4장 액티비티와 리소스

시작하면서

- **목차**
  - 액티비티 이해
  - 리소스 이해
  - 리소스 응용
  - 애플리케이션 디버깅

액티비티 이해

애플리케이션 개념

- **Java 언어로 작성**
- **자신의 리눅스 프로세스 내에서 실행**
  - 각 프로세스는 자신의 자바 가상머신을 가짐

    → 각 프로세스는 다른 프로세스와 격리되어 실행
  - 진입점(entry point)이 없고 인텐트(intent)에 의하여 컴포넌트 활성화
- **고유한 리눅스 ID가 부여**
  - 기본적으로 애플리케이션을 구성하는 파일들은 해당 사용자와 해당 애플리케이션에게만 접근 허용
  - 시스템 자원을 절약하기 위하여 2개의 애플리케이션에 대하여 동일한 사용자 ID 공유 가능

    → 동일한 리눅스 프로세스 안에서 실행되며 동일한 가상머신을 공유

애플리케이션 개념

- **느슨하게 결합된 컴포넌트로 구성**
  - 애플리케이션 컴포넌트
    - 액티비티(activity)
    - 서비스(services)
    - 컨텐트 제공자(content provider)
    - 방송 수신자(broadcast receiver)
  - 안드로이드 매니페스트 파일
    - 각 컴포넌트간 상호 작용하는 방법을 기술
    - 애플리케이션 구성 요소 선언

      → APK 외부에서 패키지 구성 요소를 알 수 있게 함
    - 활용하고자 하는 라이브러리 지정
    - 애플리케이션에게 필요한 접근 권한(permission) 등록
    - 인텐트 필터(intent-filter)를 통해 애플리케이션의 진입점으로 액티비티를 지정

애플리케이션 개념

프로세스 생명주기

메모리가 부족할 경우 중요하지 않은 순서대로 프로세스를 메모리에서 제거

- 전경(foreground) 프로세스: 현재 초점을 갖고 있는 프로세스. 가장 마지막에 제거됨
- 가시(visible) 프로세스: 촛점은 없지만 화면에 보이는 프로세스
- 서비스(service) 프로세스
- 배경(background) 프로세스: 화면에 보이지 않는 프로세스
- 공백(empty) 프로세스: 액티비티를 가지고 있지 않은 프로세스. 메모리가 부족할 경우 바로 제거

애플리케이션 개념

액티비티와 태스크

- 액티비티: 같은 애플리케이션 내에 존재하는 액티비티 뿐만 아니라 다른 애플리케이션 내에 존재하는 액티비티까지 호출 가능

  예) 서적 관리 애플리케이션 = 서적 관리 일반 기능 + 특정 기능 (바코드 스캔)

- 한 애플리케이션에서 다른 애플리케이션의 컴포넌트를 거의 자유자재로 접근 가능

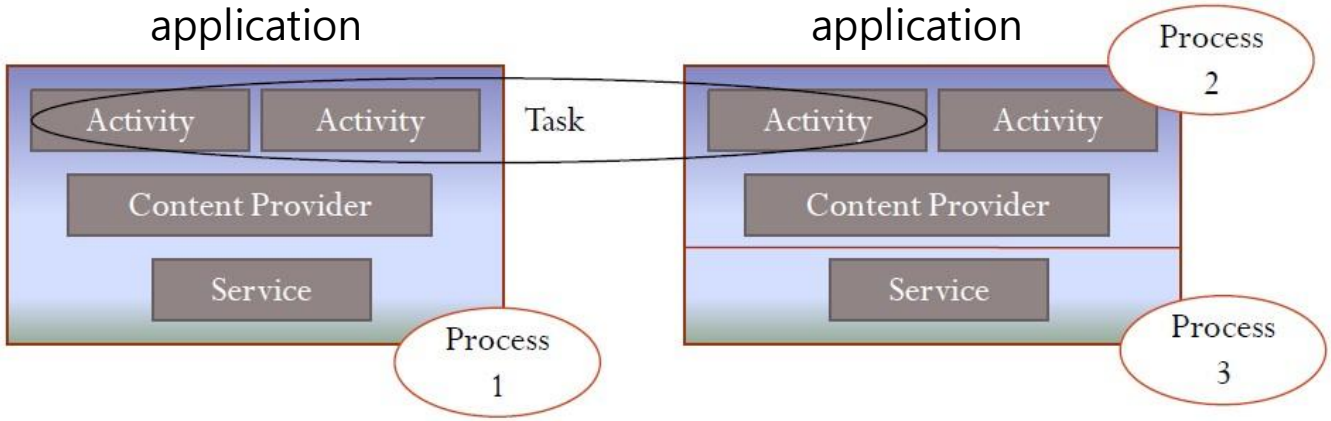
  → 파일의 구성 면에서는 애플리케이션의 경계가 뚜렷, 실제로 애플리케이션의 실행 면에서는 애플리케이션간 경계가 없음

- 각 컴포넌트들, 특히 화면에 표시되면서 사용자와 상호작용하는 액티비티는 애플리케이션 단위보다 태스크(Task) 단위로 관리
- 태스크(Task): 사용자가 실질적으로 “하나의 애플리케이션처럼” 느끼는 액티비티들의 집합임.

애플리케이션 개념

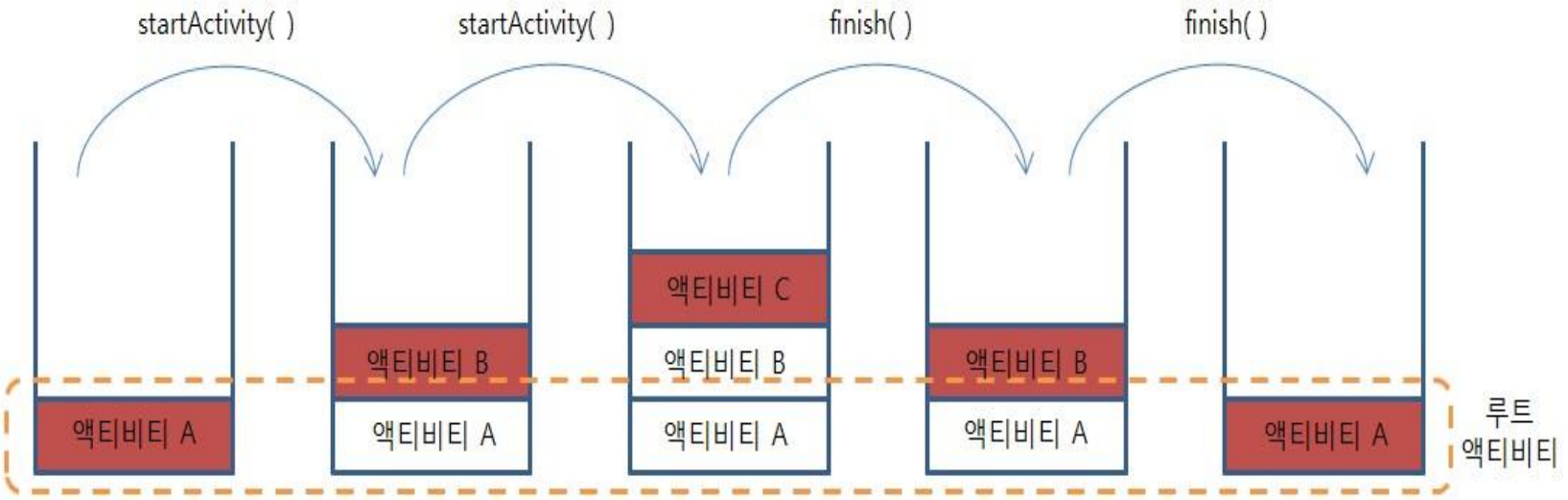
태스크와 프로세스

- 태스크
  - 연관된 액티비티의 집합
  - 다수의 프로세스와 APK에 걸쳐 존재 가능
  - 다른 APK의 액티비티 호출 가능
- 프로세스
  - 커널 프로세스
  - 기본적으로 APK는 하나의 프로세스에서 동작
  - 하나의 APK에서 다수의 프로세스 매핑 가능



