



안드로이드 앱 프로그래밍

Chapter 07

서비스와 수신자 이해하기



이번 장에서는 무엇을 다룰까요?



화면이 없이도 동작하는 기능을 만들고 싶어요



- 화면이 없는 기능은 어떻게 만드는 것이 좋을까요?
- SMS 문자를 수신할 때 앱에서 알 수 있는 방법이 있나요?
- 위험 권한이라는 것은 어떻게 부여하나요?
- 매니페스트 파일에 들어있는 내용을 좀 더 알고 싶어요.





이번 장에서는 무엇을 다룰까요?



서비스는 어떻게 사용하나요?

- 서비스 살펴보기



수신자는 어떻게 사용하나요?

- 수신자 살펴보기



위험권한을 위한 코드는 어떻게 추가하나요?

- 위험권한 부여하기



리소스에 대해 좀 더 알고 싶어요

- 리소스와 매니페스트 이해하기



그래들이 무엇인가요?

- 그래들 이해하기





강의 주제

서비스와 브로드캐스트 수신자에 대한 이해



1

서비스

2

브로드캐스트 수신자 이해하기

3

위험권한 부여하기

4

리소스와 매니페스트 이해하기

5

그래들 이해하기

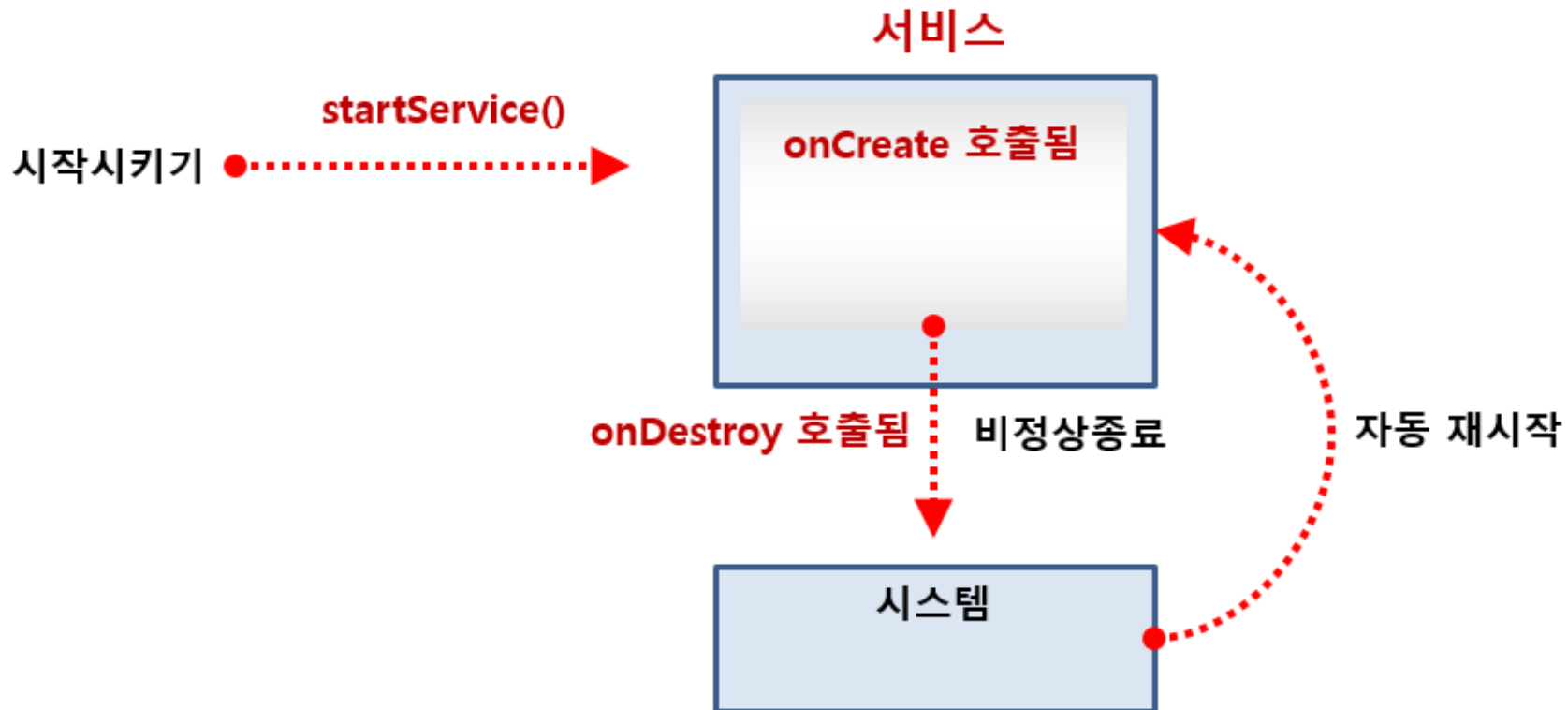
1.

서비스



자동으로 재시작되는 서비스

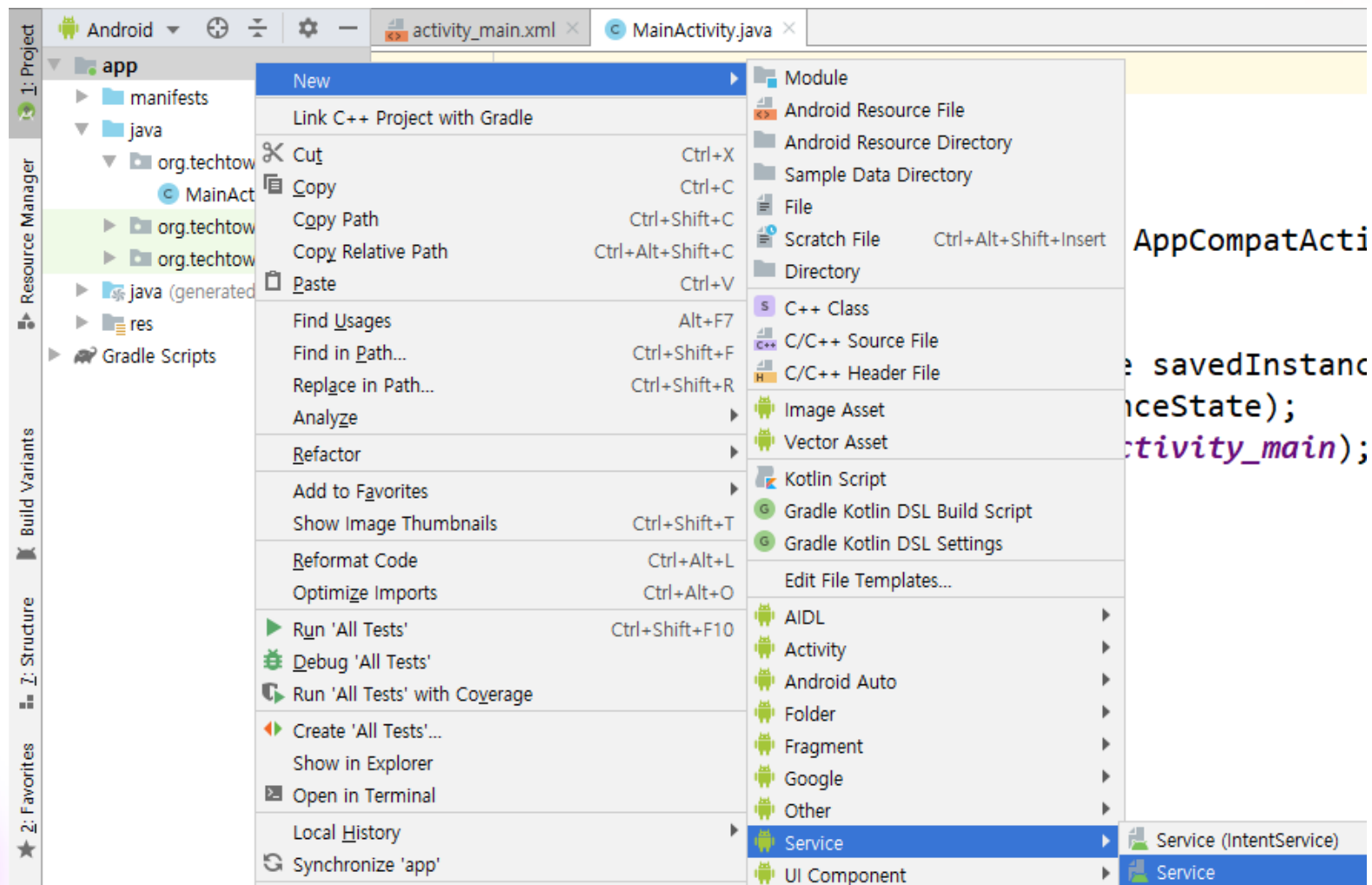
- 서비스는 화면이 없는 상태에서 백그라운드로 실행됨
- 서비스는 프로세스가 종료되어도 시스템에서 자동으로 재시작함





