6-2 애플리케이션 구성

1. 서비스
2. 브로드캐스트 수신자
3. 리소스와 매니페스트
4. 토스트와 대화상자

1. 서비스

- **서비스(Service)**
  - 일반적으로 화면 없이 동작하는 프로그램을 뜻함
  - 백그라운드 프로세스(Background Process)라고도 함
  - 액티비티 응용프로그램은 화면(액티비티)이 종료되면 동작하지 않지만 서비스는 백그라운드에서 실행되므로 화면과 상관없이 계속 동작함

서비스 생명 주기

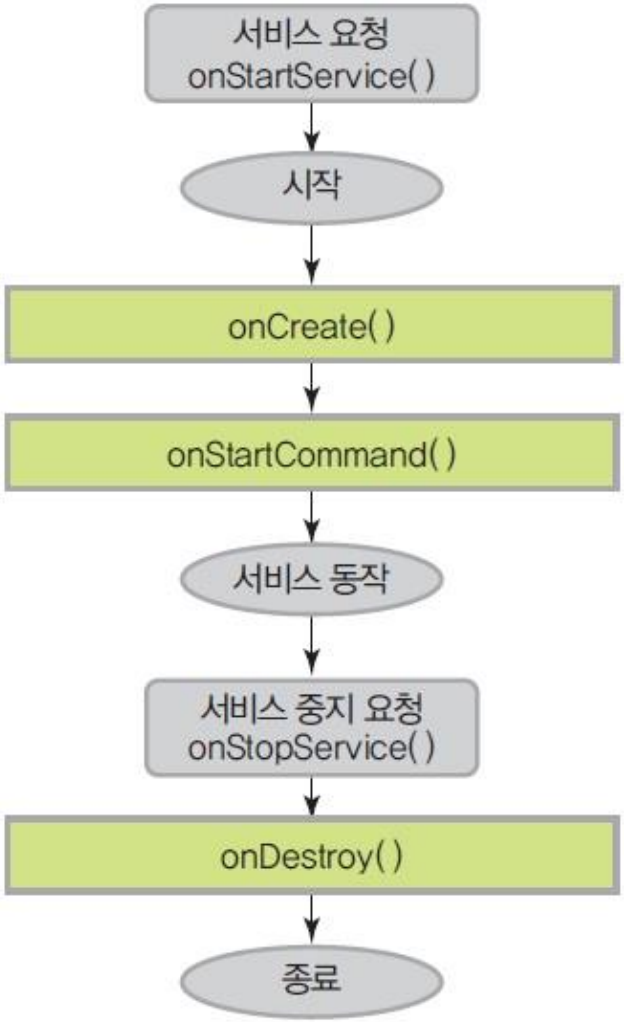


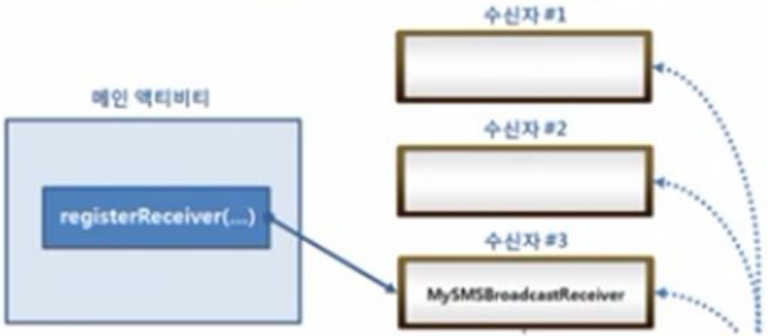
그림 14-1 서비스 생명 주기

실습예제 " p236 참고

2. 브로드캐스트 수신자

- **브로드캐스트 리시버(BR, Broadcast Receiver)**
  - 안드로이드는 문자 메시지 도착, 배터리 방전, SD카드 탈부착, 네트워크 환경 변화 등이 발생하면 방송(Broadcast) 신호를 보내는데, 이런 신호를 받아서 처리하는 것이 리시버임

