

Android Study

#1

오 영 택

안드로이드 개발환경 구축

1.1 Java 설치하기

- <http://www.oracle.com> -> downloads -> Popular Downloads
- Java SE 선택
- JDK부분 DOWNLOADS 버튼 클릭
- Accept License Agreement 체크
- 본인 PC에 맞는(32/64bit) 파일 선택

안드로이드 개발환경 구축

1.2 환경변수 설정하기

- <http://blog.naver.com/yuniejunie?Redirect=Log&logNo=220539239866> 참고하세요.

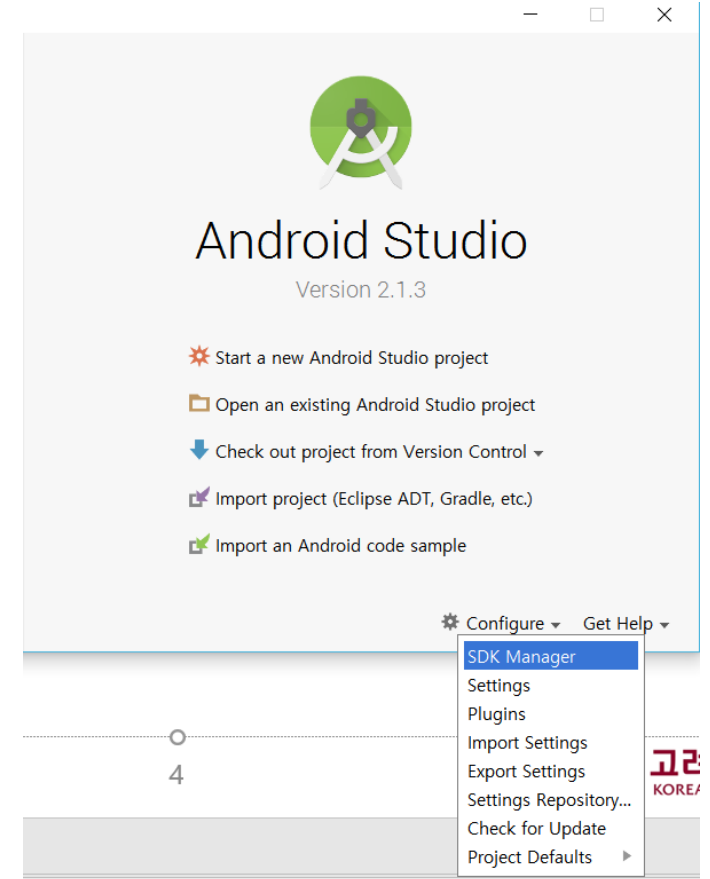
안드로이드 개발환경 구축

2.1 AndroidStudio 설치하기

- <https://developer.android.com/studio/index.html?hl=ko> 접속
후 다운로드, 설치

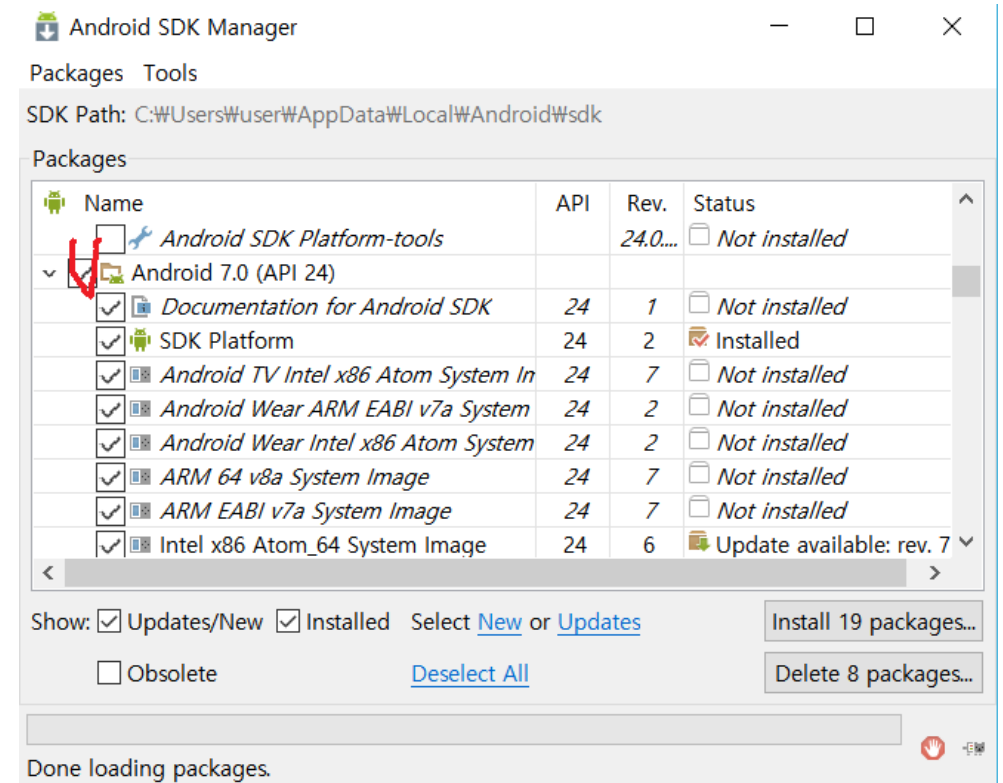
(주의! 사용자 계정명 포함하여 설치경로에 한글이 있으면 안됨!)

- AndroidStudio 실행 Configure -> SDK Manager 클릭
- 새로 뜬 창 하단의 Launch Standalone SDK Manager 클릭



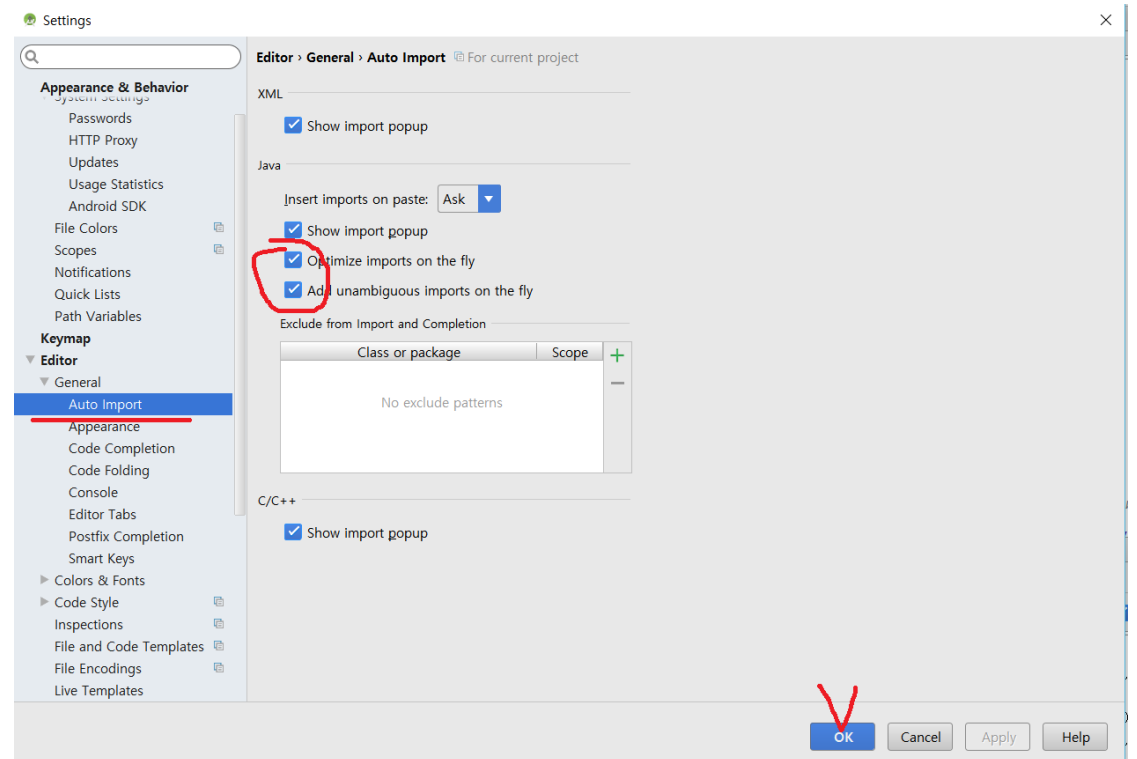
안드로이드 개발환경 구축

- 기존 체크 되어 있는 것 no touch
- Android 7.0 (API24) 체크
- 같은 방식으로 아래쪽 Extra 항목 체크
- 인스톨 진행



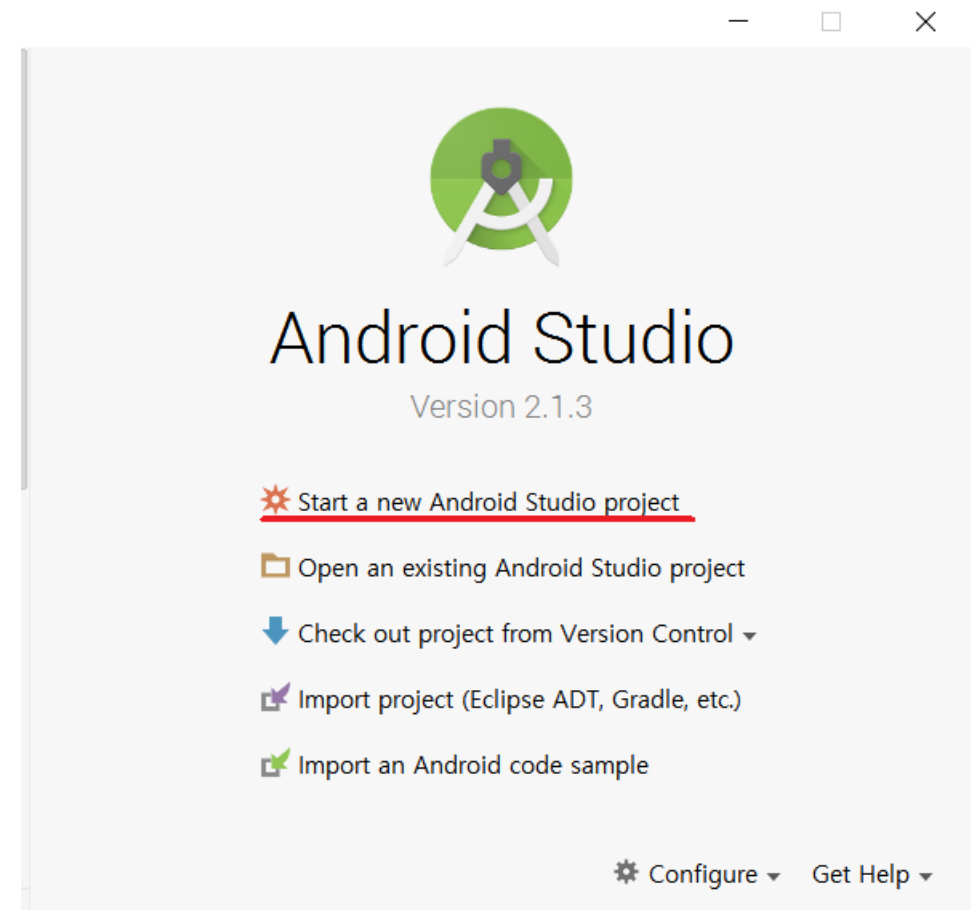
Auto Import 설정하기

- Auto Import : 코드가 입력되었을 때 필요한 import 구문을 자동으로 넣어주는 기능
- 상단 메뉴에서 File -> Settings
- Editor -> General -> Auto Imports 선택
- Optimize imports on the fly, Add unambiguous imports on the fly 체크

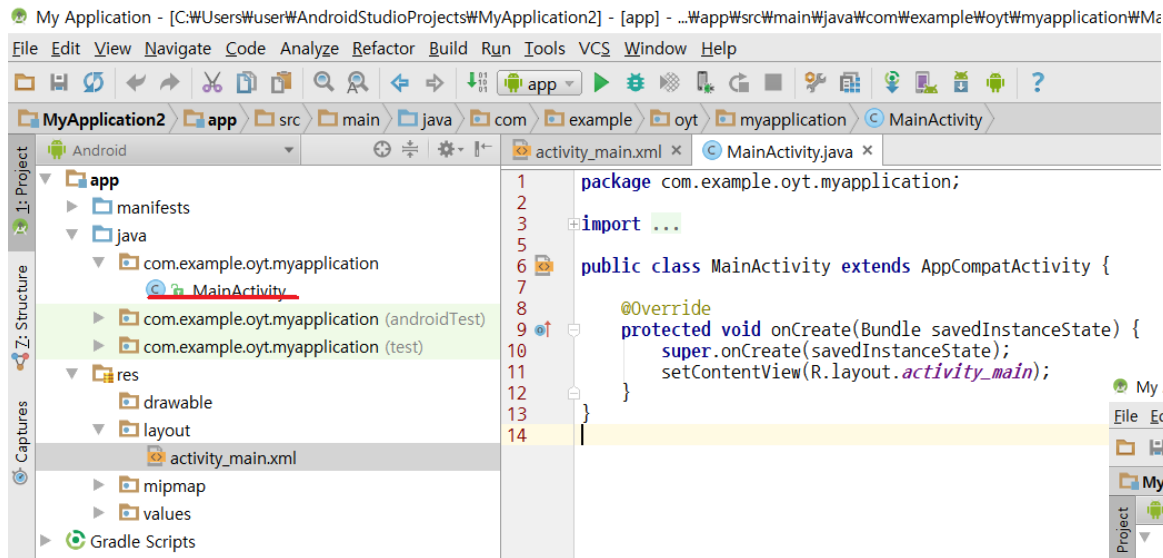


첫 프로젝트 만들어보기

- Start a new Android Studio Project
- Next
- Next
- Next
- Finish!