새로운 서비스 추가

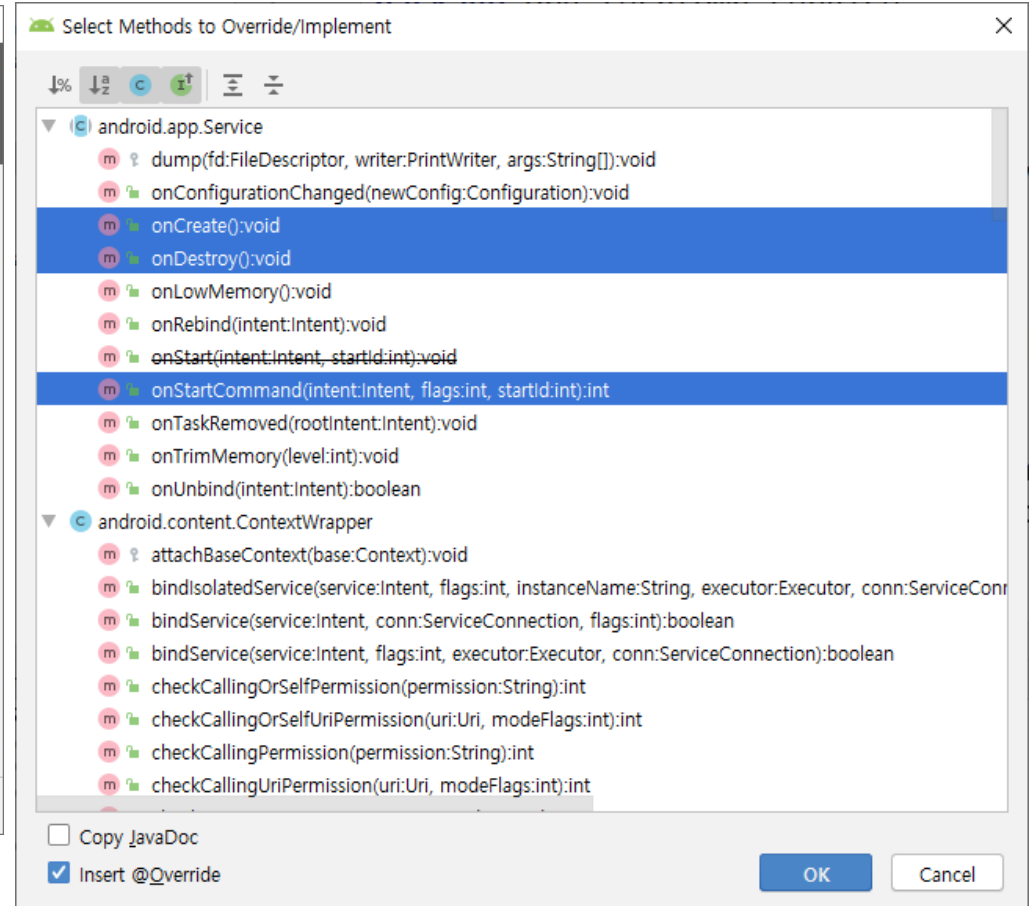
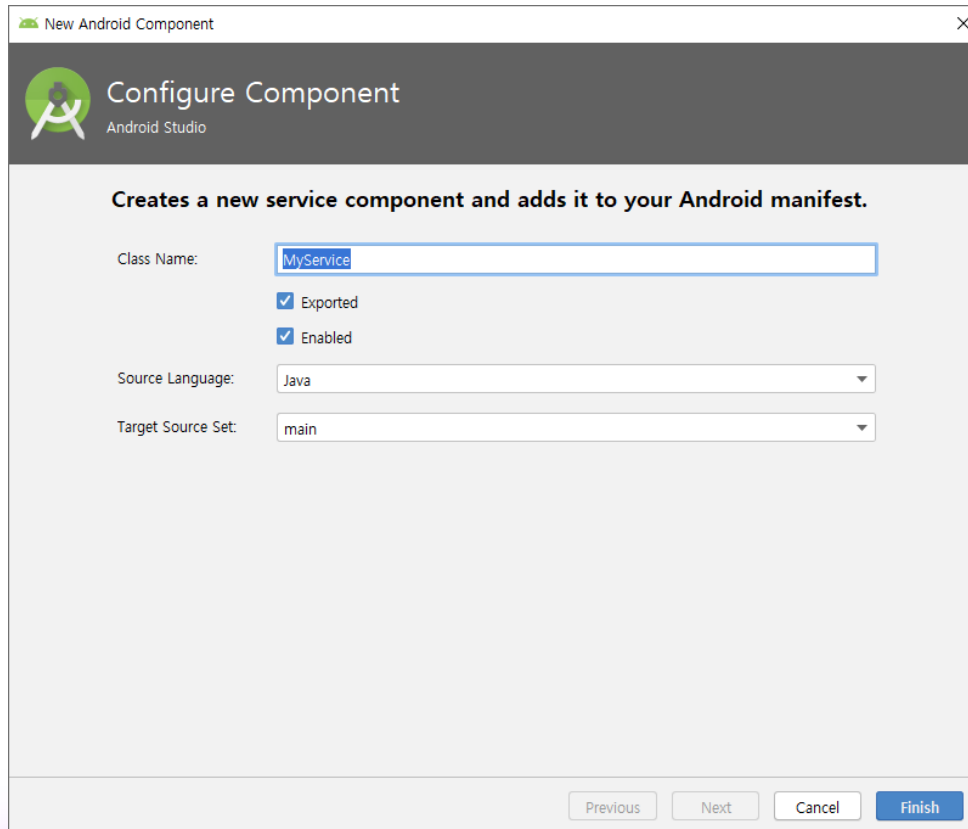
- 프로젝트 영역에서 New → Service → Service 메뉴를 이용해 서비스 추가





