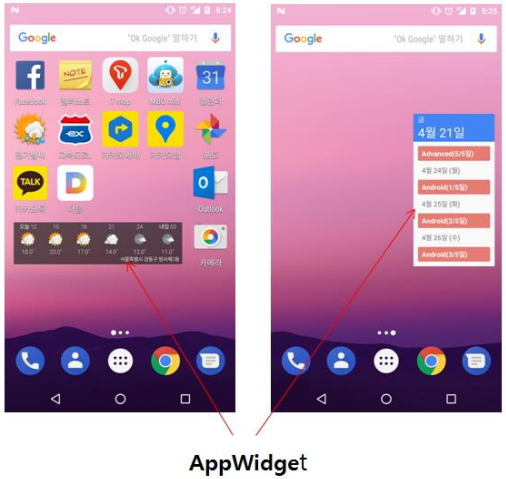
29장. AppWidget 개발

29.1 앱 위젯 기본 구조

29.1.1. 앱 위젯 개념

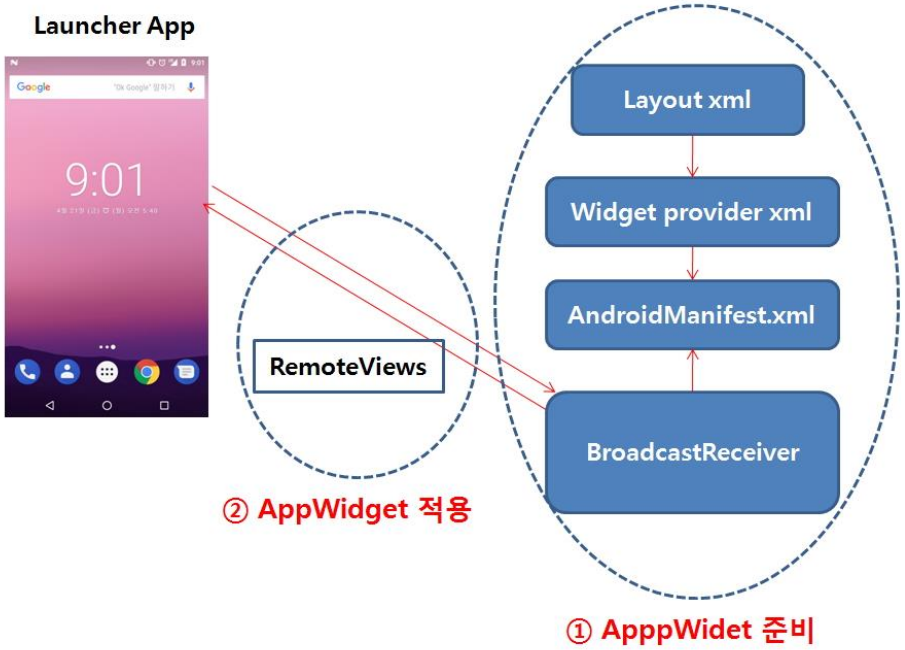
- 런처 화면에 사용자에 의해 추가되는 작은 애플리케이션



- 런처 화면에 추가할 때 각각의 앱 위젯을 식별하기 위한 고유 ID가 발급
- 하나의 앱 위젯을 런처 화면에 여러 개 추가 가능

29.1 앱 위젯 기본 구조

29.1.2. 앱 위젯 개발



- 앱 위젯 개발 layout xml
- LinearLayout, RelativeLayout, FrameLayout, GridLayout
- Button, TextView, ImageView, ImageButton, ProgressBar, ListView, GridView, StackView, AdapterViewFilipper, AnalogClock, Chronometer, ViewFlipper

29.1 앱 위젯 기본 구조

- Widget Provider XML
- 운용규칙 XML

```
<appwidget-provider xmlns:android="http://schemas.android.com/apk/res/android"
  android:minWidth="150dp"
  android:minHeight="30dp"
  android:updatePeriodMillis="1800000"
  android:initialLayout="@layout/mywidget" />
```