애플리케이션 개념

- 액티비티 스택



액티비티 상태와 생명주기

개요

- 안드로이드는 모바일 기기에서 구동되기 때문에 데스크탑 애플리케이션에 비하여 더욱 효율적 메모리 관리가 요구
- 애플리케이션 컴포넌트의 중요 요소인 액티비티도 효율적인 메모리 관리를 위하여 액티비티 생성 및 소멸 과정인 생명주기가 있음
- 액티비티가 중지 혹은 정지 상태일 경우, 해당 프로세스가 메모리에서 제거될 수 있음. 사용자에게 다시 보여질 때 이전 상태로 복구
- 액티비티는 액티비티 스택(activity stack)을 통해 관리

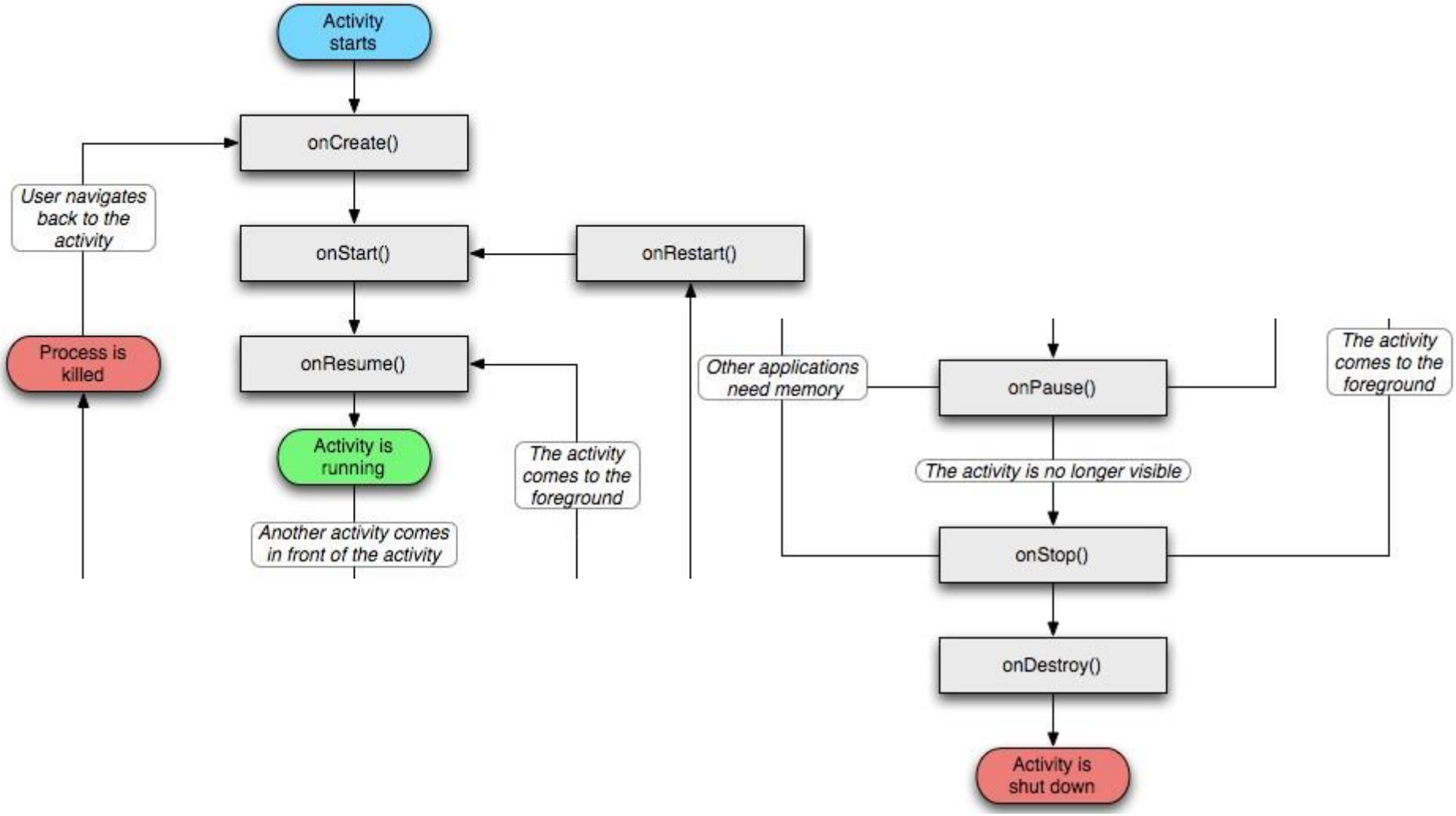
액티비티는 사용자와 상호 작용하는 단위이며 일반적으로 전체 화면을 차지

액티비티 상태와 생명주기

- **액티비티의 3가지 상태**
  - 활성(active) 혹은 실행(running) 상태
    - 전경 화면에 있을 경우
    - 해당 액티비티가 사용자 동작에 대한 초점을 가짐
    - 사용자와 상호 작용 가능
  - 중지(paused) 상태
    - 초점를 보유하지 않았지만 사용자에게 일부 보임
    - 메모리가 극도로 부족할 경우 시스템에 의하여 강제 종료
    - 사용자와 상호 작용 불가
  - 정지(stopped) 상태
    - 사용자에게 전혀 보이지 않지만 여전히 모든 상태와 멤버 정보는 유지
    - 다른 컴포넌트가 메모리를 요청하면 시스템에 의하여 강제 종료

액티비티 상태와 생명주기

- 액티비티를 위한 메소드



액티비티 상태와 생명주기

- **액티비티를 위한 콜백 메소드**
  - onCreate()
    - 액티비티가 생성될 때 처음으로 호출되는 메소드
    - 전역 상태의 모든 리소스를 초기화. 예) layout과 data binding 등
    - 액티비티는 기본적으로 버튼, 리스트, 체크박스, 입력 표시줄 등과 같은 위젯들이 배치되어 있는 레이아웃을 구성하고, 웨젯들이 사용자와 상호작용을 하는 코드를 포함
  - onStart()
    - 액티비티가 초기화 과정을 마친 후 사용자에게 보여줄 준비가 되었을 때 호출
  - onResume()
    - 액티비티가 사용자에게 보여지고 사용자의 입력을 처리할 수 있음
    - 액티비티 스택의 최상위에 위치

액티비티 상태와 생명주기

- **액티비티를 위한 콜백 메소드**
  - onPause()
    - 절전 상태 혹은 새로운 액티비티가 시작될 경우
    - 초점 상실
    - 재개(resume)되기 전 데이터 저장, 애니메이션 중지, 프로세서를 소비하는 작업 중단
  - onStop()
    - 더 이상 액티비티가 사용자에게 보이지 않음
    - 더 이상 액티비티 스택의 최상위에 위치하지 않음
  - onDestory()
    - 존재하는 모든 리소스를 해제
    - 시스템 내에 액티비티가 존재하지 않게 됨