서비스 추가를 위한 대화상자

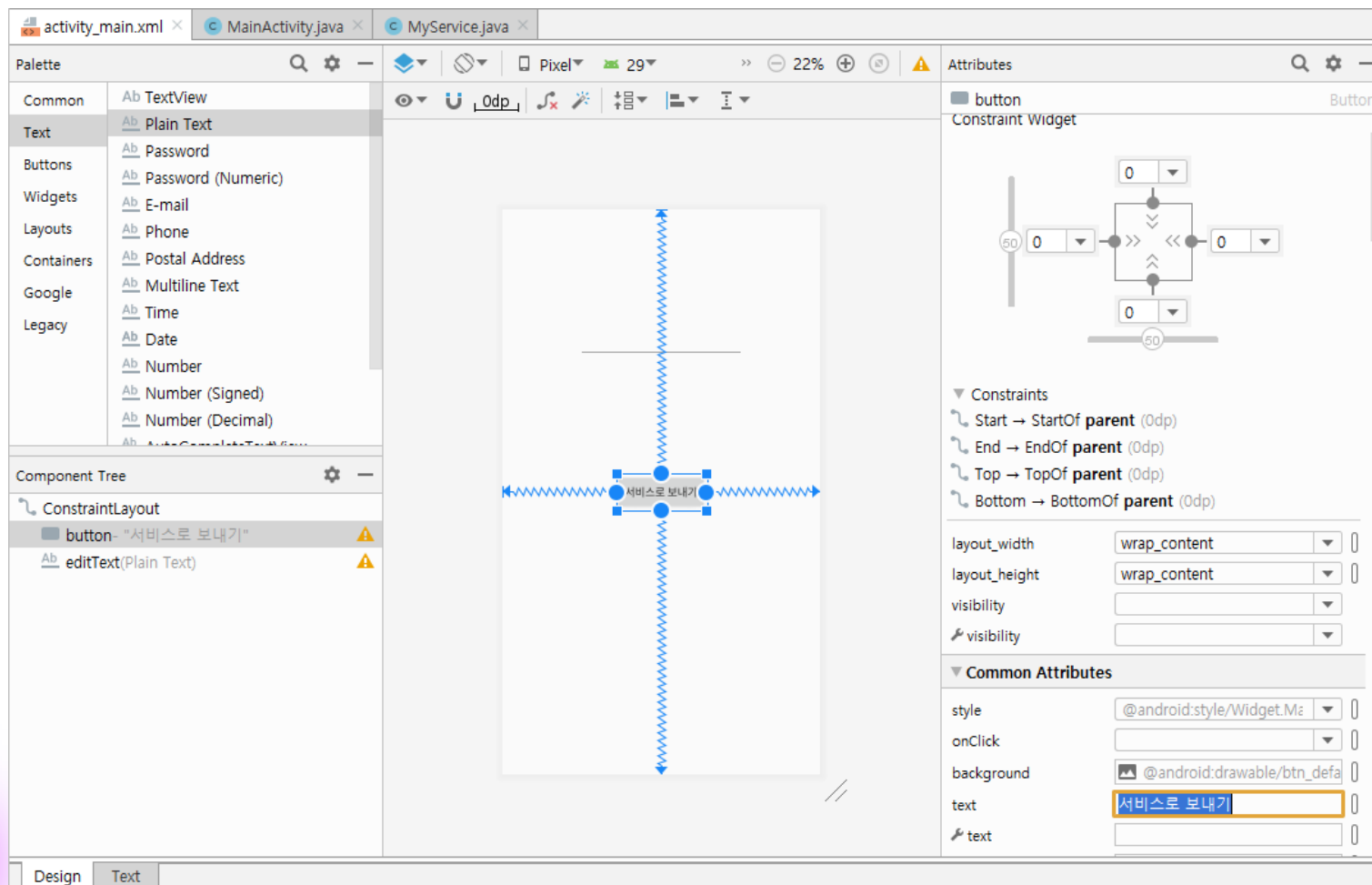
- 대화상자에서 서비스명을 입력하고 [Finish] 버튼을 누름
- 대표적인 세 개 메소드, onCreate, onDestroy, onStartCommand 를 재정의함





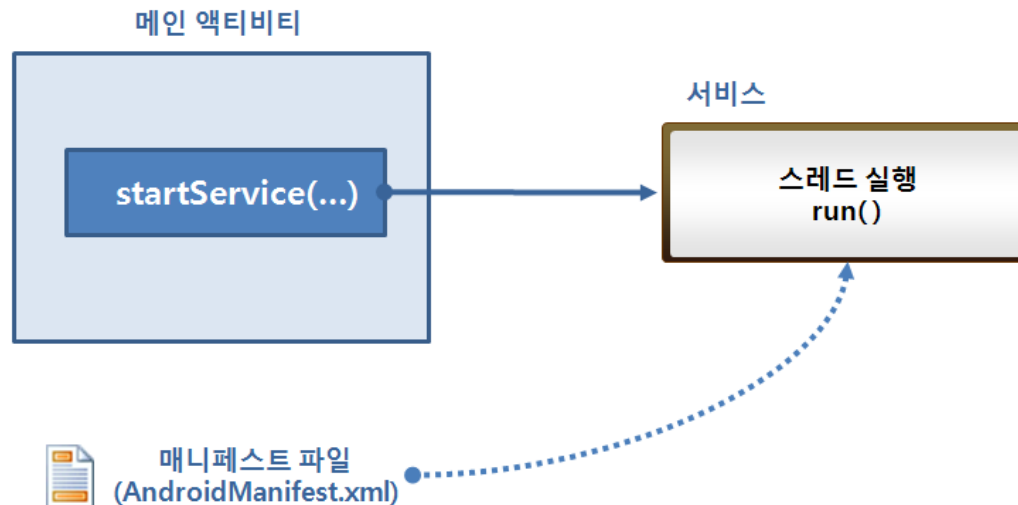
액티비티에서 서비스 시작

- 서비스 시작을 위해 startService 메소드를 호출
- 메인 액티비티에서 버튼 누르면 시작하도록 할 수 있음





서비스를 시작, 중지시키는 메소드



- 서비스는 **백그라운드**에서 실행되는 애플리케이션 구성 요소
- 서비스는 매니페스트 파일(AndroidManifest.xml) 안에 **<service>** 태그를 이용하여 선언
- 서비스를 시작/중지시키는 메소드
 - Context.startService()
 - Context.bindService()
 - stopService(...)
 - unbindService(...)
- 서비스는 다른 구성 요소들처럼 메인 스레드에서 동작
따라서 CPU를 많이 쓰거나 대기 상태(blocking)를 필요로 하는 작업들은 **스레드를 새로** 만들어 주어야 함



서비스 클래스 정의

```
public class MyService extends Service {  
    ...  
    @Override  
    public void onCreate() {  
        super.onCreate();  
        Log.d(TAG, "onCreate() 호출됨.");  
    }  
  
    @Override  
    public int onStartCommand(Intent intent, int flags, int startId) {  
        Log.d(TAG, "onStartCommand() 호출됨.");  
  
        if (intent == null) {  
            return Service.START_STICKY;  
        } else {  
            processCommand(intent);  
        }  
  
        return super.onStartCommand(intent, flags, startId);  
    }  
    ...  
}
```

Continued..



서비스 클래스 정의

...

```
private void processCommand(Intent intent) {  
    String command = intent.getStringExtra("command");  
    String name = intent.getStringExtra("name");  
    Log.d(TAG, "command : " + command + ", name : " + name);  
  
    for (int i = 0; i < 5; i++) {  
        try {  
            Thread.sleep(1000);  
        } catch (Exception e) {}  
  
        Log.d(TAG, "Waiting " + i + " seconds.");  
    }  
}
```

Continued..



메인 액티비티 코드 만들기

참조파일 SampleService>/app/java/org.techtown.service/MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    EditText editText;

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

        editText = findViewById(R.id.editText);

        Button button = findViewById(R.id.button);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String name = editText.getText().toString();

                Intent intent = new Intent(getApplicationContext(), MyService.class);
                intent.putExtra("command", "show");
                intent.putExtra("name", name);

