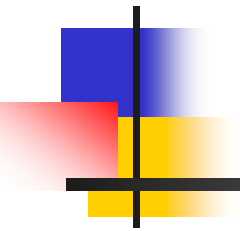


Shared Preferences(수업)

배 희호 교수
경복대학교
소프트웨어융합과

Data 저장 기법

- Android에서 제공하는 Data 저장 방법
  - Shared Preference(프리퍼런스)
  - 내부/외부 Storage를 사용한 File
  - SQLite Database
  - Network를 사용한 Web 상에서의 Data 저장 방법이 있음

Data 저장 기법

- File 저장 (Internal/External Storage)
  - Internal Storage
    - 용도
      - 민감한 Data를 App 전용으로 저장
    - 특징
      - App 전용 공간, 다른 App 접근 불가
      - App이 삭제되면 Data도 함께 삭제
      - 보안 수준이 높음
  - External Storage
    - 용도
      - 대용량 Data(Image, Video, 문서 등)를 저장
    - 특징
      - 다른 App도 Data에 접근 가능(권한 필요)
      - App 삭제 후에도 Data 유지 가능

Data 저장 기법

- SQLite Database
  - 용도
    - 구조화된 Data를 Relational DB 형태로 저장
  - 특징
    - SQL Query를 사용해 Data 삽입, 수정, 삭제, 조회 가능
    - App 내에서 Data 구조화 및 복잡한 관계 관리에 적합
    - App 전용 Database
  - 장점
    - 강력한 Data 관리 기능 제공
  - 단점
    - 구현 복잡성 증가

Data 저장 기법

- Room Persistence Library
  - 용도
    - SQLite를 더 쉽게 사용할 수 있도록 Android에서 제공하는 Library
  - 특징
    - SQLite에 비해 더 높은 수준의 추상화를 제공
    - Database 작업을 객체 지향적으로 처리
    - 데이터 접근 객체(DAO)를 통해 Database에 접근
  - 장점
    - Code 작성이 간단하고 유지보수 용이

Data 저장 기법

- Content Provider
  - 용도
    - App 간 Data를 공유
  - 특징
    - Data를 다른 App이나 System에서 접근 가능하게 함
    - URI를 통해 Data 접근
  - 장점
    - 보안성 및 Data 제어 가능
  - 단점
    - 설정 복잡

Data 저장 기법

- Network 저장 (Cloud)
  - 용도
    - Data를 Cloud Server(예: Firebase, REST API 등)에 저장
  - 특징
    - Server에 Data 저장, 여러 Device에서 동기화 가능
    - 보안 인증 필요
  - 장점
    - 동기화, 대규모 Data 관리 가능
  - 단점
    - Internet 연결 필요

Data 저장 기법

- Shared Preferences(공유 프리퍼런스)
  - 용도
    - <Key-Value> 쌍 형태로 간단한 Data를 저장
    - 주로 설정값, 사용자 환경 설정 등을 저장하는 데 사용
  - 특징
    - 내부 저장소에 자동 저장
    - 단순한 Data 구조(String, 숫자, boolean, List 등)에 적합
    - Data는 App이 삭제되기 전까지 유지
  - 장점
    - 가볍고 빠름
  - 단점
    - 구조화된 Data 관리에는 부적합

SharedPreference

- Application에 간단한 설정 Option을 추가하고 사용자가 이를 변경할 때, 변경된 Data를 저장하고 관리하게 하려면 어떻게 해야 할까요?
  - File을 따로 만들어야 할까요?

- Android에서는 App이나 그 Component(Activity, Service 등)의 환경 설정 정보를 영구적으로 저장하고 관리할 수 있게끔 Shared Preference(공유 프리퍼런스)라는 장치를 제공
- Preference는 DataBase처럼 복잡한 자료가 아닌 간단한 정보를 저장하는 일에 사용
  - 예) 인사말, Font, 소리같은 설정 정보를 저장하고 복원하는 용도로 사용
- Windows 환경의 Registry나 LINUX 환경의 Setting File 정도에 대응되는 개념임

Shared Preference

- App을 개발하다 보면 각종 환경 설정 값들을 보관해야 할 경우가 있음
  - 기초적인 자료형(Boolean형, Float형, Integer형, String 등)을 Key-Value 쌍으로 저장하고 복원할 수 있는 방법
  - 예) Game App인 경우 Sound On/Off, Vibration On/Off, 화면 밝기 설정 등을 보관 해둬야 함
- 이를 위해 Android에서는 SharedPreferences라는 매우 편리한 클래스를 제공
- 해당 클래스는 마치 HashMap 클래스를 이용하여 원하는 Data를 Key와 Value 형태로 손쉽게 추가 및 제거할 수 있음
- 이뿐만 아니라 내장 Memory에 File로 저장되기 때문에 영구적으로 보관할 수도 있음
- 여러 개의 Activity들이 이 Preference를 공유할 수 있으므로 Activity 간의 Data 교환 목적으로도 사용이 가능함

SharedPreference

- SharedPreferences는 Android에서 간단한 Data 저장을 위해 사용되는 API
- 주로 간단한 설정 값이나 작은 Data 조각들을 저장하고 불러올 때 사용
- 내부적으로 XML File에 Data를 저장하며, <Key-Value>쌍으로 Data를 관리
- 주요 기능
  - Data 저장
    - SharedPreferences.Editor를 사용하여 Data를 저장
  - Data 읽기
    - SharedPreferences 객체를 통해 Data를 읽음
  - Data 삭제
    - 특정 Key에 해당하는 Data를 삭제하거나, 전체 Data를 초기화할 수 있음

SharedPreference

- 특징
  - <Key-Value> 저장 방식
    - Data를 <Key-Value> 쌍으로 저장하며, Data를 조회할 때는 Key를 사용
  - 영구 저장
    - App이 종료되더라도 Data가 유지됨. 하지만 App이 삭제되면 SharedPreferences Data도 함께 삭제됨
  - 경량화
    - 소량의 Data(몇 Kb 정도) 저장에 적합하며, 대량 Data를 저장하는 데는 부적합
  - File 저장 위치
    - SharedPreferences Data는 내부 저장소에 XML File 형태로 저장
    - File 경로는 /data/data/<패키지 이름>/shared_prefs/

Shared Preference

- Application에서 정보를 유지하기 위해 Preference를 사용할 때는 Preference를 저장하는 위치와 저장된 Preference를 읽어오는 위치 및 Preference를 사용하는 방법을 알아야 함
- Preference는 /data/data에 위치하며 각 Application마다 /data/data [애플리케이션의 패키지명]/shared_prefs folder 내에 저장되어 있음
- Preference는 키 이름(Key)-저장된 값(Value)의 쌍으로 저장
- XML Format의 Text File에 정보를 저장
  => Setting File이나 INI File에 더 가까움

Shared Preference

- Preference 용도
  - Application의 환경 설정 정보를 영구적으로 저장
    - 사용자의 Option 선택 사항이나 Program 자체의 구성 정보를 주로 저장
  - 하나의 Application내에서도 정보를 유지하기 위해 Activity간의 정보 교환 및 공유
    - 한쪽 Activity에서 Preference의 정보를 수정하면 다른 Activity도 수정된 값이 적용
    - 그러나 Preference는 Application의 고유 정보이기 때문에 외부에서는 읽을 수 없음
  - UI 화면 정보를 저장했다가 복원하는 용도로 사용 하기도 함

Shared Preference

- Preference 장점
  - Preference는 상대적으로 용량이 작고 Activity간 정보 교환에도 활용이 가능함
  - Data 보관이 가장 손쉽고 간편하며 다른 Program에서는 접근할 수 없다는 장점이 있음
  - Preference는 “Application의 환경 설정 정보를 저장”하는 용도와 “Activity 간의 정보 교환”에 사용됨
  - UI 화면 정보를 저장했다가 복원하는 용도로 사용하기도 함

- Preference 단점
  - Preference는 DB나 File에 비해 그 사용법이 매우 간단하고 쉬움. 하지만 Preference의 가장 큰 단점은 “설정값을 XML 형태로 저장하므로 처리속도가 느림”

Shared Preference

- Preference를 사용하는 3가지 방법
  - PreferenceActivity를 상속받지 않고 Preference를 사용할 경우
  - PreferenceActivity를 상속받아 Preference를 사용할 경우
  - 허니콤(3.0)이후 버전에서 Preference를 사용할 경우

Shared Preference

- PreferenceActivity를 상속받지 않고 Preference를 사용할 경우
  - Preference 객체 생성
  - Preference 가져 오기
  - Preference 읽어 오기
  - Preference에 기록하기

Shared Preference

- 사용 방법
  - SharedPreferences 객체 가져오기
    - getSharedPreferences() 메소드를 사용하여 SharedPreferences 객체를 가져올 수 있음
  - Data 저장하기
    - SharedPreferences.Editor 객체를 사용하여 Data를 저장
  - Data 불러오기
    - getString(), getInt() 등 메소드를 통해 저장된 Data를 가져올 수 있음
  - Data 삭제
    - 특정 key의 Data를 삭제하려면 remove() 메소드 사용
  - 모든 Data 삭제
    - 전체 Data를 삭제하려면 clear() 메소드 사용

Shared Preference

- SharedPreferences를 통해 Data를 저장하기 위해, 우선 SharedPreferences의 Instance를 얻어와야 함
- SharedPreferences Instance를 얻는 3가지 방법
  - getPreferences(int mode)
    - Activity내에서 오직 한 개의 Preference File만 필요한 경우에 사용
    - 생성되는 SharedPreferences File은 해당 Activity의 이름으로 생성
    - 하나의 Activity에서 사용할 목적으로 생성하였지만, 생성한 SharedPreferences의 이름을 getSharedPreferences() 메소드에 인자 값으로 전달하면 다른 Activity에서도 저장된 Data에 접근 가능

Shared Preference

- SharedPreferences Instance를 얻는 3가지 방법
  - getSharedPreferences("String name", int mode)
    - 특정 이름을 가지는 SharedPreferences를 생성

```
SharedPreferences prefs
    = getSharedPreferences("userPrefs", MODE_PRIVATE);
```

    - getSharedPreferences의 인수
      - 첫번째 인수는 Preference를 저장할 XML File 이름
      - mode 인수 (이 File은 공유 Mode)

mode 인수	설명
MODE_PRIVATE	자기 App 내에서 사용 외부 App에서 접근 불가
0	외부 App에서 읽기, 쓰기 가능
MODE_WORLD_READABLE	외부 App에서 읽기 공유
MODE_WORLD_WRITEABLE	외부 App에서 쓰기 공유

Shared Preference

- SharedPreferences Instance를 얻는 3가지 방법
  - PreferenceManager.getDefaultSharedPreferences(Context context)
    - 기본 SharedPreferences File을 가져오는 데 사용되는 메소드
    - 이 메소드는 App 전역에서 기본으로 사용할 SharedPreferences를 제공하며, 별도의 이름을 지정하지 않아도 됨
    - 기본 SharedPreferences File은 App의 Package 이름을 기반으로 자동으로 생성
    - File 이름은 보통 다음과 같은 형식
      - <패키지 이름>_preferences
      - 예) com.example.myapp_preferences