2. 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 UI가 없음

2. 브로드캐스트 수신자

브로드캐스트 구분

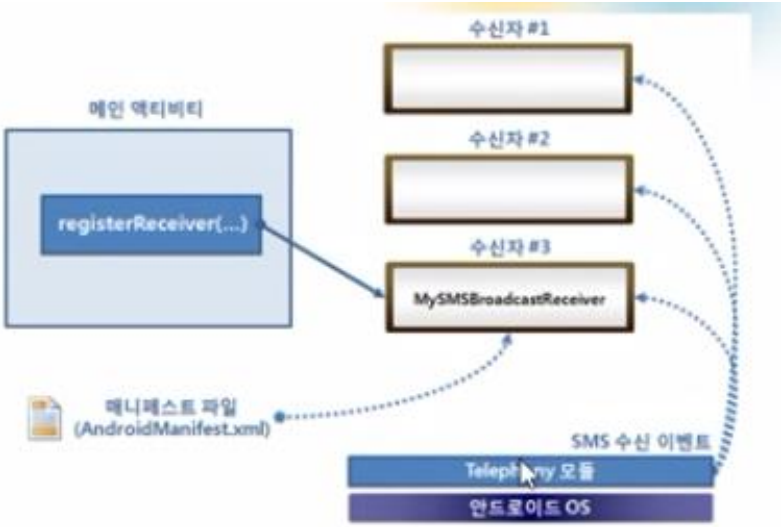
- 인텐트와 브로드 캐스트
  - 인텐트를 이용하여 액티비티를 실행하면 포그라운드(Foreground)로 실행되어 사용자에게 보여지지만
  - 브로드캐스트를 요청하여 처리하면 백그라운드(Background)로 동작하므로 사용자가 모름
  - 인텐트를 받으면 onReceive()메소드가 자동으로 호출됨
- 브로드캐스트의 구분
  - 일반 브로드캐스트(sendBroadcase() 메소드로 호출
    - 비도이적으로 실행되며 모든 수신자는 순서없이 실행됨(때로는 동시에 실행 됨)
    - 효율적이나, 한 수신자의 처리 결과를 다른 수신자가 이용할 수 없고 중간에 취소 불가.
  - 순차 브로드캐스트(sendOrderedBroadcase() 메소드로 호출)
    - 한번에 하나의 수신자에만 전달되므로 순서대로 실행됨, 중간에 취소하면 그 다음 수신자는 받지 못함. 수신자가 실행되는 순서는 인텐트필터의 속성으로 정할 수 있음. 순서가 같으면 임의로 실행됨

2. 브로드캐스트 수신자- 예제

브로드캐스트 수신자 예제

-브로드캐스트 수신자로 SMS 수신 확인하기

-브로드캐스트 수신자 정의



브로드캐스트 수신자 정의

-일정 시간간격으로 메시지를 보여주는 서비스 클래스 정의

매니페스트에 추가

-새로운 브로드캐스트 수신자를 매니페스트에 추가

2. 브로드캐스트 수신자

- 브로드캐스트 리시버의 대표적인 응용은 배터리 상태 확인

표 14-1 배터리와 관련된 액션

액션	설명
ACTION_BATTERY_CHANGED	배터리의 상태가 변경될 때
ACTION_BATTERY_LOW	배터리가 거의 방전되었을 때
CTION_BATTERY_OKAY	배터리가 방전 상태에서 정상 수준으로 올라왔을 때

안드로이드 프로젝트 생성

- 프로젝트 이름 : Project00
- 패키지 이름 : com.cookandroid.project00

디자인 및 편집

- 배터리 상태에 따라 변하는 이미지 5개를 /res/drawable에 복사

activity_main.xml 수정

- 이미지뷰 1개와 에디트텍스트 1개를 생성, id는 ivBattery, edtBattery로 함

```
<LinearLayout>
    <ImageView
        android:id="@+id/ivBattery"
        android:layout_gravity="center"
        android:src="@drawable/battery_0" />
    <EditText
        android:id="@+id/edtBattery"
        android:enabled="false" />
</LinearLayout>
```

Java 코드 작성 및 수정

- ImageView 변수 1개와 EditText 변수 2개를 전역변수로 선언
- activity_main.xml의 위젯 2개를 변수에 적용시킴

```
public class MainActivity extends Activity {
    ImageView ivBattery;
    EditText edtBattery;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        setTitle("배터리 상태 체크");

        ivBattery = (ImageView) findViewById(R.id.ivBattery);
        edtBattery = (EditText) findViewById(R.id.edtBattery);
    }
}
```

