStop()
  - 중요 메서드 → 오버라이딩 필수
    - 프래그먼트가 사용자 인터페이스 UI를 표시할 때 호출
    - UI를 그리기 위해 View 객체 반환
      - null 반환은 UI를 사용하지 않는 경우 반환



# 3. FRAGMENT 생성 (계속)

- Fragment 확장 클래스 : Fragment 클래스를 상속 받은 클래스
  - DialogFragment
    - 대화 상자 표시
  - ListFragment
    - 어댑터가 관리하는 리스트 목록 표시
    - onListItemClick() 메서드 제공 : 콜백 클릭 이벤트 처리에 사용
  - PreferenceFragmentCompat
    - Preference 객체의 계층 목록
    - 어플리케이션 설정 화면을 만들 때 사용

# 4. 사용자 인터페이스

- 사용자 인터페이스 추가
  - 액티비티의 사용자 인터페이스의 일부로 사용
  - 자체 레이아웃을 갖고 액티비티에 추가

- 중요 메서드
  - onCreateView()
    - 콜백 메서드로 구현됨
    - 프래그먼트를 레이아웃에 그릴때 Android 시스템에 의해 자동 호출
    - 프레그먼트가 상속 받은 View 클래스 반환
    - 참고
      - ListFragment의 onCreateView() 메서드는 ListView 반환

# 4. 사용자 인터페이스 (계속)

- onCreateView() 메서드 구성
  - LayoutInflater inflater
    - inflater를 이용하여 XML에 정의한 레이아웃을 inflate 시키고 해당 뷰를 반환
  - ViewGroup container
    - 상위 ViewGroup을 나타내고 프래그먼트 레이아웃을 추가
  - Bundle SavedInstanceState
    - 프래그먼트의 이전 인스턴스에 대한 데이터 제공(상태 복원 등에 사용)
- inflate() 메서드
  - XML 리소스 ID, 표시될 상위 레이아웃, 상위 레이아웃에 적용여부(false 사용)
    - true 사용시 중복 표시
    - 이미 적용 중이기 때문에 false 사용

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

# 2교시

2020년도 1학기 앱 프로그래밍 – 프래그먼트 구현

# 5. 액티비티에 프래그먼트 추가 (XML 방식)

- 액티비티 UI의 일부분으로 사용
- XML에 <fragment> 태그 사용
- 동작
  - 액티비티 레이아웃 생성 시 각 프래그먼트를 인스턴스화 진행
    - 프래그먼트 각각의 onCreateView() 메서드 호출
    - 프래그먼트가 반환하는 View를 <fragment> 태그 자리에 추가하여 표시
- 속성
  - android:name : 레이아웃 안에 생성할 Fragment 클래스 지정

```xml
<?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:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <fragment
        class="com.example.fragmentbasic.LandscapeFragment"
        android:id="@+id/frag1"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_weight="1"/>

</LinearLayout>
```

# 5. 액티비티에 프래그먼트 추가 (JAVA 코드 구현 방식)

- 자바 코드를 구현하여 기존 ViewGroup에 추가
  - 액티비티 실행 중 레이아웃에 프래그먼트 동적 추가 가능
  - FragmentManager 객체를 통해 FragmentTransaction 객체를 얻어 add() 메서드로 추가
  - commit() 메서드로 변경사항 적용

```
FragmentManager fragmentManager = getSupportFragmentManager();
FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
```

```
ExampleFragment fragment = new ExampleFragment();
fragmentTransaction.add(R.id.fragment_container, fragment);
fragmentTransaction.commit();
```

# 6. 프래그먼트 관리

- FragmentManager 를 통한 프래그먼트 관리
  - 객체 얻기 : getSupportFragmentManager() 메서드 사용

- FragmentManager의 기능
  - 액티비티 내의 프래그먼트 찾기
    - findFragmentById() 와 findFragmentByTag() 메서드 사용
  - 백 스택에서 프래그먼트 삭제
    - popBackStack() 메서드 사용
  - 백 스택 상태 확인
    - addBackStackChangedListener() 메서드 사용