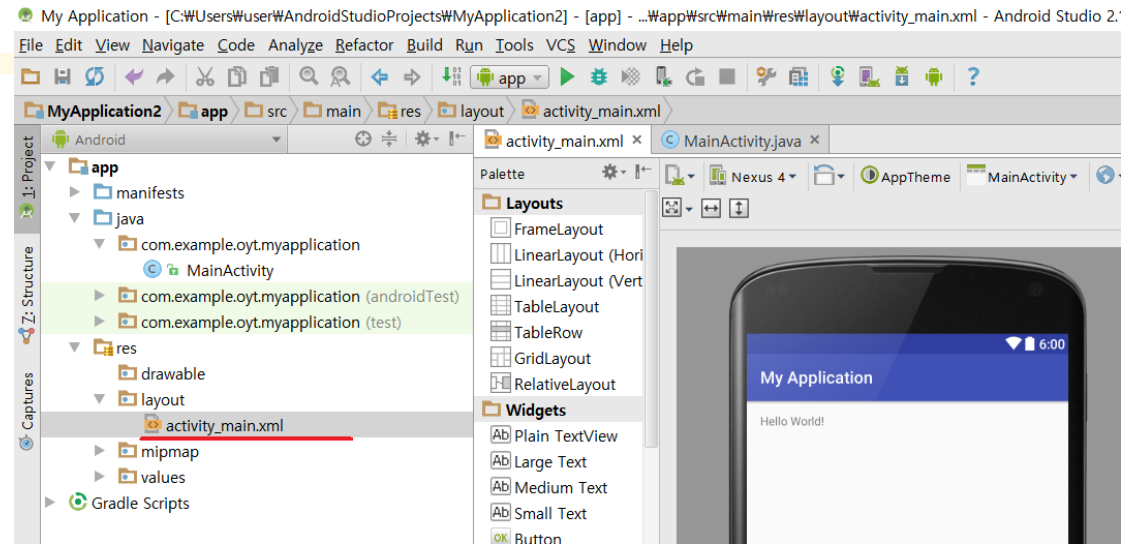


첫 프로젝트 만들어보기



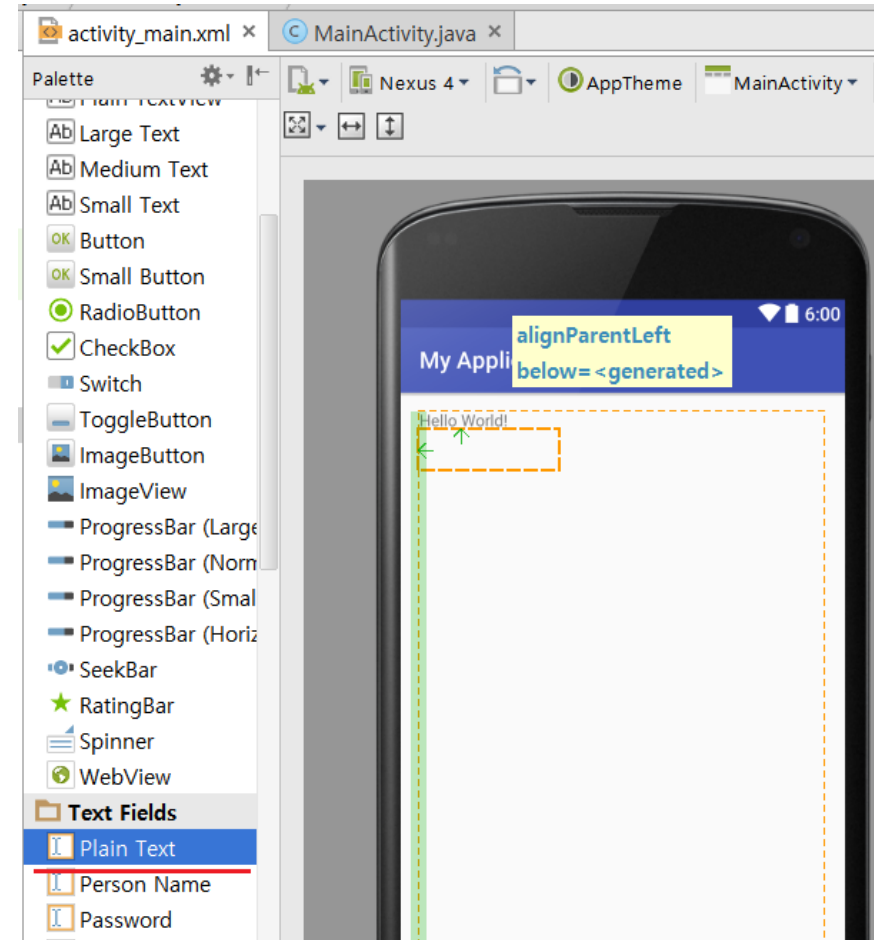
두둥!

작업할 수 있는 공간은 java파일(위)과
Xml파일 두 가지(오른쪽)



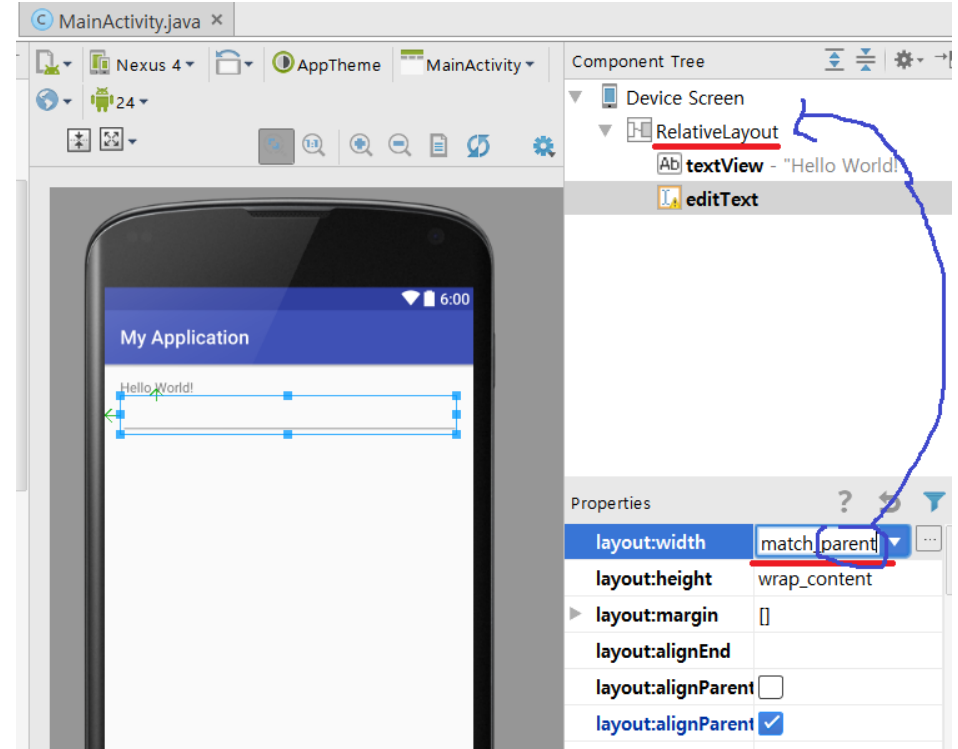
첫 프로젝트 만들어보기

- 프로젝트 창 왼쪽에서 app/res/layout/activity_main.xml을 열어보자.
- 프로젝트 생성 후 가장 첫 모습의 어플 화면이 보일 것
- Palette 탭에서 원하는 뷰들을 끌어서 화면에 추가할 수 있다 !
- 요로코롬!



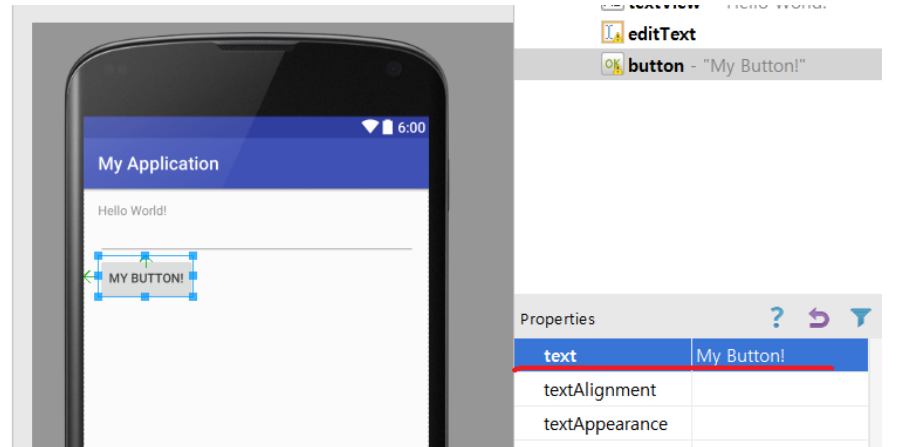
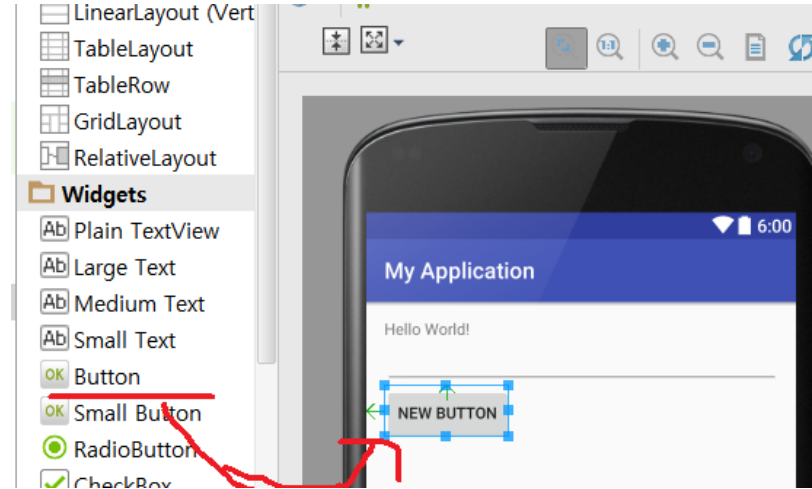
첫 프로젝트 만들어보기

- 근데 마음에 안 든다.
- EditText는 텍스트를 입력받을 수 있는 칸인데 너무 좁다!
- 오른쪽처럼 속성 변경을 원하는 객체를 마우스로 클릭한 뒤 Properties 탭에서 속성 변경할 수 있다.
(match_parent는 해당 객체가 놓여있는 객체, 여기서는 RelativeLayout의 width에 맞추겠다는 뜻이다. 즉, 폭이 화면 전체를 차지)
- 텍스트를 입력 받을 공간이 생성되었다.