액티비티 상태와 생명주기

- **액티비티 상태 저장하기**
  - 일반적으로 정지된 액티비티는 사용자가 다시 사용할 것을 대비하여 메모리에 상주. 만약 메모리가 부족하게 되면 강제 종료
  - 액티비티를 다시 호출하면,
    - 강제 종료된 상태: 다시 액티비티를 생성한 후 액티비티를 실행
    - 강제 종료되지 않은 상태: 액티비티를 다시 만들 필요가 없고 액티비티를 다시 시작하여 화면에 표시
  - 액티비티가 중지되는 것은,
    - 화면에 표시되지 않음
    - 사용자와 상호작용 하지 않음
    - 현재 액티비티의 상태를 Bundle 형태로 저장

      → 사용자가 텍스트 등을 입력하다가 액티비티가 종료될 경우 작업 내용이 날아가는 것을 방지함
    - 새로 액티비티가 시작할 경우 Bundle 객체는 null
    - 활성 상태로 변할 때 저장된 UI 상태를 복원

액티비티 상태와 생명주기

- **액티비티 상태 저장하기**
  - onSaveInstanceState()
    - onPause() 혹은 onStop() 이후 메모리가 부족할 경우 프로세스가 메모리에서 제거될 수 있음
    - 메모리에서 제거되기 전 액티비티 상태를 저장
    - 매개변수로 액티비티의 동적 상태를 기록할 수 있는 번들(Bundle) 오브젝트를 가짐
  - onRestoreInstanceState()
    - onCreate() 혹은 onStart() 이후 저장된 액티비티 상태를 복원
  - onSaveInstanceState()와 onRestoreInstanceState()는 생명주기 메소드가 아니기 때문에 항상 호출되지 않음. 개발자가 해당 코드의 메소드를 재정의하여 구현
  - 매개변수로 사용되는 savedInstanceState는 저장된 인스턴스의 상태, 즉 액티비티의 UI 상태를 의미

리소스 이해

개요

- **리소스란?**
  - 애플리케이션 = 기능 + 리소스
  - 기능은 애플리케이션 실행에 관계되는 모든 알고리즘을 포함하는 코드
  - 리소스는 애플리케이션이 사용하는 자산
    - 텍스트 문자열, 이미지, 아이콘, 오디오, 동영상 등
    - 레이아웃
  - 리소스를 소스 코드와 분리시 장점
    - 유지 보수 용이
    - 언어와 문화권에 맞는 애플리케이션의 현지화(localization) 가능

개요

- **리소스 저장**
  - 리소스는 Java 소스 코드와 별도로 외부 파일에 저장
  - 리소스 파일은 대부분 XML 파일로 저장
  - 이미지와 같은 원본 자료 파일은 그 자체로 저장
  - 모든 리소스는 ~/res 디렉토리의 하위 디렉토리에 저장되며, 하위 디렉토리 이름은 소문자+숫자+밑줄로만 구성
    - 여기서 ~는 프로젝트의 홈 디렉토리
    - 그래픽 및 애니메이션 파일 등 일부 리소스 파일은 파일 이름과 동일한 이름의 변수로 참조되기 때문에 Java 식별자 형태의 파일 이름으로 명명
  - 같은 형식의 리소스는 동일한 하위 디렉토리에 저장
  - 자동 생성되는 리소스 디렉토리: drawable, layout, values
  - aapt(Android Asset Packaging Tool)가 리소스를 모두 파악하여 자원을 접근하기 위한 변수의 정의를 담은 ~/gen/R.java 파일을 생성

개요

◉ 리소스 형식과 파일 이름

리소스 형식	디렉토리	권장 파일 이름	엘리먼트 이름
문자열	values	strings.xml	<string>
문자열 배열		arrays.xml	<string-array>
색상 값		colors.xml	<color>
크기		dimens.xml	<dimen>
단순 표시		drawables.xml	<drawable>
스타일 및 테마		styles.xml, themes.xml	<style>
그래픽	drawable	drawables.xml	
애니메이션	anim		<set>, <alpha>, <scale> 등
메뉴	menu		<menu>
XML	xml		
원본	raw		
레이아웃	layout		

리소스 접근

코드에서 문자열 참조

- R.java 파일에 정의된 R 클래스와 하위 클래스를 이용
- 코드에서 리소스를 접근하려면 R 클래스와 하위 클래스의 멤버 변수를 통하여 접근.

예) hello라는 문자열을 접근하려면

```
String myString = getResources().getString(R.string.hello);
```

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, Android!</string>
    <string name="app_name">Android</string>
</resources>
```

- getResource(): 리소스를 대표하는 Resources 객체를 반환
- getString(): 인수로 주어진 리소스 식별자에 해당하는 문자열의 실제 값을 반환
- R.string.hello는 R 클래스의 하위 클래스 string에 있는 멤버 변수 hello을 참조

리소스 접근

코드에서 문자열 배열 참조

- 문자열 배열 fruits의 접근

```
String[] fruits = getResources().getStringArray(R.array.fruits);
```

- 문자열 배열 리소스 파일(~/res/values/arrays.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="fruits">
            <item>apple</item>
            <item>banana</item>
    </string-array>
    <string-array name="animal">
            <item>tiger</item>
            <item>lion</item>
    </string-array>
</resources>
```

리소스 접근

◉ 코드에서 색상 참조

- 문자열 배열 textColor의 접근

```
int myColor = getResources().getColor(R.color.textColor);
```

- 색상 리소스 파일(~/res/values/colors.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="backgroundColor">#006600</color>
    <color name="textColor">#ffeecc</color>
</resources>
```

리소스 접근

코드에서 크기 참조

- 텍스트 크기 참조를 위한 textPointSize의 접근

```
float myTextSize = getResources().getDimension(R.dimen.textPointSize);
```

- 리소스 파일(~/res/values/dimens.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <dimen name="smallSize">6pt</dimen>
    <dimen name="textPointSize">11pt</dimen>
    <dimen name="largeSize">20pt</dimen>
</resources>
```

리소스 접근

코드에서 레이아웃 참조

- 레이아웃 내부에 정의된 ImageView01의 접근

```
ImageView iv = (ImageView)findViewById(R.id.ImageView01);
```

- 레이아웃 파일(~/res/layout/main.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#ffffffff">
<ImageView
    android:id="@+id/ImageView01"
    android:layout_width="fill_parent"
    ...
...
</LibearLayout>
```

리소스 접근

코드에서 이미지 참조

- ~/res/drawable 디렉토리에 추가한 flag.png 파일 접근

```
ImageView iv = (ImageView)findViewById(R.id.ImageView01);
iv.setImageResource(R.drawable.flag)
```

- 레이아웃 파일(~/res/layout/main.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#ffffffff">
<ImageView
    android:id="@+id/ImageView01"
    android:layout_width="fill_parent"
    ...
...
</LibearLayout>
```