- **onCreate( ) 밖에 BR 객체 생성**
- **onReceive( ) 메소드 인텐트의 액션이 ACTION_BATTERY_CHANGED인 경우 다음을 처리**
  - 인텐트의 Extra에서 배터리의 잔량을 추출
  - 그에 따라 잔량을 표시하고 배터리 이미지를 변경
  - 인텐트의 Extra에서 배터리의 전원 연결 상태를 추출한 후 표시

```
BroadcastReceiver br = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();

        if (action.equals(Intent.ACTION_BATTERY_CHANGED)) {
            int remain = intent.getIntExtra(BatteryManager.EXTRA_LEVEL, 0);
            edtBattery.setText("현재 충전량 : " + remain + "₩n");

            if (remain >= 90)
                ivBattery.setImageResource(R.drawable.battery_100);
            else if (remain >= 70)
                ivBattery.setImageResource(R.drawable.battery_80);
            else if (remain >= 50)
                ivBattery.setImageResource(R.drawable.battery_60);
            else if (remain >= 10)
                ivBattery.setImageResource(R.drawable.battery_20);
            else
                ivBattery.setImageResource(R.drawable.battery_0);
            // ....생략                                    }
        }
    }
};
```

```
BroadcastReceiver br = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            String action = intent.getAction();

            if (action.equals(Intent.ACTION_BATTERY_CHANGED)) {
                // ....생략

                int plug = intent.getIntExtra(BatteryManager.EXTRA_PLUGGED, 0);
                switch (plug) {
                case 0:
                    edtBattery.append("전원 연결 : 안됨");
                    break;
                case BatteryManager.BATTERY_PLUGGED_AC:
                    edtBattery.append("전원 연결 : 어댑터 연결됨");
                    break;
                case BatteryManager.BATTERY_PLUGGED_USB:
                    edtBattery.append("전원 연결 : USB 연결됨");
                    break;
                }
            }
        }
    };
```

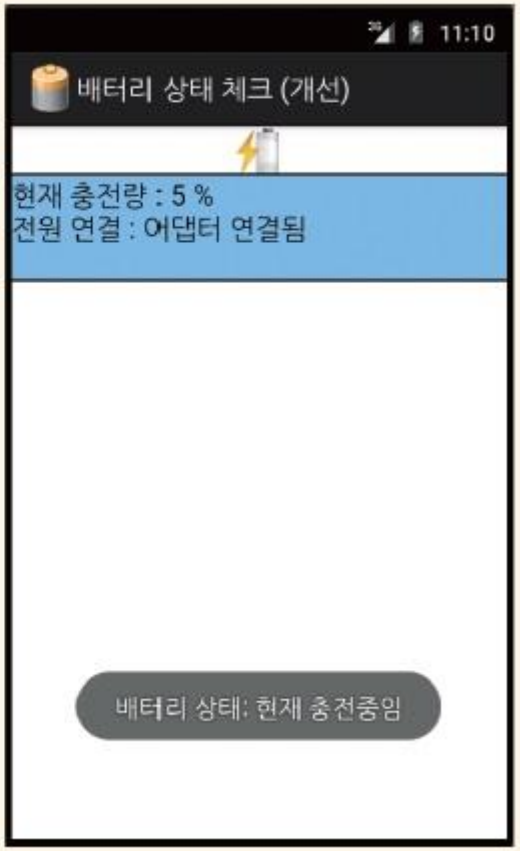
onPause( ), onResume( )를 자동완성

- onResume( ) 메소드 : 인텐트 필터를 생성하고 ACTION_BATTERY_CHANGED 액션을 추가한 후 BR에 등록
- onPause( ) 메소드 : 등록된 BR을 해제

```
@Override
protected void onPause() {
    // TODO Auto-generated method stub
    super.onPause();
    unregisterReceiver(br);
}
@Override
protected void onResume() {
    // TODO Auto-generated method stub
    super.onResume();
    IntentFilter iFilter = new IntentFilter();
    iFilter.addAction(Intent.ACTION_BATTERY_CHANGED);
    registerReceiver(br, iFilter);
}
```

직접 풀어보기 14-2

[실습 14-2]에 배터리 상태(EXTRA_STATUS)가 변경될 때마다 토스트 메시지가 나오도록 수정하자.



3. 콘텐트 프로바이더

- **콘텐트 프로바이더(Content Provider)**
  - 안드로이드는 보안상 앱에서 사용하는 데이터를 외부에서 접근할 수가 없음
  - 파일이나 데이터베이스를 외부 앱에서 사용하도록 하려면 콘텐트 프로바이더(Content Provider : 줄여서 CP)를 만들어서 외부로 제공

- **URI(Uniform Resource Identifier)**
  - URI는 콘텐트 프로바이더에서 제공하는 데이터에 접근하기 위한 주소
  - URI는 “content://패키지명/경로/아이디” 형식으로 지정