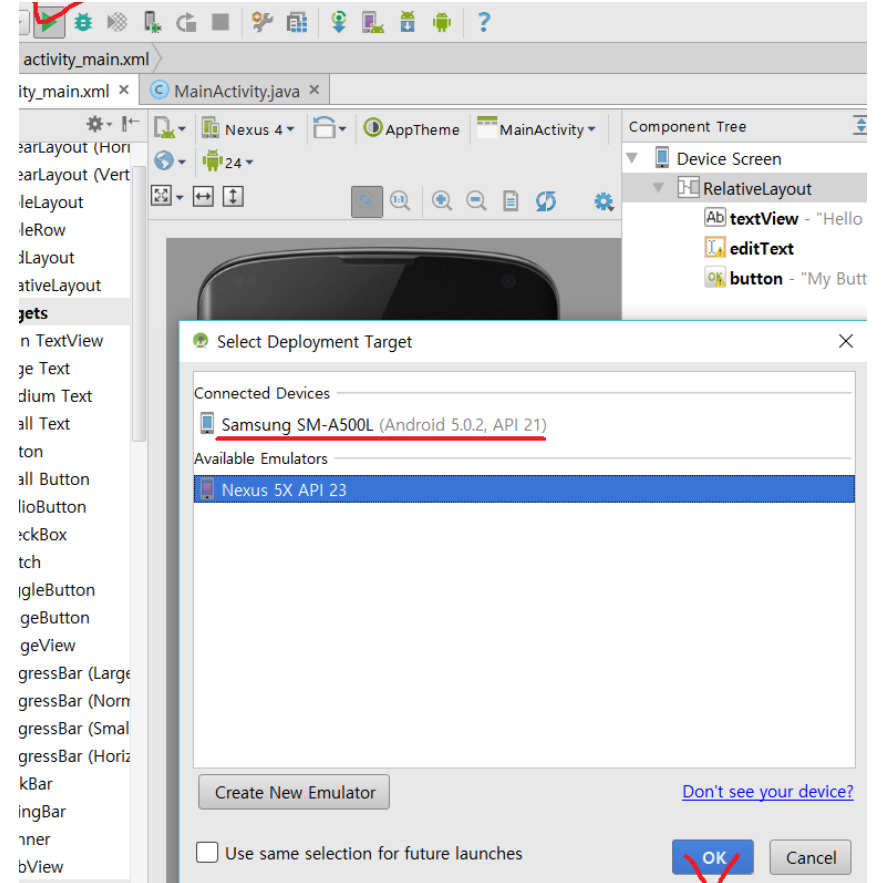
첫 프로젝트 만들어보기

- 버튼도 넣어볼까?
- 버튼 내용을 정해보자.
(Properties -> Text)
- 버튼 사이즈도 바꿔보자.
(Properties -> TextSize -> 30dp 입력)
- dp는 크기를 나타내는 척도라는 것만 알아두자.

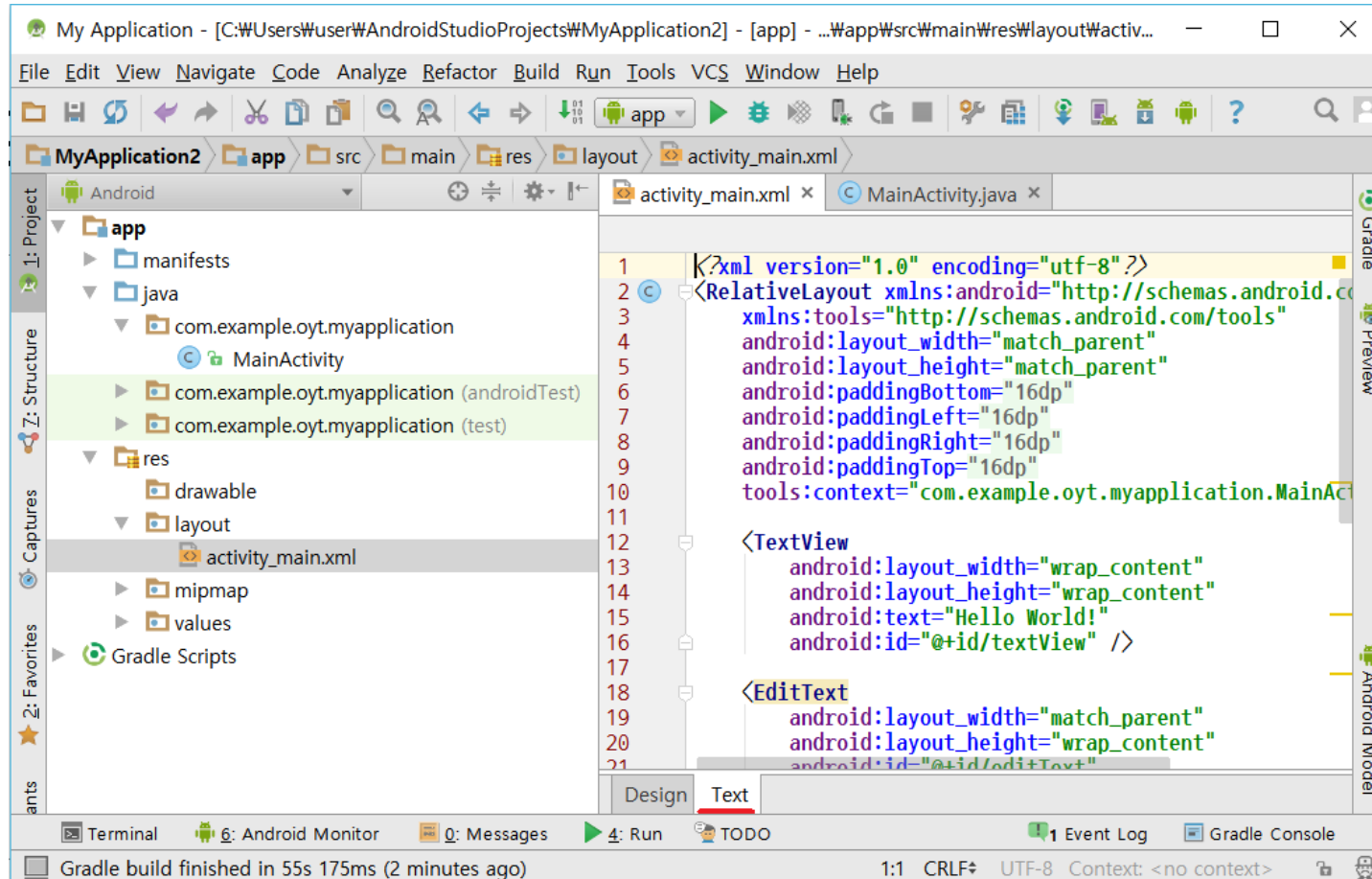


첫 프로젝트 만들어보기

- 실행해보자!
- Usb와 노트북을 연결하면 핸드폰으로 작업한 결과물을 확인할 수 있다.
- 폰 기종에 맞는 Usb 드라이버 설치 후
- 설정 -> 개발자 도구에서 USB 디버깅 허용 체크 후 가능



첫 프로젝트 만들어보기



첫 프로젝트 만들어보기

- 지금까지 한 것!

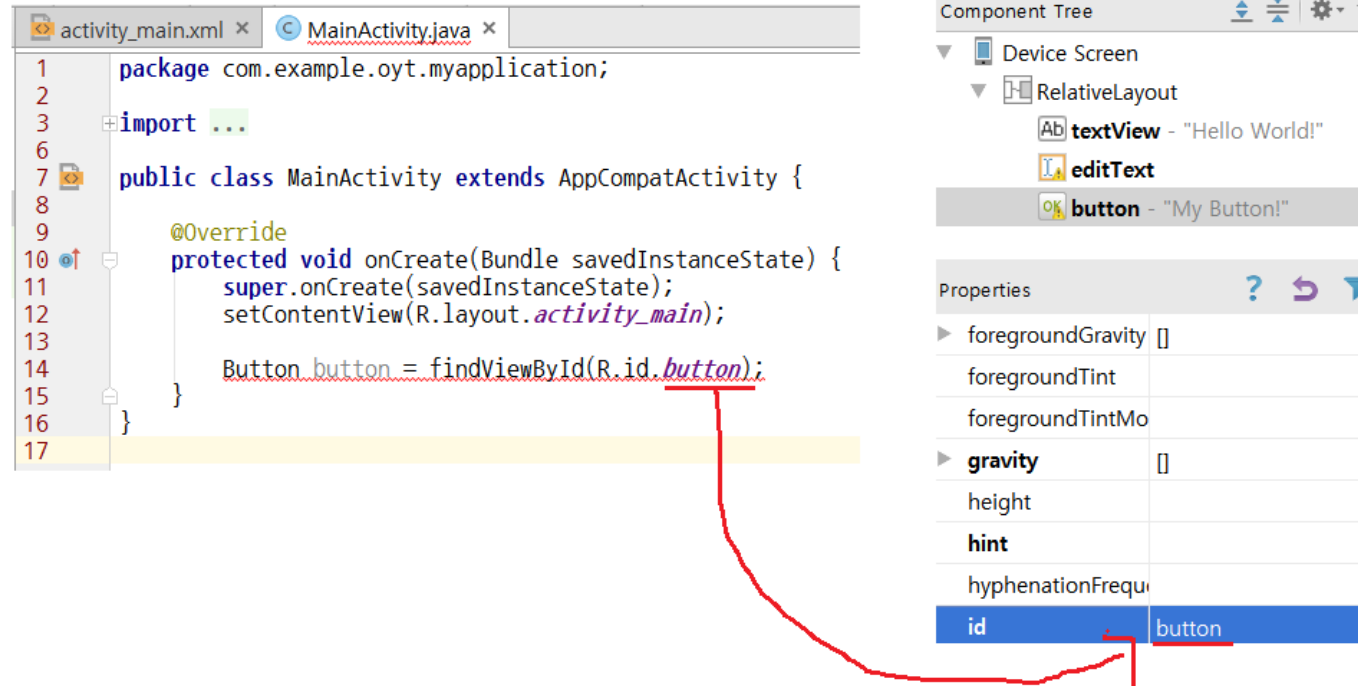
1. 안드로이드 개발 환경 구축
2. Palette 탭에서 원하는 객체 끌어다가 화면에 배치
3. Properties 탭에서 다양한 속성 변경
4. 핸드폰을 사용해 첫 프로젝트 확인
5. 끌어다 배치하는 그런 모든 방식을 xml 텍스트 상에서 동일하게 변경 가능
(나중에 복잡해지면 그렇게 해야만 한다... ㄴㄷㄴㄷ)

버튼을 누르면
메인의 텍스트를 바꾸고 싶어요!!

첫 프로젝트 만들어보기

- 지금까지 추가한 객체들(버튼, EditText, TextView)은 xml 상에서 존재한다.
- 이들을 자바 소스코드 상에서 컨트롤하고 싶다면, 이들을 찾아내야 한다.
- (View) findViewById(int id) 함수를 이용하자!
- Xml 레이아웃에 존재하는 버튼을 잡아내는 방법은...

첫 프로젝트 만들어보기



- 코드 상 R.id. 뒤에 오는 button은 버튼의 id 속성에서 온 것이다.
- 근데, 오류가 나네?

첫 프로젝트 만들어보기

```
activity_main.xml x MainActivity.java x
1 package com.example.oyt.myapplication;
2
3 import ...
6
7 public class MainActivity extends AppCompatActivity {
8
9     @Override
10     protected void onCreate(Bundle savedInstanceState) {
11         super.onCreate(savedInstanceState);
12         setContentView(R.layout.activity_main);
13
14         Button button = findViewById(R.id.button);
15     }
16 }
17
```

Incompatible types.
Required: **android.widget.Button**
Found: **android.view.View**

```
activity_main.xml x MainActivity.java x
1 package com.example.oyt.myapplication;
2
3 import ...
6
7 public class MainActivity extends AppCompatActivity {
8
9     @Override
10     protected void onCreate(Bundle savedInstanceState) {
11         super.onCreate(savedInstanceState);
12         setContentView(R.layout.activity_main);
13
14         Button button = (Button) findViewById(R.id.button);
15     }
16 }
17
```

- 그러하다. findViewById 함수는 View 객체를 리턴하는데, button은 Button형 객체이다.(좌)
- 요로코롬 타입 캐스팅을 통해 좌변과 우변의 객체 타입을 맞추어 주면 된다. (우)
- 이제, button이란 이름으로 JAVA 코드 상에서 사용할 수 있다!