- minWidth: 앱 위젯을 위한 최소한의 가로 크기
- minHeight: 앱 위젯을 위한 최소한의 세로 크기
- resizeMode: 앱 위젯의 크기가 조정. none(default), vertical, horizontal
- minResizeWidth: resizeMode가 none이 아닐 때 최소 가로 크기 지정
- minResizeHeight: resizeMode가 none이 아닐 때 최소 세로 크기 지정
- previewImage: 앱 위젯의 미리보기 이미지 지정
- configure: 앱 위젯을 위한 설정을 제공하는 액티비티 지정
- widgetCategory: 4.2 추가 내용으로 홈 화면용인지 잠금 화면용인지 지정. home_screen, keyguard(5.0 이상부터는 불가능. 5.0 이상부터는 home_screen만 허용)
- initialKeyguardLayout: 잠금 화면용 레이아웃 따로 지정 가능

29.1 앱 위젯 기본 구조

- AndroidManifest.xml

```
<receiver
   android:name=".TimeWidgetReceiver"
   android:enabled="true"
   android:exported="true">
   <intent-filter>
      <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />
   </intent-filter>
   <meta-data
      android:name="android.appwidget.provider"
      android:resource="@xml/mywidget_provider" />
</receiver>
```

- 브로드캐스트 리시버

```
public class TimeWidgetReceiver extends AppWidgetProvider {
   @Override
   public void onDisabled(Context context) {
   }
   @Override
   public void onEnabled(Context context) {
      //...   }
   @Override
   public void onUpdate(Context context, AppWidgetManager appWidgetManager,                int[] appWidgetIds) {
   }
}
```

29.1 앱 위젯 기본 구조

- ACTION_APPWIDGET_ENABLED : 위젯이 최초로 설치되는 순간
- ACTION_APPWIDGET_DISABLED : 위젯이 마지막으로 제거되는 순간
- ACTION_APPWIDGET_DELETED : 위젯이 제거되는 순간
- ACTION_APPWIDGET_UPDATE: 위젯이 설치되는 순간
- ACTION_APPWIDGET_CHANGED : 크기 및 옵션이 변경되는 순간

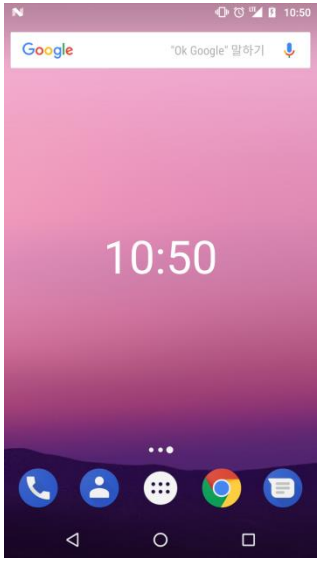
- RemoateViews를 이용한 업데이트 의뢰

```
@Override
public void onUpdate(Context context, AppWidgetManager appWidgetManager,                    int[] appWidgetIds) {

    RemoteViews remoteView = new RemoteViews(context.getPackageName(),
R.layout.mywidget);

  Calendar cal = Calendar.getInstance();
    remoteView.setTextViewText(R.id.textView, cal.get(Calendar.HOUR_OF_DAY) + ":"                    +
cal.get(Calendar.MINUTE));
    Intent intent = new Intent(context, MainActivity.class);
    PendingIntent pendingIntent = PendingIntent.getActivity(context, 0, intent,
PendingIntent.FLAG_UPDATE_CURRENT);
    remoteView.setOnClickPendingIntent(R.id.textView, pendingIntent);
    appWidgetManager.updateAppWidget(appWidgetIds, remoteView);
    super.onUpdate(context, appWidgetManager, appWidgetIds);
}
```