리소스 접근

리소스에서 리소스 참조

- 참조 방법

```
@[패키지이름:]리소스형식/리소스이름
```

예) 애플리케이션 이름을 hello 문자열과 동일하게 하는 경우: strings.xml을 수정

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, ColorSize!</string>
    <string name="app_name">@string/hello</string>
</resources>
```

리소스 접근

- **시스템 리소스 참조**
  - android.R의 하위 클래스에는 다양한 시스템 리소스 보유
    - 표준 시스템 색상
    - 시스템 스타일과 테마
    - 애플리케이션 프로그램 섬네일(thumbnail) 이미지와 아이콘
    - 오류 문자열과 표준 버튼 텍스트
    - 페이드인/아웃을 위한 애니메이션 시퀀스
    - …
  - 참조 방법
    - 코드에서 참조: R 대신 android.R
    - 리소스에서 참조: 패키지 이름으로 android 사용

리소스 접근

선택적 리소스

- 애플리케이션의 현지화 가능
- 선택적 리소스를 위한 수식어

수식어	내용
언어	kr(한글), en(영어), fr(불어) 등
화면 크기	small, normal, large
화면 방향	port(세로), land(가로)
화면 픽셀 밀도	ldpi(120dpi), mdpi(160dpi), hdpi(240dpi)

- 리소스와 참조
  - 리소스: MyApp/res/drawable-mdpi/myImage.png
  - 코드: R.drawable.myImage
  - XML: @drawable/myImage

리소스 응용

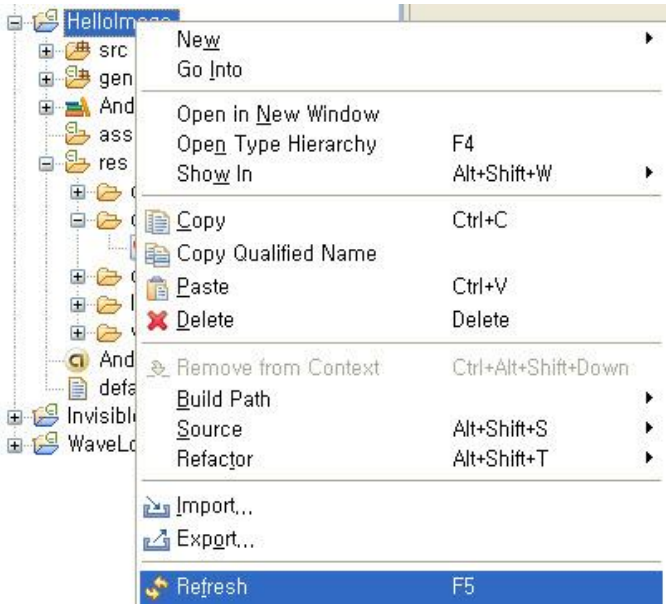
리소스 응용

- **<실습 4-1>: 간단한 이미지 출력하기**

(1) 3장에서 배운 대로 HelloImage 프로젝트 생성

(2) 이미지(android.jpg)를 res/drawable-mdpi 폴더에 복사

(3) 복사한 이미지를 이클립스에 반영하기 위하여 이클립스의 패키지 익스플로러 내부에 오른쪽 마우스를 클릭



(4) 컨텍스트 메뉴의 Refresh 항목을 선택

리소스 응용

(5) <코드 4-8>처럼 res/layout/main.xml의 내용 수정

```
<ImageView
    android:id="@+id/image"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:src="@drawable/android" />
```

(6) 소스 코드의 setContentView(R.layout.main);을 <코드 4-9>처럼 수정

```
setContentView(R.layout.main);
```

→

```
ImageView image = new ImageView(this);
image.setImageResource(R.drawable.android);
setContentView(image);
```

(7) [Ctrl] + [Shift] + [o]를 누름

리소스 응용

(8) 프로젝트를 실행

→아래와 같이 에뮬레이터에 표시

이벤트

- **이벤트 처리를 준비하는 방식**
  - 방식 1: 내장 Interface를 직접 활용하는 방식

    1)View 클래스의 내장 Interface를 이용

    2)Listener를 만듬

    3)View에 연결

    예)View.OnClickListener, View.OnTouchListener, View.OnKeyListener 등
  - 방식 2: 클래스 생성시 기 정의된 Listener를 재정의하는 방식

이벤트 처리 방식

- 방식 1

```
public class ExampleActivity extends Activity {
    private OnClickListener mListener = new OnClickListener() {
        public void onClick(View v) {
           // 클릭하면 할 일을 추가
        }
    };

    protected void onCreate(Bundle savedValues) {
        // 레이아웃에서 버튼을 찾아 오브젝트로 정의합니다.
        Button button = (Button)findViewById(R.id.corky);

        // 리스너를 버튼 오브젝트에 연결합니다.
        button.setOnClickListener(mListener);
        ...
    }
}
```

이벤트 처리 방식

- 방식 2

```
public class ExampleActivity extends Activity implements OnClickListener {
    protected void onCreate(Bundle icicle) {
        ...
        // 레이아웃에서 버튼을 찾아 오브젝트로 정의
        Button button = (Button)findViewById(R.id.corky);
        // 리스너를 버튼 오브젝트에 연결
        button.setOnClickListener(this);
    }

    // OnClickListener 콜백시에 할 일을 정의
    public void onClick(View v) {
       // 클릭하면 할 일을 추가
    }
    ...
}
```

버튼과 이벤트

- **<실습 4-2>: 버튼과 이벤트**

  (1) 3장에서 배운 대로 ButtonEvent 프로젝트 생성

  (2) res/values 폴더에 색상 리소스를 위한 colors.xml 파일 생성

  (3) <코드 4-12>와 같이 colors.xml 파일 내용 수정

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="black">#ff000000</color>
    <color name="red">#ffff0000</color>
    <color name="green">#ff00ff00</color>
    <color name="blue">#ff0000ff</color>
    <color name="white">#ffffffff</color>
</resources>
```