안드로이드에서 제공하는 CP의 사용

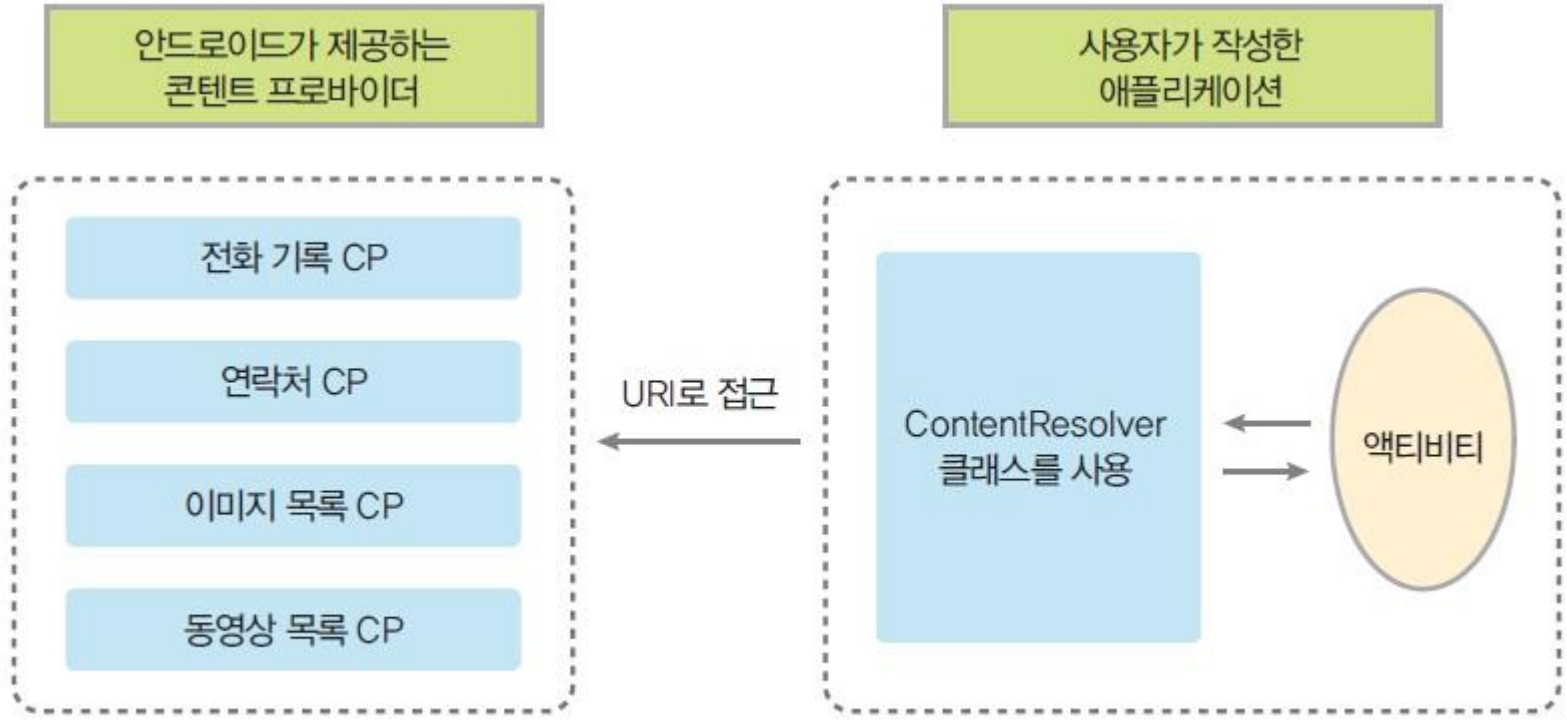


그림 14-11 안드로이드에서 제공하는 CP의 사용

안드로이드에서 제공하는 주요한 CP와 URI

콘텐트 프로바이더	URI
연락처 전화번호	android.provider.Contacts.Phones.CONTENT_URI
통화 기록	android.provider.CallLog.Calls.CONTENT_URI
브라우저 북마크	android.provider.Browser.BOOKMARKS_URI
브라우저 검색 기록	android.provider.Browser.SEARCHES_URI
시스템 설정 값	android.provider.System.CONTENT_URI
내장 미디어의 이미지	android.provider.MediaStore.Image.Media.INTERNAL_CONTENT_URI
내장 미디어의 동영상	android.provider.MediaStore.Video.Media.INTERNAL_CONTENT_URI
내장 미디어의 오디오	android.provider.MediaStore.Audio.Media.INTERNAL_CONTENT_URI
외장 미디어의 이미지	android.provider.MediaStore.Image.Media.EXTERNAL_CONTENT_URI
외장 미디어의 동영상	android.provider.MediaStore.Video.Media.EXTERNAL_CONTENT_URI
외장 미디어의 오디오	android.provider.MediaStore.Audio.Media.EXTERNAL_CONTENT_URI