Shared Preference

- Preference에 정보를 저장
  - SharedPreferences 클래스의 edit() 메소드를 호출해 SharedPreferences.Editor 객체를 얻어냄
    - SharedPreferences.Editor 객체가 Property를 사용해서 Preference에 정보를 저장할 수 있음

```
SharedPreferences.Editor prefsEditor = prefs.edit();
```

Shared Preference

- SharedPreferences.Editor
  - SharedPreferences에서 Data를 추가, 수정 또는 삭제하기 위해 사용되는 Interface
  - SharedPreferences는 읽기 전용이기 때문에 Data를 변경하려면 반드시 Editor를 통해 작업해야 함
  - 주요 역할
    - SharedPreferences.Editor는 SharedPreferences의 Data를 조작하는 데 필요한 메소드를 제공
    - 변경 작업(추가, 수정, 삭제)을 수행한 후 apply() 또는 commit() 메소드를 호출하여 변경 사항을 저장해야 함

Shared Preference

- SharedPreferences.Editor 주요 메소드

메소드	설명
putString(String key, String value)	문자열 값 저장
putInt(String key, int value)	정수 값 저장
putBoolean(String key, boolean value)	boolean 값 저장
putFloat(String key, float value)	부동소수점 값 저장
putLong(String key, long value)	Long 타입 값 저장
putStringSet(String key, Set<String> value)	문자열 집합(Set) 저장
remove(String key)	지정된 Key의 Data 삭제
clear()	모든 Data 삭제
apply()	변경 사항을 비동기적으로 저장
commit()	변경 사항을 동기적으로 저장하고 성공 여부를 반환

Shared Preference

- Preference에 정보를 저장
  - SharedPreferences.Editor 클래스의 putBoolean() 또는 putString()과 같은 putXxx(key, value) 메소드를 사용해 key와 value의 쌍으로 정보를 저장
    - 이때 key가 Property의 이름, value가 Property의 값

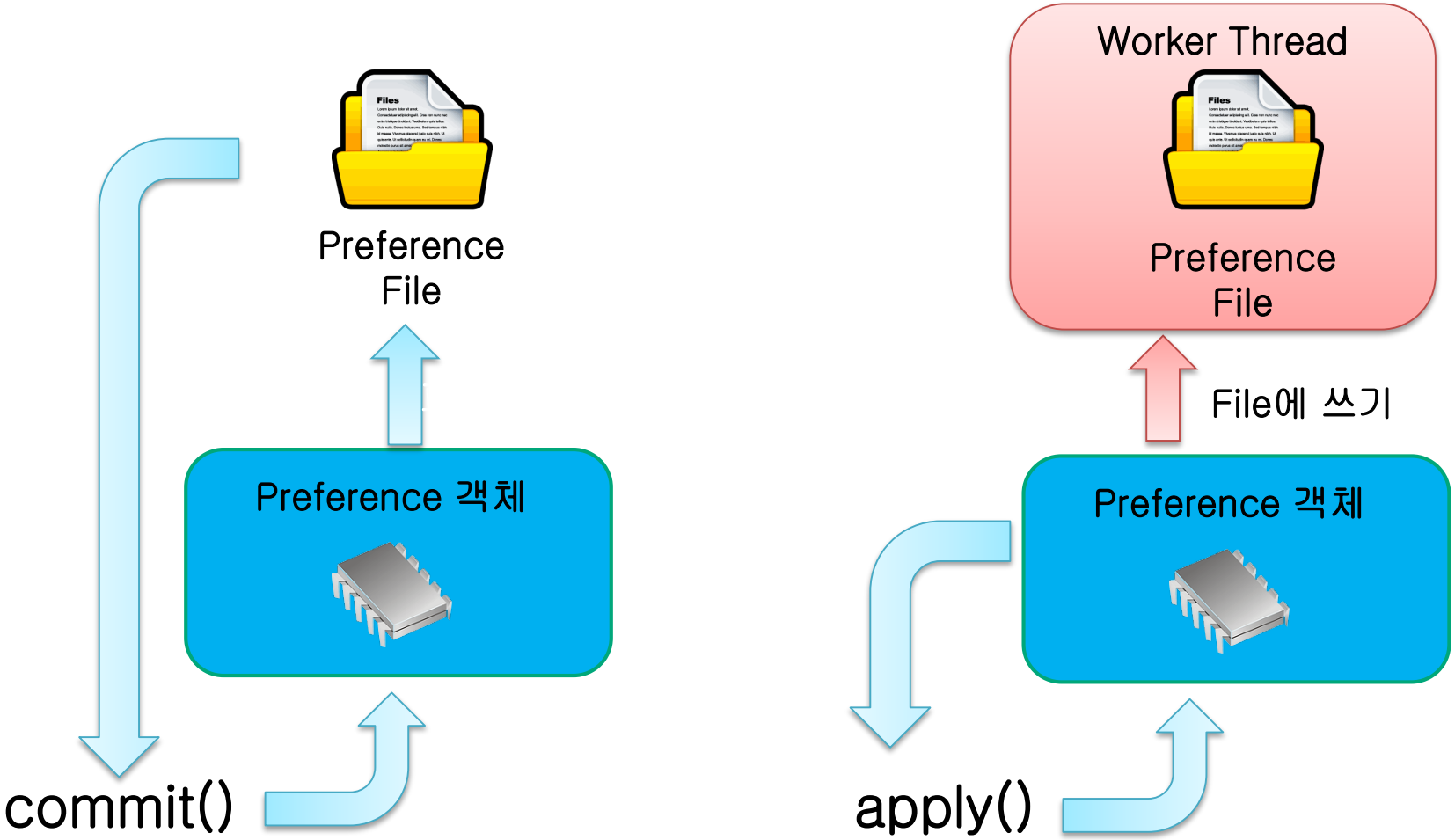
```
prefsEditor.putString("id", id);
```

  - 추가 또는 변경된 Property를 commit() 메소드를 사용해 Preference에 한 번에 반영

```
prefsEditor.commit();
```

Shared Preference

- Preference Data를 저장하는 commit()과 apply() 메소드



apply() 메소드는 Android API 9 진저브레드부터 추가

Shared Preference

- commit()과 apply()중 어떤 것을 사용해야 하나?
  - apply()는 Worker Thread에서 실제 File에 저장하기 때문에 속도가 매우 빠름
  - 아직 Worker Thread가 File에 저장하는 것을 마치지 않은 상태에서 Preference를 참조해도 전혀 문제가 되지 않음
    - Preference의 Data 참조는 이미 생성된 Preference 객체를 참조하기 때문임
  - SharedPreferences는 Process 내에서 SingleTone 방식의 Instance이기 때문에 만약 반환 값을 필요로 하지 않는다면 commit()대신에 apply()를 쓰는 것이 안전
  - apply()의 경우 Android Component의 Lifecycle 및 Disk 쓰기의 상호작용을 신경 쓸 필요가 없음
  - 구조적으로 apply()를 한 후에 이루어지는 in-flight Disk의 쓰기는 상태 전환이 이루어지기 전에 완료되도록 되어 있음

Shared Preference

- SharedPreferences.Editor 사용 시 주의점
  - apply() vs commit()
    - 대량 Data 저장이나 중요한 Data 저장 시에는 commit()을 고려해야 함
    - 일반적인 상황에서는 성능이 더 나은 apply() 사용을 권장
  - Data 형식 주의
    - 잘못된 Key나 Data 형식을 사용할 경우 Data가 저장되지 않거나 App이 충돌할 수 있음
    - 항상 get 메소드 호출 시 기본값을 지정해야 함
  - Data 삭제
    - remove()와 clear()는 apply() 또는 commit()을 호출하기 전까지 변경 사항이 반영되지 않음
  - 다중 스레드 환경
    - SharedPreferences는 Thread 안전성을 보장하지만, 너무 자주 Data를 수정하면 병목현상이 발생

Shared Preference

- Preference를 저장하는 위치와 저장된 Preference를 읽어 오는 위치
  - Preference를 저장하는 위치
    - Activity의 onPause() 메소드
  - Preference에 정보를 저장할 때
    - Activity가 Foreground에서 Background로 교체되는 지점인 onStop() 메소드에 기술되어야 함
  - 저장된 Preference를 읽어오는 위치
    - Activity의 onCreate() 메소드
  - 저장된 Preference의 정보를 얻어내려고 할 때
    - Activity가 생성 지점인 onCreate() 메소드에서 함

Shared Preference

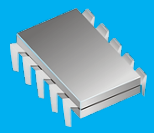
- Preference File 읽어 오기

Preference File

getSharedPreferences(String fileName, int mode)

이 메소드는 App을 실행하고 최초 호출되었을 때만 실제 File에서 읽어 옴

프레퍼런스 객체



getSharedPreferences(String fileName, int mode)

App 내에서 다시 호출

따라서 최초 getSharedPreferences 메소드가 호출될 때 저장된 Data가 많다면 느림

getSharedPreferences() 메소드는 최초 호출 시에만 느릴 수 있기 때문에 App이 실행될 때 App이 실행될 때 Loading 화면에서 미리 호출해 두면 좋음

Shared Preference

- Preference 읽어 오기
  - Preference는 SharedPreferences 클래스를 사용해서 구현
  - SharedPreferences 클래스는 Primitive Data Type의 Key와 Value의 쌍으로 이루어진 형태로 정보를 저장하고 얻어내는 것을 제공
  - SharedPreferences 클래스는 boolean, float, int, long, 그리고 String Type의 Data를 사용할 수 있음

Shared Preference

- Preference 읽어 오기
  - Preference는 Key와 Value의 쌍으로 Data를 저장
  - Key는 정보 이름이며 Value는 정보의 default 값을 저장
  - 여러 Type의 Data Type이 존재하는데, 웬만한 Primitive Type에 대해서는 읽기 메소드가 모두 제공
  - 가장 자주 사용되는 Type은 정수, 문자열, 논리형

Preference 읽기/쓰기	설명
int getInt("String key", int defValue)	정수형 읽기
String getString ("String key", String defValue)	문자열 읽기
boolean getBoolean("String key", boolean defValue)	논리형 읽기
void putInt("String key", int defValue)	정수형 쓰기
void putString ("String key", String defValue)	문자열 쓰기
void putBoolean("String key", boolean defValue)	논리형 쓰기

Shared Preference

- Preference 읽어 오기
  - 읽어올 Data의 Type이 다를 뿐이며 사용하는 방법은 모두 같음
  - Key 인수로 Data의 이름을 지정하고 defValue 인수로 값이 없을 때 적용할 Default를 지정
  - Preference에 Key가 존재하면 해당 Preference에 기록되어 있는 값이 반환되고, Key가 없을 때에는 두 번째 인수로 지정한 defValue가 반환
  - 최초 실행될 때에는 Preference가 생성되기 전이므로 Default가 반환

Shared Preference

- Preference 읽어 오기
  - getBoolean() 또는 getString()과 같은 getXxx(key, defValue) 메소드를 사용해서 정보를 얻어냄
    - 이때 Xxx는 저장된 Preference의 정보의 Data 타입으로 int 타입일 경우 getInt() 메소드를 String 타입일 경우 getString() 메소드를 사용
    - key 저장된 Property의 이름이고, defValue는 해당 Preference의 key가 존재하지 않을 경우 반환되는 값

```
String id = pref.getString("id", "");
```