                startService(intent); → ② 서비스 시작하기
            }
        });
    }
}
```

① 인텐트 객체 만들고
부가 데이터 넣기



서비스는 매니페스트에 자동 추가됨

...

<application ... >

...

<service android:name = *"MyService"* >

1

서비스 등록

</service>

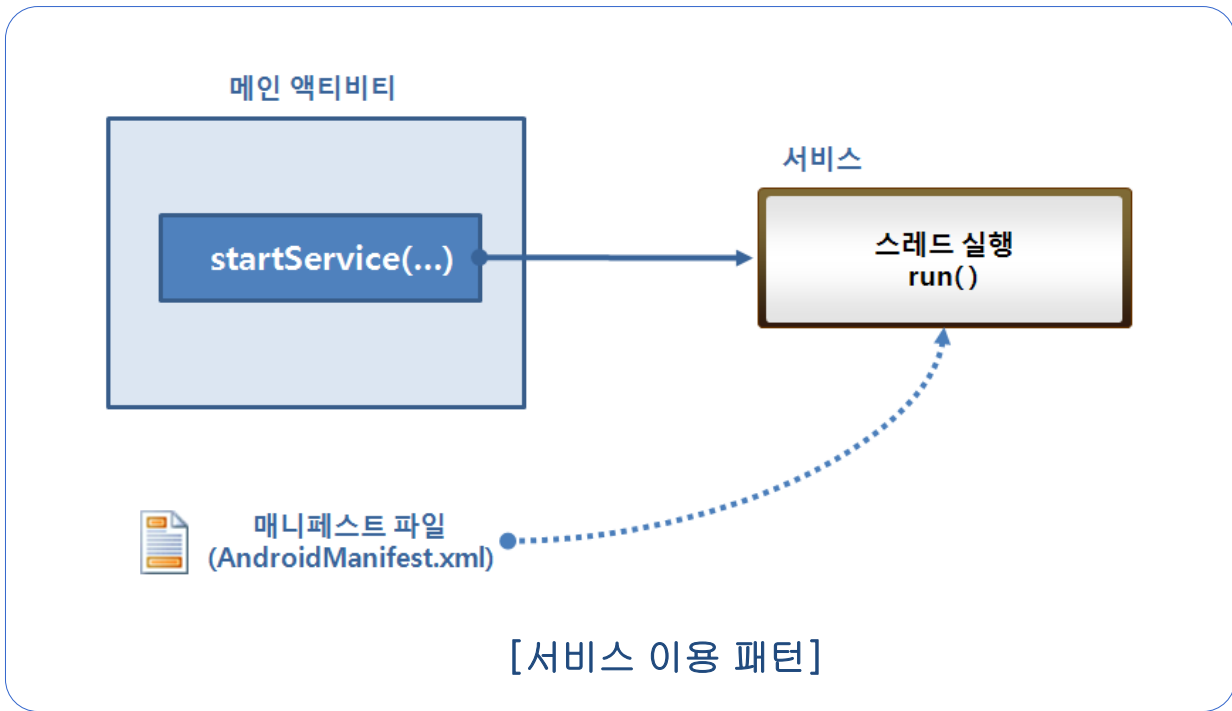
</application>

...



서비스 실행하여 로그 확인

```
Logcat
Emulator Nexus_5X_API_28_Andr org.techtown.service (9847) Verbose Q Regex Unnamed-0
2018-11-29 19:48:23.305 9847-9847/org.techtown.service D/MyService: onCreate() 호출됨.
2018-11-29 19:48:23.306 9847-9847/org.techtown.service D/MyService: onStartCommand() 호출됨.
2018-11-29 19:48:23.309 9847-9847/org.techtown.service D/MyService: command : show, name : mike
2018-11-29 19:48:24.311 9847-9847/org.techtown.service D/MyService: Waiting 0 seconds.
2018-11-29 19:48:25.313 9847-9847/org.techtown.service D/MyService: Waiting 1 seconds.
2018-11-29 19:48:26.314 9847-9847/org.techtown.service D/MyService: Waiting 2 seconds.
2018-11-29 19:48:27.326 9847-9847/org.techtown.service D/MyService: Waiting 3 seconds.
```





서비스에서 화면 띄우기

- 서비스에서 액티비티를 띄울 수 있음
- 플래그를 이용해 한 번 만들어진 액티비티를 그대로 띄움

```
private void processCommand(Intent intent) {  
    ...  
  
    Intent showIntent = new Intent(getApplicationContext(), MainActivity.class);  
    showIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK |  
        Intent.FLAG_ACTIVITY_SINGLE_TOP |  
        Intent.FLAG_ACTIVITY_CLEAR_TOP);  
    showIntent.putExtra("command", "show");  
    showIntent.putExtra("name", name + " from service.");  
    startActivity(showIntent);  
}
```




액티비티에서 인텐트 받아 처리하기

- 액티비티가 이미 메모리에 만들어져 있는 경우 `onNewIntent` 메소드 호출됨

`@Override`

```
protected void onNewIntent(Intent intent) {  
    processIntent(intent);  
    super.onNewIntent(intent);  
}  
  
private void processIntent(Intent intent) {  
    if (intent != null) {  
        String command = intent.getStringExtra("command");  
        String name = intent.getStringExtra("name");  
  
        Toast.makeText(this, "command : " + command + ", name : " + name,  
                        Toast.LENGTH_LONG).show();  
    }  
}
```



서비스에서 띄운 화면

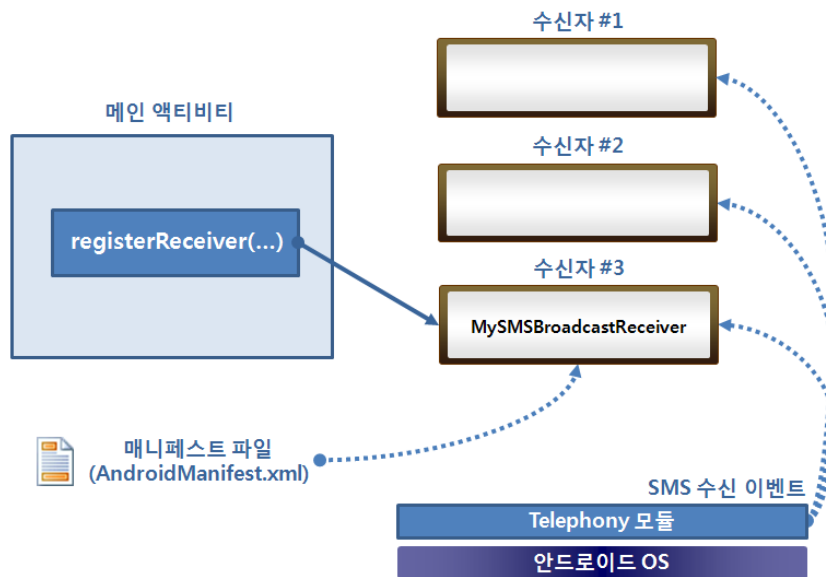


2.

브로드캐스트 수신자 이해하
기



브로드캐스트 수신자



- 애플리케이션이 글로벌 이벤트(global event)를 받아서 처리하려면 브로드캐스트 수신자로 등록
- 글로벌 이벤트란 “전화가 왔습니다.”, “문자 메시지가 도착했습니다.”와 같이 안드로이드 시스템 전체에 보내지는 이벤트
- 브로드캐스트 수신자는 인텐트필터를 포함하며, 매니페스트 파일에 등록함으로써 인텐트를 받을 준비를 함
- 수신자가 매니페스트 파일에 등록되었다면 따로 시작시키지 않아도 됨
- 애플리케이션은 컨텍스트 클래스의 registerReceiver 메소드를 이용하면 런타임 시에도 수신자를 등록할 수 있음
- 서비스처럼 브로드캐스트 수신자도 비가 없음



브로드캐스트의 구분

• 인텐트와 브로드캐스트

- 인텐트를 이용해서 액티비티를 실행하면 포그라운드(background)로 실행되어 사용자에게 보여지지만
- 브로드캐스트를 이용해서 처리하면 백그라운드(background)로 동작하므로 사용자가 모름
- 인텐트를 받으면 onReceive() 메소드가 자동으로 호출됨

• 브로드캐스트의 구분

브로드캐스트는 크게 두 가지 클래스로 구분됨

- 일반 브로드캐스트 (sendBroadcast() 메소드로 호출)

비동기적으로 실행되며 모든 수신자는 순서없이 실행됨 (때로는 동시에 실행됨)

효율적이나, 한 수신자의 처리 결과를 다른 수신자가 이용할 수 없고 중간에 취소불가

- 순차 브로드캐스트 (sendOrderedBroadcast() 메소드로 호출)

한 번에 하나의 수신자에게만 전달되므로 순서대로 실행됨. 중간에 취소하면 그 다음

수신자는 받지 못함. 수신자가 실행되는 순서는 인텐트필터의 속성으로 정할 수 있음

순서가 같으면 임의로 실행됨.



브로드캐스트 수신자 예제

브로드캐스트 수신자 예제

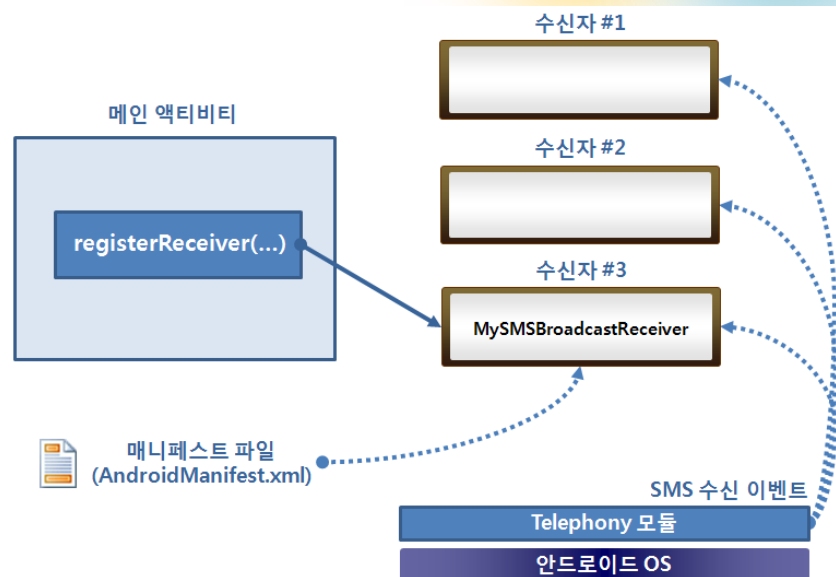
- 브로드캐스트 수신자로 SMS 수신 확인하기
- 브로드캐스트 수신자 정의

브로드캐스트 수신자 정의

- 일정 시간간격으로 메시지를 보여 주는 서비스 클래스 정의

매니페스트에 추가

- 새로운 브로드캐스트 수신자를 매니페스트에 추가





새로운 브로드캐스트 수신자 추가

- SampleReceiver 프로젝트 생성
- New → Other → Broadcast Receiver 메뉴로 브로드캐스트 수신자 추가

New Android Component

Configure Component
Android Studio

Creates a new **broadcast receiver** component and adds it to your **Android manifest**.

Class Name:

☒ Exported

☒ Enabled

Source Language:

Target Source Set:

Previous Next Cancel Finish



매니페스트에 추가