버튼과 이벤트

(4) <코드 4-13>처럼 텍스트뷰 위젯 수정하고 버튼 추가

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@color/white" >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:id="@+id/text"
    android:textColor="@color/black"
    android:text="@string/hello" />
<Button
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:id="@+id/ok"
    android:text="Button" />
</LinearLayout>
```

버튼과 이벤트

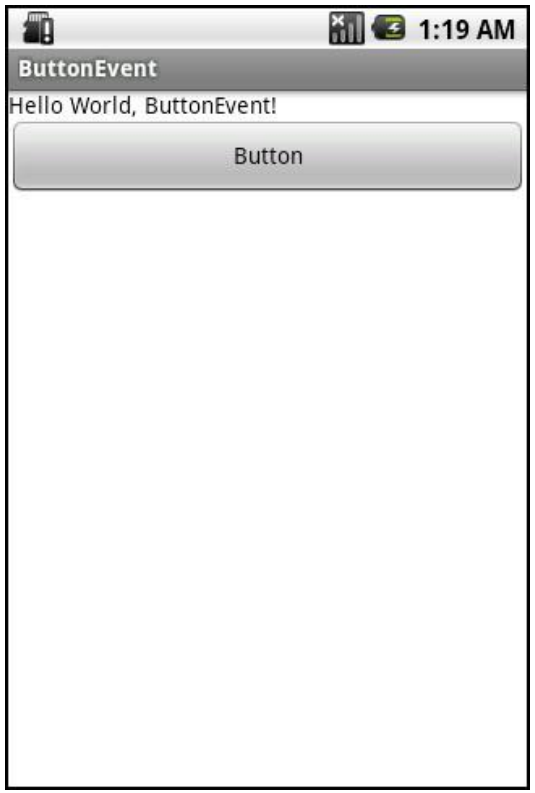
(5) <코드 4-14>처럼 자바 소스 코드의 내용 수정한 후, [Ctrl] + [Shift] + [o]를 동시에 클릭

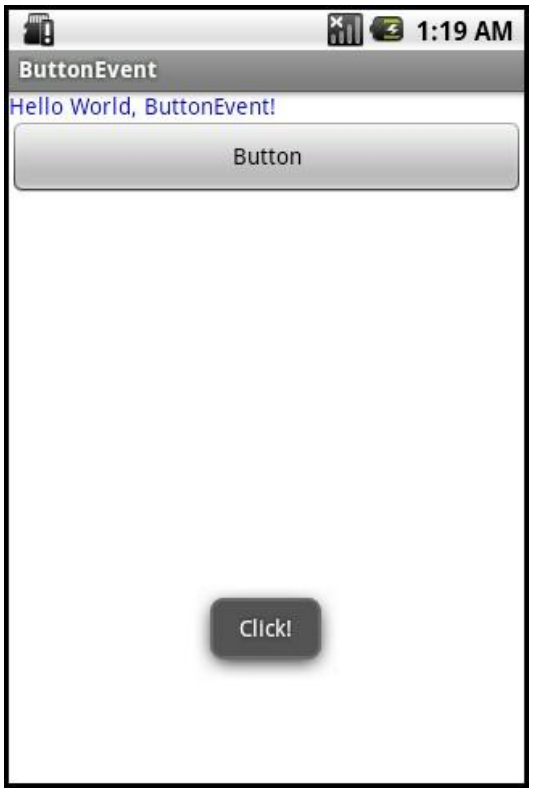
```
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        final Button button = (Button)findViewById(R.id.ok);
        button.setOnClickListener(new Button.OnClickListener() {
              public void onClick(View v) {
                        TextView view = (TextView)findViewById(R.id.text);
                        // view.setTextColor(Color.BLUE);
                        view.setTextColor(getResources().getColor(R.color.blue));
              }
        });
}
```

버튼과 이벤트

(6) 프로젝트를 실행하면, 아래와 같이 에뮬레이터에 출력





토스트

- 의미
  - 설정 변경 혹은 이벤트 발생시 사용자에게 알려주는데 방법
  - 실행 중인 액티비티의 초점을 뺏지 않고, 잠시 표시되었다가 사라짐
  - 다른 방법인 Notification에 비하여 매우 간단

토스트

사용 방법

- makeText() 메소드로 토스트 메시지를 만든 후, show() 메소드로 화면에 표시
- 메소드
  - **public static Toast makeText(Context context, int resId, int duration)**
  - **public static Toast makeText(Context context, CharSequence text, int duration)**
- 토스트를 표시할 시간 속성
  Toast.LENGTH_SHORT, Toast.LENGTH_LONG

리소스 응용

- **<실습 4-3>: 2개 국어 지원 애플리케이션**

  (1) 3장에서 배운 대로 AlternateDemo 프로젝트 생성

  (2) 패키지 익스플로러에 있는 AlternateDemo 프로젝트의 res/values 폴더를 선택

  → 마우스 오른쪽 버튼을 클릭,

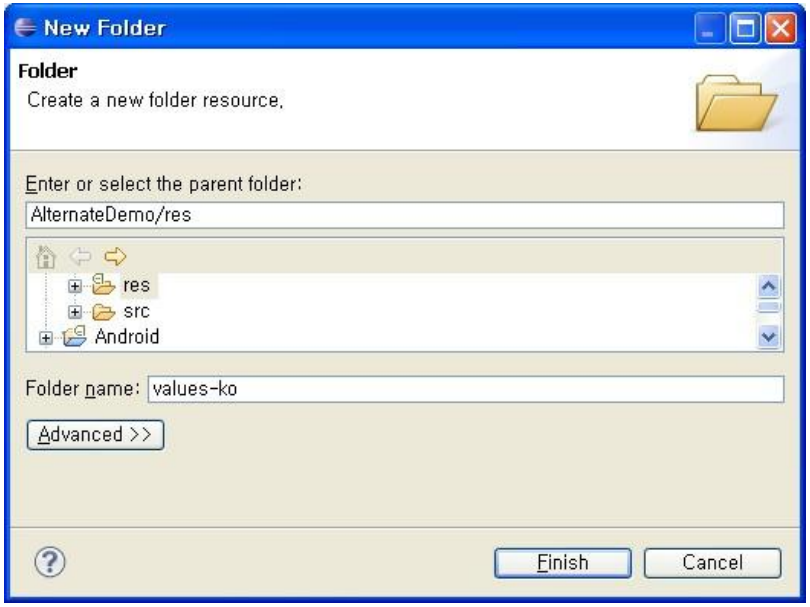
  → 나타난 컨텍스트 메뉴에서 Delete 항목 선택하여 values 폴더 삭제

  (3) res 폴더를 선택한 후 마우스 오른쪽 버튼을 클릭

  → 나타난 컨텍스트 메뉴에서 New ⟶ Folder 항목 선택

리소스 응용

(4) 폴더 이름에 values-ko로 입력한 후 [Finish] 버튼 클릭
→ values-ko라는 새로운 폴더 생성

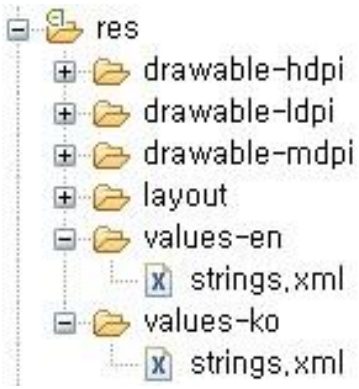


(5) values-ko 폴더에 strings.xml 파일 추가

(6) 과정 (4)~(5)를 반복 수행. 단, 과정 (4)에서 values-ko 폴더 대신에 values-en 폴더 생성

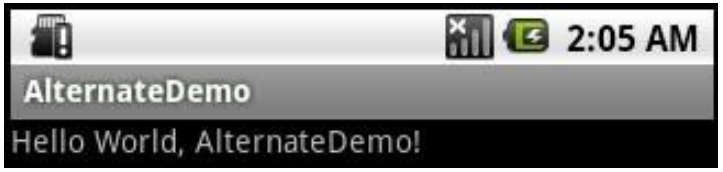
리소스 응용

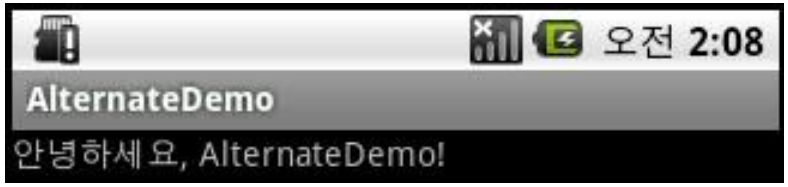
(7) values-ko/strings.xml와 values-en/strings.xml에 각각 <코드 4-15>과 <코드 4-16>처럼 입력 후 저장



(8) 실행 결과.

- 한글/영어로 설정 변경하고 애플리케이션 탭 클릭 후, AlternateDemo 아이콘을 클릭하여 프로젝트 다시 실행





애플리케이션 디버깅

개요

- **오류 종류**
  - 문법 오류(syntax error)
  - 내용 오류(semantic error)
  - 런타임 오류(runtime error)

예) n2에 n^2, n3에 n^3, m에 n / d를 저장하는 코드 부분

```
int n, n2, n3, d;