Shared Preference

- Preference 사용

```
//저장된 프리퍼런스의 정보를 읽어옴
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
                    ...생략..
   // SharedPreferences 객체를 얻어냄
   SharedPreferences prefs = getSharedPreferences(PREFS_NAME, 0);
   // 저장된 정보를 읽어옴
   currentMode = prefs.getInt("view_mode", DAY_VIEW_MODE);
}
```

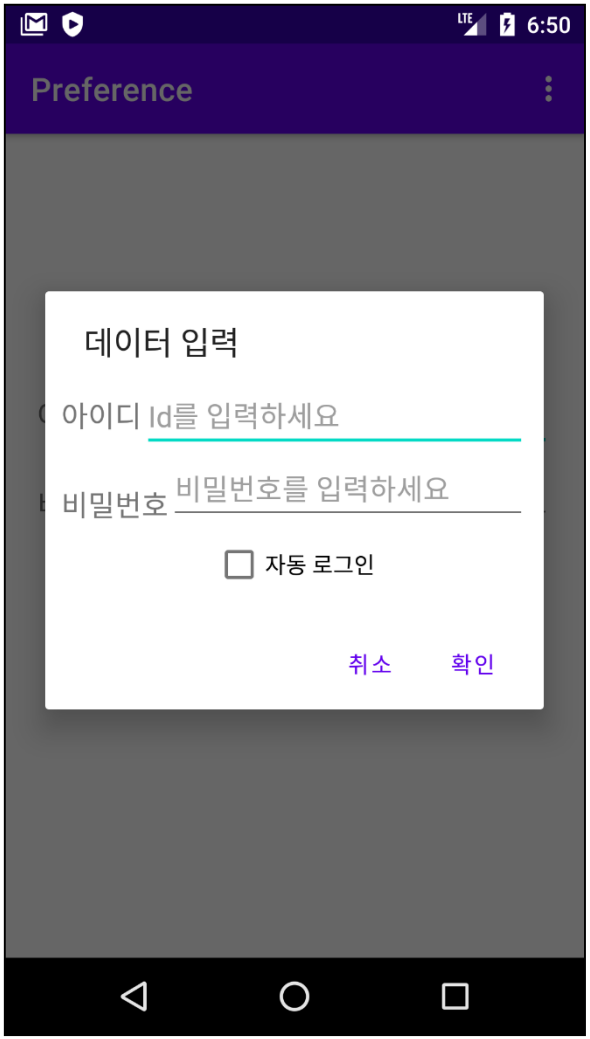
Shared Preference

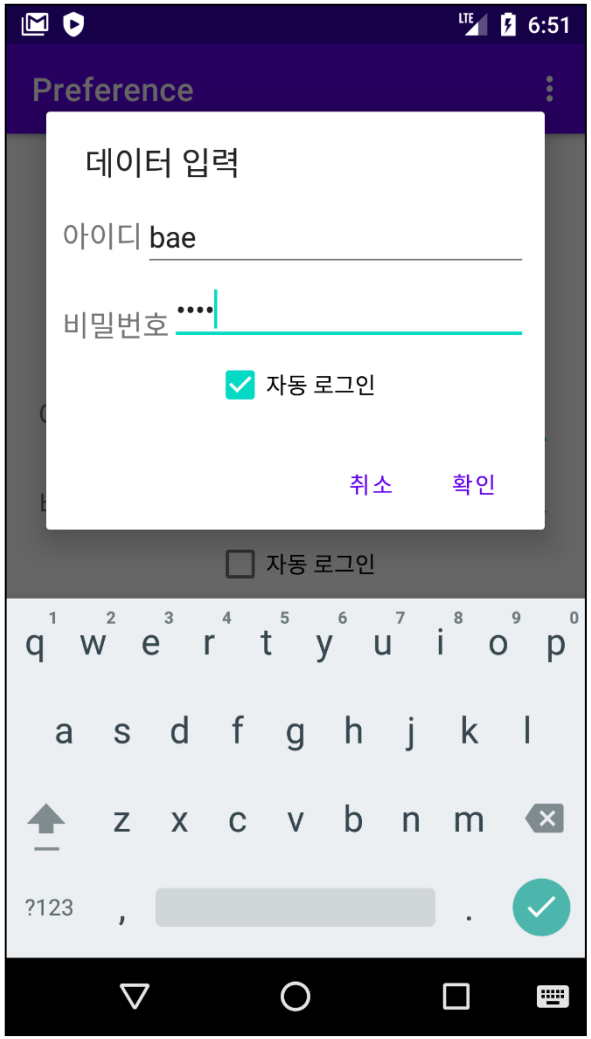
- 주의 사항
  - Data 크기 제한
    - SharedPreferences는 작은 크기의 Data를 저장하는 데 적합
    - 큰 Data를 저장하는 용도로는 적합하지 않음
  - 동시 접근 문제
    - 여러 Thread에서 동시에 SharedPreferences에 접근하는 경우 동기화 문제가 발생할 수 있음
    - 이러한 상황에서는 apply()보다는 commit()을 사용하는 것이 안전할 수 있음

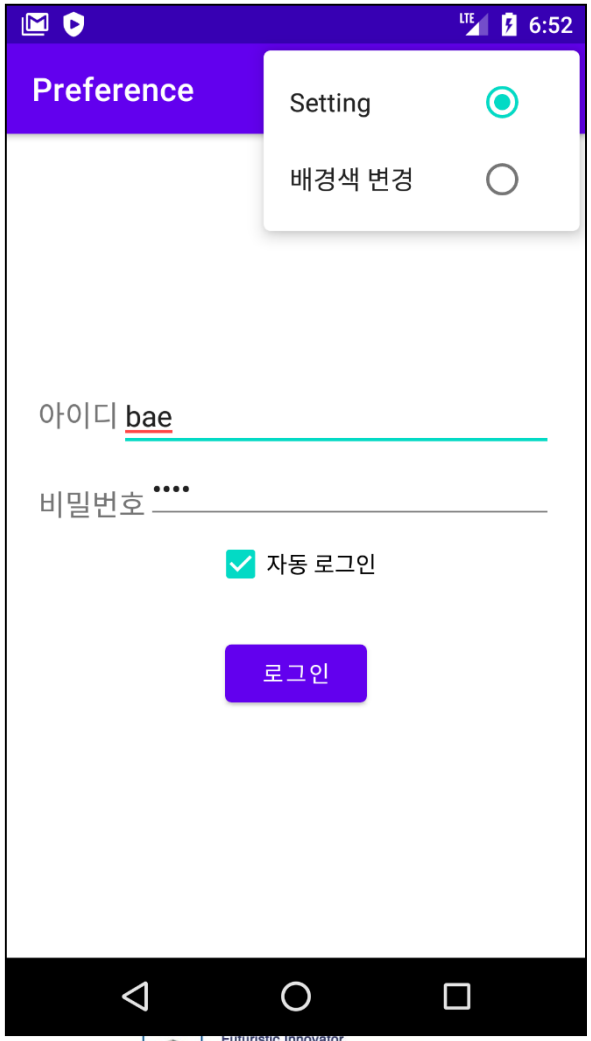
Preference 예제

- Preference를 사용해 간단한 App을 만들어보자.
  - id와 password와 1개의 CheckBox('id 저장'과 'password 저장‘을 통해 자동 로그인)를 입력 받는 App으로, 초기에 Dialog를 이용하여 Data를 입력하면 Preference에 기록하는데, 추가로 Main Activity의 배경색을 지정 내용을 함께 저장함
  - CheckBox를 선택함에 따라 id와 password 정보를 Preference에 저장된 정보를 화면에 loading하게 됨

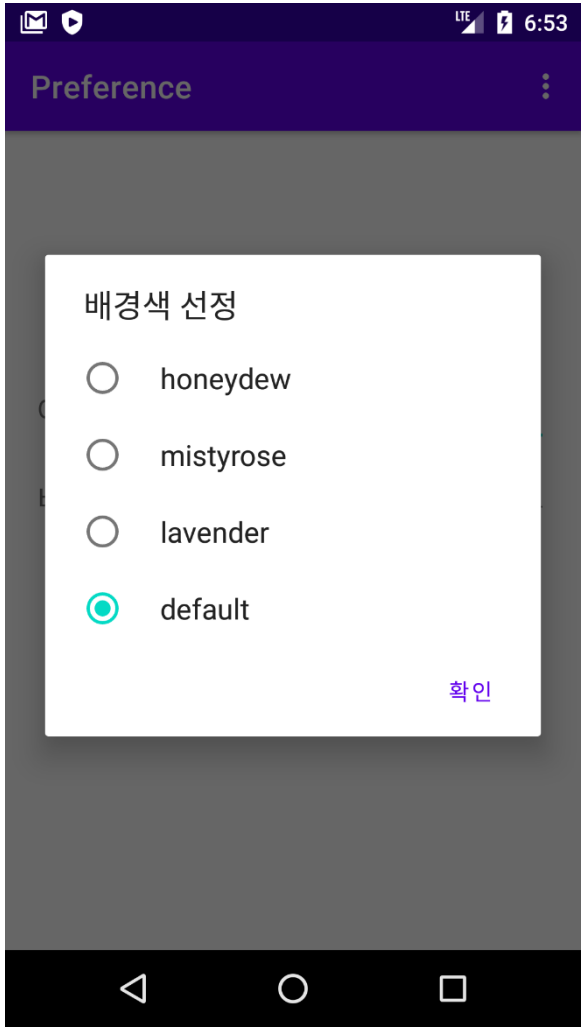
Preference 예제

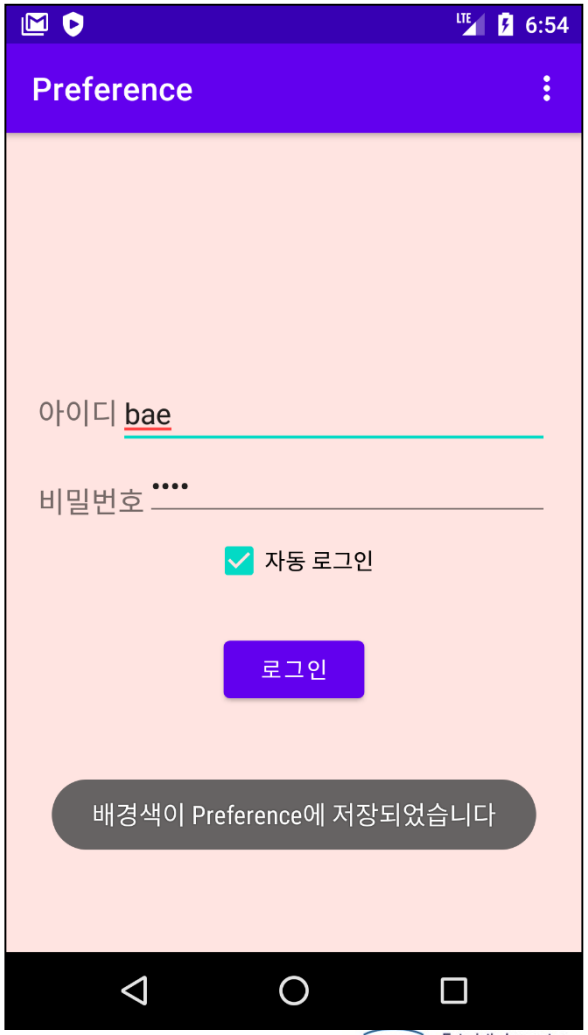






Preference 예제





Preference 예제

- activity_main.xml

```
<?xml version="1.0" encoding="utf-8" ?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical"
    android:padding="20dp"
    tools:context=".MainActivity">
```