```
<application
```

```
...
```

```
<receiver
```

```
    android:name=".SmsReceiver"
```

```
    android:enabled="true"
```

```
    android:exported="true">
```

```
    <intent-filter>
```

```
        <action android:name="android.provider.Telephony.SMS_RECEIVED"/>
```

```
    </intent-filter>
```

```
</receiver>
```

```
</application>
```




브로드캐스트 수신자 클래스 정의

```
public class SmsReceiver extends BroadcastReceiver {  
    public static final String TAG = "SmsReceiver";  
  
    @Override  
    public void onReceive(Context context, Intent intent) {  
        Log.i(TAG, "onReceive() 메소드 호출됨.");  
  
        Bundle bundle = intent.getExtras();  
        SmsMessage[] messages = parseSmsMessage(bundle);  
        if (messages != null && messages.length > 0) {  
            String sender = messages[0].getOriginatingAddress();  
            Log.i(TAG, "SMS sender : " + sender);  
            String contents = messages[0].getMessageBody().toString();  
            Log.i(TAG, "SMS contents : " + contents);  
            Date receivedDate = new Date(messages[0].getTimestampMillis());  
            Log.i(TAG, "SMS received date : " + receivedDate.toString());  
        }  
    }  
}
```

...



브로드캐스트 수신자 클래스 정의

```
...  
private SmsMessage[] parseSmsMessage(Bundle bundle) {  
    Object[] objs = (Object[]) bundle.get("pdus");  
    SmsMessage[] messages = new SmsMessage[objs.length];  
  
    int smsCount = objs.length;  
    for (int i = 0; i < smsCount; i++) {  
        // PDU 포맷으로 되어 있는 메시지를 복원합니다.  
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) { // API 23 이상  
            String format = bundle.getString("format");  
            messages[i] = SmsMessage.createFromPdu((byte[]) objs[i], format);  
        } else {  
            messages[i] = SmsMessage.createFromPdu((byte[]) objs[i]);  
        }  
    }  
  
    return messages;  
}  
...
```



권한 추가

```
<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
    package="org.androidtown.samplerceiver" >  
  
    <uses-permission android:name="android.permission.RECEIVE_SMS" />  
  
    ...
```



Sync Now 버튼 클릭

```
activity_main.xml x MainActivity.java x SmsReceiver.java x AndroidManifest.xml x app x
Configure project in Project Structure dialog. Open Project Structure Hide notification
22 allprojects {
23     repositories {
24         maven { url 'https://jitpack.io' }
25     }
26 }
27
28 dependencies {
29     implementation fileTree(dir: 'libs', include: ['*.jar'])
30     implementation 'androidx.appcompat:appcompat:1.1.0'
31     implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
32     testImplementation 'junit:junit:4.12'
33     androidTestImplementation 'androidx.test.ext:junit:1.1.1'
34     androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0'
35
36     implementation 'com.github.pedroSG94:AutoPermissions:1.0.3'
37 }
38
```



위험 권한 부여 코드 추가

참조파일 SampleReceiver>/app/java/org.techtown.receiver/MainActivity.java

```
public class MainActivity extends AppCompatActivity
    implements AutoPermissionsListener {

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

        AutoPermissions.Companion.loadAllPermissions(this, 101);

    }

    @Override public void onRequestPermissionsResult(int requestCode, String permissions[],
        int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        AutoPermissions.Companion.parsePermissions(this, requestCode, permissions, this);
    }

    @Override
    public void onDenied(int requestCode, @NotNull String[] permissions) {
        Toast.makeText(this, "permissions denied : " + permissions.length,
            Toast.LENGTH_LONG).show();
    }

