

Samsung Software Junior Membership Android CodeLab

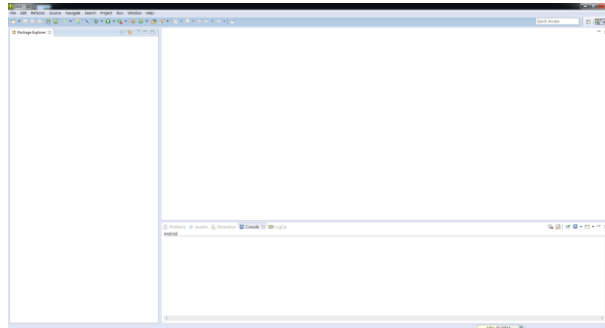
Teacher : 23-2 Jake Yoon

Date : 2014. 03. 29

#2. WeMakeFacebook

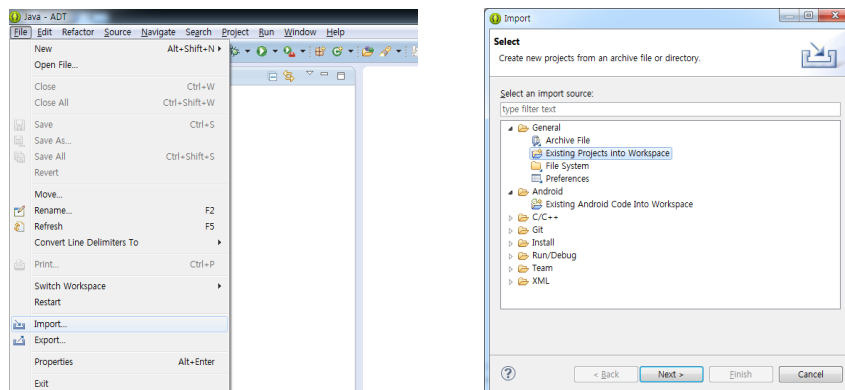
#2-1 개발환경 설정

ADT (Android Developer Tool : Eclipse) 를 실행합니다. (깨끗하군요 ^^)

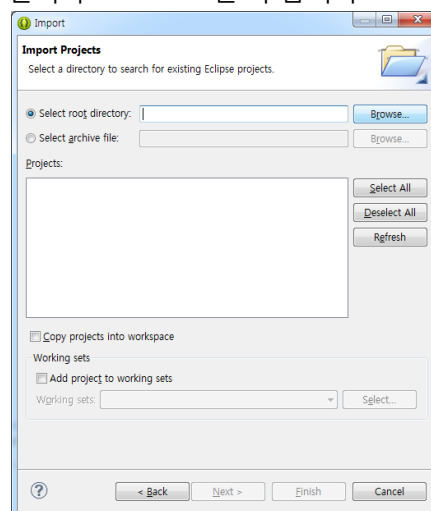


빠대가 될 프로젝트와 ActionBar Sherlock, MenuDrawer, Facebook SDK 라이브러리를 추가하도록 하겠습니다.

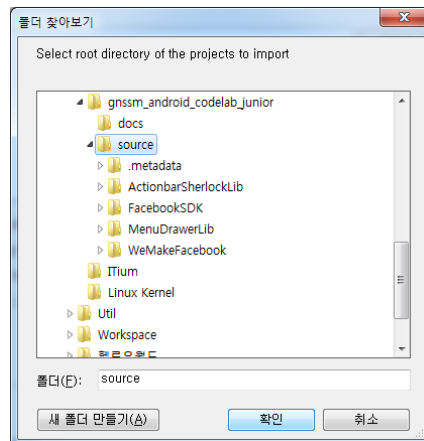
@File > Import > General > Existing Projects into Workspace



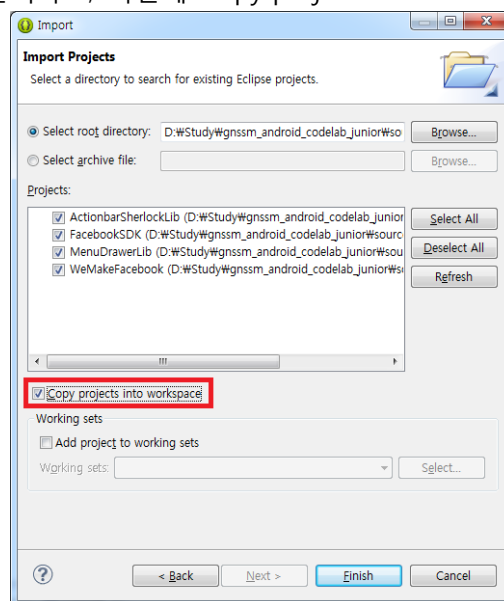
Select root directory 옵션을 선택하고 Browse를 누릅니다.



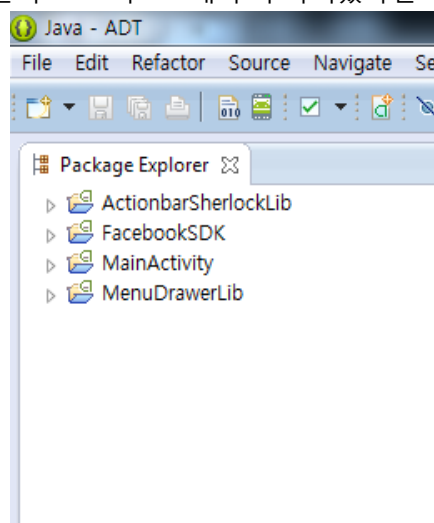
git으로 다운로드 받은 내용 중 source라는 폴더를 선택하고 확인을 누릅니다.



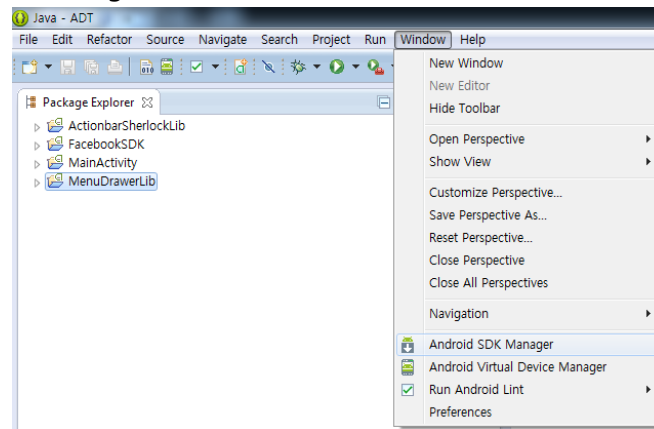
4개의 프로젝트를 모두 선택하고, 하단에 Copy projects into workspace를 눌러줍니다. (꼭!!)



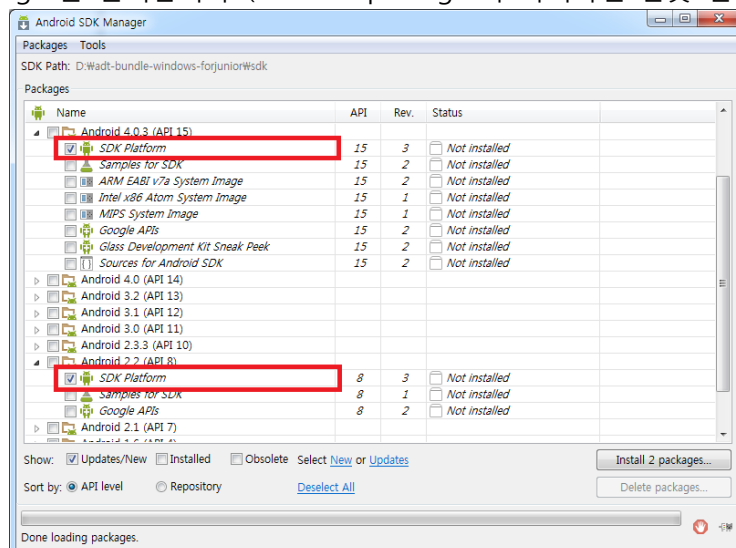
Package Explorer에 아래와 같이 프로젝트 4개가 추가되었다면 추가 성공!



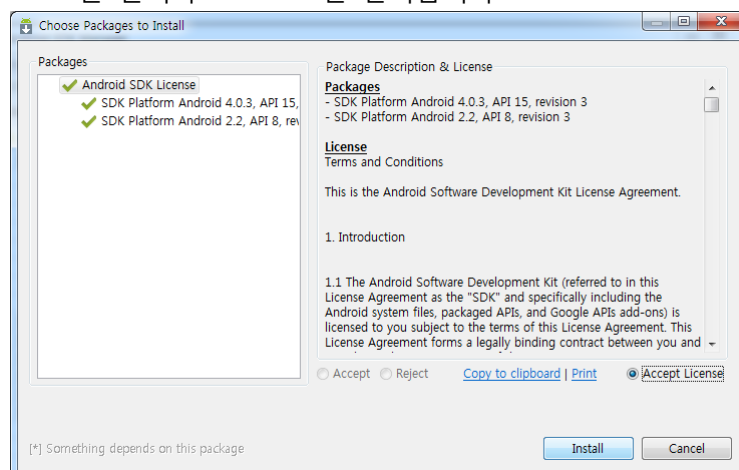
제가 제공해드린 ADT에는 Android 4.2.2 (API 17) 용 SDK만 설치되어있으므로, 타 라이브러리에 종속되어있는 Android 2.2 (API 8) SDK 와 Android 4.0.3 (API 15) SDK 를 설치하기 위해서 Window > Android SDK Manager 를 실행합니다.



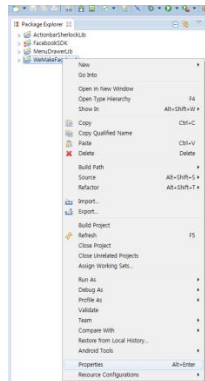
기본적으로 선택되어있는 것들이 있는데 **모두 체크를 해제**하고 아래와 같이 2개의 SDK만 선택하고 Install 2 packages를 클릭합니다. (Install 2 packages가 아니라면 잘못 선택된 것입니다.)



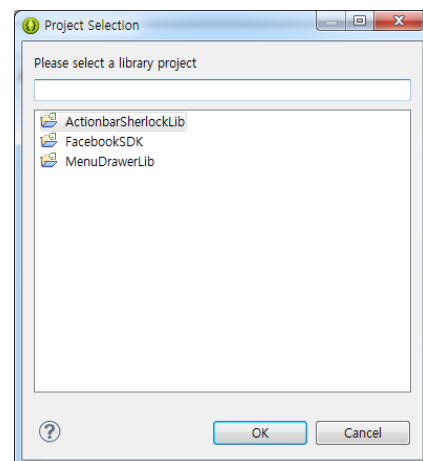
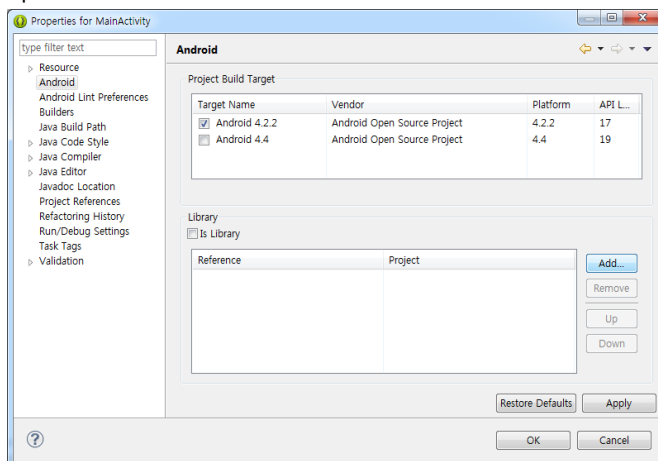
우측의 Accept License 를 클릭하고 Install 을 눌러줍니다.



지금부터는 4개의 프로젝트를 연결하여 하나의 프로젝트로 실행시킬 수 있도록 설정하는 것을 해보겠습니다. Package Explorer에서 WeMakeFacebook 프로젝트를 마우스 우 클릭해서 Properties를 실행합니다.

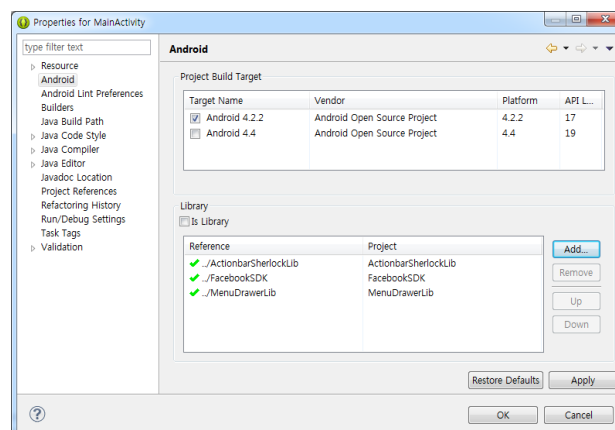


Android Tab으로 이동해서 Library 쪽에 있는 Add 버튼을 눌러서 라이브러리를 추가하겠습니다. Add를 누르면 오른쪽과 같이 Project Selection 창이 뜨는데요. 3개의 프로젝트를 하나씩 추가해줍니다.

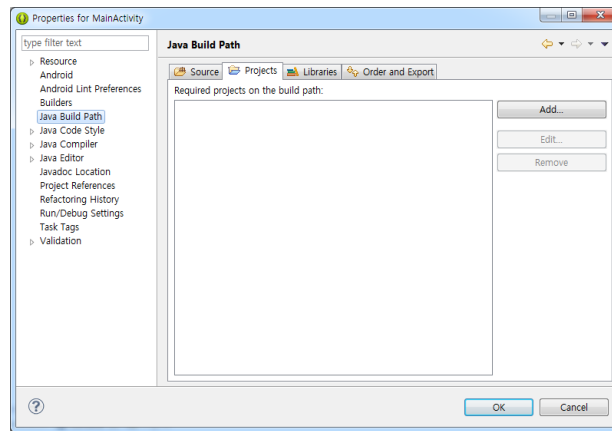


추가해주고 나면 아래와 같이 Library에 3개가 추가된 것을 확인할 수 있습니다.