n2 = n * n:                 // syntax error
n3 = n2 * n2;               // semantic error
m  = n / d;                 // if d == 0, runtime error
```

개요

- **디버깅 도구**
  - 아무리 훌륭한 프로그래머도 오류에서 자유롭지 못함
  - 이클립스 기반 안드로이드 개발 환경을 위한 디버깅 도구
    - 이클립스 디버거
    - 로그캣
    - 안드로이드 디버그 브릿지
    - DDMS
    - 트레이스 뷰

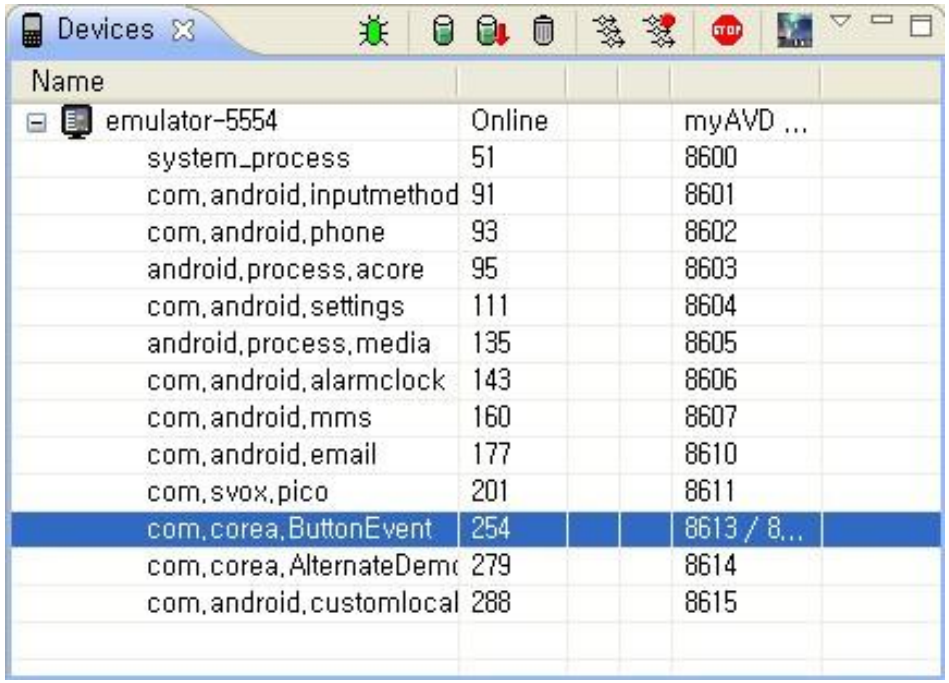
DDMS

개요

- Dalvik Debug Monitoring Service
- ADT 플러그인에서 DDMS 퍼스펙티브를 지원해주므로 이클립스 내에서 DDMS를 바로 사용 가능
- 역할
  - 프로세스 관리
  - 에뮬레이터 제어
  - 로그 관리
  - 파일 관리
  - 화면 캡쳐

DDMS: 프로세스 관리

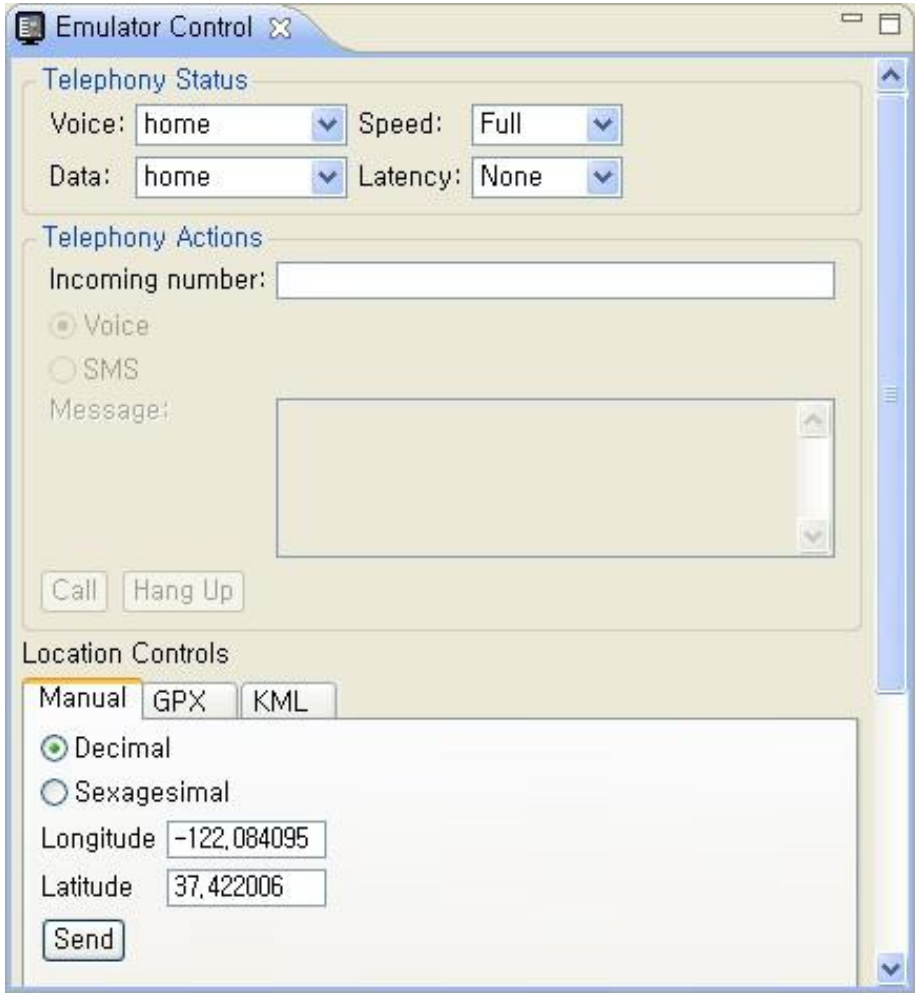
- **프로세스 강제 종료**
  - 프로세스 선택 후 stop 버튼으로 종료



- **화면 캡쳐**
  - stop 버튼 오른쪽의 Screen Capture 버튼 클릭

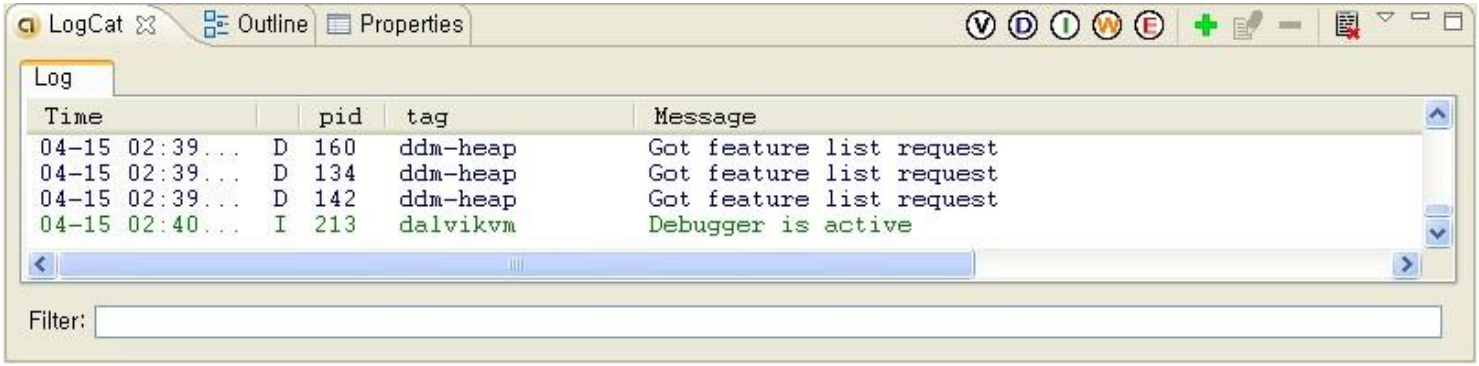
DDMS: 에뮬레이터 조작

- 에뮬레이터의 통신 상태 조절
- 가상 위치 정보 설정
- 가상 전화 발신
- SMS 전송 등



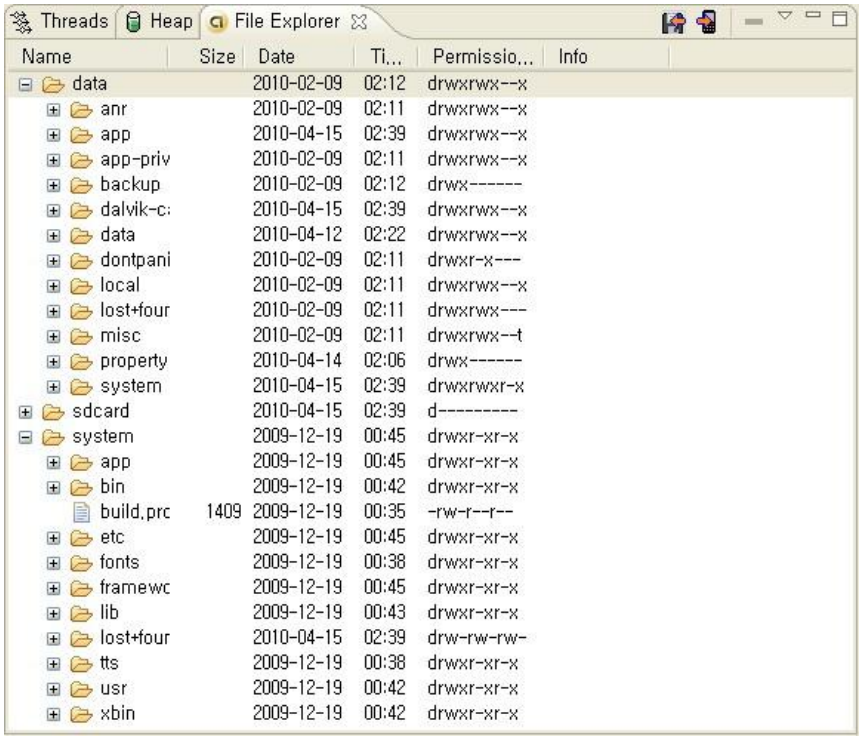
DDMS: 로그 관리

- 에뮬레이터 상태 관련 로그 출력
- 실행시간 로그 출력
  - 디버깅에 필수적



DDMS: 파일 관리

- **에뮬레이터/단말기내**
  - 파일 검색
  - 파일 저장
  - 컴퓨터로 복사
- **Linux에서 'ls –l' 처럼 내용 표시**

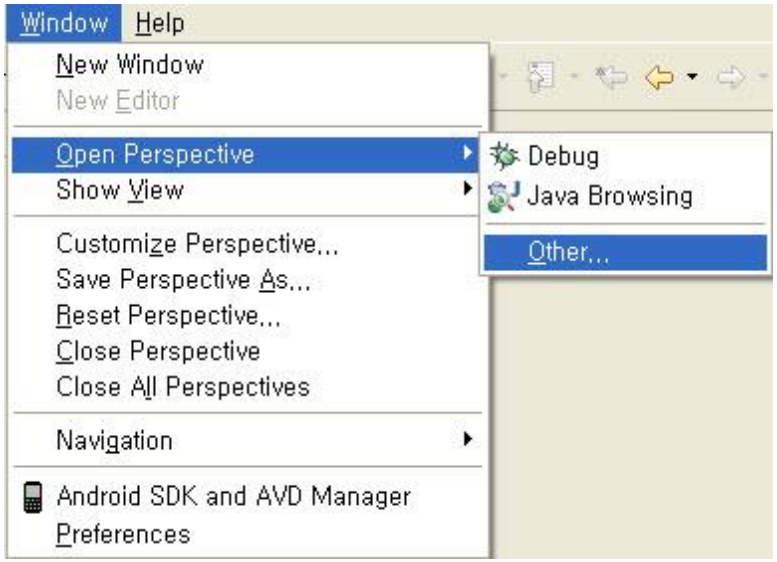


DDMS

- **<실습 4-4>: DDMS 퍼스펙티브 추가**

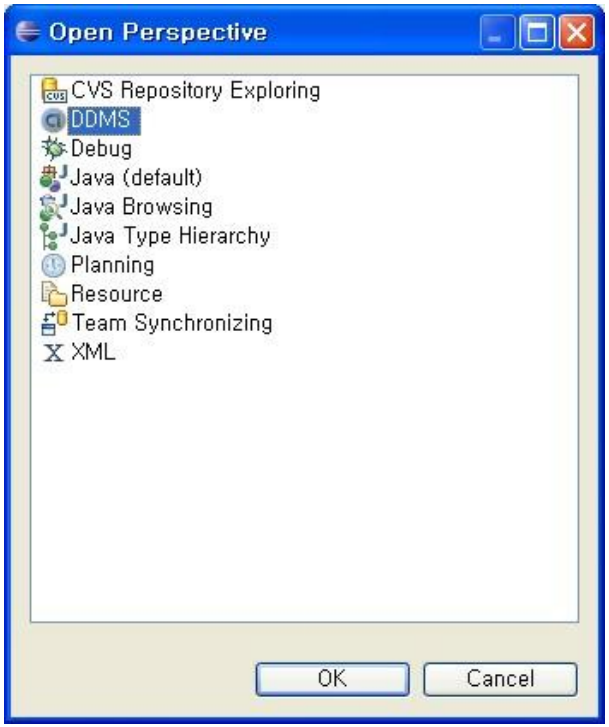
  (1) 이클립스 메뉴의 Window → Open Perspective → Other...를 선택

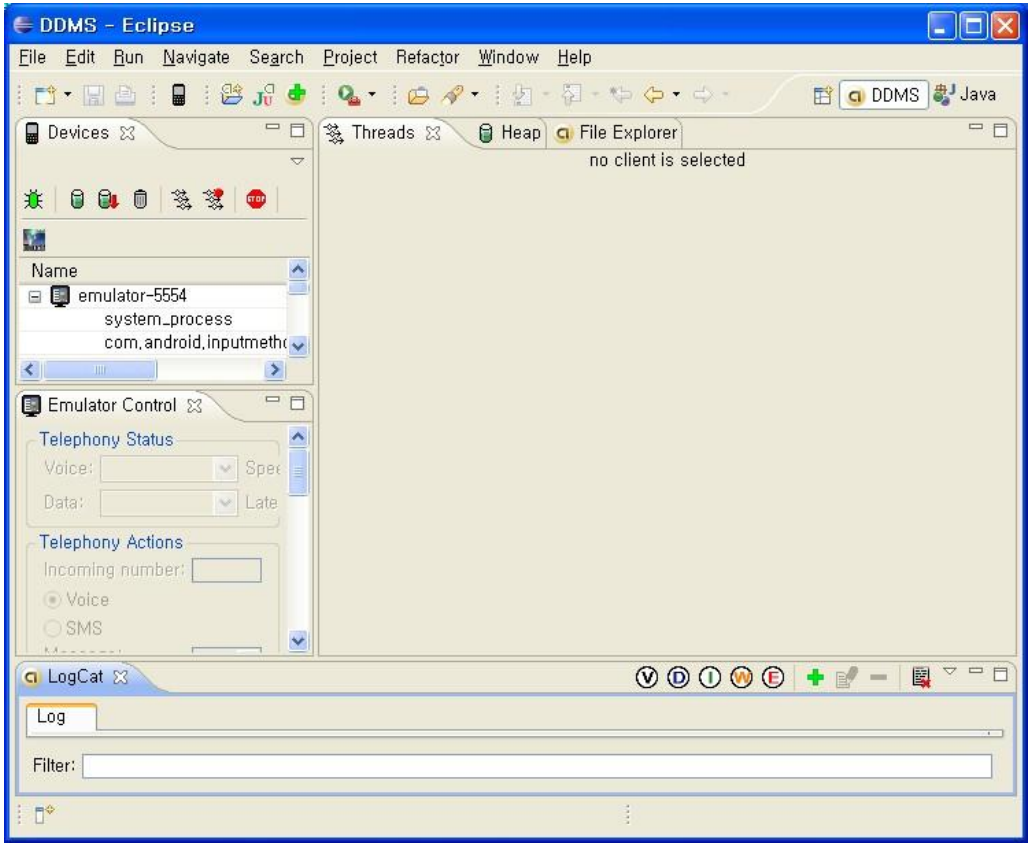