첫 프로젝트 만들어보기

The screenshot displays an IDE interface with three main panels. The left panel shows the `MainActivity.java` file with the following code:

```
1 package com.example.oyt.myapplication;
2
3 import ...
4
5
6
7 public class MainActivity extends AppCompatActivity {
8
9     @Override
10    protected void onCreate(Bundle savedInstanceState) {
11        super.onCreate(savedInstanceState);
12        setContentView(R.layout.activity_main);
13
14        Button button = (Button) findViewById(R.id.button);
15    }
16 }
17
```

The right panel is split into two views. The top view shows the **Component Tree** with the following structure:

- Device Screen
 - RelativeLayout
 - textView - "Hello World!"
 - editText
 - button - "My Button!"

The bottom view shows the **Properties** panel for the selected `textView` component:

Property	Value
height	
hint	
hyphenationFrequency	
id	textView
importantForAccessibility	

The rightmost panel shows the **Component Tree** and **Properties** panels for the `editText` component:

- Device Screen
 - RelativeLayout
 - textView - "Hello World!"
 - editText
 - button - "My Button!"

The **Properties** panel for the selected `editText` component is:

Property	Value
height	
hint	
hyphenationFrequency	
id	editText
importantForAccessibility	

이제 다른 객체들(EditText, TextView)도 동일한 방식으로 java코드 상으로 가져와볼까?

첫 프로젝트 만들어보기

```
activity_main.xml x MainActivity.java x
1 package com.example.oyt.myapplication;
2
3 import ...
9
10 public class MainActivity extends AppCompatActivity {
11
12     @Override
13     protected void onCreate(Bundle savedInstanceState) {
14         super.onCreate(savedInstanceState);
15         setContentView(R.layout.activity_main);
16
17         TextView textView = (TextView) findViewById(R.id.textView);
18         EditText editText = (EditText) findViewById(R.id.editText);
19         Button button = (Button) findViewById(R.id.button);
20         button.setOnClickListener(new View.OnClickListener() {
21             @Override
22             public void onClick(View view) {
23                 //여기에 처리할 코드가 온다.
24             }
25         });
26     }
27 }
28
```

- 버튼을 클릭했을 때 발생할 이벤트를 정의하려면 OnClickListener를 등록해야 한다.
- 코드가 길어보이지만 button.setOn(엔터) -> newOnC(엔터) 누르면 자동으로 입력된다.

첫 프로젝트 만들어보기

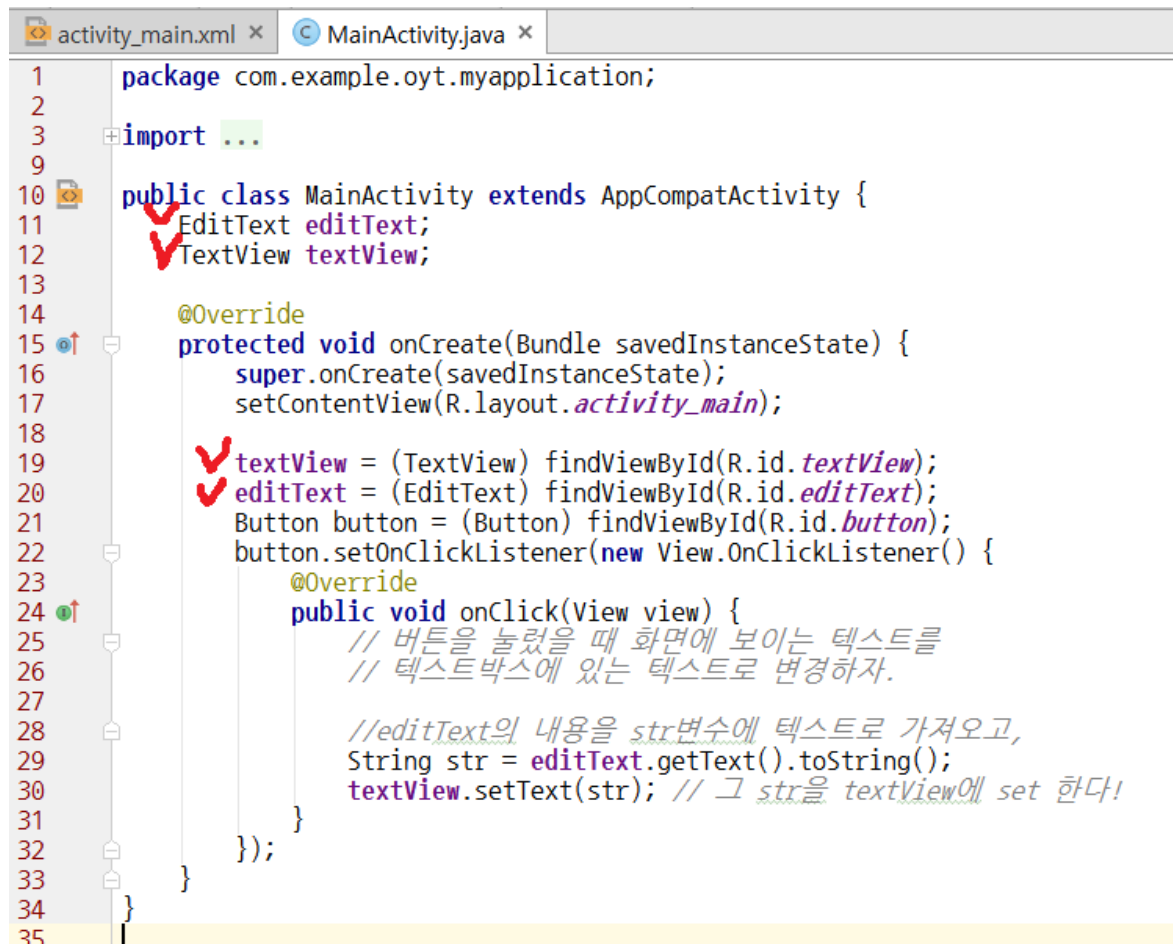
```
Button button = (Button) findViewById(R.id.button);
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        // 버튼을 눌렀을 때 화면에 보이는 텍스트를
        // 텍스트박스에 있는 텍스트로 변경하자.

        // editText의 내용을 str변수에 텍스트로 가져오고,
        String str = editText.getText().toString();
        textView.setText(str); // 그 str을 textView에 set 한다!
    }
});
```

- 근데, 오류가 난다. 확인해보니... ~~~ is accessed from within inner class, ~~
- editText와 textView는 onCreate 메소드 안에서 정의한 것이다.
- 버튼의 onClick 메소드 안에서 해당 객체들을 사용하려면 그것들을 전역으로 선언해야함.
- textView와 EditText를 어디에서든지 사용할 수 있게!!

첫 프로젝트 만들어보기

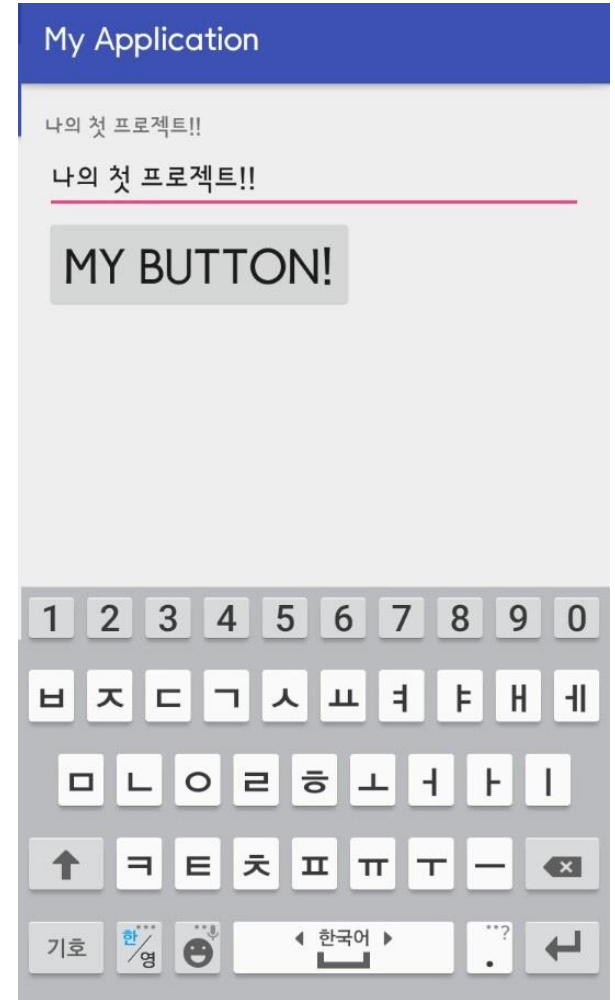
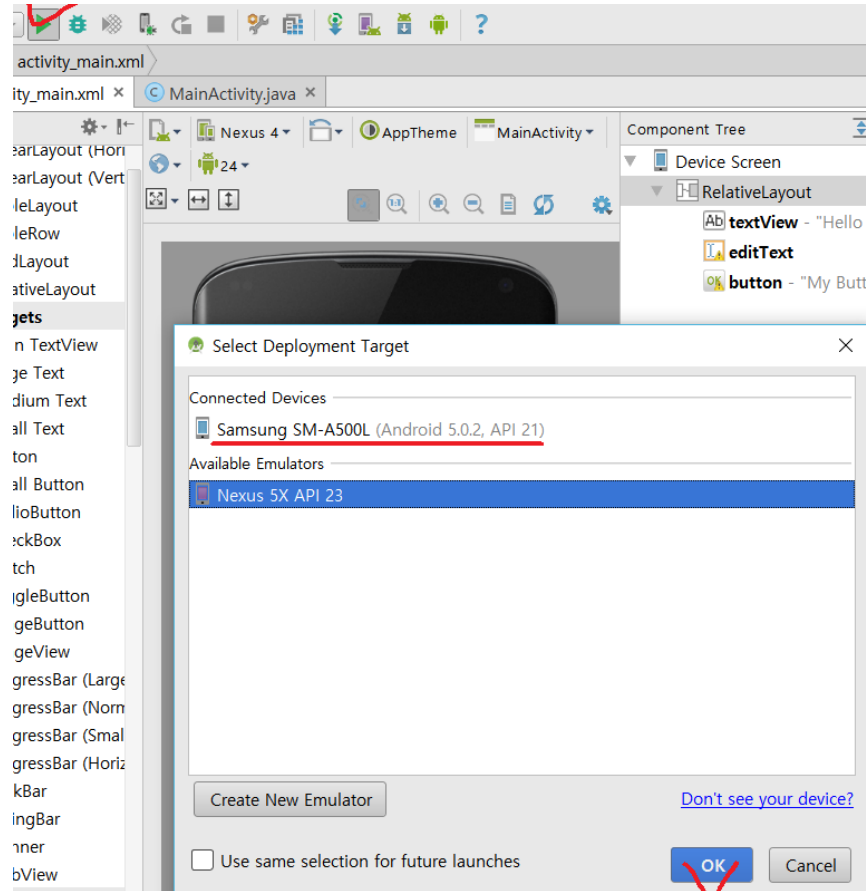
- 이렇게!



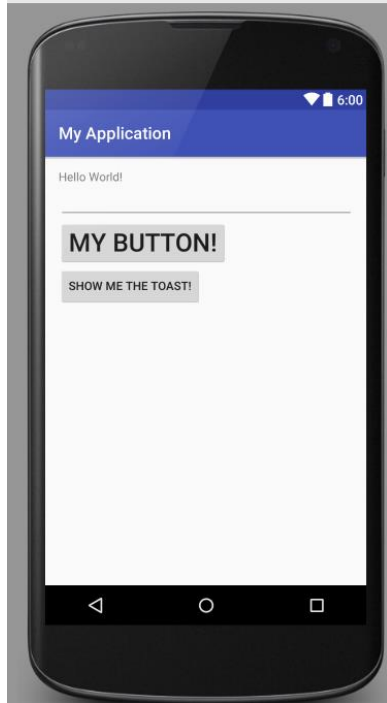
```
1 package com.example.oyt.myapplication;
2
3 import ...
4
5
6
7
8
9
10 public class MainActivity extends AppCompatActivity {
11     ✓ EditText editText;
12     ✓ TextView textView;
13
14     @Override
15     protected void onCreate(Bundle savedInstanceState) {
16         super.onCreate(savedInstanceState);
17         setContentView(R.layout.activity_main);
18
19         ✓ textView = (TextView) findViewById(R.id.textView);
20         ✓ editText = (EditText) findViewById(R.id.editText);
21         Button button = (Button) findViewById(R.id.button);
22         button.setOnClickListener(new View.OnClickListener() {
23             @Override
24             public void onClick(View view) {
25                 // 버튼을 눌렀을 때 화면에 보이는 텍스트를
26                 // 텍스트박스에 있는 텍스트로 변경하자.
27
28                 // editText의 내용을 str변수에 텍스트로 가져오고,
29                 String str = editText.getText().toString();
30                 textView.setText(str); // 그 str을 textView에 set 한다!
31             }
32         });
33     }
34 }
35
```

첫 프로젝트 만들어보기

- 디버깅!
- 사장님 나이스 샷!



