자바응용SW(앱)개발자양성

RecyclerView

백제직업전문학교
김영준 강사

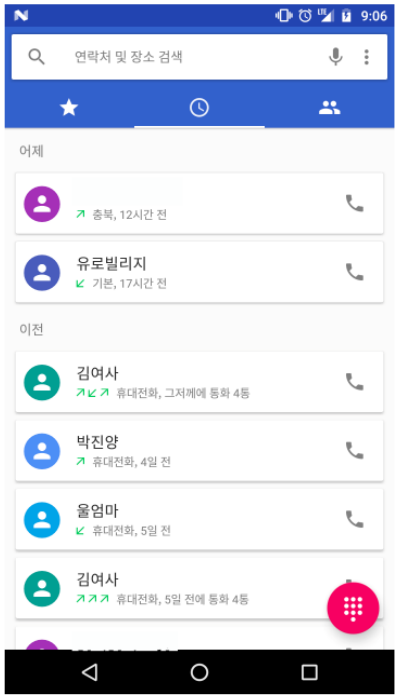
RecyclerView

1. RecyclerView 소개

- RecyclerView는 API Level 21(Android 5.0)이 나오면서 support:recyclerView-v7 라이브러리로 제공된 클래스

```
implementation 'com.android.support:recyclerview-v7:26.0.1'
```

- RecyclerView는 API Level 21(Android 5.0)이 나오면서 support:recyclerView-v7 라이브러리로 제공된 클래스



RecyclerView

- Adapter: RecyclerView 항목 구성
- ViewHolder: 각 항목 구성 뷰의 재활용을 목적으로 View Holder 역할
- LayoutManager : 항목의 배치
- ItemDecoration: 항목 꾸미기
- ItemAnimation: 아이템이 추가, 제거, 정렬될 때의 애니메이션 처리

2. Adapter, ViewHolder

- RecyclerView를 하나 준비

```
<android.support.v7.widget.RecyclerView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/lab3_recycler"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```

RecyclerView

- ViewHolder 클래스

```
private class MyViewHolder extends RecyclerView.ViewHolder {
    public TextView title;
    public MyViewHolder(View itemView) {
        super(itemView);
        title = (TextView) itemView.findViewById(android.R.id.text1);
    }
}
```

- Adapter

```
private class MyAdapter extends RecyclerView.Adapter<MyViewHolder> {
    private List<String> list;
    public MyAdapter(List<String> list) {
        this.list = list;
    }
    public MyViewHolder onCreateViewHolder(ViewGroup viewGroup, int i) {
        View view = LayoutInflater.from(viewGroup.getContext())
                .inflate(android.R.layout.simple_list_item_1, viewGroup, false);
        return new MyViewHolder(view);
    }
    public void onBindViewHolder(MyViewHolder viewHolder, int position) {
        String text = list.get(position);
        viewHolder.title.setText(text);
    }
    public int getItemCount() {
        return list.size();
    }
}
```

RecyclerView

```
private class MyAdapter extends RecyclerView.Adapter<MyViewHolder> {
    //.....................................
    public MyViewHolder onCreateViewHolder(ViewGroup viewGroup, int i) {
        View view = LayoutInflater.from(viewGroup.getContext())
          .inflate(android.R.layout.simple_list_item_1, viewGroup, false);
        return new MyViewHolder(view);
    }
    @Override
    public void onBindViewHolder(MyViewHolder viewHolder, int position) {
        String text = list.get(position);
        viewHolder.title.setText(text);
    }
}
private class MyViewHolder extends RecyclerView.ViewHolder {
    public TextView title;
    public MyViewHolder(View itemView) {
        super(itemView);
        title = (TextView) itemView.findViewById(android.R.id.text1);
    }
}
```

layout inflate ①

ViewHolder 리턴 ④

항목 구성 ⑤

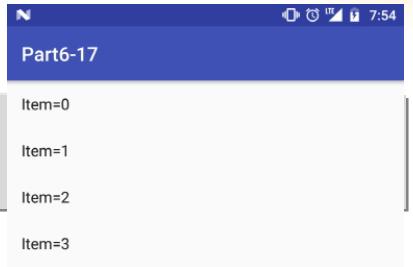
⑥ 전달된 ViewHolder 이용

② Layout 초기화후 ViewHolder 생성

③ 필요 View findViewById

RecyclerView

- Adapter를 RecyclerView에 적용




```
recyclerView.setLayoutManager(new LinearLayoutManager(this));
recyclerView.setAdapter(new MyAdapter(list));
```

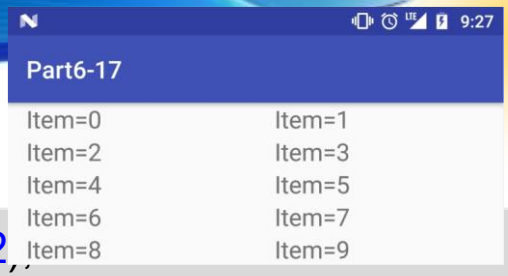
3. LayoutManager

- 항목을 어떻게 배치
- LinearLayoutManager: 수평, 수직으로 배치
- GridLayoutManager: 그리드 화면으로 배치
- StaggeredGridLayoutManager: 높이가 불규칙한 그리드 화면으로 배치

- LinearLayoutManager