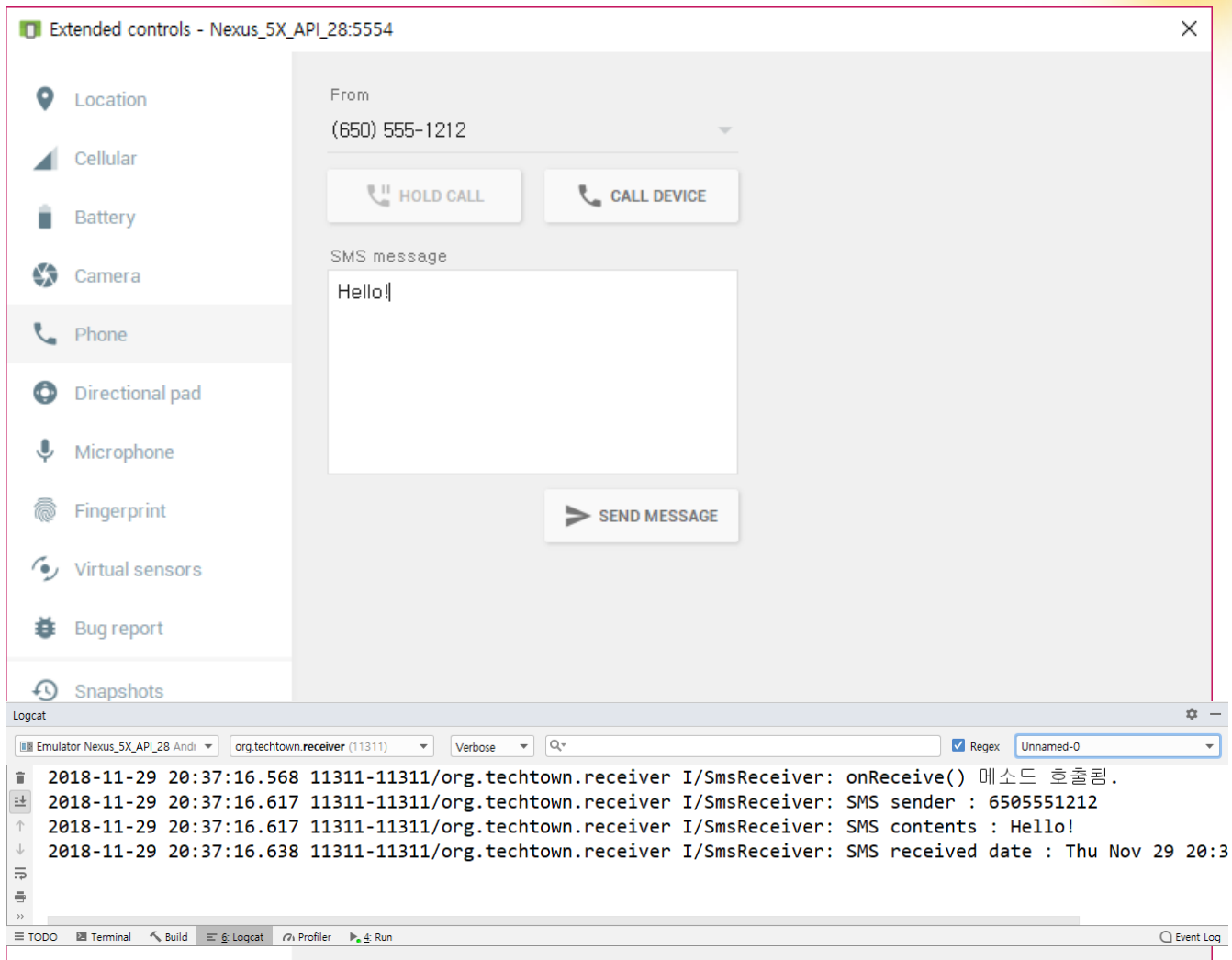
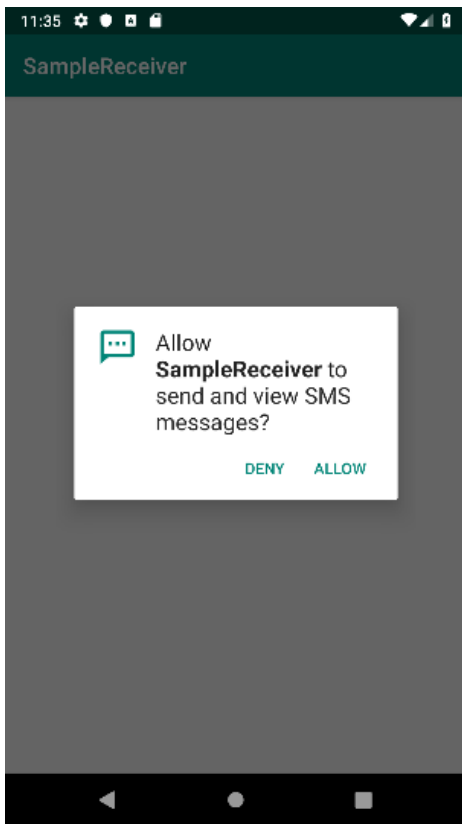
    @Override public void onGranted(int requestCode, @NotNull String[] permissions) {
        Toast.makeText(this, "permissions granted : " + permissions.length,
            Toast.LENGTH_LONG).show();
    }
}
```

① MainActivity가 인터페이스 구현하도록 하기

→ ② 모든 위험 권한을 자동 부여하도록 하는 메서드 호출하기



브로드캐스트 수신자 실행 화면



[Extended Controls 에서 에뮬레이터로 SMS를 보내고 로그가 출력된 화면]



SMS 문자를 보여줄 새로운 화면 추가

The screenshot shows the Android Studio interface. On the left, the 'Configure Activity' dialog is open, showing the configuration for a new activity named 'SmsActivity'. The dialog includes fields for Activity Name, Layout Name, Package name, Source Language, and Target Source Set. The 'Generate Layout File' checkbox is checked. Below the dialog, a preview of the activity layout is shown. On the right, the 'SmsActivity.xml' layout file is open in the Design view. The layout contains a text input field labeled '발신번호', a text input field labeled '내용', a text input field labeled '수신사', and a button labeled '확인'. The 'Attributes' panel on the right shows the properties for the selected 'editText2' widget, including its ID, declared attributes, layout constraints, and common attributes.



SMS 문자를 보여줄 새로운 화면 추가

```
...
@Override
protected void onNewIntent(Intent intent) {
    processIntent(intent);

    super.onNewIntent(intent);
}

private void processIntent(Intent intent) {
    if (intent != null) {
        String sender = intent.getStringExtra("sender");
        String contents = intent.getStringExtra("contents");
        String receivedDate = intent.getStringExtra("receivedDate");

        editText.setText(sender);
        editText2.setText(contents);
        editText3.setText(receivedDate);
    }
}
...
```



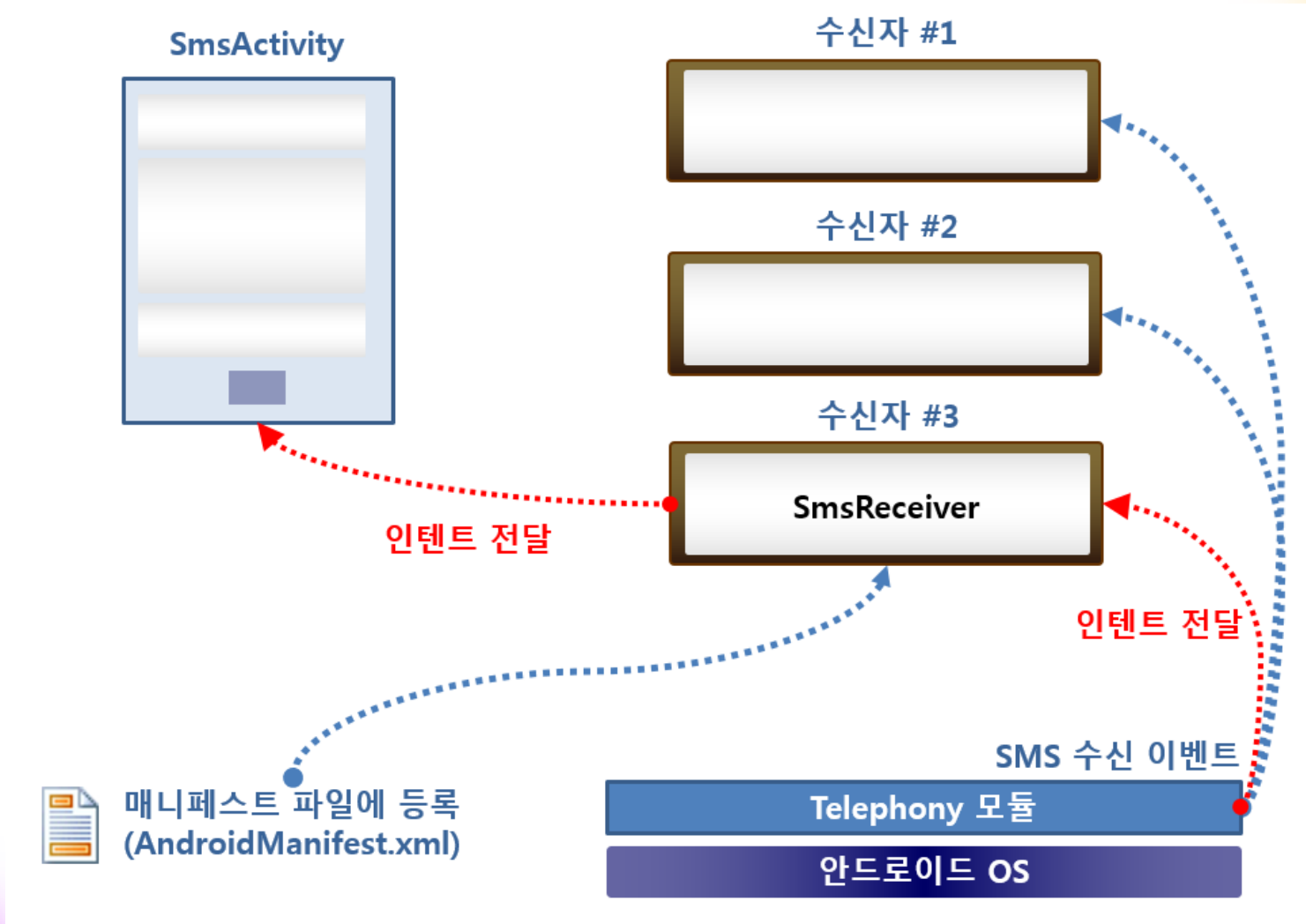

서비스에서 화면 띄우기