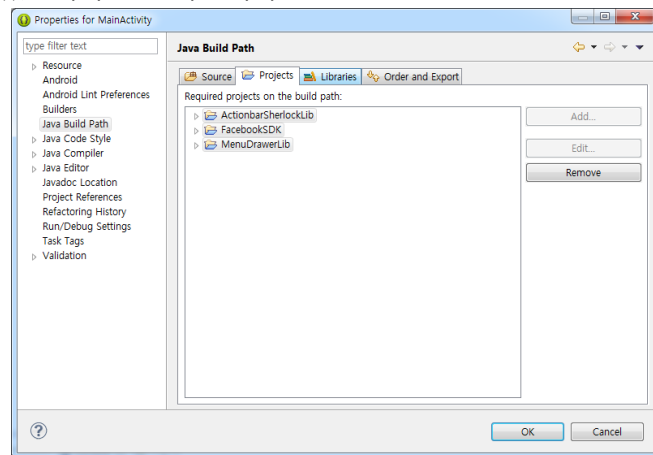
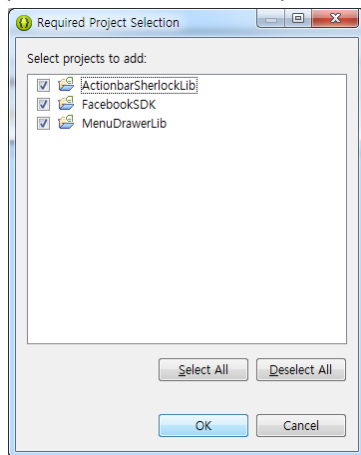
(Tip : 여기에 추가될 수 있는 프로젝트들은 Is Library 체크박스가 체크되어있는 프로젝트들입니다. 즉, MenuDrawerLib, ActionBarSherlockLib, FacebookSDK 프로젝트는 체크박스가 체크되어있을 거예요 😊)



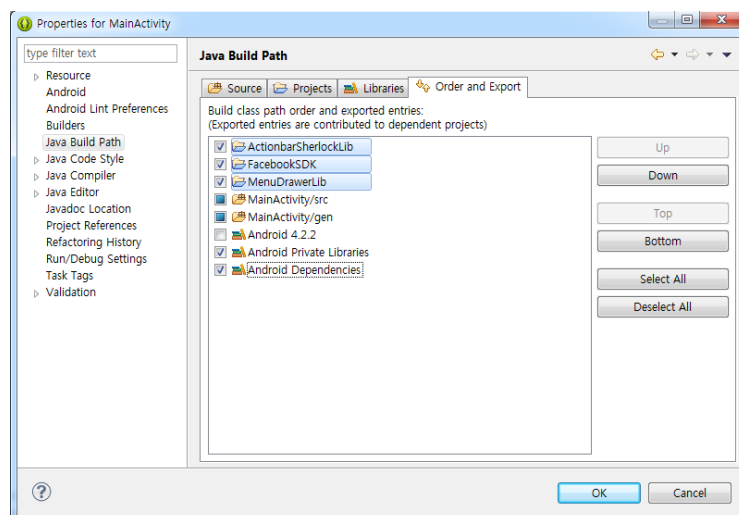
이번에는 Java Build Path 탭으로 이동하여 Projects 에 아까 그 3개의 프로젝트를 추가하도록 하겠습니다. Library로 추가되는 것 뿐만 아니라 빌드시에 함께 빌드하여, 해당 소스를 컴파일타임에 합칠 수 있도록 하기 위해서 추가하는 것입니다.



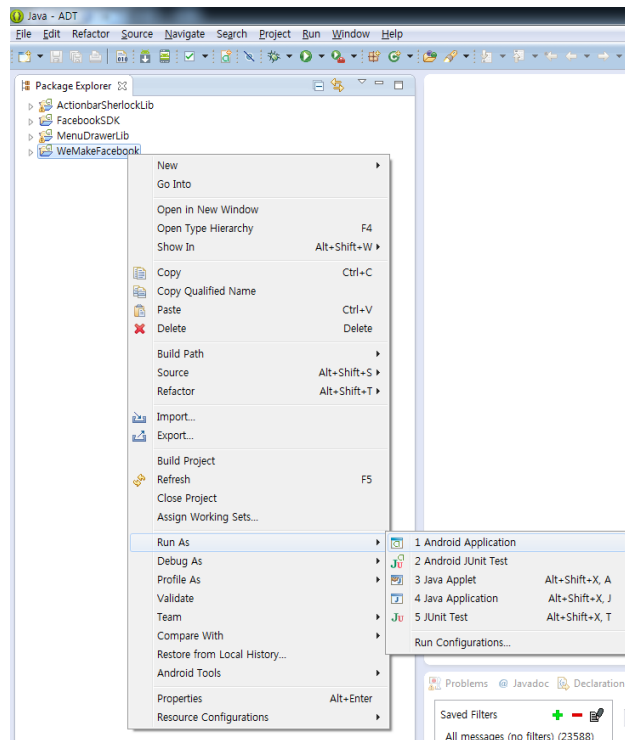
숙숙~ 참 쉽죠? 설명 안해도 할 수 있을거라고 생각합니다.



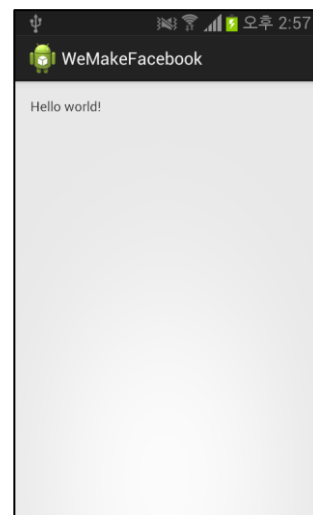
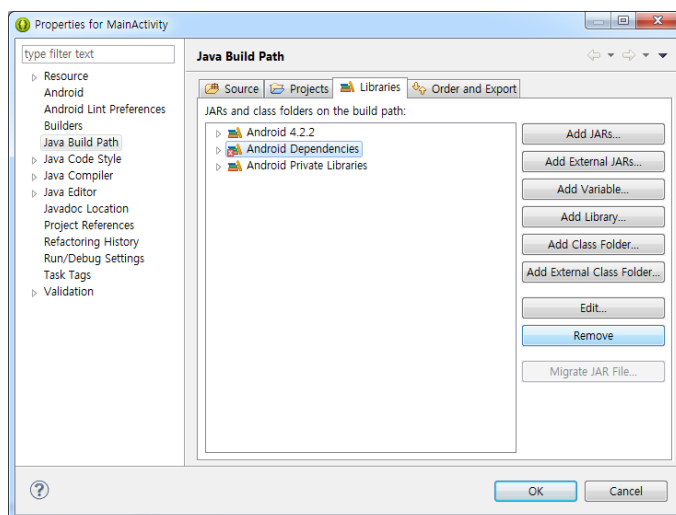
Order and Export 탭으로 이동해서, 하단에 위치한 3개의 프로젝트를 맨 위로 올려주고 체크박스를 체크해줍니다. 이렇게 함으로써 라이브러리가 빌드타임에 가장 먼저 빌드되고 사용할 수 있게 됩니다.



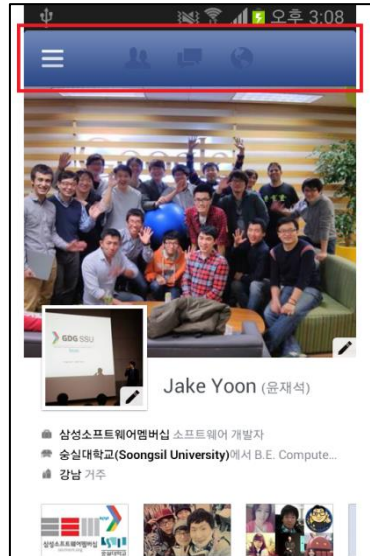
자 그럼 Package explorer에서 WeMakeFacebook 프로젝트를 우클릭하고 Run as > Android Application을 눌러서 프로젝트를 실행시킵니다. (다 된줄 알았는데 역시나 문제가 있네요 ㅎㅎ;;)



에러가 발생한다면 다시 Properties 창에 들어와서, Java Build Path > Libraries > Android Dependencies를 삭제해주고, @Project > Clean 을 한번 눌러주고 다시 실행시켜봅니다. 이제 드디어 실행이 되네요 !! 이제 본격적인 개발을 시작해봅시다.



처음으로 해볼 것은 ActionBarSherlock 이라는 라이브러리를 통해, Facebook UI 중에서 상단에 생기는 메뉴를 만들어보도록 하겠습니다.



상단에 있는 빨간 박스와 같은 메뉴를 생성하도록 하기 위해서 기존의 MainActivity를 Activity가 아닌 SherlockActivity를 상속받도록 수정하겠습니다.

```

MainActivity.java
package com.ssm.android.codelab;

import com.actionbarsherlock.app.SherlockActivity;

public class MainActivity extends SherlockActivity {

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

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}

```

SherlockActivity를 상속받도록 수정하고 나면 onCreateOptionsMenu함수에 오류가 나타는데요. 이유는 Activity의 onCreateOptionsMenu는 android.view.Menu를 함수의 인자로 받지만, SherlockActivity의 onCreateOptionsMenu는 com.actionbarsherlock.view.Menu를 함수의 인자로 받기 때문입니다.

import com.actionbarsherlock.app.SherlockActivity; 왼쪽에 +를 눌러서
import android.view.Menu;를 지우고 import com.actionbarsherlock.view.Menu;로 변경해줍니다.

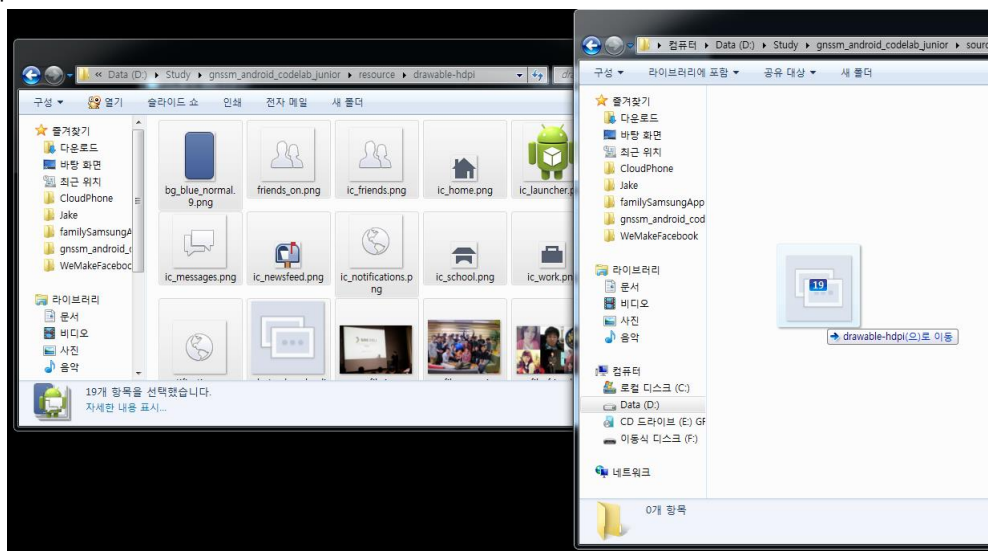
(Tip : 직접 타이핑이 귀찮다면 Ctrl + O를 눌러보세요!)

(Tip : Ctrl + Space Bar를 누르면 자동완성이 됩니다.)

onOptionsItemSelected() 함수의 getMenuInflater().inflate(R.menu.main, menu); 부분을 삭제해줍니다.

다음으로 ActionBar에 아이콘들을 추가하는 작업을 진행하도록 하겠습니다.

아이콘 리소스들을 추가하기 위해서 resource 에 있는 파일들을 /res/drawable로 복사하도록 하겠습니다.



onCreate() 함수를 아래와 같이 입력하여 줍니다.

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

    getSupportActionBar().setDisplayHomeAsUpEnabled(true); // Show MenuIcon
    getSupportActionBar().setIcon(R.drawable.ic_menu);      // Set MenuIcon like Facebook
    getSupportActionBar().setTitle("");                      // Set Title ""
}
```

주석에 쓰여있는 것처럼 MenuIcon을 보이도록 변경하고, Facebook 처럼 아이콘을 셋팅해줍니다. Title을 빈칸으로 변경합니다.

이어서, onCreateOptionsMenu() 함수에 아래와 같이 입력하여 메뉴들을 추가합니다.

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    menu.add("friends") // Add Friends Menu
        .setIcon(R.drawable.ic_friends) // Set Friends Menu Icon
        .setShowAsAction(MenuItem.SHOW_AS_ACTION_ALWAYS); // Set Friends Menu Action

    menu.add("messages") // Add Messages Menu
        .setIcon(R.drawable.ic_messages) // Set Messages Menu Icon
        .setShowAsAction(MenuItem.SHOW_AS_ACTION_ALWAYS); // Set Messages Menu Action

    menu.add("notifications") // Add Notifications Menu
        .setIcon(R.drawable.ic_notifications) // Set Notifications Menu Icon
        .setShowAsAction(MenuItem.SHOW_AS_ACTION_ALWAYS); // Set Notifications Menu Action
    return true;
}
```

크게 3개의 메뉴를 추가하기 위한 작업을 하는데요. 주석에도 쓰여있듯이 메뉴를 추가하고 메뉴의 아이콘을 설정하고 아이콘에 해당하는 Action을 정의합니다.

setShowAsAction() 함수 인자에 대해 조금 살펴보겠습니다.