Preference 예제

- activity_main.xml

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/idStr"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <EditText
        android:id="@+id/id"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10">
        <requestFocus />
    </EditText>
</LinearLayout>
```

Preference 예제

- activity_main.xml

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:text="@string/passStr"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <EditText
        android:id="@+id/pass"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="textPassword" />
</LinearLayout>
```

Preference 예제

- activity_main.xml

```
    <CheckBox
        android:id="@+id/check"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="자동 로그인" />

    <Button
        android:id="@+id/login"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginTop="20dp"
        android:text="로그인" />
</LinearLayout>
```

Preference 예제 4

- activity_main.xml

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/idStr"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <EditText
        android:id="@+id/id"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10">
        <requestFocus />
    </EditText>
</LinearLayout>
```

Preference 예제

- menu.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <group android:checkableBehavior="single">
        <item
            android:id="@+id/action_settings"
            android:checked="true"
            android:title="Setting" />

        <item
            android:id="@+id/color"
            android:title="배경색 변경" />
    </group>
</menu>
```

Preference 예제

- dialog.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="wrap_content"
    android:layout_gravity="center"
    android:orientation="vertical"
    android:padding="10dp">
```

Preference 예제

- dialog.xml

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/idStr"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <EditText
        android:id="@+id/id"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:hint="Id를 입력하세요">
        <requestFocus />
    </EditText>
</LinearLayout>
```

Preference 예제

- dialog.xml

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:text="@string/passStr"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <EditText
        android:id="@+id/pass"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:hint="비밀번호를 입력하세요"
        android:inputType="textPassword" />
</LinearLayout>
```

Preference 예제

- dialog.xml

```
    <CheckBox
        android:id="@+id/checkBox"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="자동 로그인" />
</LinearLayout>
```

Preference 예제

- dialog_cancel.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="wrap_content"
    android:background="@color/purple_200"
    android:padding="20dp">

    <ImageView
        android:id="@+id/image"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@mipmap/ic_launcher" />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="30dp"
        android:text="취소되었습니다"/>
</LinearLayout>
```

Preference 예제

- arrays.xml

```
<string-array name="background_color">
    <item>honeydew</item>
    <item>mistyrose</item>
    <item>lavender</item>
    <item>default</item>
</string-array>

<string-array name="color_values">
    <item>#F0FFF0</item>
    <item>#FFE4E1</item>
    <item>#E6E6FA</item>
    <item>#FFFFFF</item>
</string-array>
```

Preference 예제

- strings.xml

```
<string name="idStr">아이디</string>
<string name="passStr">비밀번호</string>
<string name="loginStr">로그인</string>
<string name="resultStr">저장된 프리퍼런스의 정보</string>
```

Preference 예제

- MainActivity.JAVA

```
public class MainActivity extends AppCompatActivity {
    final String PREFS_NAME = "UserID";
    private SharedPreferences prefs;
    private CheckBox checkBox;
    private LinearLayout layout;

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

        layout = findViewById(R.id.layout);
        EditText editText1 = findViewById(R.id.id);
        EditText editText2 = findViewById(R.id.pass);
        checkBox = findViewById(R.id.check);
```

Preference 예제

- MainActivity.JAVA

```
prefs = getSharedPreferences(PREFS_NAME, MODE_PRIVATE);
String id = prefs.getString("id", "");
String pass = prefs.getString("passWord", "");
boolean check = prefs.getBoolean("auto", false);
String color = prefs.getString("color", "");
if (id.equals("")) {
    preferenceDialog();
} else {
    editText1.setText(id);
    editText2.setText(pass);
    checkBox.setChecked(check);
    layout.setBackgroundColor(Color.parseColor(color));
}
```

Preference 예제

- MainActivity.JAVA

```
Button button = findViewById(R.id.login);
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String inputId = editText1.getText().toString();
        String inputPass = editText2.getText().toString();
        if (inputId.equals("") || inputPass.equals("")) {
            Toast.makeText(getBaseContext(), "입력해주세요",
                                                Toast.LENGTH_SHORT).show();
        } else {
            if (inputId.equals(prefs.getString("id", "")) &&
                    inputPass.equals(prefs.getString("passWord", ""))) {
                Intent intent = new Intent(getBaseContext(), LoginActivity.class);
                intent.putExtra("id", inputId);
                startActivity(intent);
            } else {
```

Preference 예제

- MainActivity.JAVA

```
                    Toast.makeText(getBaseContext(),
                                                    "Id와 비밀번호를 확인해주세요.",
                                            Toast.LENGTH_SHORT).show();
                }
            }
        }
    });
```

Preference 예제

```
    checkBox.setOnCheckedChangeListener(
                new CompoundButton.OnCheckedChangeListener() {
        @Override
        public void onCheckedChanged(CompoundButton buttonView,
                                     boolean isChecked) {
            SharedPreferences.Editor editor = prefs.edit();
            editor.putString("id", prefs.getString("id", ""));
            editor.putString("passWord", prefs.getString("passWord", ""));
            editor.putBoolean("auto", isChecked);
            editor.apply();
            if (!isChecked) {
                editText1.setText("");
                editText2.setText("");
            } else {
                editText1.setText(prefs.getString("id", ""));
                editText2.setText(prefs.getString("passWord", ""));
            }
        }
    });
}
```

Preference 예제

- MainActivity.JAVA

```
private void preferenceDialog() {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("데이터 입력");
    View dialog = View.inflate(this, R.layout.dialog, null);
    builder.setView(dialog);
    EditText editId = dialog.findViewById(R.id.id);
    EditText editPw = dialog.findViewById(R.id.pass);
    CheckBox checkBox1 = dialog.findViewById(R.id.checkBox);
    builder.setPositiveButton("확인", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            String inputId = editId.getText().toString();
            String inputPass = editPw.getText().toString();
            if (inputId.equals("") || inputPass.equals("")) {
                Toast.makeText(getBaseContext(), "입력해주세요",
                                                    Toast.LENGTH_SHORT).show();
            } else {
```

Preference 예제

- MainActivity.JAVA

```
SharedPreferences.Editor editor = prefs.edit();
if (checkBox.isChecked()) {
    editor.putString("id", inputId);//정보저장
    editor.putString("passWord", inputPass);//정보저장
    editor.putBoolean("auto", checkBox.isChecked());
    editor.putString("color", "");
    Toast.makeText(getBaseContext(),
                                    "프리퍼런스에 정보가 저장되었습니다.",
            Toast.LENGTH_SHORT).show();
} else {
```

Preference 예제

- MainActivity.JAVA

```
                editor.remove("id");
                editor.remove("passWord");
                editor.remove("auto");
                editor.remove("color");
            }
            editor.apply();
        }
    }
});
```

Preference 예제

- MainActivity.JAVA

```
    builder.setNegativeButton("취소", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            View cancel = View.inflate(getBaseContext(),
                                                R.layout.dialog_cancel, null);
            Display display = ((WindowManager)
                        getSystemService(WINDOW_SERVICE)).getDefaultDisplay();
            int x = (int) (Math.random() * display.getWidth());
            int y = (int) (Math.random() * display.getHeight());
            Toast toast = new Toast(getBaseContext());
            toast.setGravity(Gravity.TOP | Gravity.LEFT, x, y);
            toast.setView(cancel);
            toast.setDuration(Toast.LENGTH_SHORT);
            toast.show();
        }
    });
    builder.show();
}
```

Preference 예제

- MainActivity.JAVA

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.menu, menu);
    return true;
}
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    switch (item.getItemId()) {
        case R.id.action_settings:
            preferenceDialog();
            checkBox.setChecked(false);
            break;
        case R.id.color:
            colorDialog();
    }
    item.setChecked(true);
    return true;
}
```

Preference 예제

- MainActivity.JAVA

```
private void colorDialog() {
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("배경색 선정");
    builder.setSingleChoiceItems(R.array.background_color, 3,
                                 new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            String[] value = getResources().getStringArray(R.array.color_values);

            layout.setBackgroundColor(Color.parseColor(value[which]));
            SharedPreferences.Editor editor = prefs.edit();
            editor.putString("color", value[which]);
            editor.apply();
        }
    });
```

Preference 예제

- MainActivity.JAVA

```
        builder.setPositiveButton("확인", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                Toast.makeText(getBaseContext(),
                        "배경색이 Preference에 저장되었습니다",
                                                    Toast.LENGTH_SHORT).show();
            }
        });
        builder.show();
    }
}
```

Preference 예제

- activity_login.xml

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

    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="20dp"
        android:gravity="center"/>
</LinearLayout>
```

Preference 예제

- LoginActivity.JAVA

```
public class LoginActivity extends AppCompatActivity {

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

        Intent intent = getIntent();
        String id = intent.getStringExtra("id");

        TextView textView = findViewById(R.id.textView);
        textView.setText("안녕하십니까 " + id + "님");
    }
}
```

Preference 예제

- Preference 가져 오기

```
SharedPreference 프레퍼런스 객체명 =
                                    getSharedPreferences(
                "프레퍼런스를 저장할 xml파일 이름", int mode);
```

Preference 예제

- Preference 읽어 오기
  - 읽어올 Data의 Type이 다를 뿐이며 사용하는 방법은 모두 같음
  - Key 인수로 Data의 이름을 지정하고, defValue 인수로 값이 없을 때 적용할 Default를 지정
  - Preference에 Key가 존재하면 해당 Preference에 기록되어 있는 Value가 반환되고 Key가 없을 때에는 두 번째 인수로 지정한 defValue가 반환됨
  - 최초 실행될 때에는 Preference가 생성되기 전이므로 Default가 반환됨

Preference 예제

- Preference에 기록 하기
  - Preference에 Value를 저장하기 위한 2가지 과정
    - Preference editor 객체 가져 오기
    - Preference에 Value 기록 하기
  - Preference Editor 객체 가져 오기
    - Preference 클래스 자체에는 Value를 읽는 메소드만 제공되며 Value를 기록하는 메소드는 inner 클래스인 SharedPreferences.Editor가 제공함
    - 얻어온 Preference 객체에 edit() 메소드를 호출하여 Editor 객체를 먼저 얻어 옴

```
SharedPreference.Editor 에디터 객체명 =
                                프레퍼런스 객체명.edit();
```

Preference 예제

- Preference에 값 기록 하기
  - Editor는 모든 변경을 모아 두었다가 한꺼번에 적용 시킬 수 있으므로 일종의 트랜잭션
  - Preference에 Value를 기록하는데, Data Type에 따라 다음과 같은 메소드를 제공

✓ SharedPReference.Editor putInt("String key", int value);
✓ SharedPReference.Editor putBoolean("String key", boolean value);
✓ SharedPReference.Editor putString("String key", String value);