```
...  
private void sendToActivity(Context context, String sender, String contents, Date receivedDate) {  
    // 메시지를 보여줄 액티비티를 띄워줍니다.  
    Intent myIntent = new Intent(context, SmsActivity.class);  
  
    // 플래그를 이용합니다.  
    myIntent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK |  
        Intent.FLAG_ACTIVITY_SINGLE_TOP | Intent.FLAG_ACTIVITY_CLEAR_TOP);  
  
    myIntent.putExtra("sender", sender);  
    myIntent.putExtra("contents", contents);  
    myIntent.putExtra("receivedDate", format.format(receivedDate));  
  
    context.startActivity(myIntent);  
}  
...
```





브로드캐스트 수신자 동작 방식

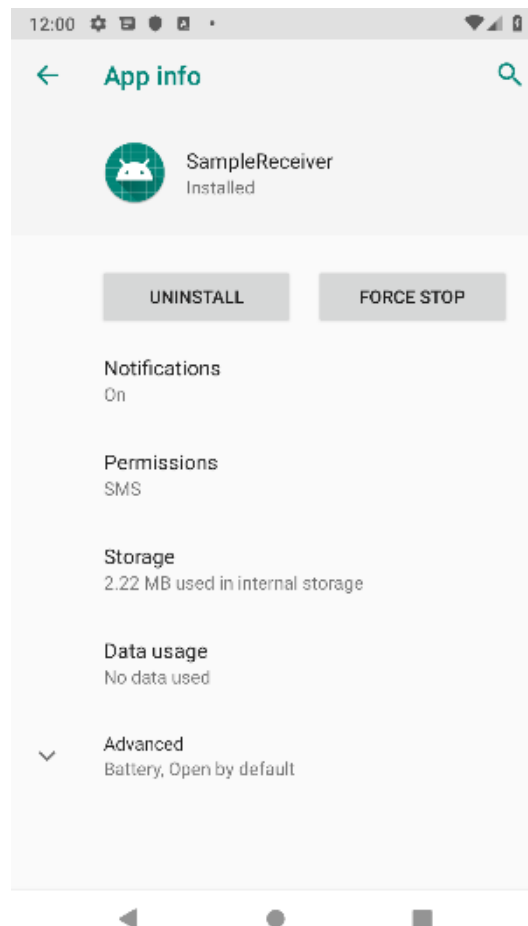




화면을 수정해도 이전 화면이 계속 보여요!

- 프로젝트를 복사하여 새로운 프로젝트를 만든 경우

- SMS를 수신하는 동일한 앱이 설치되어 있다면 SMS 수신 시 그 화면이 보여질 수 있음
- 따라서, 이전에 설치한 앱을 삭제한 후 새로운 앱을 실행해야 함
- 단말의 설정에서 앱을 삭제할 수 있음



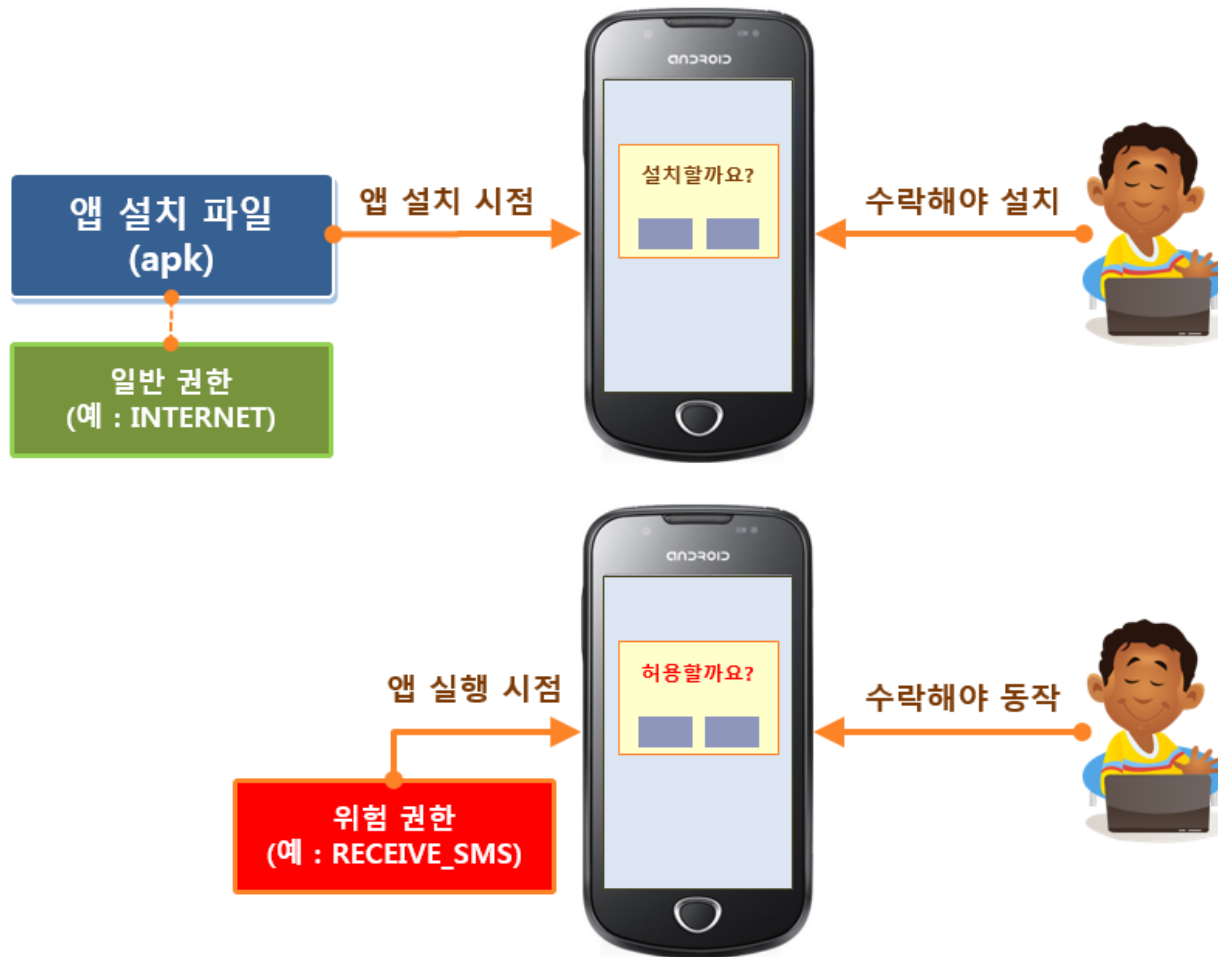
3.

위험 권한 부여하기



일반 권한과 위험 권한 (마시멜로 API23부터)

- 위험 권한은 실행 시 권한 부여





대표적인 위험 권한들

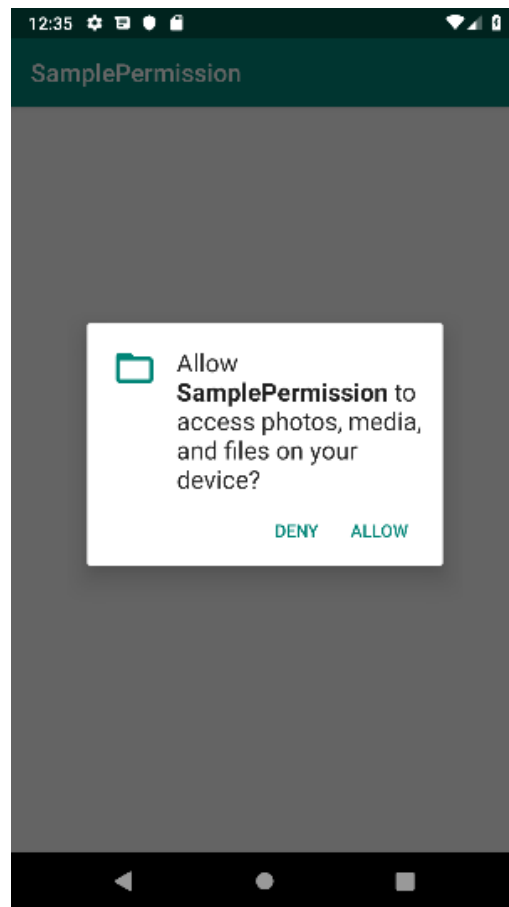


LOCATION (위치)
- ACCESS_FINE_LOCATION
- ACCESS_COARSE_LOCATION
CAMERA
- CAMERA
MICROPHONE
- RECORD_AUDIO
CONTACTS
- READ_CONTACTS
- WRITE_CONTACTS
- GET_ACCOUNTS
PHONE
- READ_PHONE_STATE
- CALL_PHONE
- READ_CALL_LOG
- WRITE_CALL_LOG
- ADD_VOICEMAIL
- USE_SIP
- PROCESS_OUTGOING_CALLS
SMS
- SEND_SMS
- RECEIVE_SMS
- READ_SMS
- RECEIVE_WAP_PUSH
- RECEIVE_MMS
CALENDAR
- READ_CALENDAR
- WRITE_CALENDAR
SENSORS
- BODY_SENSORS
STORAGE
- READ_EXTERNAL_STORAGE
- WRITE_EXTERNAL_STORAGE



실행 시 권한 부여

- 실행 시 권한 부여를 묻는 대화상자 표시





매니페스트에 권한 추가

참조파일 SamplePermission>/app/manifests/AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="org.techtown.permission">

    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

    <application
```