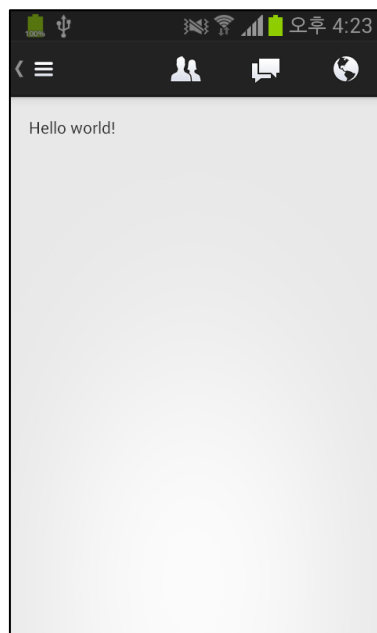
@ MenuItem.SHOW_AS_ACTION_NEVER : 해당 메뉴가 보이지 않도록 합니다.

@ MenuItem.SHOW_AS_ACTION_IF_ROOM : 해당 메뉴가 선택될 경우 새로운 액션바의 메뉴가 되도록 구현할 때 사용합니다. (ex: 검색 버튼을 누를 경우 검색어를 입력받도록 변환)

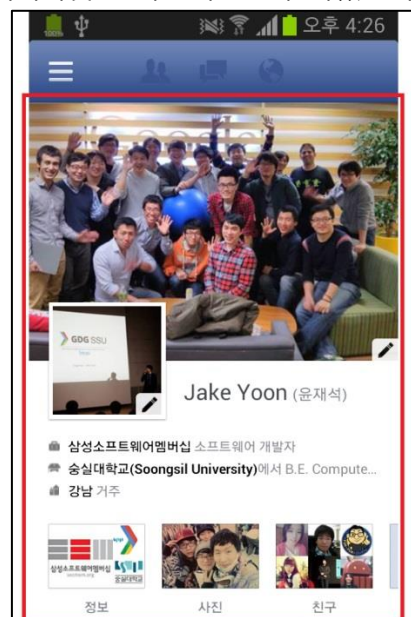
@ MenuItem.SHOW_AS_ACTION_ALWAYS : 해당 메뉴가 항상 보이도록 합니다. 너무 많은 버튼을 ALWAYS로 설정할 경우, 스크린이 작은 디바이스에서는 문제가 될 수 있다고 합니다.

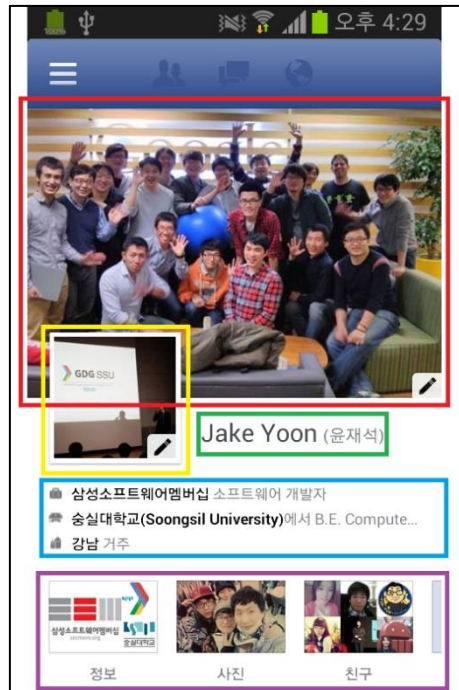
@ MenuItem.SHOW_AS_ACTION_WITH_TEXT : 메뉴 아이콘과 함께 텍스트가 보이도록 합니다.

어플리케이션을 실행해보면 조금 어설픈하지만 페이스북과 같은 상단 메뉴가 생긴 것을 확인할 수 있습니다.



이번에는 프로필 화면을 위한 레이아웃을 작성해보도록 하겠습니다.





프로필 화면 하단에 있는 뉴스피드 영역은 제외하고 보면, 프로필 화면은 크게 5가지 요소로 정의할 수 있습니다.

@ 커버사진영역 (빨간색) : 스크린 사이즈에 비해 가로는 가득 찬 크기, 세로는 절반 정도 채운 크기의 ImageView입니다.

@ 프로필사진영역 (노란색) : 스크린 화면의 세로로는 중앙에 가로로는 5% 만큼 padding을 주어 스크린 가로 사이즈의 3분의 1만큼의 크기로 커버사진을 덮어씌우고 있는 ImageView입니다.

@ 이름 영역 (초록색) : 커버사진의 아래에, 프로필사진의 우측에 존재하는 TextView입니다.

@ 정보 요약 영역 (파란색) : 프로필사진 아래에 존재하며, 가로는 가득 찬 크기로 존재하는 TextView입니다.

@ 정보 미리보기 영역 (보라색) : 정보 요약 아래에 존재하며, 가로로는 가득 찬 크기, 스크린 세로 사이즈의 4분의 1만큼의 크기의 Custom Gallery입니다.

이렇게 정리하고 보면, 위의 레이아웃은 LinearLayout과 FrameLayout을 활용하면 쉽게 디자인할 수 있습니다. LinearLayout을 활용하는 이유는 스크린의 사이즈에 비례하여 제작하기 때문이고, FrameLayout을 활용하는 이유는 커버사진영역과 프로필사진영역이 겹쳐지는 형태를 띄고 있기 때문입니다.

프로젝트의 res/layout/layout_main.xml 을 열어서 최상위의 RelativeLayout을 FrameLayout으로 고치고 속성을 아래와 같이 바꾼 뒤, TextView를 삭제합니다.

```
*layout_main.xml
<FrameLayout 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:background="#FFFFFF"
    tools:context=".MainActivity" >

</FrameLayout>
```

FrameLayout안에 android:layout_width="match_parent" android:layout_height="match_parent" android:orientation="vertical" 속성을 갖는 LinearLayout과 RelativeLayout을 생성합니다.

```
*layout_main.xml
<FrameLayout 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:background="#FFFFFF"
    tools:context=".MainActivity" >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"

    </LinearLayout>

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"

    </RelativeLayout>
</FrameLayout>
```

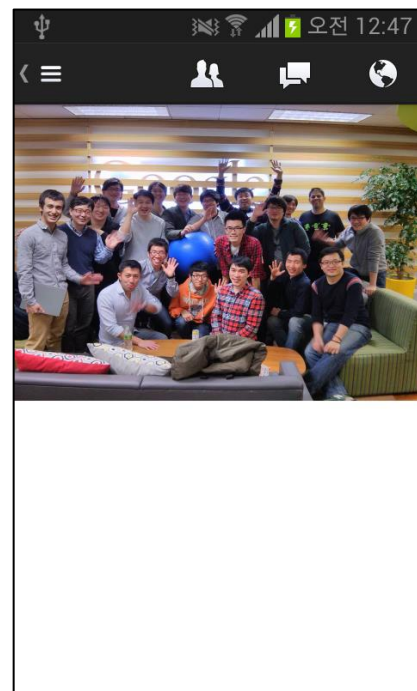
(Tip : XML 또는 Java 파일 입력 중에 자동 들여쓰기를 하고 싶다면, Ctrl + a를 눌러서 전체영역을 선택하고 Ctrl + i 를 누르면 자동 들여쓰기를 할 수 있습니다.)

첫 번째 레이아웃은 프로필사진을 제외한 나머지를 위한 영역이고, 두 번째 레이아웃은 프로필 사진을 위한 영역입니다.

커버사진을 추가하기 위한 레이아웃을 작성해보도록 하겠습니다. 커버사진영역은 스크린 사이즈에 비해 가로는 가득 찬 크기, 세로는 절반 정도 채운 크기의 ImageView이므로, LinearLayout의 layout_weight를 활용하면 쉽게 적용할 수 있습니다.

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:weightSum="2">

    <ImageView
        android:id="@+id/profile_cover"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:src="@drawable/profile_cover"
        android:scaleType="fitXY"
        android:layout_weight="1"/>
</LinearLayout>
```



다음으로 커버사진 아래에 이름 영역을 추가해보도록 하겠습니다. 이름 영역은 커버사진의 아래에, 프로필 사진의 우측에 존재하는 TextView 입니다. 따라서 프로필사진의 사이즈인 스크린의 가로사이즈 3분의 1 만큼을 띄워주고 TextView 두 개를 나란히 놓도록 구현합니다.

```
<ImageView
    android:id="@+id/profile_cover"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:scaleType="fitXY"
    android:src="@drawable/profile_cover" />

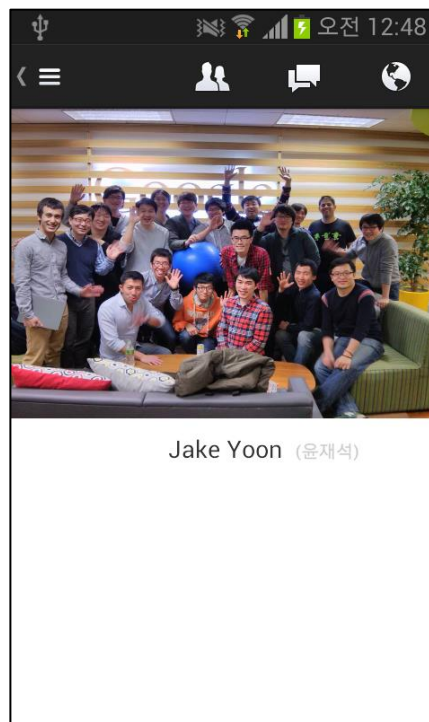
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    android:orientation="vertical" >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="right"
        android:orientation="horizontal"
        android:paddingBottom="20dp"
        android:paddingTop="10dp"
        android:weightSum="3" >

        <LinearLayout
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="2"
            android:orientation="horizontal" >

            <TextView
                android:id="@+id/profile_nickname"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginLeft="10dp"
                android:layout_marginRight="10dp"
                android:text="Jake Yoon"
                android:textSize="18dp" />



            <TextView
                android:id="@+id/profile_name"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="(주재석)"
                android:textColor="#CCCCCC"
                android:textSize="14dp" />
        </LinearLayout>
    </LinearLayout>
</LinearLayout>
```



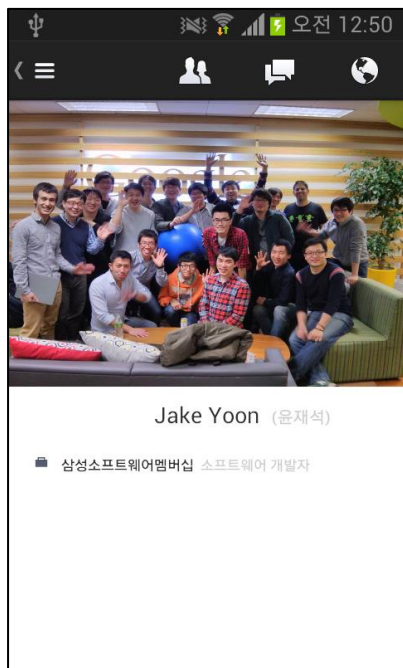
커버영역 아래에 가로 방향의 LinearLayout을 추가하고, weightSum을 3으로 주어서 화면을 3등 분하도록 하였습니다. 또한 gravity 속성을 right로 주어, 오른쪽으로 정렬되도록 속성을 주었습니다. 추가한 LinearLayout 안에 또 LinearLayout을 추가하여, layout_weight 속성을 2로 주었습니다. 그래서 3분의 2만큼 영역을 잡는 LinearLayout을 주었고, 그 레이아웃 안에 TextView를 2개 추가 하도록 구현하였습니다.

다음으로 이름 영역아래에 있는 정보 요약 영역을 추가해보도록 하겠습니다. 정보 요약 영역은 프로필사진 아래에 존재하며, 가로는 가득 찬 크기로 존재하는 TextView입니다.

정보 요약 영역은 일하는 곳, 공부하는 곳, 거주하는 곳 순서로, 3개의 요약 정보가 존재하며 동일한 패턴을 반복적으로 사용하고 있습니다. 일하는 곳 하나의 레이아웃을 집중하여 구현해보면, 크게 아래와 같은 세가지가 필요합니다.

아이콘, 기본정보, 상세정보  삼성소프트웨어멤버십  소프트웨어 개발자

이 세가지를 아래와 같은 레이아웃으로 구현합니다.



```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:paddingLeft="20dp"
    android:paddingRight="20dp" >

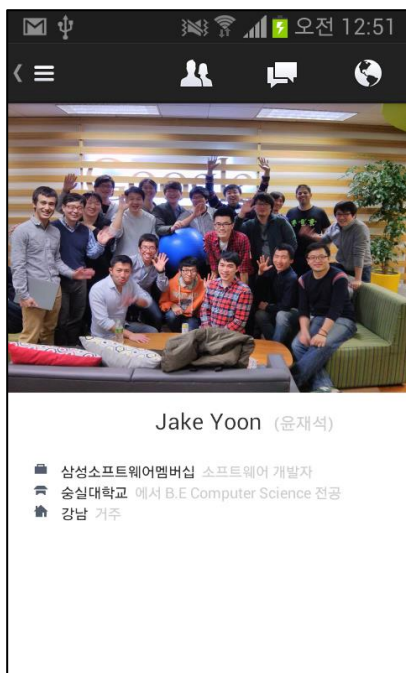
    <ImageView
        android:layout_width="12dp"
        android:layout_height="12dp"
        android:layout_marginRight="10dp"
        android:scaleType="fitXY"
        android:src="@drawable/ic_work" />

    <TextView
        android:id="@+id/profile_work"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginRight="5dp"
        android:singleLine="true"
        android:text="삼성소프트웨어멤버십"
        android:textSize="12dp"
        android:textStyle="bold" />