  - 그 외에 제공되는 메소드

✓ SharedPreferences.Editor remove("String key")
해당 키 값을 삭제
✓ boolean commit()
put() 메소드를 모두 집어 넣었다면 반드시 commit()을 사용해 주어야 실제로 xml에 저장

PreferenceActivity

- Android Programming에서 설정 Activity를 만들어야 할 때가 있음
- PreferenceActivity는 Android에서 설정 화면을 구현할 때 사용되는 Activity
- 사용자가 Application의 설정을 쉽게 변경할 수 있도록 도와주는 표준 UI Component를 제공
- PreferenceActivity는 XML Resource File을 기반으로 설정 항목을 정의하고, 자동으로 UI를 생성하여 사용자에게 표시
- 설정 Activity들은 보통 List로 구성되어 있음
- List를 Converting하면서 만든다고 해도, Text만 나오는 row가 있고, CheckBox도 혼재된 row, 입력 창이 들어가야하는 row가 있을 수 있음
  - 이럴 때 편하게 사용할 수 있는 방법이 바로 PreferenceActivity를 상속받아서 만드는 것

PreferenceActivity

- 사용자는 설정 화면에서 App의 기본 설정값을 변경할 수 있으며, 이러한 설정값은 SharedPreferences에 자동으로 저장

- 특징
  - 설정 관리
    - 설정 화면을 구현하는 데 특화되어 있으며, 설정값은 자동으로 SharedPreferences에 저장
  - 구조적 UI
    - 설정 항목을 XML Resource로 정의하여 관리 가능
  - Deprecated 경고
    - PreferenceActivity는 Android 11(API 30) 기준으로 일부 기능이 권장되지 않음
    - PreferenceFragmentCompat를 사용하는 방식으로 대체. 하지만 PreferenceActivity는 여전히 일부 구 버전 App에서 사용될 수 있음

PreferenceActivity

- 주요 구성 요소
  - Preference
    - 기본 설정 항목을 나타냄
    - 여러 종류의 Preference가 있으며, 각 항목은 사용자가 설정을 변경할 수 있는 UI 요소를 제공
  - PreferenceScreen
    - Preference 항목들의 최상위 Container로서, 전체 설정 화면을 정의
    - 여러 Preference 항목을 Group하여 계층 구조로 설정할 수 있음
  - PreferenceFragmentCompat
    - PreferenceActivity와 함께 사용되는 Fragment로, 설정 화면을 분할하여 표시할 수 있음
    - PreferenceFragmentCompat를 사용하면 설정 UI를 더 유연하게 구성할 수 있음

PreferenceActivity

- 주요 구성 요소
  - PreferenceCategory
    - Android의 설정 화면에서 여러 Preference 항목을 Group화하여 논리적으로 구분하는 데 사용
    - PreferenceCategory는 설정 화면 내에서 시각적 구분을 제공하는 용도로 사용
    - PreferenceCategory는 XML File 내에서 여러 Preference 항목을 Group화하는 Container 역할
    - PreferenceCategory를 사용하면 관련된 설정 항목들을 Group으로 묶어 사용자에게 더 직관적인 설정 화면을 제공할 수 있음

PreferenceActivity

- PreferenceActivity를 사용하기 위한 2단계
  - Layout 작성
  - Activity 작성

- Layout 작성
  - UI에 보여줄 Preference들을 XML에 정의해야 함
  - PreferenceActivity는 Activity가 들어있지만 res/layout에 XML을 저장하지 않고, res/xml folder에 File이 들어 있음
  - PreferenceActivity에 사용되는 Layout XML은 일반 Activity에 사용하는 XML과 작성법이 조금 다름
  - 일반 Activity에 사용하는 default layout이 LinearLayout 이라면, Preference Activity에 사용하는 default layout은 PreferenceScreen

PreferenceActivity

- Activity 작성
  - PreferenceActivity를 처음 작성할 때 달라진 부분은 하나 뿐임
  - 보통의 Activity에서 Layout을 설정할 때 setContentView(R.layout.파일이름)으로 설정
  - PreferenceActivity는 Layout을 설정할 때 addPreferencesFromResource(R.xml.파일이름)과 같이 함
  - 일반적인 Activity라면 Click Event를 OnClickListener로 받음
  - PreferenceActivity는 Click Event를 OnPreferenceClickListener로 받아야 함

PreferenceActivity

- XML에 Preference 정의하는 절차
  - 먼저 "res" Directory에 " xml" 이름으로 Directory를 생성
  - res/xml 안에 원하는 preference XML File을 생성
    - 일반적으로 preference.xml과 같이 생성
    - 생성한 XML File을 열어보면 아래와 같이 <PreferenceScreen>을 root로 하여 만들어진 것을 확인할 수 있음

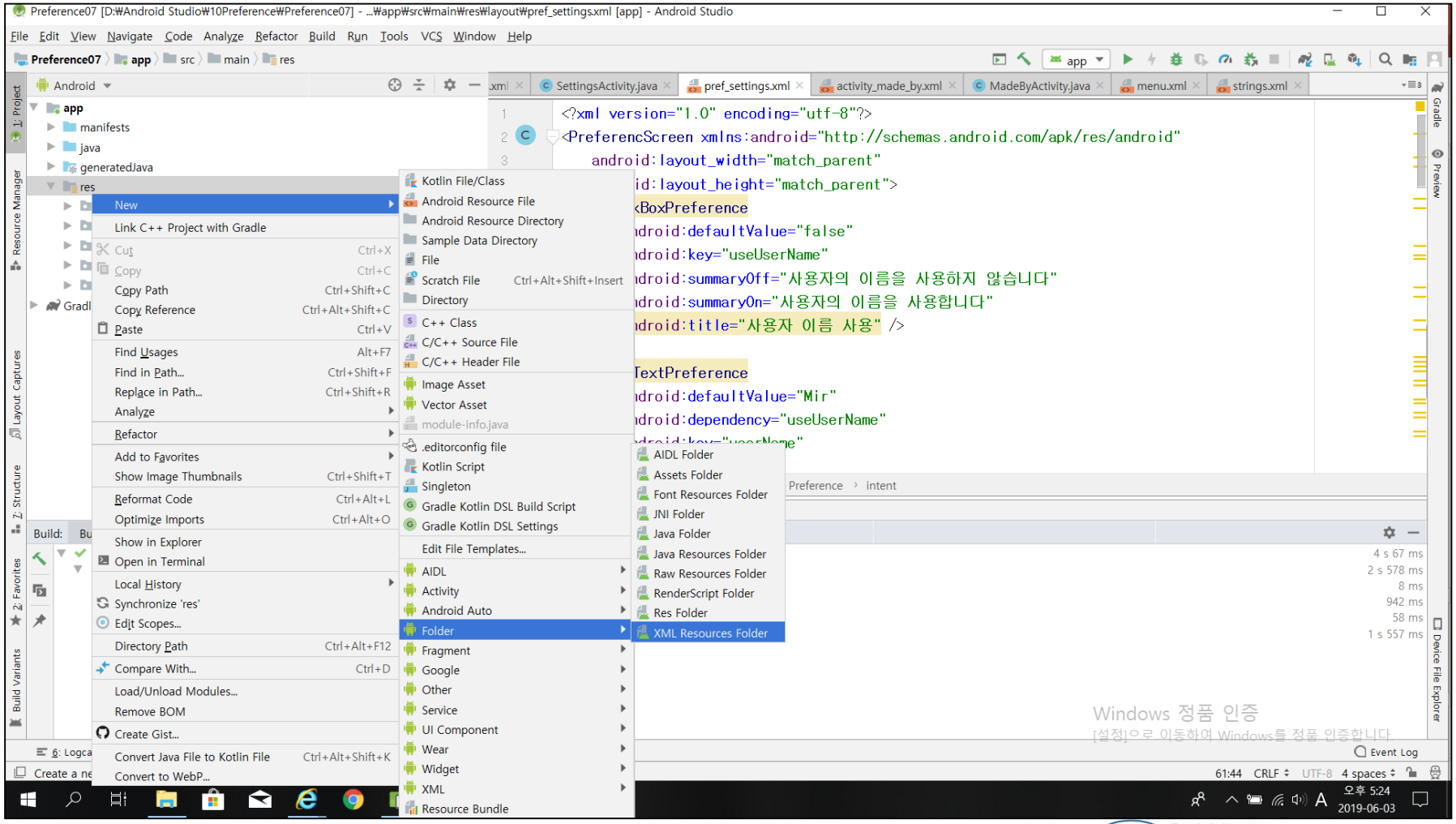
```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android">

</PreferenceScreen>
```

  - 생성한 XML File을 열어 원하는 Preference를 추가

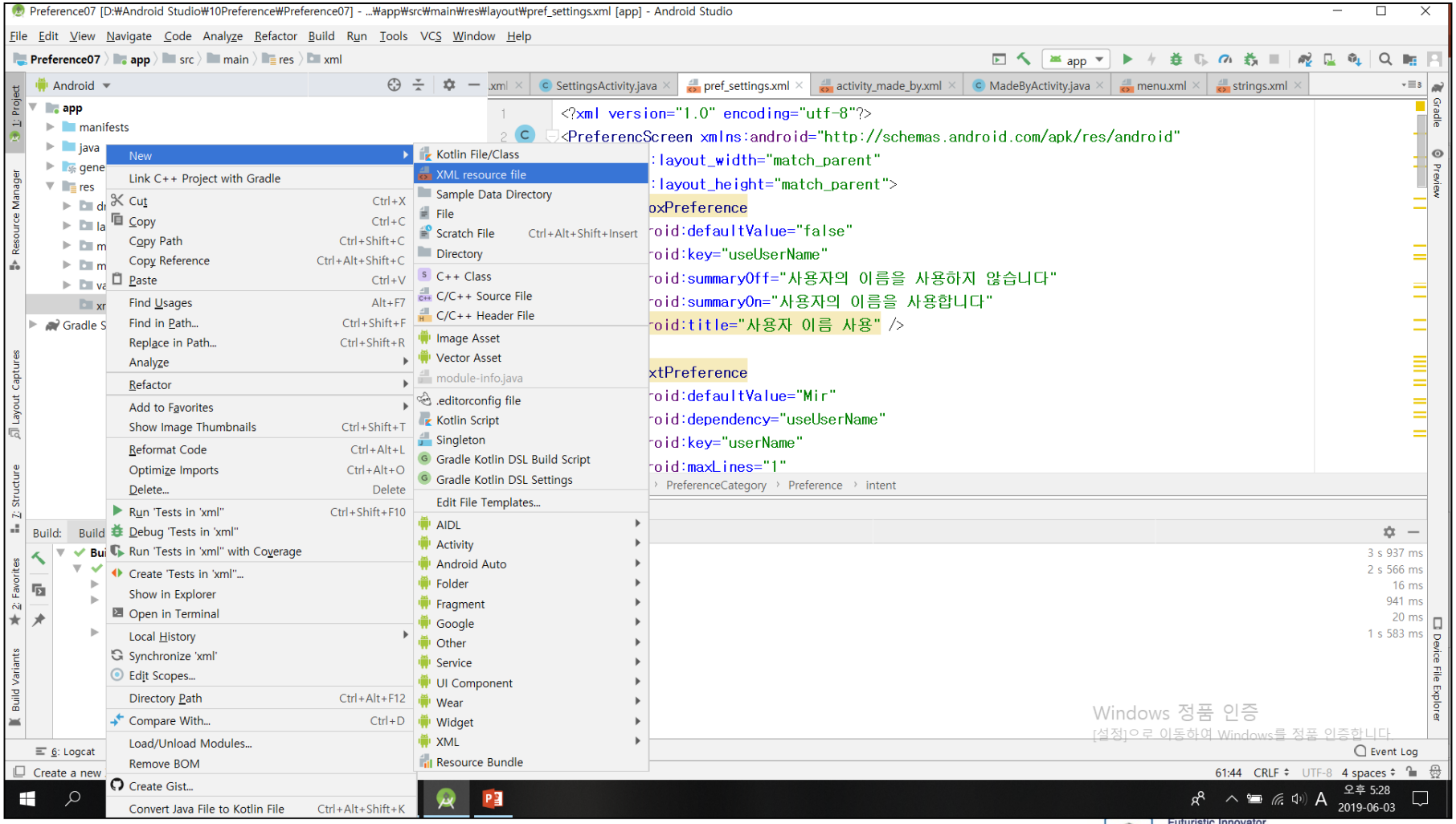
PreferenceActivity

- [New]–[Folder]–[XML Resources Folder]