종락...



위험 권한 부여 요청 코드 추가

```
public void checkPermissions(String[] permissions) {
    ArrayList<String> targetList = new ArrayList<String>();

    for (int i = 0; i < permissions.length; i++) {
        String curPermission = permissions[i];
        int permissionCheck = ContextCompat.checkSelfPermission(this, curPermission);
        if (permissionCheck == PackageManager.PERMISSION_GRANTED) {
            Toast.makeText(this, curPermission + " 권한 있음.", Toast.LENGTH_LONG).show();
        } else {
            Toast.makeText(this, curPermission + " 권한 없음.", Toast.LENGTH_LONG).show();
            if (ActivityCompat.shouldShowRequestPermissionRationale(this, curPermission)) {
                Toast.makeText(this, curPermission + " 권한 설명 필요함.",
                    Toast.LENGTH_LONG).show();
            } else {
                targetList.add(curPermission);
            }
        }
    }

    String[] targets = new String[targetList.size()];
    targetList.toArray(targets);

    ActivityCompat.requestPermissions(this, targets, 101); —→ ② 위험 권한 부여 요청하기
```



권한 요청 결과 확인 코드

```
@Override
public void onRequestPermissionsResult(int requestCode, String permissions[],
                                      int[] grantResults) {

    switch (requestCode) {  ①
        case 1: {
            if (grantResults.length > 0 &&
                grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                Toast.makeText(this, "SMS 권한을 사용자가 승인함.",
                              Toast.LENGTH_LONG).show();
            } else {
                Toast.makeText(this, "SMS 권한 거부됨.", Toast.LENGTH_LONG).show();
            }
        }

        return;
    }
}
```

 ②



위험 권한 자동 부여

참조파일 SamplePermission2>Gradle Scripts>build.gradle(Module:app)

중략...

```
allprojects {  
    repositories {  
        maven { url 'https://jitpack.io' }  
    }  
}
```

```
dependencies {
```

중략...

```
    implementation 'com.github.pedroS694:AutoPermissions:1.0.3'  
}
```



위험 권한 자동 부여 코드 추가

참조파일 SamplePermission2>/app/java/org.techtown.permission2/MainActivity.java

```
public class MainActivity extends AppCompatActivity
    implements AutoPermissionsListener {

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

        AutoPermissions.Companion.loadAllPermissions(this, 101); —→ ❶
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, String permissions[],
                                           int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        AutoPermissions.Companion.parsePermissions(this, requestCode, permissions, this);
    }

    @Override
    public void onDenied(int requestCode, @NotNull String[] permissions) {
        Toast.makeText(this, "permissions denied : " + permissions.length,
            Toast.LENGTH_LONG).show();
    }

    @Override
    public void onGranted(int requestCode, @NotNull String[] permissions) {
        Toast.makeText(this, "permissions granted : " + permissions.length,
            Toast.LENGTH_LONG).show();
    }
}
```



4.

리소스와 매니페스트 이해하
기



매니페스트의 태그 항목

[Reference]

<action> <permission>

<activity> <permission-group>

<activity-alias>

<application> <provider>

<category> <receiver>

<data> <service>

<grant_uri_permission> <uses_configuration>

<instrumentation> <uses-library>

<intent-filter> <uses-permission>

<manifest> <uses-sdk>

<meta-data>



매니페스트 파일의 역할

- 애플리케이션의 자바 패키지 이름 지정
- 애플리케이션 구성요소에 대한 정보 등록(액티비티, 서비스, 브로드캐스트 수신자, 내용 제공자)
- 각 구성요소를 구현하는 클래스 이름 지정
- 애플리케이션이 가져야 하는 권한에 대한 정보 등록
- 다른 애플리케이션이 접근하기 위해 필요한 권한에 대한 정보 등록
- 애플리케이션 개발 과정에서 프로파일링을 위해 필요한 instrumentation 클래스 등록
- 애플리케이션에 필요한 안드로이드 API의 레벨 정보 등록
- 애플리케이션에서 사용하는 라이브러리 리스트

[Code]

[매니페스트 파일의 기본 구조]

```
<manifest ... >
<application ... >
...
<service
android:name="org.techtown.service.MyService" ... >
...
</service>
...
</application>
</manifest>
```



메인 액티비티 정의

[Code]

```
<activity android:name="org.techtown.basicMainActivity"
    android:label="@string/app_name">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```




리소스 사용

- 리소스를 자바 코드와 분리하는 이유는 이해하기 쉽고 유지관리가 용이하기 때문임
- 프로젝트를 처음에 만들면 `[/res]` 폴더와 `[/assets]` 폴더가 따로 분리되어 있는데 두 가지 모두 리소스라고 할 수 있으며 대부분은 `[/res]` 폴더 밑에서 관리됨

- 애셋(Asset)은 동영상이나 웹페이지와 같이 용량이 큰 데이터를 의미함
- 리소스는 빌드되어 설치파일에 추가되지만 애셋은 빌드되지 않음



스타일과 테마

- 스타일과 테마는 여러 가지 속성들을 한꺼번에 모아서 정의한 것
- 대표적인 예로는 대화상자를 들 수 있음

[Code]

```
<style name="Alert" parent="android:Theme.Dialog">  
    <item name="android:windowBackground">@drawable/alertBackground</item>  
</style>
```

5.

그래들 이해하기



애플리케이션 빌드 설정

참조파일 SamplePermission2>/Gradle Scripts/build.gradle (Project: SamplePermission2)

```
buildscript {  
    repositories {  
        google()  
        jcenter()  
    }  
  
    dependencies {  
        classpath 'com.android.tools.build:gradle:3.3.0'  
    }  
}  
  
allprojects {  
    repositories {  
        google()  
        jcenter()  
    }  
}  
  
task clean(type: Delete) {  
    delete rootProject.buildDir  
}
```



앱 빌드 설정

참조파일 SamplePermission2>/Gradle Scripts/build.gradle (Module: app)

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 28
    defaultConfig {
        applicationId "org.techtown.permission"
        minSdkVersion 15
        targetSdkVersion 28
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles
                getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }
}

allprojects {
    repositories {
        maven { url 'https://jitpack.io' }
    }
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
}
```



모듈 포함 정보

참조파일 SamplePermission2>/Gradle Scripts/settings.gradle

```
include ':app'
```