    <TextView
        android:id="@+id/profile_work_position"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:singleLine="true"
        android:text="소프트웨어 개발자"
        android:textColor="#CCCCCC"
        android:textSize="12dp" />
</LinearLayout>
```

위와 같은 방법으로 3번을 반복하면 아래와 같은 화면을 완성할 수 있습니다.

각 정보 요약을 나타내는 뷰의 id 값은 아래와 같이 설정해줍니다.



profile_work	: 직장 정보
profile_work_position	: 직책 정보
profile_school	: 학교 정보
profile_school_dept	: 학과 정보
profile_home	: 거주지역 정보
profile_home_state	: 거주상태 정보

이제 프로필 화면의 마지막 부분인 정보 미리보기 영역을 구현해보도록 하겠습니다.

정보 미리보기 영역은 정보 요약 아래에 존재하며, 가로로는 가득 찬 크기, 스크린 세로 사이즈의 4분의 1만큼의 크기의 Custom Gallery입니다. Custom Gallery가 이미 구현되어있다고 가정하고 Main화면에 위치시키도록 하겠습니다.

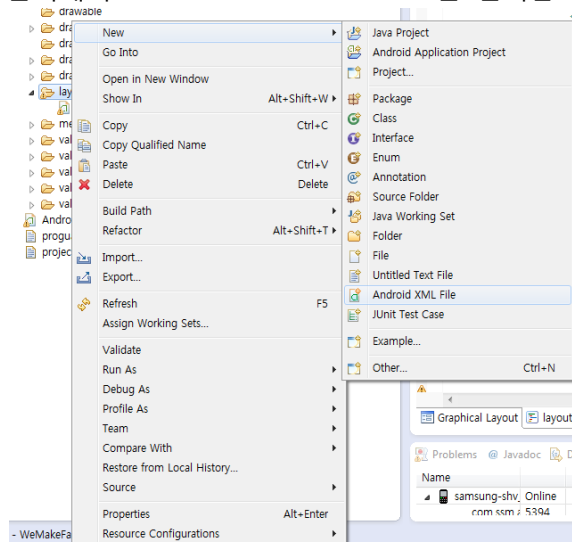
```
<TextView
    android:id="@+id/profile_home_state"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:singleLine="true"
    android:text="> <"
    android:textColor="#CCCCCC"
    android:textSize="12dp" />

</LinearLayout>

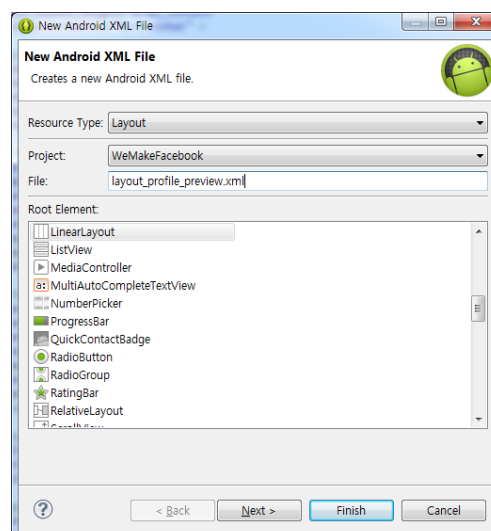
<Gallery
    android:id="@+id/profile_preview_gallery"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:spacing="2dp" />

</LinearLayout>
</LinearLayout>
```

Custom Gallery를 구현하기 위해서는 각 아이템 하나에 해당하는 레이아웃을 설계해야 합니다. 따라서, res / layout 에 layout_profile_preview.xml 을 추가하도록 하겠습니다. Package Explorer 에서 res / layout 폴더를 우클릭해서 new > Android XML File 을 선택합니다.



New Android XML File 창이 뜨면, File 명을 layout_profile_preview.xml 을 입력하고 Finish를 누릅니다.



layout_profile_preview.xml 파일을 열어서, 아래와 같이 입력합니다.

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="100dp"
    android:layout_height="80dp"
    android:orientation="vertical" >

    <ImageView
        android:id="@+id/profile_preview_image"
        android:layout_width="80dp"
        android:layout_height="55dp"
        android:layout_marginBottom="5dp"
        android:layout_marginLeft="10dp"
        android:layout_marginRight="10dp"
        android:layout_marginTop="5dp"
        android:background="#CCCCCC"
        android:padding="1dp"
        android:scaleType="fitXY"
        android:src="@drawable/photo_downloading" />

    <TextView
        android:id="@+id/profile_preview_text"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center_horizontal"
        android:text="???"
        android:textColor="#CCCCCC"
        android:textSize="12dp" />

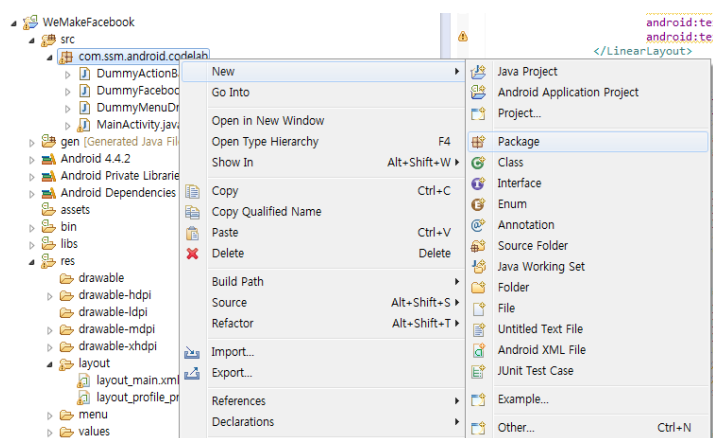
</LinearLayout>

```

각 하나의 정보 미리보기를 나타내는 아이템은 ImageView 하나와 TextView로 이루어져 있습니다.

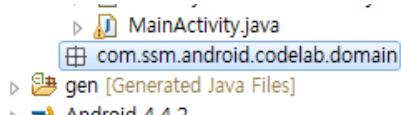
지금부터는 Custom Gallery를 만들기 위해 오랜만에 Java 코딩을 하도록 하겠습니다. 조금 어려울 수 있으므로 집중하고 따라오세요 ☺

Gallery의 한 Item의 정보를 저장하기 위한 클래스를 하나 생성하기 전에, Domain 정보들을 담고 있는 Package를 생성하도록 하겠습니다. Package Explorer에서 src > com.ssm.android.codelab을 우 클릭해서 new > Package를 눌러서 패키지를 생성합니다.

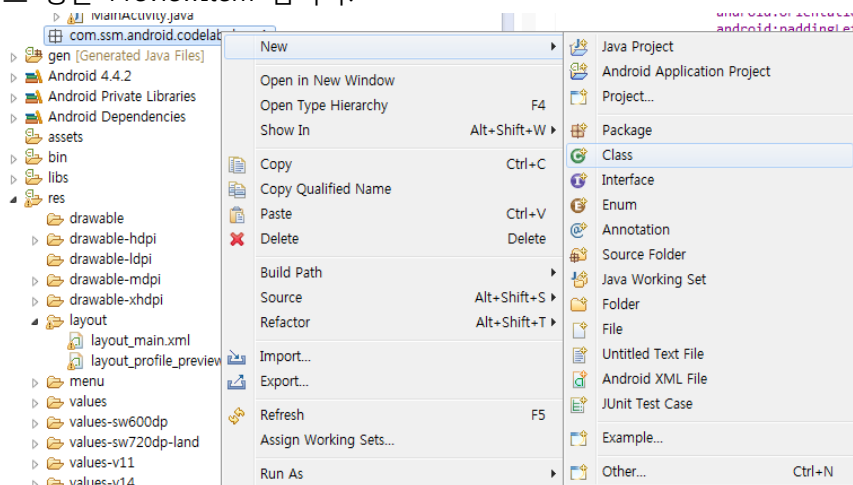


Package 명을 com.ssm.android.codelab.domain로 입력해줍니다.

(Tip : 패키지는 사실 생성하지 않아도 상관없습니다만, 용도에 따라, 기능에 따라 패키지를 구분하는 습관을 가지면, 후에 프로젝트를 관리할 때도 용의합니다.)

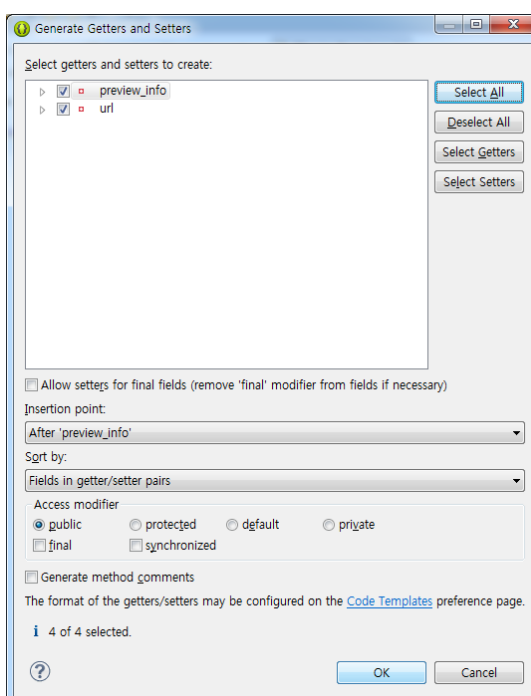
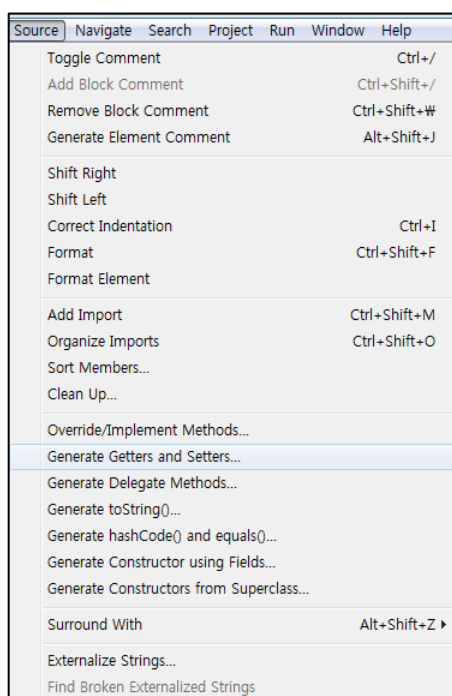


새롭게 생성한 패키지에 Gallery Preview Item의 정보를 저장하기 위한 클래스를 생성하도록 하겠습니다. 클래스명은 PreviewItem 입니다.



Item이 갖는 속성은 뿌려질 이미지의 url정보와 미리보기하는 정보를 표현하는 문자열 이므로 private 형태로 String url, String preview_info 를 추가합니다. 그리고 url과 preview_info를 외부에서 접근하기 위한 getter / setter 함수가 필요한데, 이클립스에서는 getter / setter 함수를 자동으로 생성할 수 있습니다.

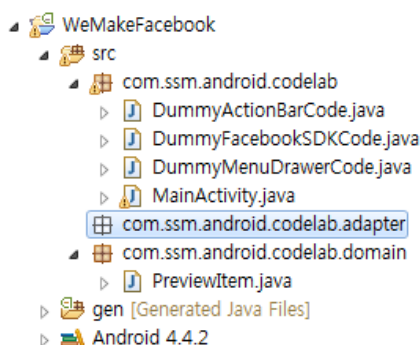
@ Source > Generate Getters and Setters를 선택합니다. Generate Getters and Setters 창에서 Select All을 선택하고 OK를 누릅니다.



getUrl(), setUrl(), getPreview_info(), setPreview_info() 함수가 자동으로 생성된 것을 확인할 수 있습니다. 또, PreviewItem(String url, String preview_info) 형태의 생성자 함수를 추가합니다.

지금부터는 Gallery를 위한 Adapter를 생성하는 작업을 하도록 하겠습니다. Adapter란 ListView, Gallery와 같이 동적으로 생성될 수 있는 View에 Data와 UI를 연결해주는 것으로, Adapter하나를 만들면 ListView에 붙이면 List형태로 Gallery에 붙이면 Gallery형태로 표현할 수 있습니다.

Adapter들을 가지고 있는 패키지를 하나 생성하겠습니다. Package Explorer에서 `src > com.ssm.android.codelab` 을 우 클릭해서 `new > Package`를 눌러서 `adapter`라는 패키지를 하나 생성합니다.



`adapter` 패키지에 `PreviewAdapter` 클래스를 만듭니다. 대부분의 Adapter는 `ArrayAdapter`를 상속받도록 구현하면, 2개의 함수만 오버라이드하면 되므로 간편하게 제작할 수 있습니다.

(Tip : 오버라이드란, 상속받은 클래스의 부모에게 있는 함수를 다른 방식으로 사용할 수 있도록 재정의하는 것을 말합니다. `abstract`와 같은 키워드를 가진 함수의 경우 무조건 적으로 재정의의를 해야하지만, 그렇지 않은 함수들은 재정의하지 않을 수 있습니다. 재정의하지 않을 경우 부모에서 구현된 내용을 그대로 따르게 됩니다.)

`ArrayAdapter<PreviewItem>`을 상속받고, 생성자 함수와, `getView` 함수를 재정의하여 아래와 같이 구현합니다.

```
package com.ssm.android.codelab.adapter;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.TextView;

import com.ssm.android.codelab.domain.PreviewItem;

public class PreviewAdapter extends ArrayAdapter<PreviewItem> {
    private LayoutInflater inflater = null;

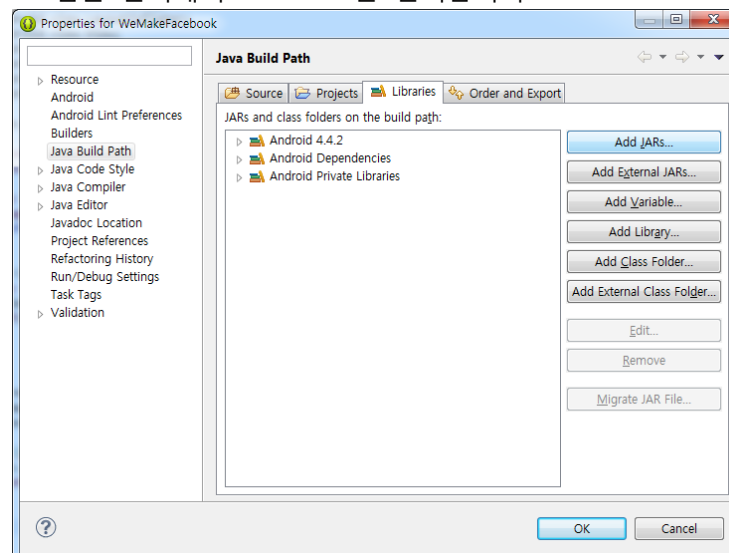
    public PreviewAdapter(Context context, int textViewResourceId) {
        super(context, textViewResourceId);
        inflater = LayoutInflater.from(context);
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        View v = convertView;
        if (null == v) {
            v = inflater.inflate(R.layout.layout_profile_preview,
                parent,
                false);
        }
        PreviewItem preview = getItem(position);
        ImageView iv = (ImageView)v.findViewById(R.id.profile_preview_image);
        TextView tv = (TextView)v.findViewById(R.id.profile_preview_text);

        //iv에 이미지를 넣어줌
        tv.setText(preview.getPreview_info());
        return v;
    }
}
```

빨간 박스 부분은 ImageView에 이미지를 바인딩 하는 곳으로, HTTP를 통해 URL에 해당하는 이미지를 다운로드 하는 오픈소스 라이브러리인 UniversalImageLoader를 활용하도록 하겠습니다. UniversalImageLoader를 프로젝트에 추가하기 위해서 library 폴더에 있는 universal-image-loader-1.7.0.jar 파일을 프로젝트의 libs 폴더로 복사합니다.