Step by Step 29-1 – 앱 위젯

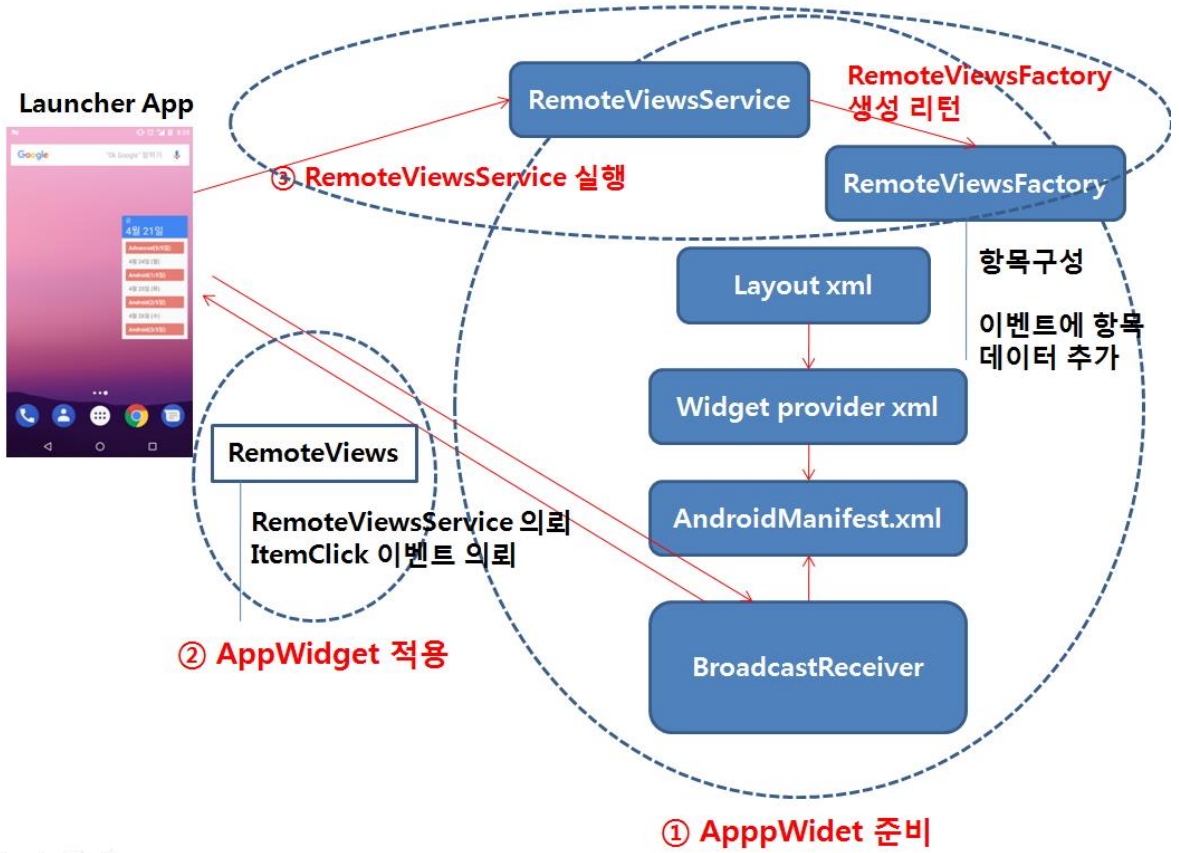
앱위젯을 이용하여 Launcher 화면에 시간을 출력하는 간단한 앱위젯 작성 테스트