첫 프로젝트 만들어보기



```
Button btnToast = (Button) findViewById(R.id.btnToast);
btnToast.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String str = editText.getText().toString();
        ✓ Toast.makeText(getApplicationContext(), str, Toast.LENGTH_LONG).show();
        ✓ Toast tst = Toast.makeText(getApplicationContext(), str, Toast.LENGTH_LONG);
        tst.show();
    }
});
```

- 번외편: 토스트 메시지 띄워보기
- Xml 파일 상에서 새로운 버튼을 추가하고, 텍스트를 Show me the Toast!, id값을 btnToast로 주자.
- 첫 번째 체크와 두 번째 체크는 사실상 동일한 기능을 한다.

첫 프로젝트 만들어보기

- Toast.makeText()는 Toast 객체를 반환해주고, 이 객체를 show() 메소드를 이용하여 화면 상에 메시지를 띄울 수 있다.

- makeText()의 첫 번째 인자는 컨텍스트, 관례적으로 들어간다고만 알아두자. 코드는 `getApplicationContext()` .

- 두 번째 인자는 넣고 싶은 텍스트가 들어간다.

- 세 번째 인자는 메시지를 띄울 시간(duration)이다.

상수가 미리 정의되어 있고, `Toast.LENGTH_LONG`은 길게, `Toast.LENGTH_SHORT`는 그 것보다 짧게 메시지가 지속된다.

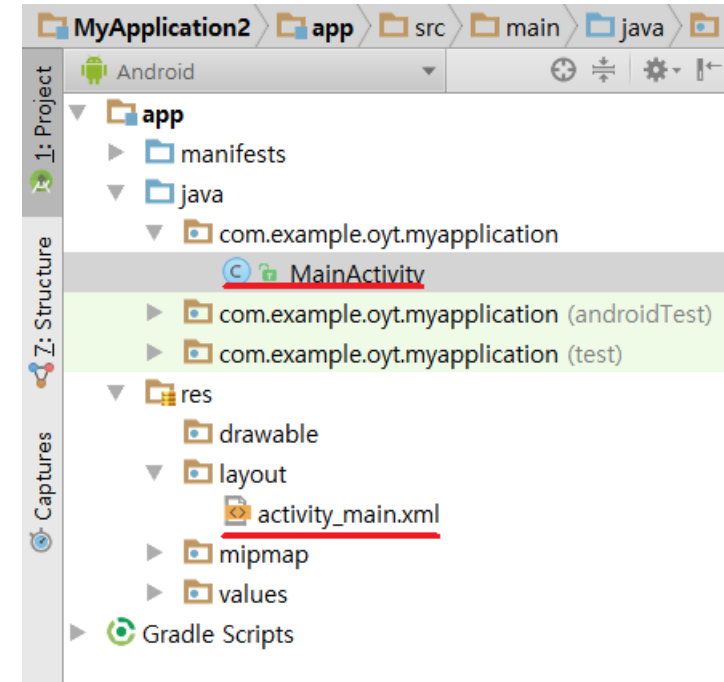


어플리케이션 다듬기

새로운 창을 띄우고 싶어요 !

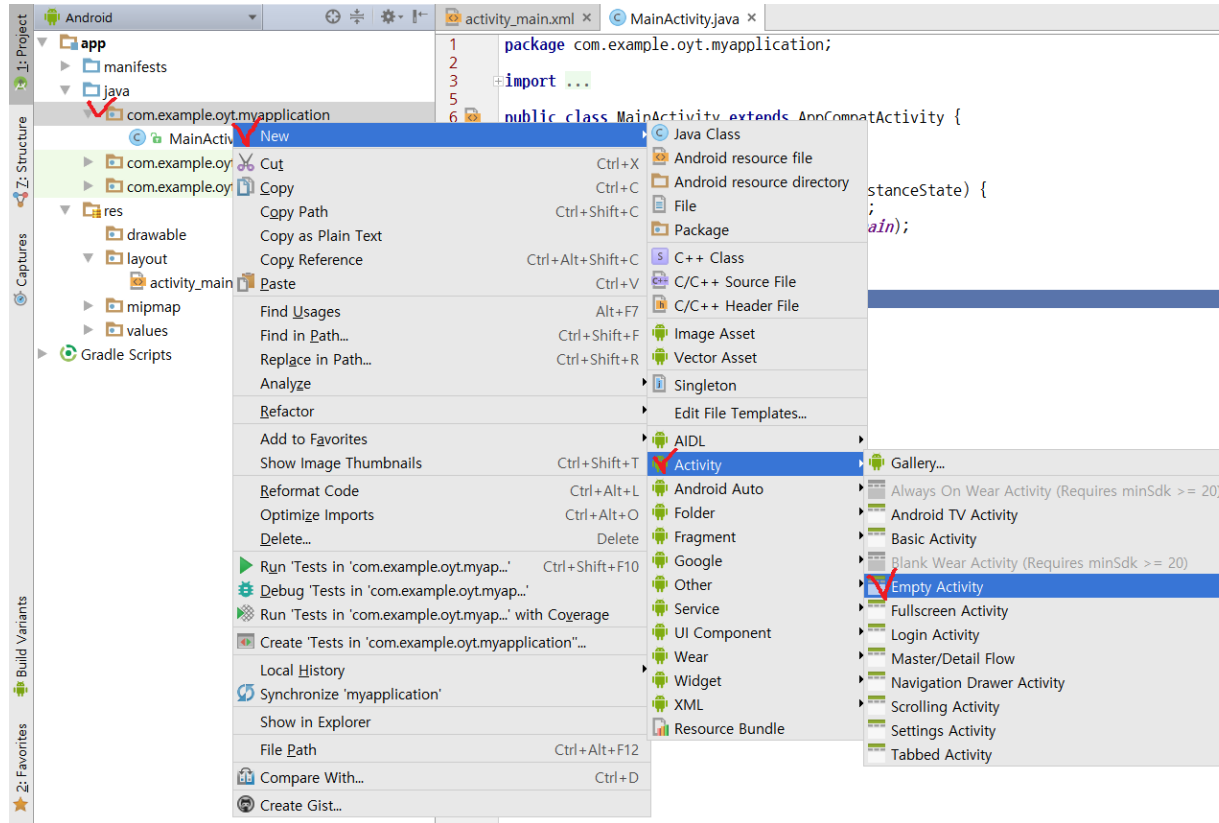
어플리케이션 다듬기

- 안드로이드에서 하나의 화면을 담당하는 것은 하나의 액티비티.
- 어플리케이션에서 어떤 화면이 보인다는 것은 해당 액티비티가 열려있다는 것과 같다.
- 그림에서 볼 수 있듯 액티비티는 java 파일과 xml 파일의 조합으로 구성됨
- Xml은 화면 상의 위젯 요소들을 배치해주고, java 파일은 그것들을 원하는대로 제어할 수 있는 역할.
(사실 java파일에서 버튼같은 객체들을 직접 추가할수도 있음)



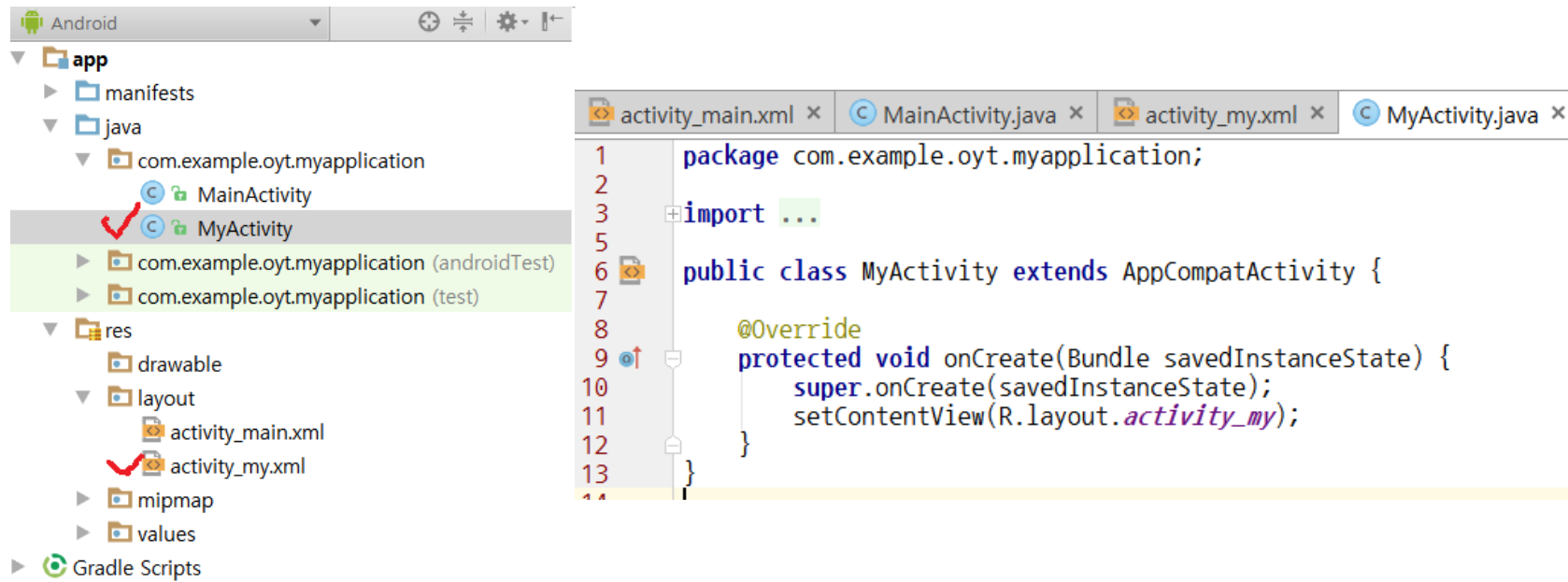
그렇다! 우리는 지금까지 MainActivity를 만지고, 그곳에서 작업했던 것인 것이다!

어플리케이션 다듬기



- 새로운 액티비티를 추가해 보자. 이름은, MyActivity. 입력후 Finish!

어플리케이션 다듬기

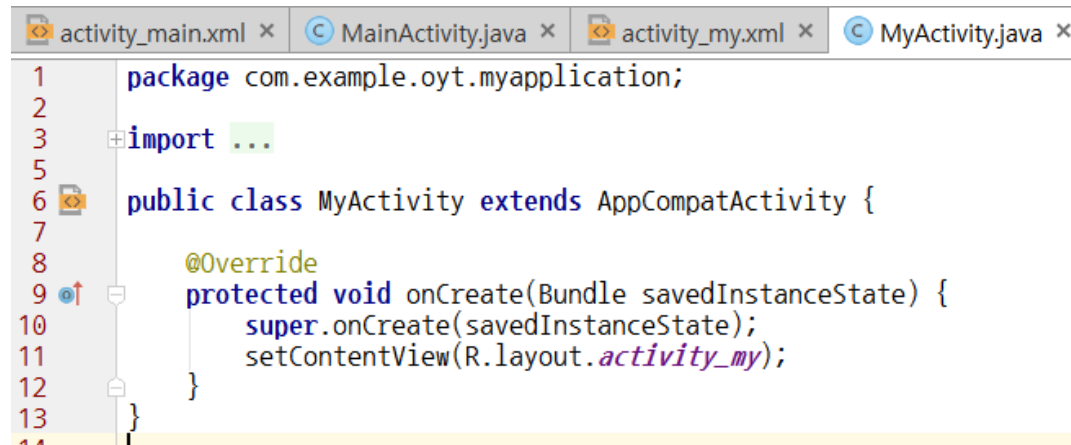


- 추가되었다 !
- 한 쌍의 java 파일과 xml 파일이 생성됨!