  만약 메뉴 내에 포함되어 있다면 DDMS 항목을 선택하고, 과정 (2)부터 수행 불필요



DDMS

(2) Open Perspective 마법사에서 DDMS를 선택 후 OK 버튼 클릭





(3) 오른쪽 상단 퍼스펙티브 선택 버튼에 DDMS 추가 확인

로그캣

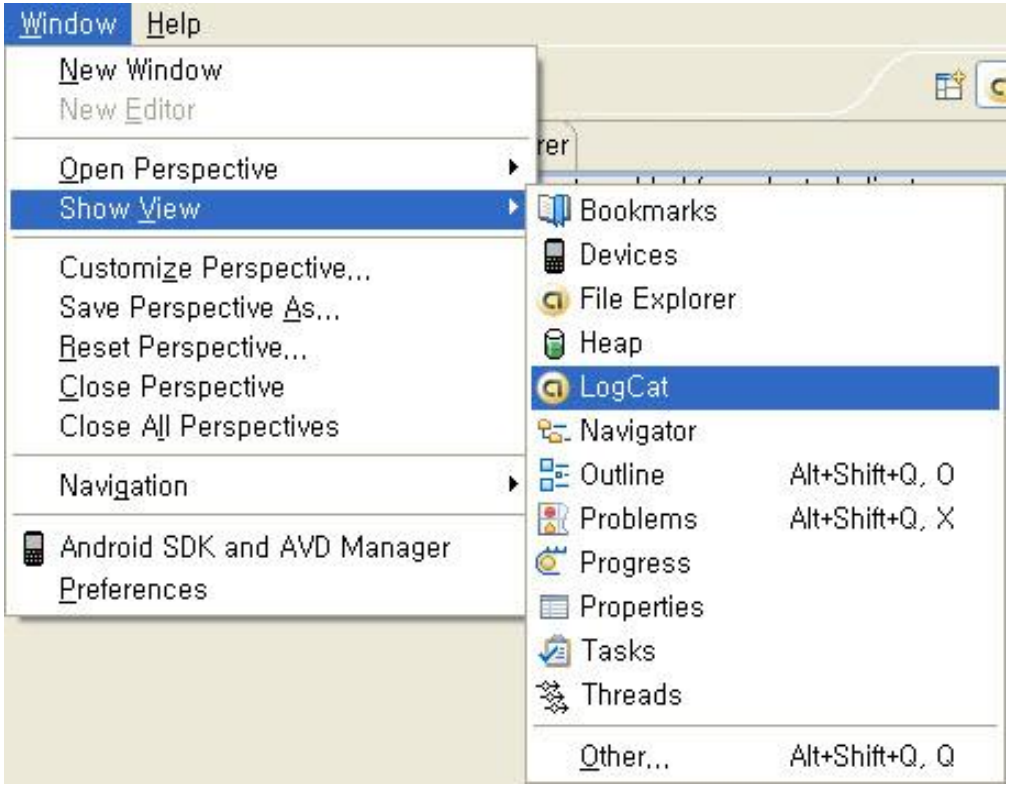
개요

- 디버깅을 위한 범용 로그 패키지
- DDMS 퍼스펙티브를 변환하면 하단부에 로그캣 창이 나타남

- 애플리케이션을 실행하면 로그캣 창에 부팅을 포함한 모든 실행 과정에 대한 로그 메시지가 출력

로그캣

- 로그캣 열기


Window
Help
New Window
New Editor
Open Perspective
Show View
Customize Perspective...
Save Perspective As...
Reset Perspective...
Close Perspective
Close All Perspectives
Navigation
Android SDK and AVD Manager
Preferences
Bookmarks
Devices
File Explorer
Heap
LogCat
Navigator
Outline Alt+Shift+Q, O
Problems Alt+Shift+Q, X
Progress
Properties
Tasks
Threads
Other... Alt+Shift+Q, Q

로그캣

로그캣 이용

- 수준별 로그 메시지를 출력하려면 툴바에 있는 V, D, I, W, E를 사용
  - V(Verbose): 모든 메시지 보기
  - D(Debug): Debug, Information, Warning, Error 수준의 메시지만 보기
  - I(Information): Information, Warning, Error 수준의 메시지만 보기
  - W(Warning): Warning, Error 수준의 메시지만 보기
  - E(Error): Error 수준의 메시지만 보기
- 관심있는 로그 메시지만을 출력하려면 하단부의 Filter 필드를 사용
  → 원하는 문자만 포함하는 로그 메시지만 추출 가능
- 로그 메시지 내용
  - 시간(time)
  - 우선 순위(priority)
  - 프로세스 식별자(process identifier)
  - 태그(tag)
  - 로그 메시지(log message)

로그캣

- **로그캣에 직접 로그 메시지 출력하기**
  - 다음 메소드를 사용