1. Module 생성
2. 파일 복사
3. Widget provider xml 작성
4. Component 생성
5. TimeWidgetReceiver 작성
6. 실행



29.2 Collection 앱 위젯 개발

29.2.1. Collection 앱 위젯 구조



29.2 Collection 앱 위젯 개발

29.2.2. Collection 앱 위젯 개발

- BroadcastReceiver
- RemoteViews에 RemoteViewsService를 실행하기 위한 인텐트와 사용자의 항목 선택 이벤트 처리를 위한 인텐트를 담아 의뢰

```
@Override
public void onUpdate(Context context, AppWidgetManager appWidgetManager, int[] appWidgetIds) {

    //RemoteViewsService 실행 등록
    Intent svcIntent = new Intent(context, MyRemoteViewsService.class);
    RemoteViews widget = new RemoteViews(context.getPackageName(),
R.layout.my_collection_widget);
    widget.setRemoteAdapter(R.id.list, svcIntent);

    //항목 선택 이벤트 지정
    Intent clickIntent = new Intent(context, DetailActivity.class);
    PendingIntent clickPI = PendingIntent.getActivity(context, 0, clickIntent,
PendingIntent.FLAG_UPDATE_CURRENT);
    widget.setPendingIntentTemplate(R.id.list, clickPI);

    //의뢰
    appWidgetManager.updateAppWidget(appWidgetIds, widget);
    super.onUpdate(context, appWidgetManager, appWidgetIds);
}
```

29.2 Collection 앱 위젯 개발

- 항목의 변경사항이 발생하면 notifyAppWidgetViewDataChanged( ) 함수를 호출

```
AppWidgetManager appWidgetManager = AppWidgetManager.getInstance(context);
int appWidgetIds[] = appWidgetManager.getAppWidgetIds(new ComponentName(context,
MyCollectionWidgetReceiver.class));
//데이터 변경. RemoteViewsFactory 의 onDataChanged 함수 호출
appWidgetManager.notifyAppWidgetViewDataChanged(appWidgetIds, R.id.list);
```

- RemoteViewsService
- RemoteViewsFactory를 얻을 목적으로 인텐트 발생에 의해 실행

```
public class MyRemoteViewsService extends RemoteViewsService {
    @Override
    public RemoteViewsFactory onGetViewFactory(Intent intent) {
        return(new MyRemoteViewsFactory(this.getApplicationContext()));
    }
}
```

29.2 Collection 앱 위젯 개발

- RemoteViewsFactory
- ListView를 구성하기 위한 Adapter 역할

- onCreate: 최초 한 번 호출
- onDestroy: 마지막에 한 번 호출
- getCount: 항목 개수를 판단하기 위해서 호출
- getViewAt: 각 항목을 구현하기 위해 호출
- getLoadingView: 로딩 뷰를 표현하기 위해 호출. 없으면 null 반환
- getViewTypeCount: 항목의 타입 개수를 판단하기 위해 호출
- getItemId: 각 항목의 식별자 값을 얻기 위해 호출
- onDataSetChanged: 항목 추가, 제거 등의 변경이 발생 시 호출

- 항목 선택 이벤트 발생 시 인텐트에 담겨야 할 항목 데이터를 추가

```
RemoteViews row = new RemoteViews(context.getPackageName(), R.layout.item_collection);

Intent i = new Intent();
i.putExtra("item_id",arrayList.get(position)._id);
i.putExtra("item_data", arrayList.get(position).content);
row.setOnClickFillInIntent(R.id.text1, i);
```

Step by Step 29-2 – Collection 타입 앱 위젯

Collection 타입의 앱위젯을 테스트

•간단한 Todo 성격의 앱위젯으로 Activity 화면에서 등록한 Todo 항목을 앱위젯에서 최근 5개 추출해서 나열
•Todo 항목은 DB에 저장하여 사용
•새로운 Todo 가 추가된 경우 앱위젯에 반영되도록 처리

1. Activity 생성
2. Component 생성
3. 파일 복사
4. Widget provider xml 작성
5. AndroidManifest.xml 작업
6. MyRemoteViewsFactory 작성
7. MyRemoteViewsService 작성
8. MyCollectionWidgetReceiver 작성
9. Lab29_2Activity.java 실행