어플리케이션 다듬기

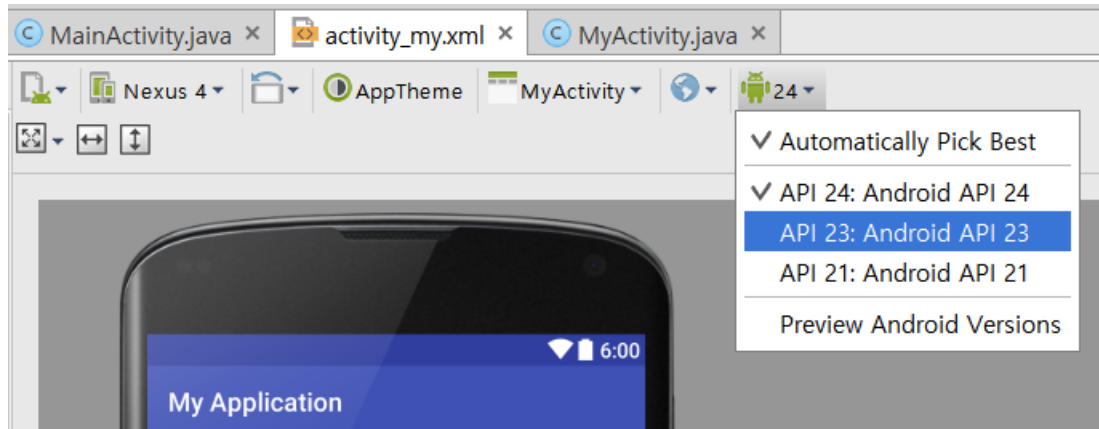
- 이젠 해석할 수 있어요!
- onCreate는 뭘지는 모르겠지만 어플리케이션이 실행된 뒤 자동으로 그 안의 코드를 실행시켜주는구나!
- setContentView 이하 문장은 말 그대로, 미리 짜여진 xml layout 파일인 activity_my라는 파일을 java 파일과 연결하여 화면에 내용으로 보여준다는 말이구나!



```
1 package com.example.oyt.myapplication;
2
3 import ...
4
5
6 public class MyActivity extends AppCompatActivity {
7
8     @Override
9     protected void onCreate(Bundle savedInstanceState) {
10         super.onCreate(savedInstanceState);
11         setContentView(R.layout.activity_my);
12     }
13 }
```

어플리케이션 다듬기

- 적당히 꾸며준다.
- 한글이 깨지면 아래와 같이 API 버전을 낮추어 설정해본다.



<activity_my.xml>



어플리케이션 다듬기

- btnBack이라는 id를 가진 버튼을 xml 레이아웃에 추가한 뒤, 다음과 같이 MainActivity.java를 코딩한다.

```
public class MainActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_my);  
  
        Button btnBack = (Button) findViewById(R.id.btnBack);  
        btnBack.setOnClickListener(new View.OnClickListener() {  
            @Override  
            public void onClick(View view) {  
                finish();  
            }  
        });  
    }  
}
```


어플리케이션 다듬기

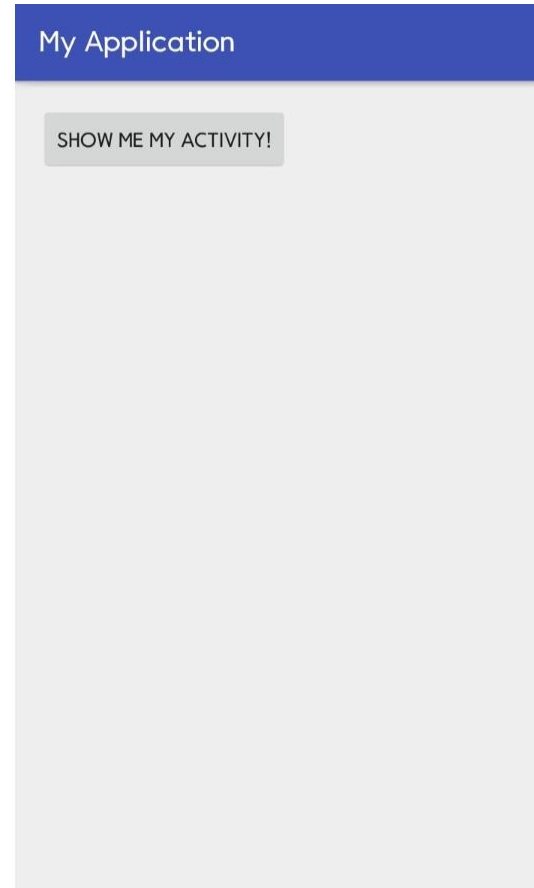
```
public class MainActivity extends AppCompatActivity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        Button btnShow = (Button) findViewById(R.id.btnShow);  
        btnShow.setOnClickListener(new View.OnClickListener() {  
            @Override  
            public void onClick(View view) {  
                Intent intent = new Intent(getApplicationContext(), MyActivity.class);  
                startActivity(intent);  
            }  
        });  
    }  
}
```

- activity_main.xml에 btnShow라는 아이디를 가진 버튼을 추가한다.
- MainActivity.java를 위와 같이 코딩한다.

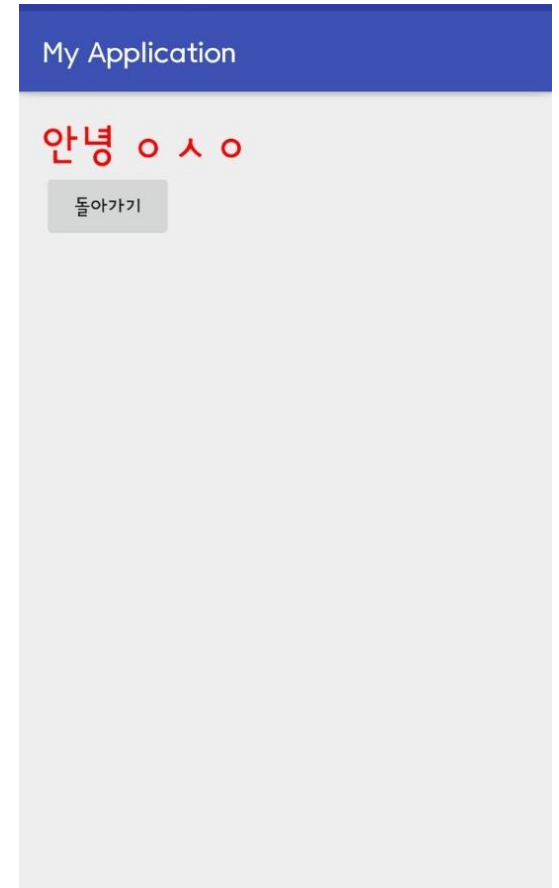
어플리케이션 다듬기

- 디버깅!

MainActivity



MyActivity

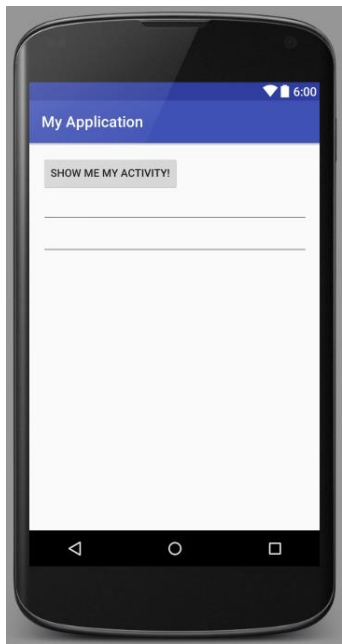


인텐트! 그것이 알고싶다.

- 인텐트(Intent)는 시스템이 이해할 수 있는 객체를 정의한 것으로 생각할 수 있다.
- 안드로이드에서 화면을 띄우려면(=새로운 Activity를 띄우려면) 안드로이드 시스템에 요청해야 한다.
- 여기서 인텐트는 안드로이드 시스템에게 "화면 좀 띄워줘!"라 하는 메시지라고 보자.
- Intent의 첫 번째 인자인 `getApplicationContext()` 메소드는 이 어플리케이션의 Context 객체를 참조 후 시스템에 전달하는 역할을 한다.
- 인텐트를 보낼 때 추가적인 메시지(내용)들을 넣어서 보낼 수 있다.

인텐트! 그것이 알고싶다.

- MainActivity에서 MyActivity로 이름과 나이 데이터를 전달해보자.
- MainActivity에 editName, editAge의 아이디를 가진 EditText 객체를 더 추가하자.
- editAge 텍스트의 inputType 속성을 Number로 바꿔주자. 다음과 같이 MainActivity.java를 코딩한다.



```
10 public class MainActivity extends AppCompatActivity {
11     EditText editName, editAge;
12     @Override
13     protected void onCreate(Bundle savedInstanceState) {
14         super.onCreate(savedInstanceState);
15         setContentView(R.layout.activity_main);
16         editName = (EditText) findViewById(R.id.txtName);
17         editAge = (EditText) findViewById(R.id.editAge);
18         Button btnShow = (Button) findViewById(R.id.btnShow);
19         btnShow.setOnClickListener(new View.OnClickListener() {
20             @Override
21             public void onClick(View view) {
22                 String name = editName.getText().toString();
23                 int age = Integer.parseInt(editAge.getText().toString()); // editAge에서 정수를 뽑아낸다.
24                 Intent intent = new Intent(getApplicationContext(), MyActivity.class); // MyActivity를 띄우는 intent 정의
25                 Bundle bundle = new Bundle(); // 이름과 나이 정보를 넣을 bundle 객체 정의
26                 bundle.putInt("Age", age); // bundle에 Age라는 key로 age 값 투입
27                 bundle.putString("Name", name); // bundle에 Name이라는 key로 name 값 투입
28                 intent.putExtra("MyData", bundle); // intent에 MyData라는 key를 가진 bundle 투입
29                 startActivity(intent); // intent를 시스템에 전달 -> 새로운 액티비티가 뜬다.
30             }
31         });
32     }
33 }
```

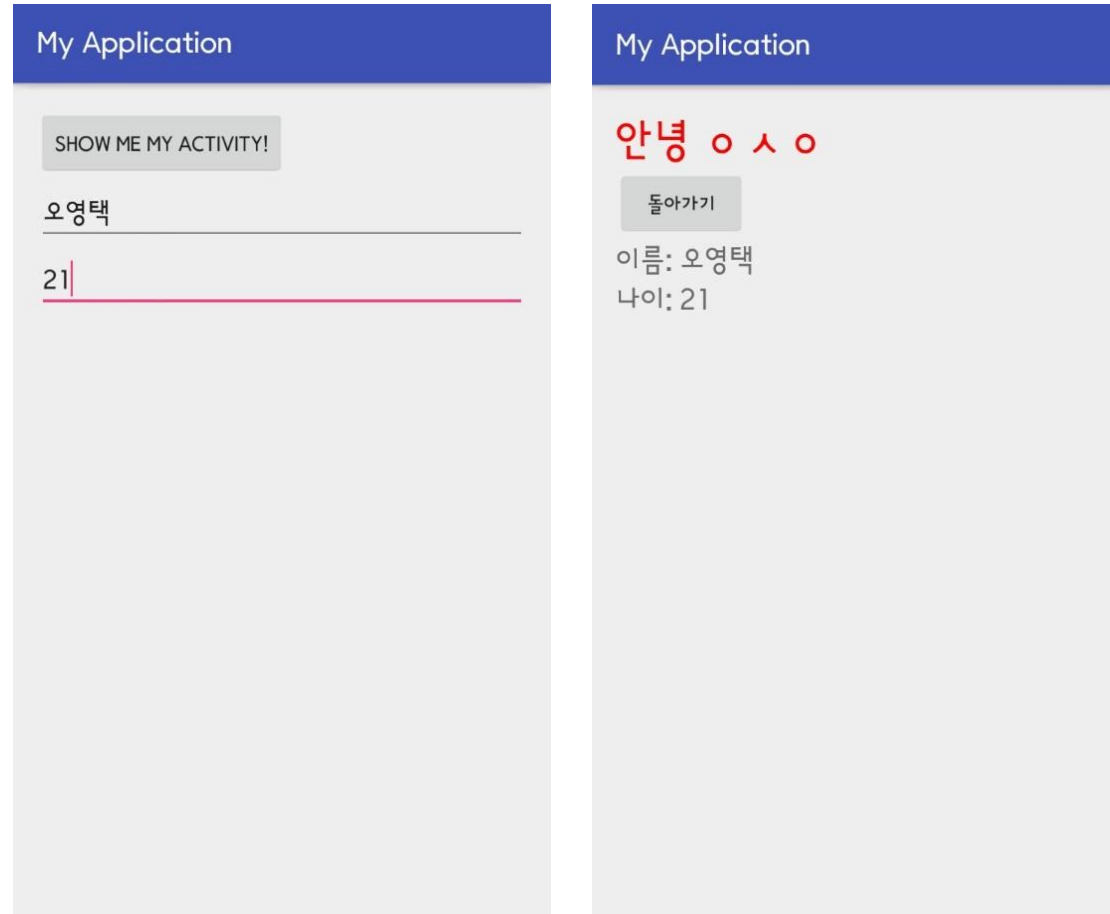
인텐트! 그것이 알고싶다.