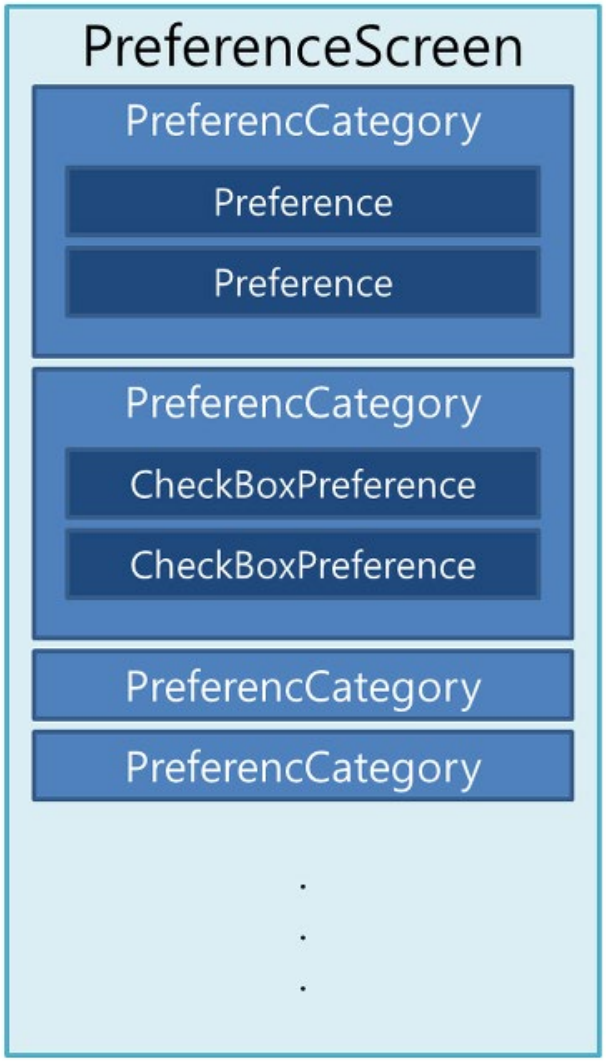
PreferenceActivity

- [New]-[XML resource file]



PreferenceActivity

- PreferenceScreen Tag 안에는 복수의 PreferenceCategory가 들어갈 수 있음
- 또 PreferenceCategory 안에는 복수의 Preference들이 들어갈 수 있음



PreferenceActivity

- Tag 종류

태그 종류	설명
PreferenceScreen	설정화면 단위, 중첩 가능하며 중첩된 내용은 별도의 화면에 나옴
PreferenceCategory	설정 여러 개를 시각적으로 묶어서 표현
CheckboxPreference	체크박스가 나오는 설정
EditTextPreference	글 입력을 위한 설정
ListPreference	항목 다이얼로그를 위한 설정
MultiSelectListPreference	항목 다이얼로그인데 체크박스가 자동 추가되어 여러 선택 가능
RingtonPreference	알림음 선택을 위한 설정
SwitchPreference	스위치를 이용한 설정

PreferenceActivity

XML 속성

XML 속성	설명
android:key	Preferences 데이터 저장/불러올 때 사용되는 키(key) 지정
android:title	Preference 항목의 제목을 지정
android:summary	Preference 항목을 자세히 설명하는 문자열 지정
android:enabled	Preference 항목의 활성화 비활성화 여부 지정 (true/false)
android:selectable	Preference 항목의 선택 가능 여부 결정 (true/false)
android:order	Preference 항목이 표시되는 순서를 결정 (0-based 정수 사용: 낮은 값 먼저 보임) 순서가 명시적으로 지정되지 않는다면 XML에 정의된 순서대로 표시됨

PreferenceActivity

메소드

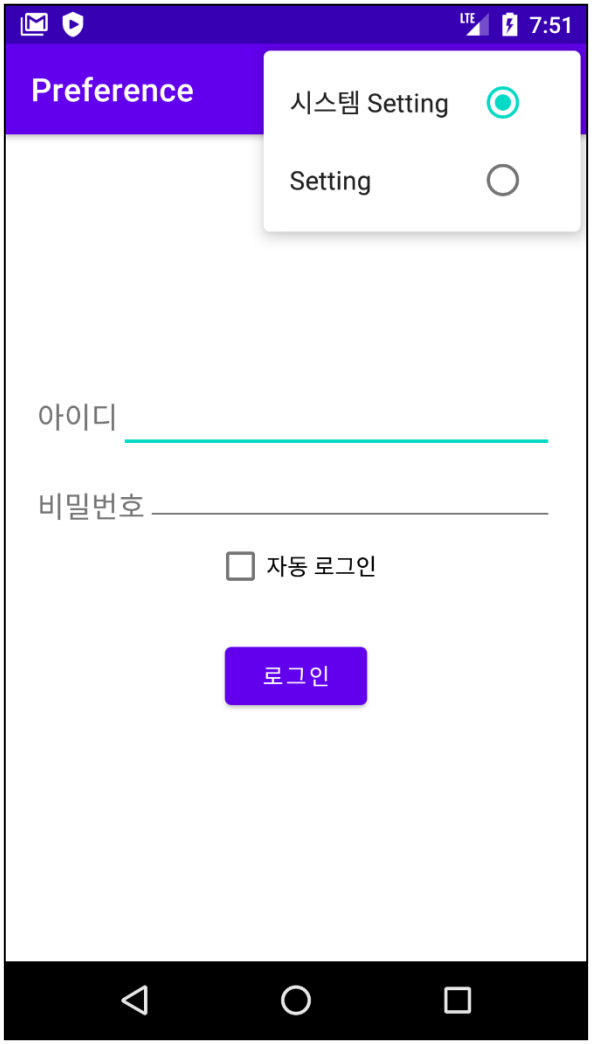
메소드	설명
void setKey/setTitle/setSummary	Preference 항목의 키, 제목, 설명을 지정
void setEnabled(boolean)	Preference 항목의 활성화/비활성화 여부 지정
void setSelectable(boolean)	Preference 항목의 선택 가능 여부 지정
void setLayoutResource()	Preference 항목에 표시할 view를 'R.layout.레이아웃파일이름' 형식으로 파라메터로 넘김
void setOnPreferenceChangedListener()	Preference 항목에 변화가 있으면 호출할 callback 메소드 지정
void setOnPreferenceClickListener()	Preference 항목에 click 이벤트가 발생하면 호출할 callback 메소드 지정

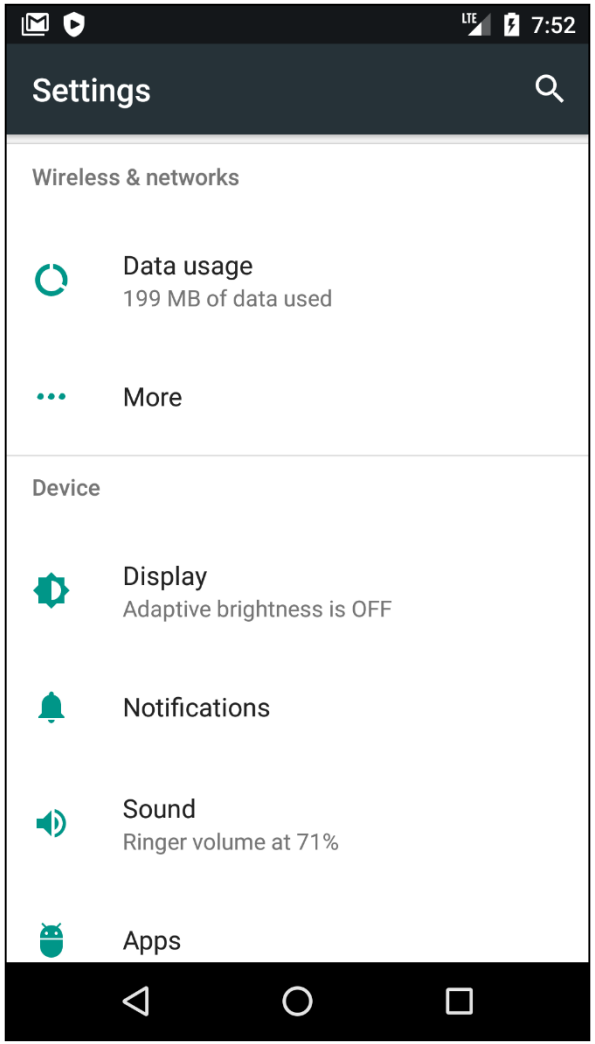
PreferenceActivity

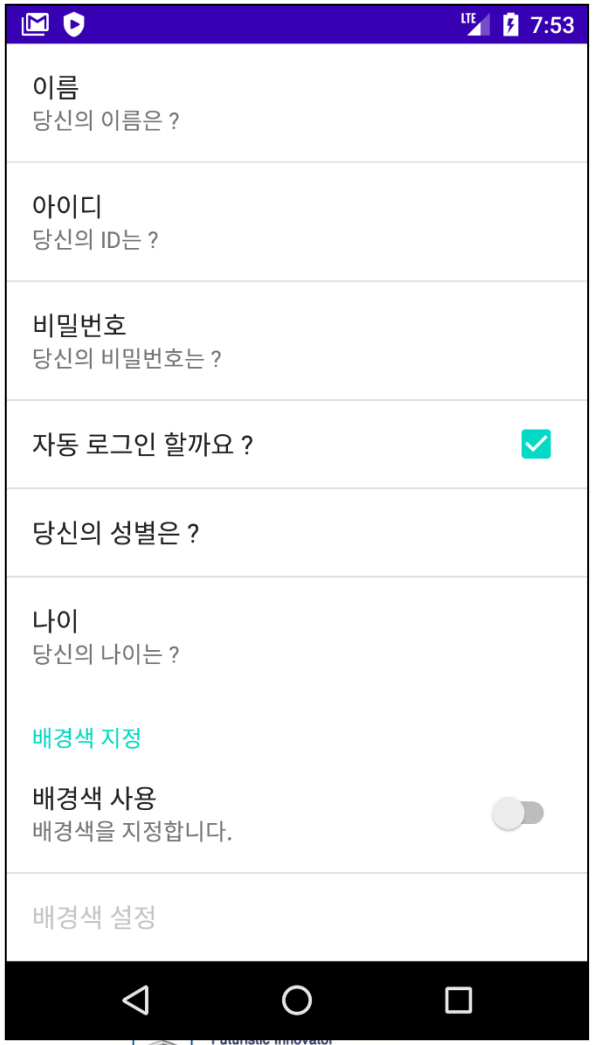
- 이 Preference의 tag의 속성들은 간단하게는 "키(KEY)"와 "타이틀(TITLE)" / "키(KEY)"와 "타이틀(TITLE)", 그리고 "값(VALUE)"로 이루어져 있음
- 다른 속성들이 많이 있지만, 이것들만 사용해도 어렵지 않게 설정의 기능들을 구현해 낼 수 있음