다시 이클립스로 돌아와서 Package Explorer에서 WeMakeFacebook 프로젝트를 클릭하고 F5버튼을 눌러서 새로고침 합니다. WeMakeFacebook 프로젝트를 오른쪽 클릭해서 Properties로 들어갑니다. Java Build Path 탭을 선택해서 Add JARs를 선택합니다.



WeMakeFacebook 프로젝트 libs 폴더에 있는 universal-image-loader-1.7.0.jar를 추가하면 프로젝트에서 UniversalImageLoader 라이브러리를 활용할 수 있게 됩니다. UniversalImageLoader를 사용하기 위한 옵션들을 설정해주어야 합니다. MainActivity 클래스의 onCreate() 함수에 아래와 같은 내용을 추가합니다.

```
public class MainActivity extends SherlockActivity {

    public static ImageLoader imageLoader;
    public static DisplayImageOptions options;

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

        // ImageLoader
        imageLoader = ImageLoader.getInstance();
        imageLoader.init(ImageLoaderConfiguration.createDefault(this));

        // Options
        options = new DisplayImageOptions.Builder()
            .showImageForEmptyUri(R.drawable.photo_downloading) // URL에 이미지가 없을 경우 뿌려지는 이미지
            .showStubImage(R.drawable.photo_downloading) // 로딩 중에 뿌려지는 이미지
            .cacheInMemory() // 메모리 캐시 사용
            .bitmapConfig(Bitmap.Config.RGB_565) // bitmap
            .build();
    }
}
```

DisplayImageOptions의 옵션들을 하나씩 확인해보자.

- @ showImageForEmptyUri : URL에 이미지가 없을 경우 뿌려지는 이미지
- @ showStubImage : 로딩 중에 뿌려지는 이미지
- @ cacheInMemory : 메모리 캐시 사용
- @ cacheOnDisc : 디스크 캐시 사용
- @ bitmapConfig : 뿌려질 이미지의 속성

위와 같이 설정하고 나면 우리는

MainActivity.imageLoader.displayImage(Image URL, ImageView, MainActivity.options);

이렇게 1줄로 원격지에 있는 이미지를 Http를 통해 ImageView에 표현할 수 있습니다.

아까 전에 작성하던 PreviewAdapter.java 파일을 열어서 getView함수의 주석부분을 아래와 같이 수정해줍니다.

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {
    View v = convertView;
    if(null == v) {
        v = inflater.inflate(R.layout.layout_profile_preview,
            parent,
            false);
    }
    PreviewItem preview = getItem(position);
    ImageView iv = (ImageView)v.findViewById(R.id.profile_preview_image);
    TextView tv = (TextView)v.findViewById(R.id.profile_preview_text);

    MainActivity.imageLoader.displayImage( preview.getUrl(), iv, MainActivity.options);
    tv.setText(preview.getPreview_info());
    return v;
}
```

MainActivity의 View들을 연결시킬 멤버변수를 선언하고, findViewById() 함수를 통해 XML에 선언된 뷰와 Java의 클래스를 연결시키는 작업을 합니다.

선언 =>

```
public class MainActivity extends SherlockActivity {

    public static ImageLoader imageLoader;
    public static DisplayImageOptions options;

    private ImageView profile_cover;           // 커버 영역
    private ImageView profile;                 // 프로필 사진
    private TextView profile_nickname;         // 닉네임
    private TextView profile_name;             // 이름
    private TextView profile_work;             // 직장
    private TextView profile_work_position;     // 직책
    private TextView profile_school;           // 학교
    private TextView profile_school_dept;       // 학과
    private TextView profile_home;             // 거주지
    private TextView profile_home_state;       // 거주상태
    private Gallery profile_preview_gallery;    // 정보 미리보기
```

연결 =>

```
profile_cover = (ImageView)findViewById(R.id.profile_cover);
profile = (ImageView)findViewById(R.id.profile);
profile_nickname = (TextView)findViewById(R.id.profile_nickname);
profile_name = (TextView)findViewById(R.id.profile_name);
profile_work = (TextView)findViewById(R.id.profile_work);
profile_work_position = (TextView)findViewById(R.id.profile_work_position);
profile_school = (TextView)findViewById(R.id.profile_school);
profile_school_dept = (TextView)findViewById(R.id.profile_school_dept);
profile_home = (TextView)findViewById(R.id.profile_home);
profile_home_state = (TextView)findViewById(R.id.profile_home_state);
profile_preview_gallery = (Gallery)findViewById(R.id.profile_preview_gallery);
```

MainActivity에 PreviewAdapter와 Adapter에서 사용할 더미데이터를 추가합니다.

(현재는 Facebook SDK를 연동하는 작업 없이 UI 작업만 진행하고 있으므로, 일단은, 제 개인서버에 올려놓은 이미지 데이터를 다운로드 받도록 구현하겠습니다.)

```
MainActivity.imageLoader.displayImage("http://yjaeseok.cafe24.com:8080/facebook/profile.jpg",
    profile, MainActivity.options);
MainActivity.imageLoader.displayImage("http://yjaeseok.cafe24.com:8080/facebook/profile_cover.jpg",
    profile_cover, MainActivity.options);

adapter = new PreviewAdapter(this, 0);
adapter.add(new PreviewItem("http://yjaeseok.cafe24.com:8080/facebook/profile_info.png", "정보"));
adapter.add(new PreviewItem("http://yjaeseok.cafe24.com:8080/facebook/profile_photo.png", "사진"));
adapter.add(new PreviewItem("http://yjaeseok.cafe24.com:8080/facebook/profile_friends.png", "친구"));
profile_preview_gallery.setAdapter(adapter);
profile_preview_gallery.setSelection(1);
```

프로필 화면의 마지막으로, 프로필 사진을 레이아웃에 추가하도록 하겠습니다. layout_main.xml 을 열어서 가장 마지막 하단에 RelativeLayout과 ImageView를 아래와 같이 추가합니다.

```

        </LinearLayout>
    </LinearLayout>

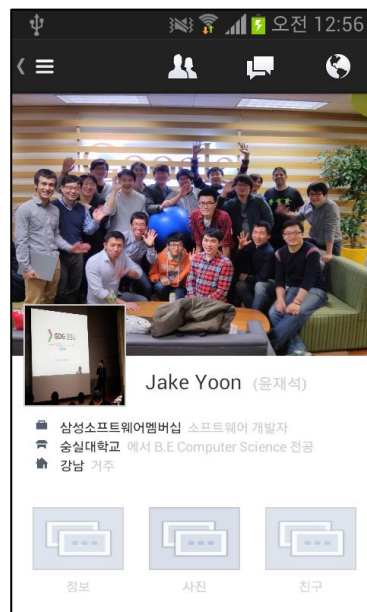
    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent" >

        <ImageView
            android:id="@+id/profile"
            android:layout_width="90dp"
            android:layout_height="90dp"
            android:background="#CCCCCC"
            android:padding="1dp"
            android:layout_alignParentLeft="true"
            android:layout_centerVertical="true"
            android:layout_marginLeft="10dp"
            android:scaleType="fitXY"
            android:src="@drawable/profile" />
    </RelativeLayout>

</FrameLayout>

```

짹짹!! 드디어 프로필 화면이 완성되었습니다. 실행해보면 아래와 같은 화면을 확인할 수 있습니다. !!!!! ㅎㅎㅎㅎ



그러나 이상하게도 이미지를 서버로부터 다운로드 받지 않는 것 같군요. AndroidManifest.xml 파일을 열어서 Internet 권한을 추가해줍니다.

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.ssm.android.codelab"
    android:versionCode="1"
    android:versionName="1.0" >

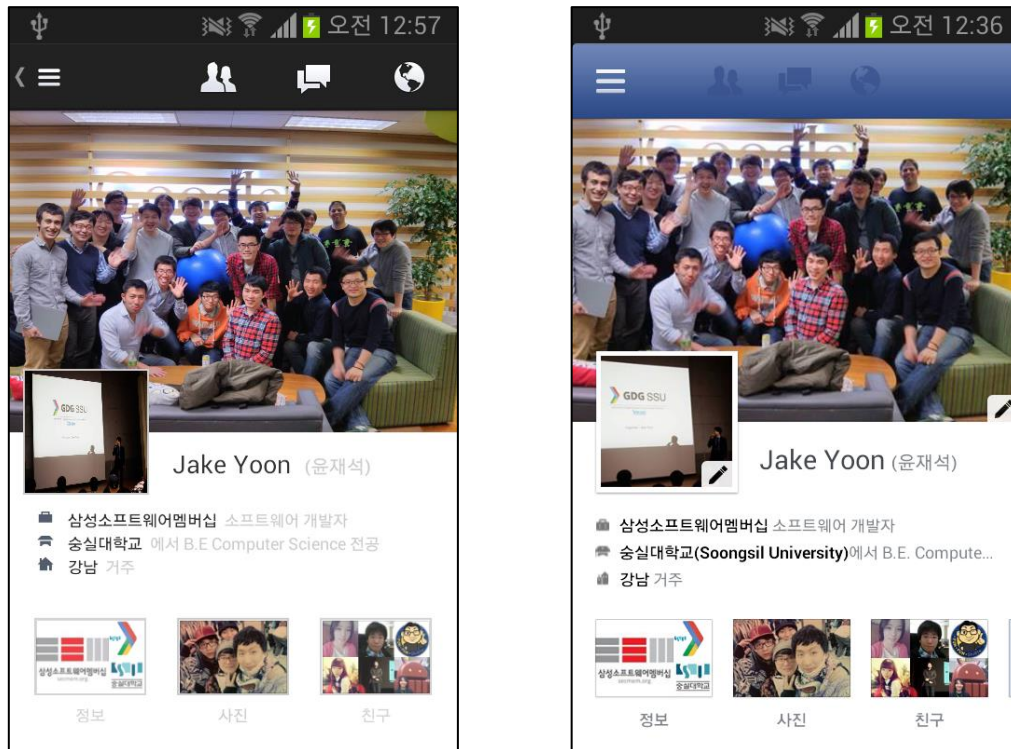
    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="18" />
    <uses-permission android:name="android.permission.INTERNET"/>

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/Theme.SherLock.Light.DarkActionBar" >
        <activity
            android:name="com.ssm.android.codelab.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

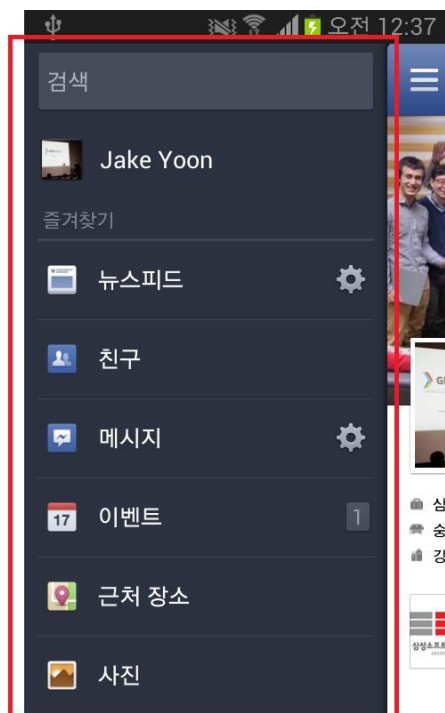
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>

```

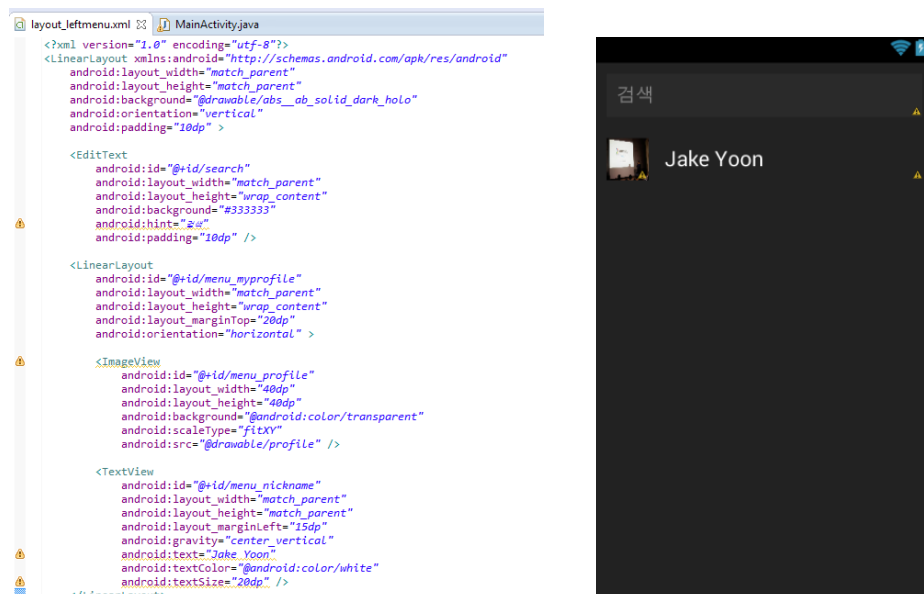
퍼미션 추가 후 다시 실행해보면 다음과 같이 이미지를 다운로드 받아오는 것을 확인할 수 있습니다. 조금은 어색하지만, Facebook의 프로필화면과 유사한 UI를 완성했습니다.