- MyActivity에 txtName, txtAge라는 아이디를 가진 TextView를 추가한다.
- MyActivity를 다음과 같이 코딩한다. (btnBack 항목은 빼두고, setContentView ~~ 아래줄)

```
public class MyActivity extends AppCompatActivity {
    TextView txtName, txtAge;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_my);
        txtName = (TextView) findViewById(R.id.txtName);
        txtAge = (TextView) findViewById(R.id.txtAge);
        Intent intent = getIntent();
        if(intent != null){ //전달된 intent가 있으면
            Bundle data = intent.getBundleExtra("MyData"); // MyData라는 key를 가진 bundle 추출
            if(data != null){
                // 아까 key가 Age, Name이었던 데이터를 각각 꺼낸다.
                txtAge.setText("나이: " + data.getInt("Age"));
                txtName.setText("이름: " + data.getString("Name"));
            }
        }
    }
}
```

인텐트! 그것이 알고싶다.

- 디버깅!



StartActivity vs. StartActivityResult

- 새로운 액티비티를 띄우기 위해 사용했던 메소드가 startActivity인데, 이는 단순히 액티비티를 띄워 화면에 보이도록 만든다.
- 실제 어플리케이션을 구성할 경우 메인 액티비티에서 띄워야 할 화면들이 여러 개가 될 수도 있고,
- 띄웠던 화면을 닫고 원래의 메인 화면으로 돌아올 때 데이터를 새로 적용해야 하는 경우도 생김.
- 즉 어떤 액티비티를 띄운 것인지, 띄웠던 액티비티로부터의 응답을 받아 처리하는 코드 필요.

`startActivityResult(Intent intent, int requestCode)`

startActivityResult Method

- MainActivity에 다음과 같이 REQUEST_CODE_MYACTIVITY 상수 선언

```
public class MainActivity extends AppCompatActivity {  
    EditText editName, editAge;  
    public static final int REQUEST_CODE_MYACTIVITY = 1001; //임의로 코드 정의(실행한 액티비티를 지칭함)
```

커서를 onCreate메소드 밖에 위치한 채 Ctrl+o를 눌러서 onActivityResult 메소드 추가, 코딩

```
@Override  
protected void onActivityResult(int requestCode, int resultCode, Intent data) {  
    super.onActivityResult(requestCode, resultCode, data);  
  
    if(requestCode == REQUEST_CODE_MYACTIVITY){ //띄운 액티비티가 MainActivity일 경우  
        Toast toast = Toast.makeText(getApplicationContext(),  
            "MainActivity로부터 응답이 왔습니다. 결과코드: " + resultCode, Toast.LENGTH_LONG);  
        toast.show();  
  
        if(resultCode == RESULT_OK){ //resultCode가 RESULT_OK일 경우  
            String name = data.getExtras().getString("Name");  
            toast = Toast.makeText(getApplicationContext(), "응답으로 전달된 name->" + name, Toast.LENGTH_LONG);  
            toast.show();  
        }  
    }  
}
```

startActivity 메소드를 대체 : startActivityForResult(intent, REQUEST_CODE_MYACTIVITY);

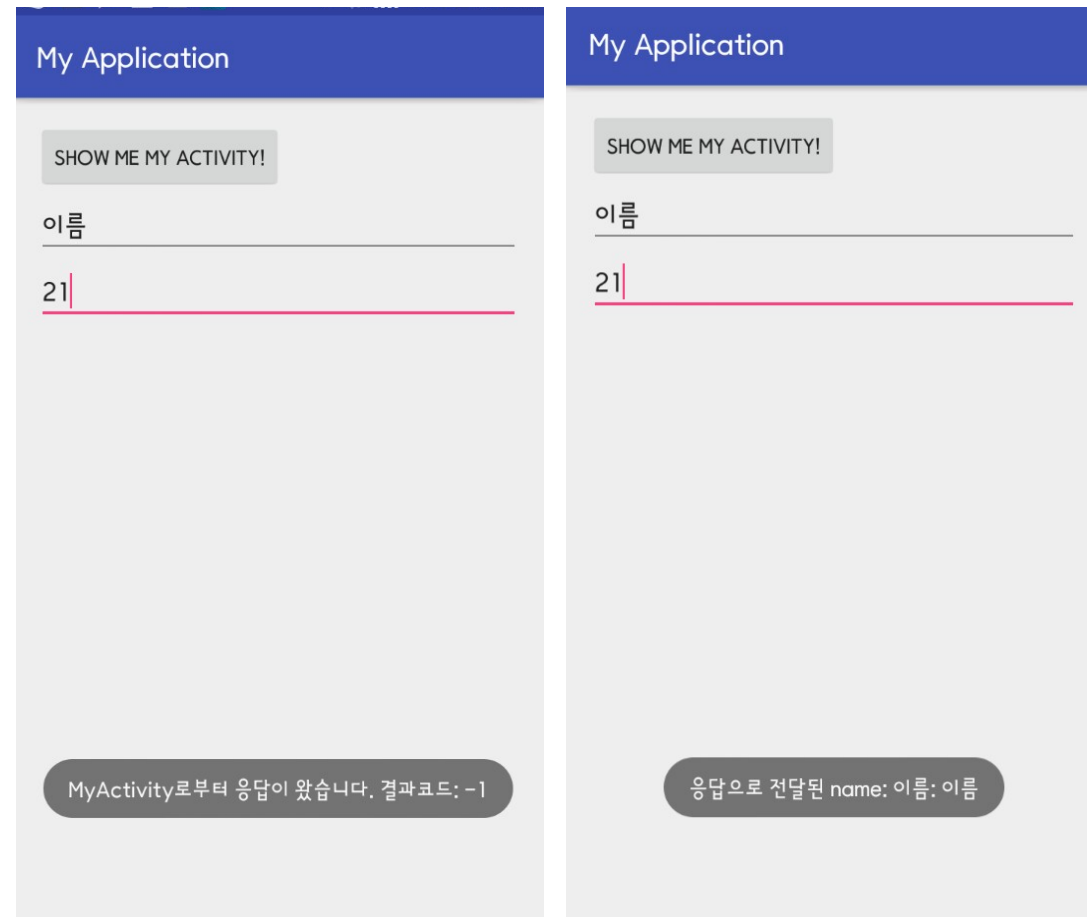
startActivityForResult

- MyActivity.java의 onClick 메소드 안의 코드를 다음과 같이 수정

```
Button btnBack = (Button) findViewById(R.id.btnBack);
btnBack.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent(); //MainActivity로 전달할 intent 생성
        intent.putExtra("Name", txtName.getText().toString()); //intent에 Name이라는 키를 가진 값 투척
        setResult(RESULT_OK, intent); //resultcode 설정
        finish();
    }
});
```

startActivityForResult

- 디버깅!



액티비티 띄우기와 Intent 정리!

- 새로운 액티비티를 띄우기 위해서는 Intent 객체를 통하여 시스템에 알려야 한다.
- 이 때 사용하는 메소드는 startActivity와 startActivityForResult 메소드.
- 또, 인텐트에 부가 정보를 넣어서 시스템에 전달할 수 있다. (새로운 액티비티에서 수신)
- Bundle 객체를 이용하면 여러 정보를 한꺼번에 담아서 보낼 수 있다.

(굳이 그러지 않아도 괜찮음)

- startActivityForResult 메소드는 요청코드와 결과코드를 처리한다. (onActivityResult)
- requestCode를 통해 내가 어떤 액티비티를 띄웠는지 확인할 수 있으며,
- resultCode를 통해 해당 액티비티가 어떤 result를 보냈는지 확인할 수 있다.(setResult)

숙제

- 액티비티 수명 주기에 대해 조사(만) 하시오.
- onCreate(), onStart(), onResume, onRestart()
- onPause(), onStop(), onDestroy()

Layout – Relative, Frame, Linear

- Relative Layout : 해당 레이아웃에선 배치될 수 있는 위젯들의 위치가 상대적임.
어떤 위젯이 다른 위젯의 바로 위나 아래, 옆에 위치하도록 조정할 수 있음.
혹은, 좌표나 parent 속성을 조정하여 위젯의 위치를 직접 지정 가능
- Linear Layout: 배치될 수 있는 위젯들은 오직 한 방향만을 가짐. (수직 / 수평)
Xml 파일을 뜯어보면 Linear Layout에는 방향성분(orientation)을 갖고있음.
수직과 수평 Linear Layout을 조합하여 원하는 화면을 구성할 수 있다.
- Frame Layout : Frame, 즉 어떤 개체들을 담아둘 수 있는 컨테이너 역할을 한다.
(나중에 만나게 될 것)

Example: Linear Layout

- Linear Layout을 이용하여 로그인 레이아웃 배치
- Empty Activity를 가진 새로운 프로젝트를 생성한다.
- Activity_main.xml 파일을 열어서 Hello World!라 쓰인 텍스트뷰 제거
- 빈 화면에 LinearLayout(Horizontal) 추가
- 방금 추가한 Layout '안에' LinearLayout(Vertical)과 Button 추가.
(이 때 LinearLayout들의 layout:width, height는 모두 wrap_content로 맞추어준다.)
- Vertical LinearLayout 안에 editText 두 번 추가

Example: Linear Layout

- 짤!
- EditText의 layout:width와 버튼의 layout:height 수치를 적절히 조절하면 오른쪽 사진과 같은 레이아웃을 구현할 수 있다!
- 이러한 구성은 xml파일을 연 상태에서 Text 탭을 누르면 텍스트 형태로 작성된 xml 파일을 확인할 수도 있다.



Example: Linear Layout

- 가장 상위의 Relative Layout 아래로 여러 레이아웃들이 포함되어 있음을 알 수 있다.

(우리는 이 작업을 단지 드래그&드롭만으로 한 것이다.)