PreferenceActivity 예제

실행 화면

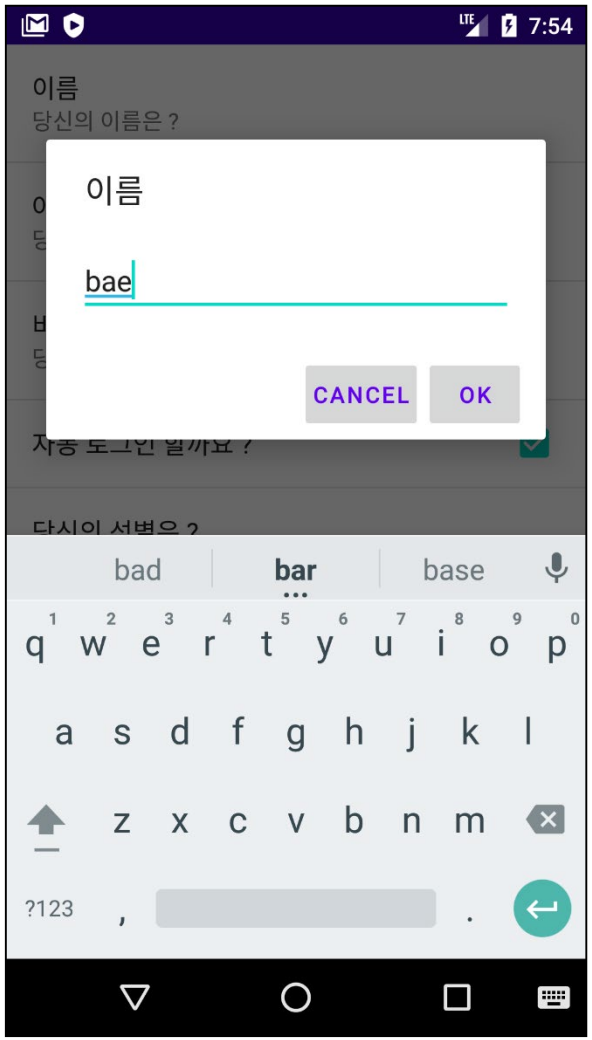


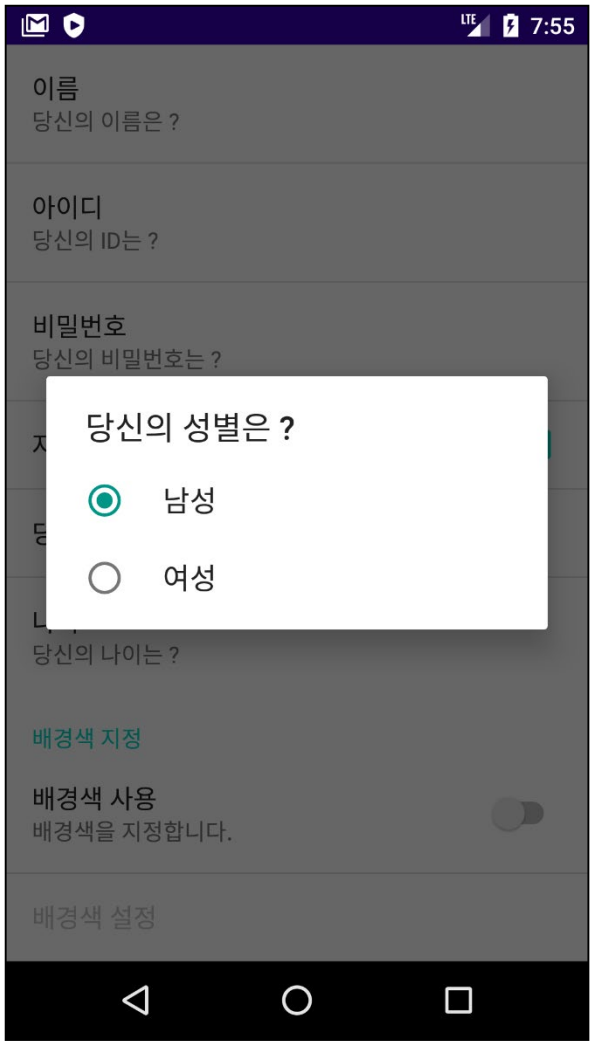


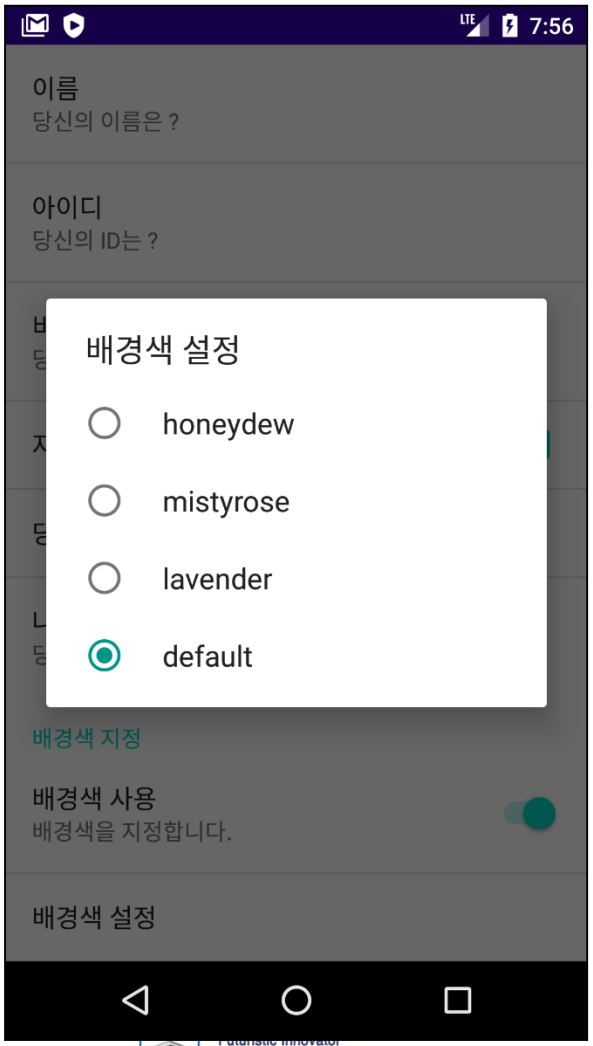


PreferenceActivity 예제

실행 화면







PreferenceActivity 예제

- 사용자 인터페이스
  - 앞의 예제 재 사용

PreferenceActivity 예제

- MainActivity.JAVA

```
public class MainActivity2 extends AppCompatActivity {
    private LinearLayout layout;
    private CheckBox checkBox;

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

        layout = findViewById(R.id.layout);
        EditText editText1 = findViewById(R.id.id);
        EditText editText2 = findViewById(R.id.pass);
        checkBox = findViewById(R.id.check);
```

PreferenceActivity 예제

- MainActivity.JAVA

```
SharedPreferences prefs = getDefaultSharedPreferences(this);
String id = prefs.getString("id", "");
String pass = prefs.getString("passWord", "");
boolean check = prefs.getBoolean("auto", false);
if (check) {
    editText1.setText(id);
    editText2.setText(pass);
} else {
    editText1.setText("");
    editText2.setText("");
}
checkBox.setChecked(check);
```

PreferenceActivity 예제

- MainActivity.JAVA

```
Button button = findViewById(R.id.login);
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String inputId = editText1.getText().toString();
        String inputPass = editText2.getText().toString();
        if (inputId.equals("") || inputPass.equals("")) {
            Toast.makeText(getBaseContext(), "입력해주세요",
                                                Toast.LENGTH_SHORT).show();
        } else {
            if (inputId.equals(prefs.getString("id", "")) &&
                    inputPass.equals(prefs.getString("passWord", ""))) {
                Intent intent = new Intent(getBaseContext(), LoginActivity.class);
                intent.putExtra("id", inputId);
                startActivity(intent);
            } else {
```

PreferenceActivity 예제

- MainActivitry.JAVA

```
                Toast.makeText(getBaseContext(),
                                                "Id와 비밀번호를 확인해주세요.",
                        Toast.LENGTH_SHORT).show();
            }
        }
    }
});
```

PreferenceActivity 예제

```
    checkBox.setOnCheckedChangeListener(
                        new CompoundButton.OnCheckedChangeListener() {
        @Override
        public void onCheckedChanged(CompoundButton buttonView,
                                                        boolean isChecked) {
            SharedPreferences.Editor editor = prefs.edit();
            editor.putString("id", prefs.getString("id", ""));
            editor.putString("passWord", prefs.getString("passWord", ""));
            editor.putBoolean("auto", isChecked);
            editor.apply();
            if (!isChecked) {
                editText1.setText("");
                editText2.setText("");
            } else {
                editText1.setText(prefs.getString("id", ""));
                editText2.setText(prefs.getString("passWord", ""));
            }
        }
    });
}
```

PreferenceActivity 예제

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.menu1, menu);
    return true;
}
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    Intent intent = null;
    switch (item.getItemId()) {
        case R.id.action_settings:
            intent = new Intent(getBaseContext(), CustomSetting.class);
            break;
        case R.id.system_settings:
            intent = new Intent(Settings.ACTION_SETTINGS);
    }
    item.setChecked(true);
    startActivity(intent);
    return true;
}
```

PreferenceActivity 예제

- MainActivity.JAVA

```
    public CheckBox getCheckBox() {
        return checkBox;
    }

    public LinearLayout getLayout() {
        return layout;
    }
}
```

PreferenceActivity 예제

- menu1.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <group android:checkableBehavior="single">
        <item
            android:id="@+id/system_settings"
            android:checked="true"
            android:title="시스템 Setting" />

        <item
            android:id="@+id/action_settings"
            android:title="Setting" />
    </group>
</menu>
```

PreferenceActivity 예제

- xml/settings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android">
    <EditTextPreference
        android:defaultValue=""
        android:key="name"
        android:summary="당신의 이름은 ?"
        android:title="이름" />

    <EditTextPreference
        android:defaultValue=""
        android:key="id"
        android:summary="당신의 ID는 ?"
        android:title="아이디" />