```
log.x(String tag, String message, [Throwable exception])
```

  - x는 수준별 로그 메시지 출력을 위한 툴바에 있는 V, D, I, W, E 중의 하나 지정

로그캣 이용하기

- **<실습 4-5>: 로그캣 이용하기**

(1) 3장에서 배운 대로 LogcatDemo 프로젝트 생성

(2) 패키지 익스플로러에 있는 Logcat.java를 더블 클릭하여 소스 코드를 자바 편집기로 불러옴

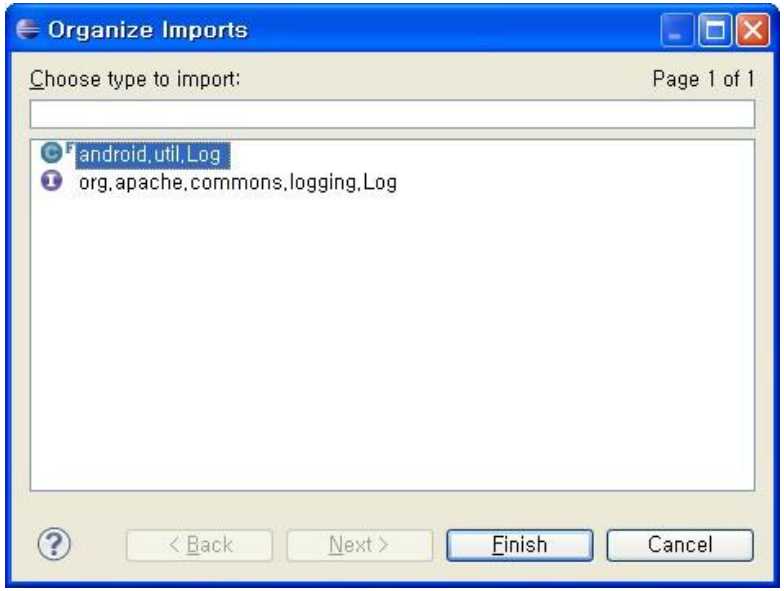
(3) <코드 4-18>과 같이 소스 수정

```
10 public void onCreate(Bundle savedInstanceState) {
11    Log.i("demo", "Before super.onCreate()");
12    super.onCreate(savedInstanceState);
13    Log.i("demo", "After super.onCreate()");
14    setContentView(R.layout.main);
15 }
```

(4) [Ctrl] + [Shift] + [o]를 클릭

로그캣 이용하기

(5) 'Organize Imports' 마법사에 나타난 항목 중 android.util.Log를 선택



(6) 자바 소스 코드 저장 후 실행

로그캣 이용하기

(7) 실행 결과.

DDMS 퍼스펙티브로 화면 변경 후 아래의 로그캣 창을 관찰