어때요 좀 비슷한가요? 이어서 MenuDrawer 라이브러리를 활용해서 아래와 같은 서랍메뉴 UI를 구현해보도록 하겠습니다.

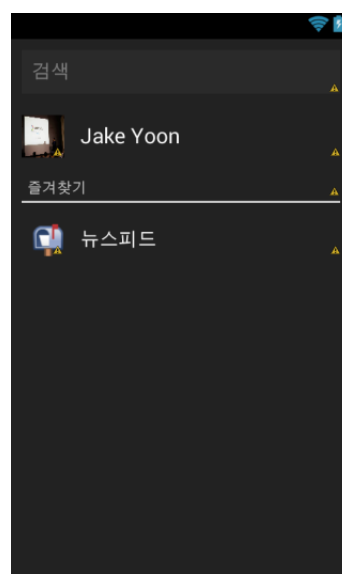


서랍메뉴 레이아웃을 구성하기 위한 layout_leftmenu.xml 파일을 생성하고 아래와 같이 작성합니다.



위의 그림과 같이 검색을 위한 검색창과 프로필 사진, 닉네임을 보여주기 위한 ImageView와 TextView를 추가하였습니다. 하단에 Facebook의 즐겨찾기 메뉴 중 뉴스피드 메뉴만 추가하도록 하겠습니다.

```
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/search_hint"
    android:textColor="@color/white"
    android:layout_marginTop="20dp"
    android:layout_marginLeft="5dp"
    android:textSize="14dp" />
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="2dp"
    android:layout_marginTop="3dp"
    android:background="@color/white" />
<LinearLayout
    android:id="@+id/menu_newsfeed"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dp"
    android:orientation="horizontal" >
    <ImageView
        android:layout_width="30dp"
        android:layout_height="30dp"
        android:layout_marginLeft="10dp"
        android:background="@android:color/transparent"
        android:scaleType="fitXY"
        android:src="@drawable/ic_newsfeed" />
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginLeft="15dp"
        android:gravity="center_vertical"
        android:text="@string/newsfeed"
        android:textColor="@android:color/white"
        android:textSize="18dp" />
</LinearLayout>
</LinearLayout>
```



이제 MenuDrawer를 사용하기 위한 레이아웃 작성을 마무리하였습니다. 지금부터는 레이아웃을 MainActivity에 적용해보도록 하겠습니다. MainActivity에 MenuDrawer를 적용하기 위해선 다음과 같은 2개의 멤버변수가 필요합니다.

```
private static final String STATE_MENUDRAWER = "MenuDrawer"; // 메뉴상태를 저장하기 위한 Key값
private MenuDrawer mMenuDrawer; // 메뉴 객체
```

STATE_MENUDRAWER 변수는 onSaveInstanceState() 함수에서 메뉴의 현재상태를 저장하거나 onRestoreInstanceState() 함수에서 메뉴의 현재상태를 불러오기 위해 Key값으로 사용하기 위해 선언하였습니다.

아래와 같이 onSaveInstanceState() 함수와 onRestoreInstanceState() 함수를 재정의해서, 메뉴의 현재상태를 저장하도록 수정합니다. (이렇게 해주면, 화면 이동 후에도 메뉴의 상태가 유지됩니다.)

```
@Override
protected void onRestoreInstanceState(Bundle savedInstanceState) {
    super.onRestoreInstanceState(savedInstanceState);
    mMenuDrawer.restoreState(savedInstanceState.getParcelable(STATE_MENUDRAWER));
}

@Override
protected void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);
    outState.putParcelable(STATE_MENUDRAWER, mMenuDrawer.saveState());
}
```

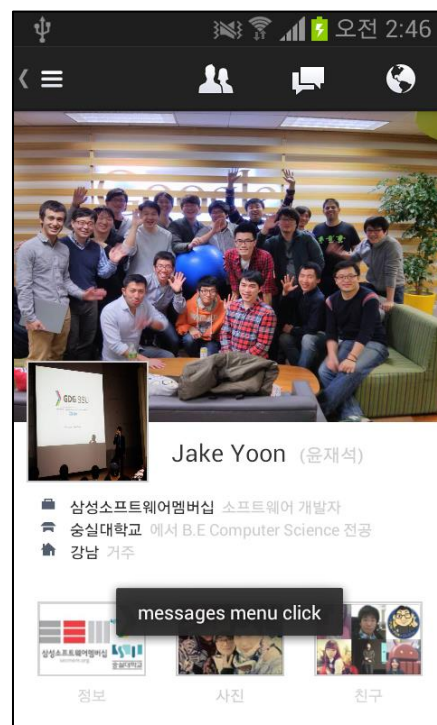
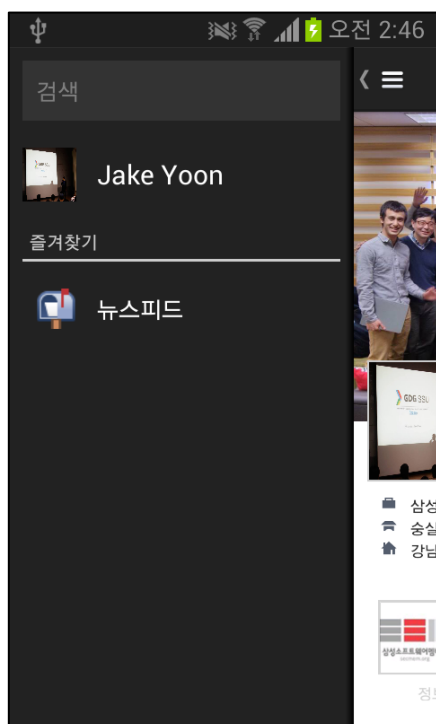
ActionBar에 있는 메뉴가 클릭될 때 콜백을 받기 위해서 아래와 같이 onOptionsItemSelected() 함수를 재정의합니다.

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch(item.getItemId()) {
        case android.R.id.home: // 메뉴 버튼이 눌리면
            mMenuDrawer.toggleMenu(); // 메뉴 상태를 토글시킨다
            break;

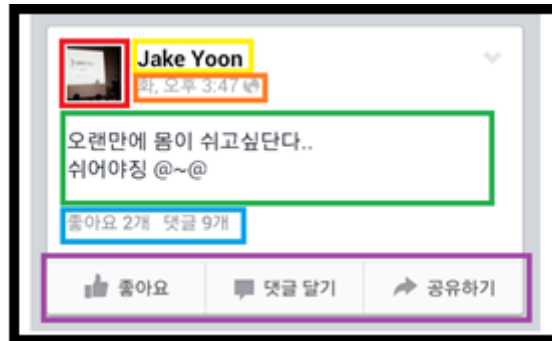
        if(item.getTitle().equals("friends")) { // 친구 메뉴가 눌리면
            Toast.makeText(this, "friends menu click", Toast.LENGTH_SHORT).show();
        } else if (item.getTitle().equals("messages")) { // 메시지 메뉴가 눌리면
            Toast.makeText(this, "messages menu click", Toast.LENGTH_SHORT).show();
        } else if (item.getTitle().equals("notifications")) { // 알림 메뉴가 눌리면
            Toast.makeText(this, "notifications menu click", Toast.LENGTH_SHORT).show();
        }

        return super.onOptionsItemSelected(item);
    }
}
```

실행해보면 다음과 같은 화면을 확인할 수 있습니다.



지금부터는 뉴스피드 화면을 구현해보도록 하겠습니다. 뉴스피드 하나의 아이템이 어떻게 구성되는지 쪼개서 보도록 하겠습니다.



뉴스피드의 한 아이템은 다음과 같이 구성됩니다.

@ 배경영역 (검은색) : 패딩을 가지고 있는 LinearLayout입니다.

@ 프로필사진영역 (빨간색) : 글쓴이의 프로필 사진이 표현되는 ImageView입니다.

@ 닉네임영역 (노란색) : 프로필 사진 우측 상단에 위치하며 글쓴이의 닉네임이 표현되는 TextView입니다.

@ 게시날짜 영역 (주황색) : 글을 쓴 날짜가 표현되며, 프로필 사진 우측에, 닉네임 하단에 표현되는 TextView입니다.

@ 콘텐츠 영역 (초록색) : 글 내용이 담기는 부분으로 프로필 사진 아래에 표현되는 TextView 입니다.

@ 좋아요/댓글 (파란색) : 글에 대한 좋아요/댓글 수를 나타내는 것으로 콘텐츠 영역 아래에 표현되는 TextView입니다.

@ 좋아요/댓글/공유하기 버튼 : 글에 대해서 좋아요/댓글/공유하기 액션을 할 수 있도록 해주는 버튼으로 하단에 가로로 존재합니다.

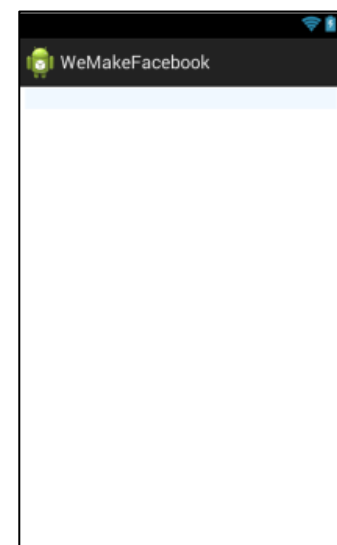
뉴스피드의 아이템을 작성하기 위해 layout_newsfeed_item.xml 을 만들고 배경영역을 위해 아래와 같이 레이아웃을 작성합니다.

```
<?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="@android:color/white"
    android:orientation="vertical"
    android:paddingBottom="2dp"
    android:paddingLeft="5dp"
    android:paddingRight="5dp"
    android:paddingTop="2dp" >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="#F0F8FF"
        android:padding="10dp" >

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:background="@android:color/white"
            android:orientation="vertical" >

            </LinearLayout>
        </LinearLayout>
    </LinearLayout>
```



3단의 LinearLayout을 활용했는데, 첫 번째 LinearLayout은 가장 바깥의 테두리영역(하얀색)을 위

해서, 두 번째 LinearLayout은 그 안을 감싸는 테두리영역(하늘색)을 위해서, 세 번째 LinearLayout은 콘텐츠(글쓴이 정보, 글 정보, 글 내용, 버튼)를 담기 위해 사용할 것입니다.

세 번째 LinearLayout에 1개의 RelativeLayout과 1개의 LinearLayout을 추가합니다. RelativeLayout은 버튼을 제외한 영역을 위해서, LinearLayout은 버튼을 위한 영역으로 활용할 것입니다.

(Tip: RelativeLayout은 이름에도 쓰여있듯이 상대적으로 위치시키기에 유용한 Layout입니다.)

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#F0F8FF"
    android:padding="10dp" >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@android:color/white"
        android:orientation="vertical" >

        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:padding="5dp" >

        </RelativeLayout>

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal" >

        </LinearLayout>
    </LinearLayout>
</LinearLayout>
```

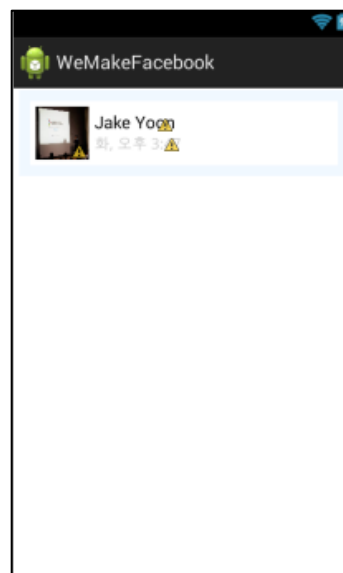
RelativeLayout에 글쓴이 사진을 위한 ImageView를 좌측 상단에 위치시키고, 글 작성자 닉네임과 글 작성 시간을 위한 LinearLayout을 ImageView 우측에 위치시킵니다. LinearLayout에는 TextView 2개를 추가하여 RelativeLayout의 내용을 완성시킵니다. 완성시킨 코드는 아래와 같습니다.

```
<ImageView
    android:id="@+id/author_icon"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="true"
    android:scaleType="fitXY"
    android:src="@drawable/profile" />

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="50dp"
    android:layout_marginLeft="5dp"
    android:layout_toRightOf="@id/author_icon"
    android:gravity="center_vertical"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/author_nickname"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="2dp"
        android:text="Jake Yoon"
        android:textColor="@android:color/black"
        android:textSize="16dp" />

    <TextView
        android:id="@+id/author_date"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="3:47"
        android:textColor="#CCCCCC"
        android:textSize="14dp" />
</LinearLayout>
```



이어서, 글 콘텐츠를 표현할 `TextView` 와 좋아요 / 댓글을 표현할 `TextView` 를 추가해보도록 하겠습니다.

```
<TextView
    android:id="@+id/author_date"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="화, 오후 3:47"
    android:textColor="#CCCCCC"
    android:textSize="14dp" />

</LinearLayout>

<TextView
    android:id="@+id/author_content"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/author_icon"
    android:minHeight="30dp"
    android:paddingBottom="8dp"
    android:paddingTop="12dp"
    android:singleLine="false"
    android:text="오랜만에 몸이 쉬고싶단다... \n쉬어야지 @~@"
    android:textColor="@android:color/black"
    android:textSize="14dp" />

<TextView
    android:id="@+id/like_and_reply"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@id/author_content"
    android:text="좋아요 2개 댓글 9개"
    android:textColor="#CCCCCC"
    android:textSize="12dp" />

</RelativeLayout>
```



상단에 있는 `RelativeLayout` 을 위한 작업은 모두 끝났습니다. 조금 더 다듬기 위해서 `RelativeLayout` 의 padding 값을 10 으로 조정하였습니다.

이번에는 `RelativeLayout` 아래에 있는 `LinearLayout` 에 버튼 3 개를 추가하도록 하겠습니다.