```
LinearLayoutManager linearManager = new LinearLayoutManager(this);
linearManager.setOrientation(LinearLayoutManager.HORIZONTAL);
recyclerView.setLayoutManager(linearManager);
```

RecyclerView



- GridLayoutManager

```
GridLayoutManager gridManager=new GridLayoutManager(this, 2);
recyclerView.setLayoutManager(gridManager);
```

- 방향 지정

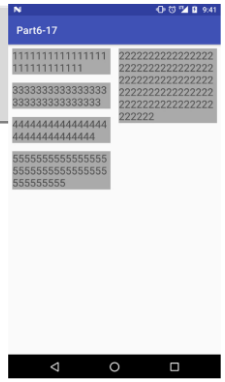
```
GridLayoutManager gridManager=new GridLayoutManager(this, 2, GridLayoutManager.HORIZONTAL, false);
```

- 아래부터 나열

```
GridLayoutManager gridManager=new GridLayoutManager(this, 2,
GridLayoutManager.VERTICAL, true);
```

- StaggeredGridLayoutManager

```
StaggeredGridLayoutManager sgManager=new StaggeredGridLayoutManager(2,
StaggeredGridLayoutManager.VERTICAL);
```



4. ItemDecoration

각 항목을 다양하 게 꾸미기

- onDraw: 항목을 배치하기 전에 호출
- onDrawOver: 모든 항목이 배치된 후에 호출
- getItemOffsets: 각 항목을 배치할 때 호출

```
class MyItemDecoration extends RecyclerView.ItemDecoration {

    @Override
    public void getItemOffsets(Rect outRect, View view, RecyclerView parent,
                               RecyclerView.State state) {
        super.getItemOffsets(outRect, view, parent, state);
        //항목의 index 값 획득
        int index=parent.getChildAdapterPosition(view)+1;

        if(index % 3 == 0)
            //left, top, right, bottom
            outRect.set(20, 20, 20, 60 );
        else
            outRect.set(20, 20, 20, 20 );

        view.setBackgroundColor(0xFFECE9E9);
        ViewCompat.setElevation(view, 20.0f);
    }
}
```

RecyclerView

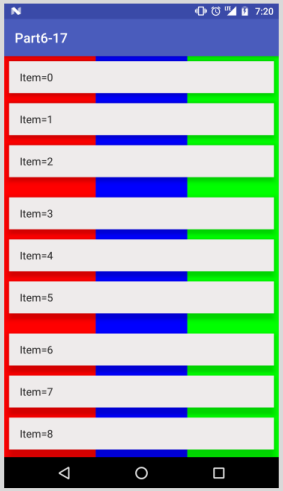
- ItemDecoration을 RecyclerView에 적용

```
recyclerView.addItemDecoration(new MyItemDecoration());
```

- onDraw( ) 함수

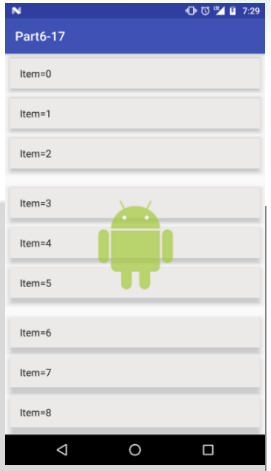
```
public void onDraw(Canvas c, RecyclerView parent, RecyclerView.State state) {
    super.onDraw(c, parent, state);
    //RecyclerView의 사이즈 계산
    int width=parent.getWidth();
    int height=parent.getHeight();

    Paint paint=new Paint();
    paint.setColor(Color.RED);
    c.drawRect(0, 0, width/3, height, paint );
    paint.setColor(Color.BLUE);
    c.drawRect(width/3, 0, width/3*2, height, paint);
    paint.setColor(Color.GREEN);
    c.drawRect(width/3*2, 0, width, height, paint);
}
```



RecyclerView

- onDrawOver( )




```
public void onDrawOver(Canvas c, RecyclerView parent, RecyclerView.State state) {
    super.onDrawOver(c, parent, state);
    //RecyclerView의 사이즈 계산
    int width=parent.getWidth();
    int height=parent.getHeight();
    //이미지 사이즈 계산
    Drawable dr= ResourcesCompat.getDrawable(getResources(),   R.drawable.android, null);
    int drWidth=dr.getIntrinsicWidth();
    int drHeight=dr.getIntrinsicHeight();

    int left=width/2 - drWidth/2;
    int top=height/2 - drHeight/2;
    c.drawBitmap(BitmapFactory.decodeResource(getResources(),  R.drawable.android), left,top,null);
}
```

RecyclerView

RecyclerView 테스트