    <EditTextPreference
        android:defaultValue=""
        android:key="passWord"
        android:summary="당신의 비밀번호는 ?"
        android:title="비밀번호" />
```

PreferenceActivity 예제

- xml/settings.xml

```
<CheckBoxPreference
    android:defaultValue="true"
    android:key="auto"
    android:title="자동 로그인 할까요 ?" />

<ListPreference
    android:defaultValue="0"
    android:dialogTitle="당신의 성별은 ?"
    android:entries="@array/gender"
    android:entryValues="@array/gender_values"
    android:key="gender"
    android:negativeButtonText="@null"
    android:positiveButtonText="@null"
    android:title="당신의 성별은 ?" />
<EditTextPreference
    android:defaultValue="0"
    android:key="age"
    android:summary="당신의 나이는 ?"
    android:title="나이" />
```

PreferenceActivity 예제

- xml/settings.xml

```
<PreferenceCategory android:title="배경색 지정">
    <SwitchPreference
        android:defaultValue="false"
        android:key="color"
        android:summary="배경색을 지정합니다."
        android:title="배경색 사용" />

    <ListPreference
        android:defaultValue="0"
        android:dependency="color"
        android:entries="@array/background_color"
        android:entryValues="@array/color_values"
        android:key="backgrounder"
        android:negativeButtonText="@null"
        android:positiveButtonText="@null"
        android:title="배경색 설정" />
</PreferenceCategory>
</PreferenceScreen>
```

PreferenceActivity 예제

- arrays.xml

```
<string-array name="gender">
    <item>남성</item>
    <item>여성</item>
</string-array>

<string-array name="gender_values">
    <item>0</item>
    <item>1</item>
</string-array>
```

PreferenceActivity 예제

- CustomSetting.JAVA

```
public class CustomSetting extends PreferenceActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        addPreferencesFromResource(R.xml.settings);

        PreferenceScreen screen = getPreferenceScreen();
        ListPreference backgrounder = (ListPreference)
                                    screen.findPreference("backgrounder");
        backgrounder.setOnPreferenceChangeListener(
                                    new Preference.OnPreferenceChangeListener() {
```

PreferenceActivity 예제

- CustomSetting.JAVA

```
        @Override
        public boolean onPreferenceChange(Preference preference,
                                          Object newValue) {
            String values = (String) newValue;
            ListPreference listPreference = (ListPreference) preference;
            int index = listPreference.findIndexOfValue(values);
            MainActivity2 activity = (MainActivity2) getApplicationContext();
            activity.getLayout().setBackgroundColor(Color.parseColor(
                (String) backgrounder.getEntryValues()[index]));
            return true;
        }
    });
```

PreferenceActivity 예제

- CustomSetting.JAVA

```
        CheckBoxPreference boxPreference = (CheckBoxPreference)
                                                screen.findPreference("auto");
        boxPreference.setOnPreferenceClickListener(
                                new Preference.OnPreferenceClickListener() {
            @Override
            public boolean onPreferenceClick(Preference preference) {
                CheckBoxPreference checkBoxPreference =
                                                (CheckBoxPreference) preference;
                MainActivity2 activity = (MainActivity2) getApplicationContext();
                if (checkBoxPreference.isChecked())
                    activity.getCheckBox().setChecked(true);
                else
                    activity.getCheckBox().setChecked(false);
                return true;
            }
        });
    }
}
```

PreferenceActivity 예제

- Android Setting 화면 호출 방법
  - Settings의 메소드
    - ACTION_SETTINGS : 기본 설정 화면
    - ACTION_ACCESSIBILITY_SETTINGS : 접근성 설정 화면
    - ACTION_AIRPLANE_MODE_SETTINGS : 비행기 모드 설정 화면
    - ACTION_APPLICATION_SETTINGS : 앱 관련 설정 화면
    - ACTION_BLUETOOTH_SETTINGS : 블루투스 설정 화면
    - ACTION_DATE_SETTINGS : 날짜 및 시간 설정 화면
    - ACTION_DISPLAY_SETTINGS : 디스플레이 설정 화면
    - ACTION_FINGERPRINT_ENROLL : 지문 등록 설정 화면
    - ACTION_INTERNAL_STORAGE_SETTINGS : 내부 저장소 설정 화면
    - ACTION_SOUND_SETTINGS : 사운드 및 볼륨 설정 화면
    - ACTION_WIFI_SETTINGS : 와이파이 설정 화면

PreferenceActivity 예제

- PreferenceActivity
  - Setting 정보를 쉽게 저장하고 관리할 수 있는 Preference 임
  - Data를 설정하는 과정을 UI로도 제공하며 일일이 commit()할 필요 없이 사용할 수 있어 편리하지만 UI 설정을 위한 xml을 만들어주어야 함
- PreferenceActivity 설정 File
  - PreferenceActivity는 Activity가 있지만 기존의 res/layout이 아닌 xml 폴더에 xml File을 생성하여 저장
  - res/xml 폴더에 PreferenceActivity xml File을 생성

PreferenceFragmentCompat

- PreferenceActivity와 Fragment의 차이
  - PreferenceActivity
    - Android 5.0(API 21) 이전에 주로 사용
    - 설정 화면을 한 번에 모두 표시
    - Multi Panel UI를 지원하지 않음
    - API 28 이상에서는 PreferenceFragmentCompat 사용을 권장
  - PreferenceFragmentCompat
    - AndroidX의 PreferenceFragmentCompat를 사용하면 보다 현대적인 설정 화면을 구현할 수 있음
    - Multi Panel UI를 지원하여 Tablet 등 큰 화면에서 유리
    - 더 많은 기능과 유연성을 제공

PreferenceFragmentCompat

- 주요 설정 항목

설정 항목	설명
EditTextPreference	사용자가 Text를 입력하여 값을 설정할 수 있음
SwitchPreferenceCompat	Switch를 켜고 끌 수 있는 설정 항목
CheckBoxPreference	CheckBox를 사용하여 값을 설정 (SwitchPreferenceCompat 사용 권장)
ListPreference	DropDown 목록에서 값을 선택할 수 있는 설정 항목
PreferenceCategory	설정 항목을 Group화하여 시각적으로 구분
SeekBarPreference	Slider를 사용하여 숫자 값을 설정
MultiSelectListPreference	다중 선택 가능한 목록 설정 항목

PreferenceFragmentCompat 예제

- MainActivity.JAVA

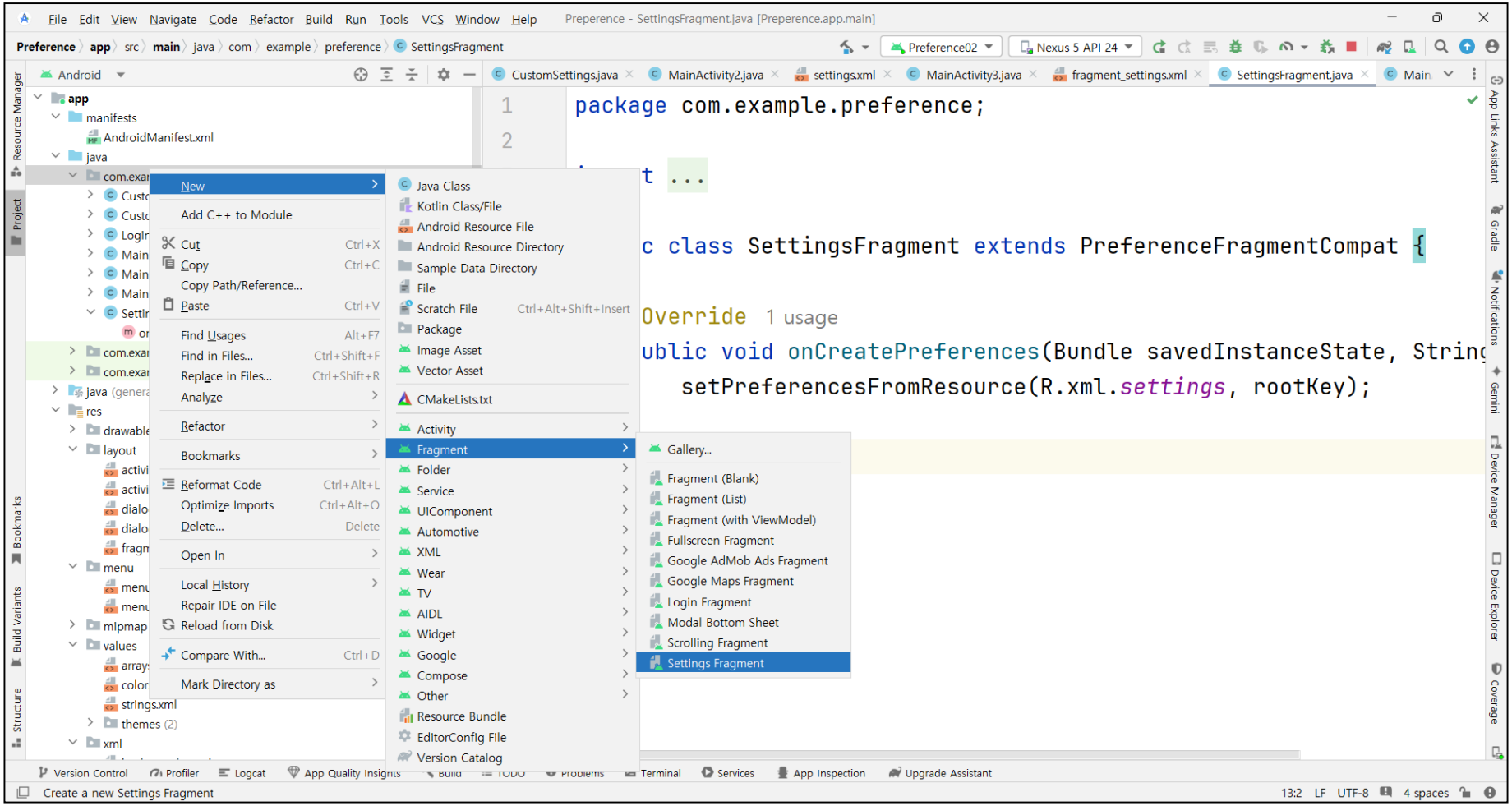
```
public class MainActivity3 extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        FragmentManager manager = getSupportFragmentManager();
        FragmentTransaction transaction = manager.beginTransaction();
        transaction.replace(android.R.id.content, new SettingsFragment());
        transaction.commit();
    }
}
```

PreferenceFragmentCompat 예제

- Settings Fragment



PreferenceFragmentCompat 예제

- SettingsFragment.JAVA

```
public class SettingsFragment extends PreferenceFragmentCompat {

    @Override
    public void onCreatePreferences(Bundle savedInstanceState, String rootKey) {
        setPreferencesFromResource(R.xml.root_preferences, rootKey);
    }
}
```

PreferenceFragmentCompat 예제

- CustomSettings.JAVA

```
public class CustomSettings extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        FragmentManager manager = getSupportFragmentManager();
        FragmentTransaction transaction = manager.beginTransaction();
        transaction.replace(android.R.id.content, new SettingsFragment());
        transaction.commit();
    }
}
```