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#708090"
    android:orientation="horizontal" >

    <Button
        android:id="@+id/btn_like"
        android:layout_width="0dp"
        android:layout_height="25dp"
        android:layout_margin="1dp"
        android:layout_weight="1"
        android:background="#F5F5F5"
        android:drawableLeft="@drawable/ic_like"
        android:paddingLeft="10dp"
        android:paddingRight="8dp"
        android:text="좋아요"
        android:textColor="#778899"
        android:textSize="11dp" />

    <Button
        android:id="@+id/btn_reply"
        android:layout_width="0dp"
        android:layout_height="25dp"
        android:layout_margin="1dp"
        android:layout_weight="1"
        android:background="#F5F5F5"
        android:drawableLeft="@drawable/ic_reply"
        android:paddingLeft="10dp"
        android:paddingRight="8dp"
        android:text="댓글 달기"
        android:textColor="#778899"
        android:textSize="11dp" />

    <Button
        android:id="@+id/btn_share"
        android:layout_width="0dp"
        android:layout_height="25dp"
        android:layout_margin="1dp"
        android:layout_weight="1"
        android:background="#F5F5F5"
        android:drawableLeft="@drawable/ic_share"
        android:paddingLeft="10dp"
        android:paddingRight="8dp"
        android:text="공유하기"
        android:textColor="#778899"
        android:textSize="11dp" />

</LinearLayout>
```



어때요? 좀 비슷하나요?

지금부터는 뉴스피드 화면을 위한 NewsFeedItem 클래스를 작성하도록 하겠습니다. (이전에 작성했던 PreviewItem 와 유사하므로 소스만 첨부하겠습니다. (마찬가지로 domain 패키지에 추가하는 것 잊지마세요 ☺)

```
package com.ssm.android.codelab.domain;

public class NewsFeedItem {
    private String id;
    private String author_id;
    private String author_name;
    private String content;
    private String type;
    private int likes;
    private int replies;
    private String updated_time;

    public String getId() {
        return id;
    }
    public void setId(String id) {
        this.id = id;
    }
    public String getAuthor_id() {
        return author_id;
    }
    public void setAuthor_id(String author_id) {
        this.author_id = author_id;
    }
    public String getAuthor_name() {
        return author_name;
    }
    public void setAuthor_name(String author_name) {
        this.author_name = author_name;
    }
    public String getContent() {
        return content;
    }
    public void setContent(String content) {
        this.content = content;
    }
    public String getType() {
        return type;
    }
    public void setType(String type) {
        this.type = type;
    }
    public int getLikes() {
        return likes;
    }
    public void setLikes(int likes) {
        this.likes = likes;
    }
    public int getReplies() {
        return replies;
    }
    public void setReplies(int replies) {
        this.replies = replies;
    }
    public String getUpdated_time() {
        return updated_time;
    }
    public void setUpdated_time(String updated_time) {
        this.updated_time = updated_time;
    }
}
```

다음으로 뉴스피드 화면을 위한 Adapter 클래스를 작성하도록 하겠습니다. 이 Adapter 클래스의 경우도 PreviewAdapter 와 유사하므로 자세한 설명은 생략하겠습니다.

```
package com.ssm.android.codelab.adapter;

import android.content.Context;

public class NewsFeedAdapter extends ArrayAdapter<NewsFeedItem> {
    private LayoutInflater inflater = null;

    public NewsFeedAdapter(Context context, int textViewResourceId) {
        super(context, textViewResourceId);
        inflater = LayoutInflater.from(context);
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        View v = convertView;
        if (null == v) {
            v = inflater.inflate(R.layout.layout_newsfeed_item,
                               parent,
                               false);
        }
        NewsFeedItem newsfeed = getItem(position);
        ImageView author_icon = (ImageView)v.findViewById(R.id.author_icon);
        TextView author_nickname = (TextView)v.findViewById(R.id.author_nickname);
        TextView author_date = (TextView)v.findViewById(R.id.author_date);
        TextView author_content = (TextView)v.findViewById(R.id.author_content);
        TextView like_and_reply = (TextView)v.findViewById(R.id.like_and_reply);

        author_nickname.setText(newsfeed.getAuthor_name());
        author_date.setText(newsfeed.getUpdated_time());
        author_content.setText(newsfeed.getContent());
        like_and_reply.setText("좋아요 " + newsfeed.getLikes() + "개 " +
                              "댓글 " + newsfeed.getReplies() + "개");

        return v;
    }
}
```

NewsFeed 를 위한 화면을 하나 만들도록 하겠습니다. NewsFeedActivity 클래스를 생성하고 아래와 같이 작성합니다.

```
import android.os.Bundle;

import com.ssm.android.codelab.adapter.NewsFeedAdapter;

public class NewsFeedActivity extends ListActivity {
    private NewsFeedAdapter adapter;

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

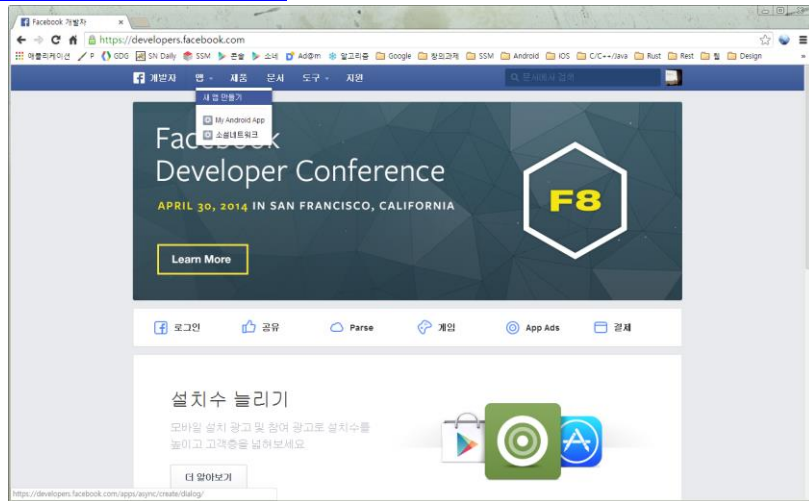
        adapter = new NewsFeedAdapter(this, 0);
        setListAdapter(adapter);
    }
}
```

화면 설계를 위한 작업은 여기까지만 하고 Facebook SDK 연동을 통해, 실제 페이스북에 올려진 데이터들이 올라올 수 있도록 변형하는 작업을 시작하도록 하겠습니다. !! 으아아!!

Facebook SDK 를 활용해서, 연동하려고 하는 데이터는 크게 2 가지 입니다.

1. 페이스북에 로그인하지 않았을 경우 로그인 창을 띄워 하도록 유도.
2. 메인화면을, 페이스북에 로그인 한 유저정보로 연동
3. 메뉴에서 뉴스피드를 클릭하면, 피드들의 목록을 연동

먼저 <https://developers.facebook.com> 사이트로 이동해서, 앱 > 새 앱 만들기를 눌러주세요.

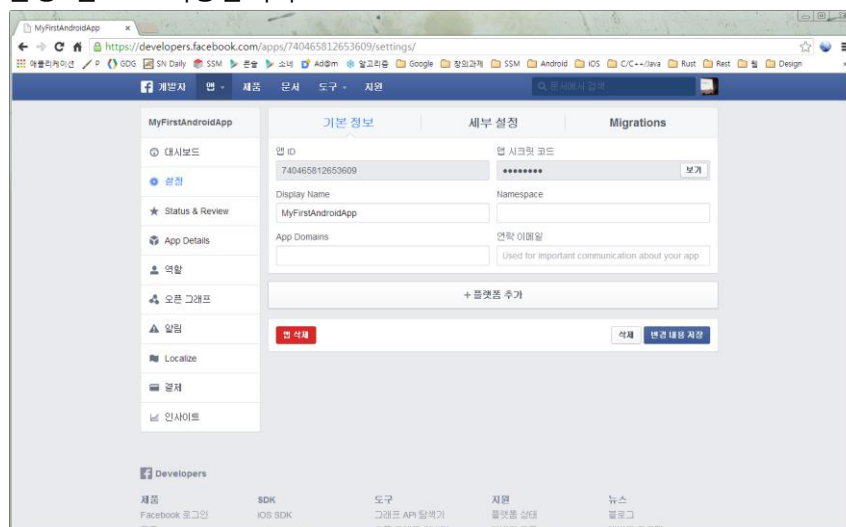


Display Name 에 MyFirstAndroidApp 을 입력하고 카테고리는 라이프스타일을 선택한 뒤 어플리케이션 만들기를 누릅니다.

 A screenshot of the 'Create a New App' form. The form has the title 'Create a New App' and a subtitle 'Get started integrating Facebook into your app or website'. It contains three input fields: 'Display Name' with the value 'MyFirstAndroidApp', 'Namespace' with the placeholder 'A unique identifier for your app (optional)', and a '분류' (Category) dropdown menu set to '라이프스타일' (Lifestyle). At the bottom, there is a checkbox for 'By proceeding, you agree to the Facebook Platform Policies' and two buttons: '취소' (Cancel) and '어플리케이션 만들기' (Create Application).

잠시 기다리면 페이스북 연동을 위한 App 하나가 생성됩니다.

좌측에 있는 설정 탭으로 이동합니다.



중앙에 있는 플랫폼 추가를 누르고 Android 앱을 추가합니다.

Package Name : com.ssm.android.codelab

Class Name : com.ssm.android.codelab.MainActivity

HashKey 는 임시로 123456 이라고 입력하고 Single Sign On 을 On 으로 바꾸고 변경내용 저장을 누릅니다.

>다시 이클립스로 이동해서,

res / values / string.xml 에 Facebook 관리자 페이지에 있는 앱 ID 를 값으로 갖는 app_id 을 추가합니다. `<string name="app_id">740465812653609</string>`

AndroidManifest.xml 파일을 열어서 application 안에 아래 세 정보를 추가합니다.

```
<meta-data android:name="com.facebook.sdk.ApplicationId" android:value="@string/app_id"/>
<activity android:name="com.facebook.LoginActivity"></activity>
<activity android:name=".NewsFeedActivity"></activity>
```

layout_main.xml 파일을 열어서 프로필 이미지를 표현하기 위해 만든 ImageView 를 com.facebook.widget.ProfilePictureView 로 변경하고 facebook:preset_size="normal" 속성을 추가합니다.

(Tip: ProfilePictureView 뷰는 페이스북 SDK 에서 제공해주는 View 로 사용자 id 값을 세팅해주면 비동기적으로 Background 에서 프로필 이미지를 불러와주는 유용한 뷰 입니다.)

```
<com.facebook.widget.ProfilePictureView
    android:id="@+id/profile"
    android:layout_width="90dp"
    android:layout_height="90dp"
    android:layout_alignParentLeft="true"
    android:layout_centerVertical="true"
    android:layout_marginLeft="10dp"
    android:background="#CCCCCC"
    android:padding="1dp"
    android:scaleType="fitXY"
    android:src="@drawable/photo_downloading"
    facebook:preset_size="normal" />
```

MainActivity.java 파일을 열어서 profile ImageView 를 ProfilePictureView 로 변경합니다.

```
private ImageView profile_cover; // 커버 영웅
private ProfilePictureView profile; // 프로필 사진 => 페이스북 API를 통해서 가져옴
private TextView profile_nickname; // 닉네임
private TextView profile_name; // 이름
private TextView profile_work; // 직장
private TextView profile_work_position; // 직책
private TextView profile_school; // 학교
private TextView profile_school_dept; // 학과
private TextView profile_home; // 거주지
private TextView profile_home_state; // 거주상태
private Gallery profile_preview_gallery; // 갤러리 미리보기
```

onCreate() 함수 내에 mMenuDrawer 선언 부를 아래와 같이 수정해줍니다.

```
// MenuDrawer
mMenuDrawer = MenuDrawer.attach(this, MenuDrawer.MENU_DRAG_WINDOW);
mMenuDrawer.setContentview(R.layout.layout_main);
mMenuDrawer.setMenuView(R.layout.layout_leftmenu);
LinearLayout menu_newsfeed = (LinearLayout)mMenuDrawer.findViewById(R.id.menu_newsfeed);
menu_newsfeed.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        startActivity(new Intent(MainActivity.this, NewsFeedActivity.class));
    }
});
```


onCreate() 함수내에 findViewById 호출하는 부분도 ImageView 를 ProfilePictureView 로 수정해줍니다.

```
profile_cover = (ImageView)findViewById(R.id.profile_cover);
profile = (ProfilePictureView)findViewById(R.id.profile);
profile_nickname = (TextView)findViewById(R.id.profile_nickname);
profile_name = (TextView)findViewById(R.id.profile_name);
profile_work = (TextView)findViewById(R.id.profile_work);
profile_work_position = (TextView)findViewById(R.id.profile_work_position);
profile_school = (TextView)findViewById(R.id.profile_school);
profile_school_dept = (TextView)findViewById(R.id.profile_school_dept);
profile_home = (TextView)findViewById(R.id.profile_home);
profile_home_state = (TextView)findViewById(R.id.profile_home_state);
profile_preview_gallery = (Gallery)findViewById(R.id.profile_preview_gallery);
```

또, ImageView 를 활용하기 때문에 작성하였던 profile 이미지를 불러오는 부분을 삭제합니다.

```
// 삭제 MainActivity.imageLoader.displayImage("http://yjaeseok.cafe24.com:8080/facebook/profile.jpg",
//      profile, MainActivity.option      s);
```

아래 부분에 Facebook 개발자 사이트에서 임시로 입력하였던 HashKey 를 알려줄 함수를 작성합니다. (로그캣으로 HashKey 가 출력됩니다.)

```
// Hash Key를 얻기 위한 코드
try {
    PackageInfo info = getPackageManager().getPackageInfo(
        "com.ssm.android.codelab",
        PackageManager.GET_SIGNATURES);
    for (Signature signature : info.signatures) {
        MessageDigest md = MessageDigest.getInstance("SHA");
        md.update(signature.toByteArray());
        Log.d("KeyHash:", Base64.encodeToString(md.digest(), Base64.DEFAULT));
    }
} catch (NameNotFoundException e) {
    Log.d("KeyHash:", "NameNotFoundException");
} catch (NoSuchAlgorithmException e) {
    Log.d("KeyHash:", "NoSuchAlgorithmException");
}
}
```

이어서 실제 Facebook SDK 를 활용하여 Facebook 서버에 MainActivity 를 위한 유저정보를 요청할 함수를 작성합니다.