# 7. 프래그먼트 트랜잭션 처리

- 작업 처리 (Transaction)
  - 프래그먼트 추가, 삭제, 교체 작업의 처리
  - 트랜잭션 : commit() 적용 이후의 변경사항의 집합
  - FragmentTransaction 클래스 내의 API 사용

```
FragmentManager fragmentManager = getSupportFragmentManager();
FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
```

  - 동시 처리할 변경 사항의 집합
- 트랜잭션 메서드
  - add(), remove(), replace()
- 트랜잭션 적용
  - commit() 메서드 호출

# 7. 프래그먼트 트랜잭션 처리 (계속)

- 작업 처리 (Transaction) 특징
  - addToBackStack(null) 메서드로 백 스택에 트랜잭션 추가
  - commit() 메서드 호출을 마지막으로 처리 진행
  - commit() 호출 시 트랜잭션이 즉시 처리되지 않음
    - 처리를 위한 스레드가 준비되면 트랜잭션을 수행하도록 예약하는 것이다!!!
  - executePendingTransactions()
    - 즉시 트랜잭션을 처리하도록 하는 기능
    - 사용하지 않음!!!

# 8. 액티비티와의 통신

- 액티비티와의 관계
  - Fragment는 FragmentActivity 로부터 독립적인 객체로 구현
  - 여러 액티비티에서 사용 가능
  - 프래그먼트를 포함하고 있는 액티비티에 직접적으로 연결 됨
- 액티비티 접근
  - getActivity() 메서드를 통해 FragmentActivity 인스턴스에 접근
  - 액티비티 레이아웃에서 뷰를 찾는 작업이 가능
  - getActivity().findViewById() 를 통해 레이아웃 내의 View에 접근

# 9. 액티비티에 대한 이벤트 생성

- 콜 백 메서드
  - 프래그먼트 내에 구현 가능한 콜 백 메서드
- ListFragment 클래스의 콜 백
  - 특정 항목이 선택되었을 때 시스템이 호출하는 메서드

```
@Override
public void onListItemClick(ListView l, View v, int position, long id) {
    // Append the clicked item's row ID with the content provider Uri
    Uri noteUri = ContentUris.withAppendedId(ArticleColumns.CONTENT_URI, id);
    // Send the event and Uri to the host activity
    listener.onArticleSelected(noteUri);
}
```

```
@Override
public void onListItemClick(ListView l, View v, int position, long id) {
    if(getResources().getConfiguration().orientation
            == Configuration.ORIENTATION_LANDSCAPE && mDualPane){
        showDetails(position);
    }
    else {
        Intent intent = new Intent();
        intent.setClass(getActivity(), DetailsActivity.class);
        intent.putExtra( name: "index", mCurCheckPosition);
        startActivity(intent);
    }
}
```

# 9. 액티비티에 대한 이벤트 생성 (계속)

- OnArticleSelectedListener() 클래스의 콜 백
  - 하나의 액티비티에 두개의 프래그먼트가 구성된 경우
  - 두 프래그먼트 사이의 이벤트 처리를 위한 메서드

```
// Container Activity must implement this interface
public interface OnArticleSelectedListener {
    public void onArticleSelected(Uri articleUri);
}
```

# 3교시

2020년도 1학기 앱 프로그래밍 – 프래그먼트 실습 및 복습 과제

# 8. 앱 프로그래밍 학습을 위한 복습 (과제)

- 실습 및 복습 문제
  - https://developer.android.com/guide/components/fragments 에 작성된 예제 구현
  - 영상에서 설명한 기능을 구현하세요!

- 제출 방법
  - http://ctl.gtec.ac.kr 의 과제 제출란에 제출
  - Java 파일과 xml 파일을 제출한다.
  - 실행 화면을 캡쳐하여 위 파일들과 함께 압축 파일하여 제출한다.
  - 압축파일은 자신의 "반_학번_이름.zip" 로 한다.
  - 제출 일자와 시간을 엄수하여 제출하세요!