안드로이드에서 통화기록을 가져오는 예제

- AVD에서 통화 버튼을 눌러서 통화 기록을 몇 건 남겨놓음
- 통화 기록에 접근하기 위해 <application> 위에 AndroidManifest.xml의 다음 코드를 추가하여 접근 권한을 줌

```
<uses-permission android:name="android.permission.READ_CALL_LOG"/>
```

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <Button
        android:id="@+id/btnCall"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="통화 기록 가져오기" />

    <EditText
        android:layout_weight="1"
        android:id="@+id/edtCall"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="top"
        android:background="#eee8aa" />

</LinearLayout>
```

```
public class MainActivity extends Activity {
    Button btnCall;
    EditText edtCall;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        btnCall = (Button) findViewById(R.id.btnCall);
        edtCall = (EditText) findViewById(R.id.edtCall);
        btnCall.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                edtCall.setText(getCallHistory());
            }
        });
    }
    // 이하 생략
}
```

```
public String getCallHistory() {
    String[] callSet = new String[] { CallLog.Calls.DATE,
            CallLog.Calls.TYPE, CallLog.Calls.NUMBER,
            CallLog.Calls.DURATION };
    Cursor c = getContentResolver().query(CallLog.Calls.CONTENT_URI,
            callSet, null, null, null);
    if (c == null)  return "통화기록 없음";

    StringBuffer callBuff = new StringBuffer(); // 최대 100 통화 저장
    callBuff.append("₩n날짜 : 구분 : 전화번호 : 통화시간₩n₩n");
    c.moveToFirst();
    do {
        long callDate = c.getLong(0);
        SimpleDateFormat datePattern = new SimpleDateFormat("yyyy-MM-dd");
        String date_str = datePattern.format(new Date(callDate));
        callBuff.append(date_str + ":");
        if (c.getInt(1) == CallLog.Calls.INCOMING_TYPE)
            callBuff.append("착신 :");
        else
            callBuff.append("발신 :");
        callBuff.append(c.getString(2) + ":");
        callBuff.append(c.getString(3) + "초₩n");
    } while (c.moveToNext());

    c.close();
    return callBuff.toString();
}
```

직접 풀어보기 14-3

콘텐트 프로바이더를 활용해서 북마크를 가져와서 화면에 뿌리는 앱을 작성하자.

HINT 북마크 퍼미션

com.android.browser.permission.READ_HISTORY_BOOKMARKS

4. 리소스와 매니페스트

- **매니페스트**
  - 애플리케이션이 실행되기전에 알아야하는 내용들을 정의
  - 매니페스트 파일의 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>

매니패스트의 주요 역할

- 애플리케이션의 | 자바 패키지 이름 지정
- 애플리케이션 구성요소에 대한 정보 등록
  - 액티비티, 서비스, 브로드캐스트 수신자, 내용재공자
- 각 구성요서를 구성하는 클래스 이름 지정
- 애플리케이션이 가져야하는 권한에 대한 정보 등록
- 다른 애플리케이션이 접근하기위해 필요한 권한 정보 등록
- 애플리케이션개발 과정에 프로파일링을 위해 필요한 Instrumentation 클래스 등록
- 애플리케이션에 필요한 안드로이드 API 레벨정보 등록
- 애프리케이션에서 사요하는 라이브러리 리스트

매니패스트의 기본 구조

[Code] [매니페스트 파일의 기본 구조]

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

매인 액티비티 정의

```
<activity android:name="org.androidtown.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>
```

리소스의 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. 토스트와 대화상자

- 토스트

  - 간단한 메시지를 잠깐 보여주었다가 없어지는 뷰로 애플리케이션 위에 떠 있는 뷰라 할 수 있음

[Code]

```
Toast.makeText(Context context, String message, int duration)
```

[Code]

```
public void setGravity(int gravity, int xOffset, int yOffset)
public void setMargin(float horizontalMargin, float verticalMargin)
```

토스트 만들기 예제

교재 p248- 참고

토스트 만들기 예제

-토스트의 색상이나 모양을 직접 구성

-새로운 레이아웃 정의

메인 액티비티
XML 레이아웃 정의

-메인 액티비티의 레이아웃 정의

메인 액티비티 코드 작성

-메인 액티비티에서 위치 설정

토스트를 위한
XML 레이아웃 정의

-토스트의 모양을 XML 레이아웃으

메인 액티비티 코드 작성

-메인 액티비티에서 모양 설정

대화상자 만들기 예제

대화상자 만들기 예제

-대화상자 보여주기

-이벤트 처리

메인 액티비티
XML 레이아웃 정의

-메인 액티비티의 레이아웃 정의

메인 액티비티 코드 작성

-메인 액티비티에서 대화상자 보여주기

교재 P 253