```
// Session을 열고 Session을 통해서 이름, 프로필사진 얻어오기
Session session = Session.getActiveSession();
if (session != null && !session.isOpened() && !session.isClosed()) {
    session.openForRead(new Session.OpenRequest(this)
        .setPermissions(Arrays.asList("basic_info", "user_location"))
        .setCallback(mCallback));
} else {
    Session.openActiveSession(this, true, mCallback);
}
}
```

함수의 구성요소를 보면 세션을 활용하고, 세션이 연결되어있지 않다면,

setPermissions() 함수를 통해 권한 정보를 셋팅 하고

setCallback() 함수를 통해 콜백 형태로 받을 함수를 지정하는 것을 볼 수 있습니다. 콜백 함수는 아래와 같은 형태를 가지고 있습니다.

```
private StatusCallback mCallback = new StatusCallback() {
    @Override
    public void call(Session session, SessionState state, Exception exception) {
        if (session.isOpened()) {
            new Request(session, "/me?access_token=CAACEdEose0cBAHZBGZCh8oZBmSvlyleQPB5aZBoklivC4eNLGhpSUzGNIU0mt93hV0ZAHpAwBw8hBF5Z
                "&fields=id,name,username,work,location,education&locale=ko_KR", null, HttpMethod.GET, new Request.Callback() {
                    @Override
                    public void onCompleted(Response response) {
                        // ...
                    }
                }).executeAsync();
        }
    }
};
```


콜백이 발생하면 `call()`이라는 함수가 호출됩니다. 먼저 `session` 이 열려있는지 확인을 하고, Request 객체를 생성하여, 실제 요청하고 싶은 정보들을 요청하게 되는데요. 요청하기 위해서는 `session` 객체와, 요청할 Graph API 주소, 파라미터 정보, Http 요청방식, Request 를 통해 받을 콜백 함수 이렇게 5 개가 필요합니다.

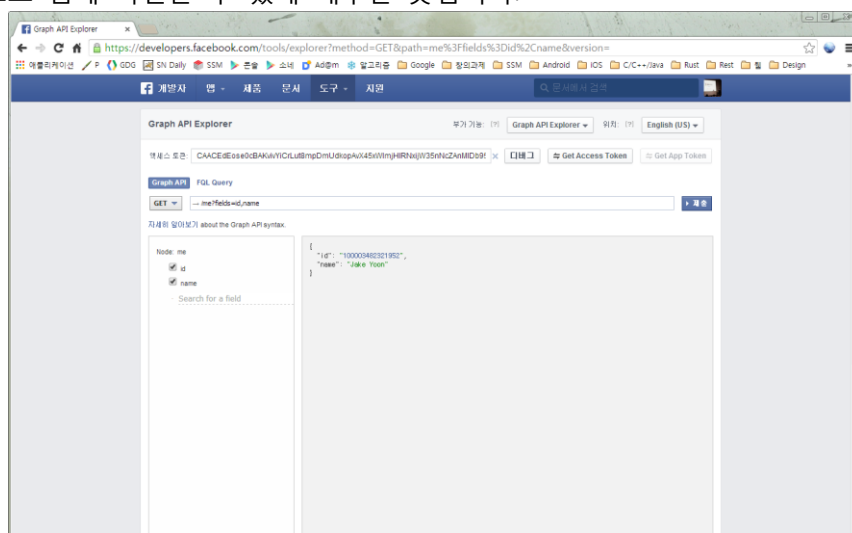
여기서 Facebook 에서 쓰이는 Graph API 라는 것을 알아야 하므로, Graph API 사이트로 이동하겠습니다. <https://developers.facebook.com/docs/graph-api/>

Graph API 는 Facebook 의 데이터를 읽거나 쓰거나 요청하기 위해 쓰이는 중요한 방법입니다. Rest 와 유사하게 데이터를 주고 받을 때 JSON 을 활용합니다.

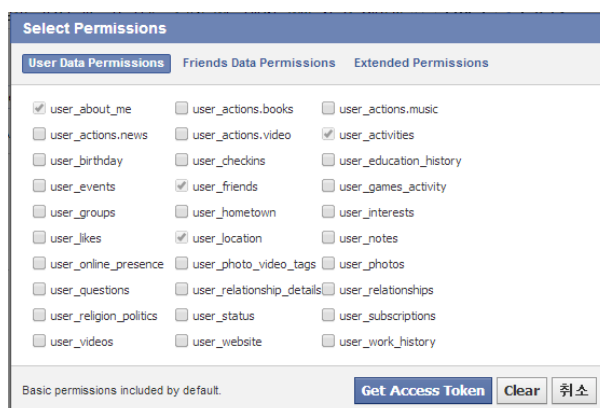
실습보다 쉽게 취득할 수 있는 방법은 없다고 생각하기에 Graph API 를 실습해보도록 하겠습니다.

Graph API Explorer : <https://developers.facebook.com/tools/explorer>

Graph API Explorer 는 Graph API 에서 동작하는 원리로 제작된 사이트로 우리가 요청하고자 하는 데이터를 눈으로 쉽게 확인할 수 있게 해주는 곳입니다.



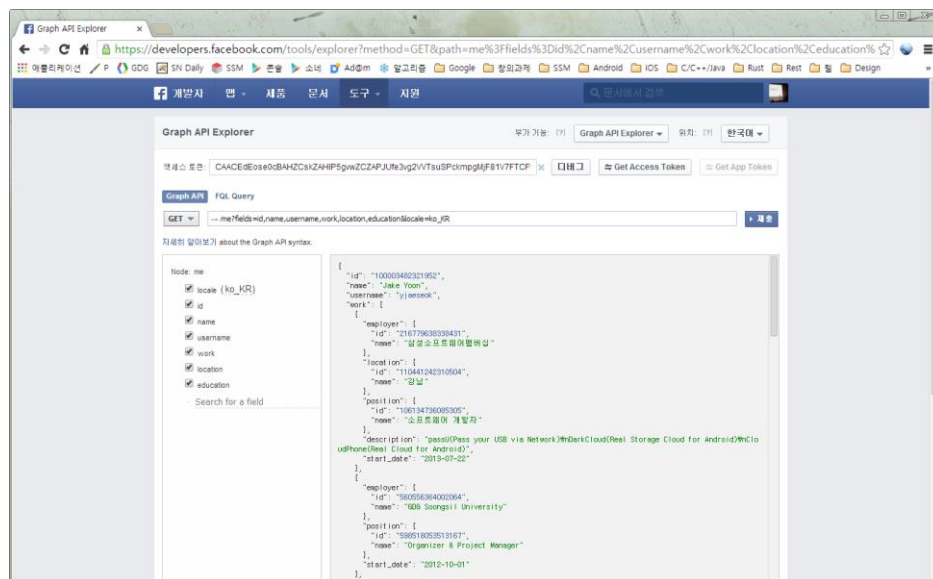
Get Access Token 은 퍼미션을 관리할 수 있는 것으로 Get Access Token 을 누르면 `user_about_me`, `user_friends`, `user_location`, `user_activities` 가 선택되어있는 것을 확인할 수 있습니다.



여기서 퍼미션을 어떻게 주느냐에 따라서 사용할 수 있는 Graph API 의 범위도 달라집니다. 저희는 MainActivity 를 위하여 사용할 것이므로 현재 퍼미션을 유지하기 위해 취소 버튼을 눌러서 빠져나옵니다.

Get Access Token 버튼 옆에 있는 English(US) 버튼을 눌러서 한국어로 바꿔줍니다.

좌측 트리에 id, name, username, work, location, education 을 추가하고 제출버튼을 누릅니다.



페이스북 유저정보들이 JSON 형태로 제공되는 것을 확인할 수 있습니다.

지금부터는 MainActivity.java 로 돌아와서 콜백함수의 세부 내용을 작성하도록 하겠습니다. 좀 길기 때문에 이번에는 소스를 드리겠습니다.

```
private StatusCallback mCallback = new StatusCallback() {
    @Override
    public void call(Session session, SessionState state, Exception exception) {
        if(session.isOpened()) {
            new Request(session, "/me?access_token=[ACCESS_TOKEN]" +
                "&fields=id,name,username,work,location,education&locale=ko_KR", null, HttpMethod.GET, new Request.Callback() {
                    @Override
                    public void onCompleted(Response response) {
                        Log.w("onCompleted", response + "");
                        JSONObject user = response.getGraphObject().getInnerJSONObject();
                        try {
                            String id = user.getString("id");
                            String name = user.getString("name");
                            String username = user.getString("username");
                            String work_name = "";
                            String work_position = "";
                            String school_name = "";
                            String school_dept = "";
                            String home = user.getJSONObject("location").getString("name");
                            int school_level = 0;
                            JSONArray works = user.getJSONArray("work");
                            if(works.length() > 0) {
                                JSONObject work = works.getJSONObject(0);
                                work_name = work.getJSONObject("employer").getString("name");
                                work_position = work.getJSONObject("position").getString("name");
                            }
                            JSONArray schools = user.getJSONArray("education");
                            for(int i = 0; i < schools.length(); i++) {
                                int level = -1;
                                JSONObject school = schools.getJSONObject(i);
                                if(school.getString("type").equals("College")) {
                                    level = 3;
                                } else if(school.getString("type").equals("High School")) {
                                    level = 2;
                                } else if(school.getString("type").equals("Middle School")) {
                                    level = 1;
                                }
                                if(school_level < level) {
                                    school_level = level;
                                    school_name = school.getJSONObject("school").getString("name");
                                    if(level == 3)
                                        school_dept = school.getJSONArray("concentration").getJSONObject(0).getString("name");
                                    else
                                        school_dept = "";
                                }
                            }
                            profile.setProfileId(id);
                            profile_nickname.setText(name);
                            profile_name.setText(username);
                            profile_work.setText(work_name);
                            profile_work_position.setText(work_position);
                            profile_school.setText(school_name);
                            profile_school_dept.setText(school_dept);
                            profile_home.setText(home);
                        } catch (JSONException e) {
                            e.printStackTrace();
                        }
                    }
                })
                .executeAsync();
        }
    }
};
```

JSON 을 파싱해서 정보들을 각 화면 뷰 들에게 넣도록 구현하였습니다.

소스내에 [ACCESS_TOKEN]이라고 되어있는 부분은 Graph API 에서 보았던 Access Token 을 넣어줍니다.

마지막으로 onActivityResult()를 재정의하여 Session 이 유지되도록 합니다.

```
@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    Session.getActiveSession().onActivityResult(this, requestCode, resultCode, data);
}
```

자 오랜만에 어플리케이션을 실행해보도록 하겠습니다. 어플리케이션을 실행하고 LogCat 에 뜨는 HashKey 를 복사하겠습니다. 복사한 HashKey 를 Facebook 개발자 사이트에 등록하고 저장해줍니다.

마지막으로 NewsFeedActivity 를 작성하고 마무리하도록 하겠습니다. NewsFeedActivity 의 소스를 다음과 같이 수정해줍니다. (역시나 소스가 길어서 소스를 드리겠습니다 ☺)

```
package com.ssm.android.codelab;

import java.util.Arrays;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import android.app.ListActivity;
import android.os.Bundle;

import com.facebook.HttpMethod;
import com.facebook.Request;
import com.facebook.Response;
import com.facebook.Session;
import com.facebook.Session.StatusCallback;
import com.facebook.SessionState;
import com.ssm.android.codelab.adapter.NewsFeedAdapter;
import com.ssm.android.codelab.domain.NewsFeedItem;

public class NewsFeedActivity extends ListActivity {
    private NewsFeedAdapter adapter;

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

        adapter = new NewsFeedAdapter(this, 0);
        setListAdapter(adapter);

        Session session = Session.getActiveSession();
        if (session != null && !session.isOpened() && !session.isClosed()) {
            session.openForRead(new Session.OpenRequest(this)
                .setPermissions(Arrays.asList("read_stream"))
                .setCallback(mCallback));
        } else {
            session.openActiveSession(this, true, mCallback);
        }
    }

    private StatusCallback mCallback = new StatusCallback() {
        @Override
        public void call(Session session, SessionState state, Exception exception) {
            if (session.isOpened()) {
                new Request(session, "/me?access_token=[ACCESS_TOKEN]" +
                    "&fields=home.limit(100).fields(id,from,message,type,updated_time,comments,likes)&locale=ko_KR", null, HttpMethod.GET, new Request.Callback() {
                    @Override
                    public void onCompleted(Response response) {
                        try {
                            JSONObject home = response.getGraphObject().getInnerJSONObject().getJSONObject("home");
                            JSONArray datas = home.getJSONArray("data");
                            for (int i = 0; i < datas.length(); i++) {
                                JSONObject data = datas.getJSONObject(i);
                                if ("status".equals(data.getString("type"))) {
                                    NewsFeedItem item = new NewsFeedItem();
                                    item.setAuthor_id(data.getJSONObject("from").getString("id"));
                                    item.setAuthor_name(data.getJSONObject("from").getString("name"));
                                    if (data.has("message"))
                                        item.setContent(data.getString("message"));
                                    if (data.has("type"))
                                        item.setType(data.getString("type"));
                                    if (data.has("updated_time"))
                                        item.setUpdated_time(data.getString("updated_time"));
                                    if (data.has("likes")) {
                                        item.setLikes(data.getJSONObject("likes").getJSONArray("data").length());
                                    } else {
                                        item.setLikes(0);
                                    }
                                    if (data.has("comments")) {
                                        item.setReplies(data.getJSONObject("comments").getJSONArray("data").length());
                                    } else {
                                        item.setLikes(0);
                                    }
                                }
                            }
                        } catch (JSONException e) {
                            e.printStackTrace();
                        }
                    }
                });
            }
        }
    }
}
```

```

        adapter.add(item);
    }
} catch (JSONException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
}).executeAsync();
}
};
}

```

끝!!.. 이제 실행을 해보도록 하겠습니다.

짜잔! 수고많이하셨습니다. (과연 여기까지 할 수 있을랑가..?)