- 텍스트 상에서 속성을 바꾸면, design 탭으로 바로 반영이 된다.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout 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:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.ojt.myapplication.MainActivity">

    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true">

        <LinearLayout
            android:orientation="vertical"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content">

            <EditText
                android:layout_width="200dp"
                android:layout_height="wrap_content"
                android:id="@+id/editText" />

            <EditText
                android:layout_width="200dp"
                android:layout_height="wrap_content"
                android:id="@+id/editText2" />

        </LinearLayout>

        <Button
            android:layout_width="wrap_content"
            android:layout_height="80dp"
            android:text="New Button"
            android:id="@+id/button" />

    </LinearLayout>

</RelativeLayout>
```


숙제!

- Required

- 메인 액티비티: TextView와 Button을 하나 배치한다.

(Button의 텍스트는 Login, TextView는 아직 아무 처리도 하지 않는다.)

- 메인 액티비티의 Login 버튼을 누르면 두번째 액티비티를 띄운다. (startActivityForResult 이용)

(이 액티비티에는 아이디와 비밀번호를 입력할 editText 객체가 두 개 있고, 로그인을 시도하는 button이 있다.)

- 두 번째 액티비티의 로그인 버튼을 누를 시 입력한 아이디값을 intent로 MainActivity에 전달한다.

(setResult 메소드 이용, 건들지 않았던 TextView에 '환영합니다 OOO님'이라는 텍스트를 표시한다.)

(onActivityResult 메소드 역시 이용.)

(로그인 실패여부는 고려하지 않는다.)

Layout Inflation

- 화면을 구성하기 위해서는 xml 레이아웃을 먼저 정의 -> 액티비티에 적용하는 과정
- onCreate() 내의 setContentView 메소드가 어플리케이션이 xml레이아웃의 내용을 이해하도록 만드는 역할을 함.
- 이렇게, xml 레이아웃에 정의된 내용이 메모리 상에 객체화되는 과정을 인플레이션(Inflation)이라 함.
- Xml 레이아웃 파일의 경우 프로젝트가 빌드되는 시점에 컴파일되어 어플리케이션에 포함되긴 하지만 실행 시점에서야 로드되어 메모리상에 객체화됨.
- setContentView 이전에는 findViewById로 객체를 참조할 수 없다!

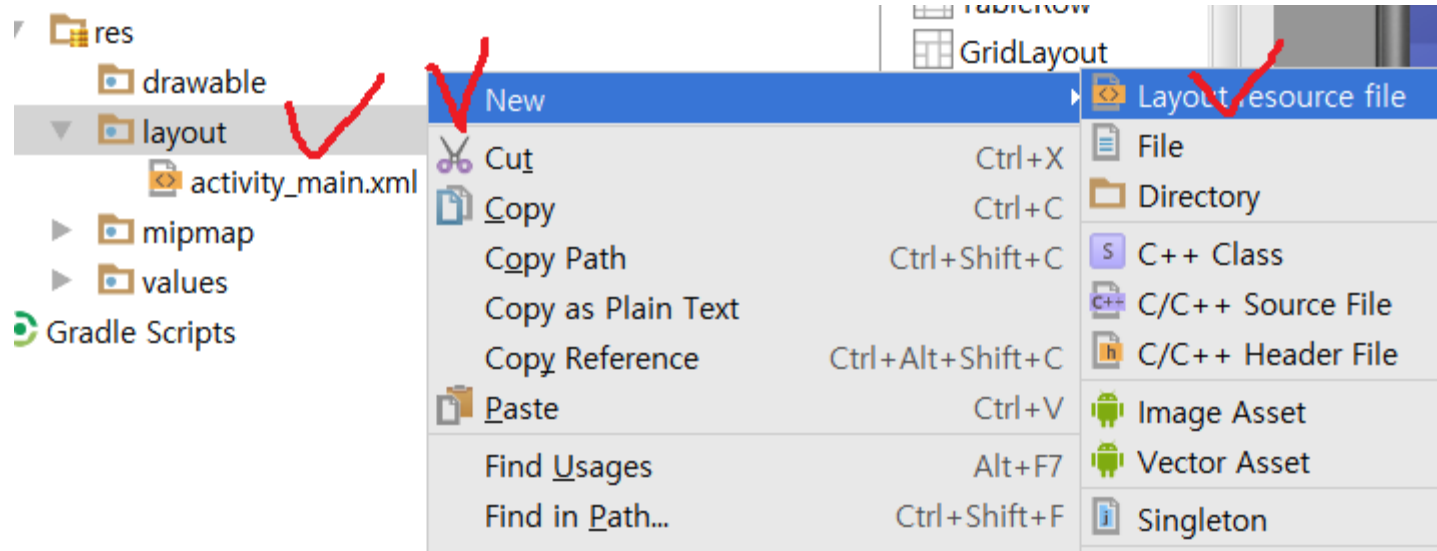
Layout Inflation

- 새로운 xml 파일을 정의한 뒤 이를 MainActivity에 띄워보자!
- 새로운 프로젝트를 만들고, activity_main.xml에 Button과 FrameLayout을 추가한다.
- FrameLayout의 layout:width와 height는 모두 wrap_content로 맞추어 준다.
- FrameLayout의 id값을 편의상 container로 설정한다.



LayoutInflation

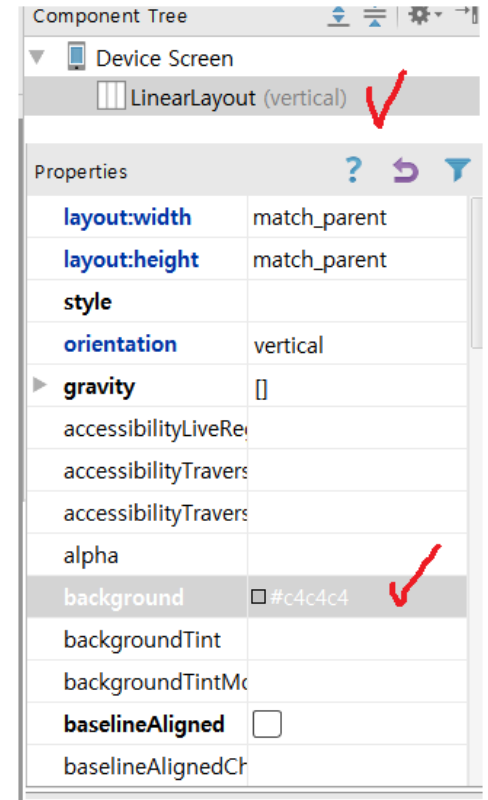
- res/layout에서 우클릭, New -> Layout resource file을 클릭, 새로운 xml layout 파일을 만들어준다.
- 이름은 sub_layout.xml로 설정해주자. (대문자 불가능)



Layout Inflation

- sub_layout.xml 레이아웃 파일을 연 뒤
마음대로 꾸며준다.
- 단, MainActivity의 배경과 구분짓기 위해
background 색을 마음대로 바꿔준다.

sub_layout.xml ->



Layout Inflation

- MainActivity.java를 다음과 같이 코딩한다.

```
public class MainActivity extends AppCompatActivity {
    FrameLayout container;

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

        container = (FrameLayout) findViewById(R.id.container);
        Button button = (Button) findViewById(R.id.button);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                LayoutInflater inflater = (LayoutInflater) getSystemService(Context.LAYOUT_INFLATER_SERVICE);
                inflater.inflate(R.layout.sub_layout, container, true);
            }
        });
    }
}
```

Layout Inflation

- setContentView() 메소드는 액티비티의 화면 전체를 설정하는 역할이므로, 부분적인 뷰를 위한 xml Layout을 Inflation 하려면(=xml layout을 메모리 상에 객체로 로드 하려면) 별도의 인플레이션 객체를 사용해야 한다.
- 안드로이드에서, 시스템 서비스로 제공되는 LayoutInflater 클래스 사용!

getService(Context.LAYOUT_INFLATER_SERVICE)

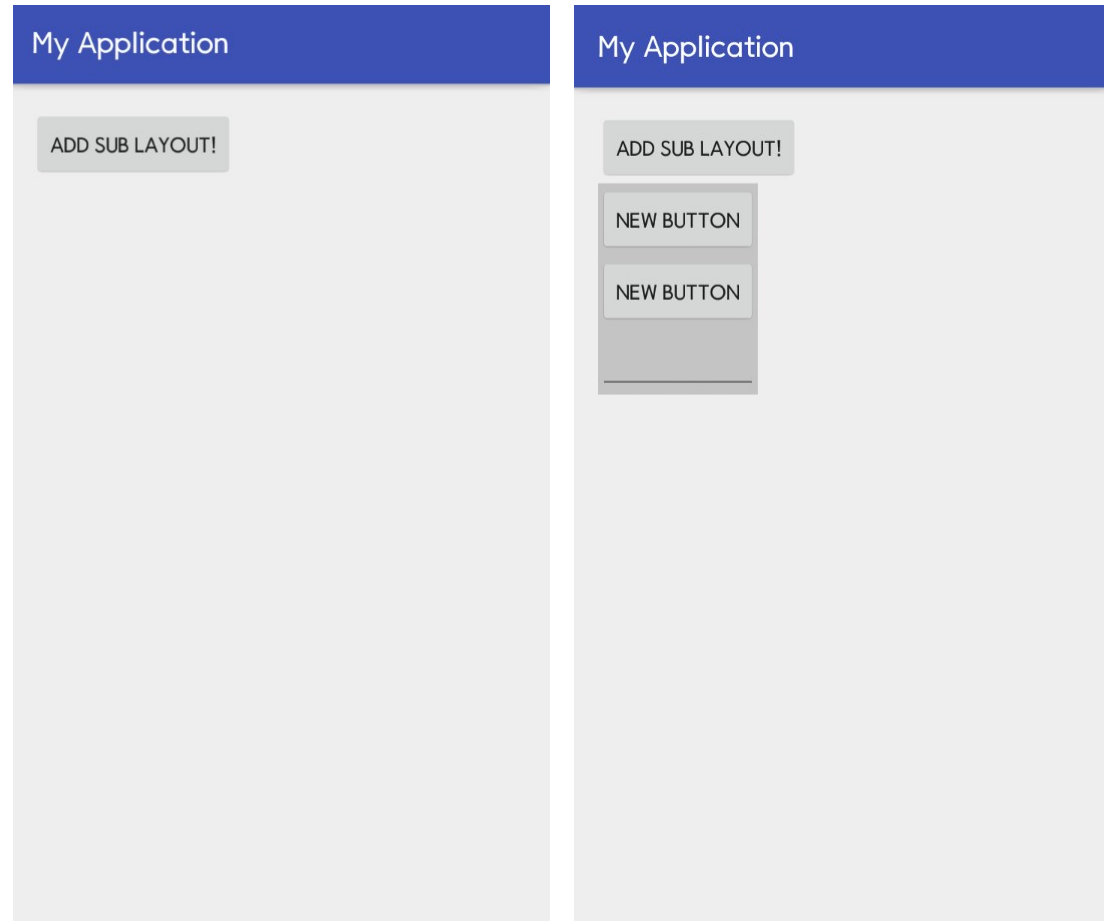
- 이 함수는 Object 형 객체를 리턴하므로, LayoutInflater형으로 형변환 해서 받는다.
- > `LayoutInflater inflater = (LayoutInflater) getSystemService(Context.LAYOUT_INFLATER_SERVICE);`

Layout Inflation

- LayoutInflater형 inflater 객체를 inflate 메소드로 인플레이션을 진행한다.
-> `inflater.inflate(R.layout.sub_layout, container, true);`
- Inflate의 첫 인자는 인플레이션 할 xml 레이아웃 파일(이전에 추가한 sub_layout.xml)이며
- 두 번째 인자는 어디에 추가할 것인지 정하는 역할을 한다.
(container라는 아이디를 가진 FrameLayout)에 sub_layout.xml을 로드한다.
- 세 번째 인자 true는 언제 로드할 것인지 정한다. (바로 로드할 것이므로 true를 적는다.)

Layout Inflation

- 짤
- 새로운 부분화면이 container에 추가됨을 확인!



Layout Inflation

- 왜 쓸까?
- 다음의 소스코드를 보자.

```
button.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
        LayoutInflater inflater = (LayoutInflater) getSystemService(Context.LAYOUT_INFLATER_SERVICE);  
        //inflater.inflate(R.layout.sub_layout, container, true);  
  
        ViewGroup subView = (ViewGroup) inflater.inflate(R.layout.sub_layout, container, true);  
  
        Button sub_btn1 = (Button) subView.findViewById(R.id.button2);  
        sub_btn1.setText("버튼의 이름을 바꿔보자.");  
    }  
});
```

Layout Inflation

- 오른쪽처럼, sub_layout의 요소들을 찾아서 마음대로 변경할 수 있다.
- Inflater.inflate 메소드가 리턴하는 객체를 ViewGroup형으로 받을 수 있음.
- 그렇다면, 해당 뷰에 존재하는 다양한 버튼같은 객체들을 참조하고, 변경하고, 어떤 동작을 정의할 수 있음!
- 그렇다면, sub_layout으로 데이터도 전달할 수 있겠지!



Layout Inflation 정리!

- 사실, 지금 본 Inflation 예제는 xml 레이아웃으로 정의된 파일이 Layout Inflation 과정을 거쳐서 메모리 상에 객체화 된 후 화면에 보여지는 과정을 이해하기 위한 것임.
- Xml 파일과 java 파일이 분리되어 있기 때문에 이러한 과정이 필요하다는 것을 깨닫자.
- 실제로 부분화면을 구성하는 방법은 CustomView를 정의하거나,
 - Fragment를 사용하여 새로운 부분화면을 추가하거나, Tab Layout처럼 화면을 전환하는 기능을 구현함.
- Layout Inflation 과정은 매우매우 중요하다!
- ListView를 구현할 때든지, Tab Layout, View Pager를 구현할 때 다시 찾을 것이다.

끝

- Thank You!
-

