1. 리스트뷰 사용하기

왜 굳이 선택위젯이라는 이름으로 구분할까?

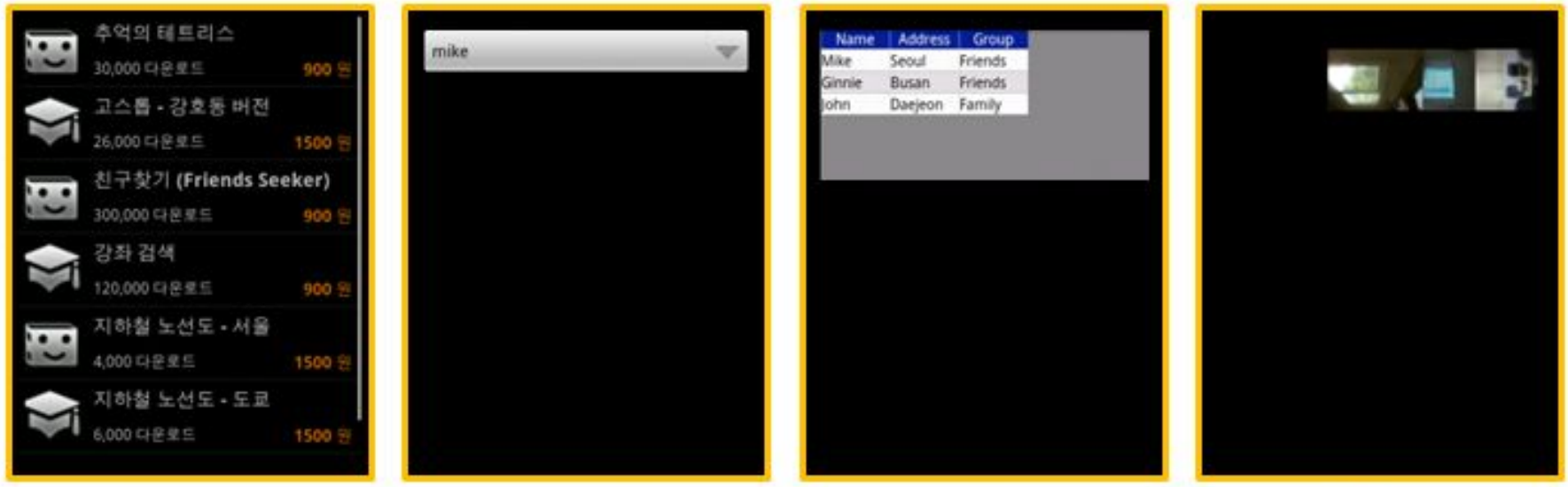
● 안드로이드에서는 여러 아이템 중의 하나를 선택하는 선택위젯은 별도의 패턴을 사용함

- 여러 개의 아이템 중에서 하나를 선택하는 방식의 선택 위젯은 어댑터를 사용하여야 함
- 이 어댑터에서 데이터를 관리하도록 해야 할 뿐만 아니라 화면에 보여지는 뷰도 어댑터의 **getView()** 메소드에서 결정함
- 선택위젯의 가장 큰 특징은 원본 데이터를 위젯에 직접 설정하지 않고 어댑터라는 클래스를 사용하도록 되어 있다는 점으로 이 패턴을 잘 기억해 두어야 함

[선택 위젯과 어댑터]

대표적인 선택 위젯

- 안드로이드에서 제공하는 대표적인 선택 위젯들은 그 사용방법을 잘 알아두어야 함



리스트뷰로 보여줄 때 해야 할 일들

(1) 아이템을 위한 XML 레이아웃 정의하기

- 리스트뷰에 들어갈 각 아이템의 레이아웃을 **XML**로 정의함

(2) 아이템을 위한 뷰 정의하기

- 리스트뷰에 들어갈 각 아이템을 하나의 뷰로 정의. 이 뷰는 여러 개의 뷰를 담고 있는 뷰그룹이어야 함
- 이것은 부분화면과 같아서 **(1)**번에서 정의한 **XML** 레이아웃을 인플레이션 후 설정해야 함

(3) 어댑터 정의하기

- 데이터 관리 역할을 하는 어댑터 클래스를 만들고 그 안에 각 아이템으로 표시할 뷰를 리턴하는 **getView()** 메소드를 정의함

(4) 리스트뷰 정의하기

- 화면에 보여줄 리스트뷰를 만들고 그 안에 데이터가 선택되었을 때 호출될 리스너 객체를 정의함

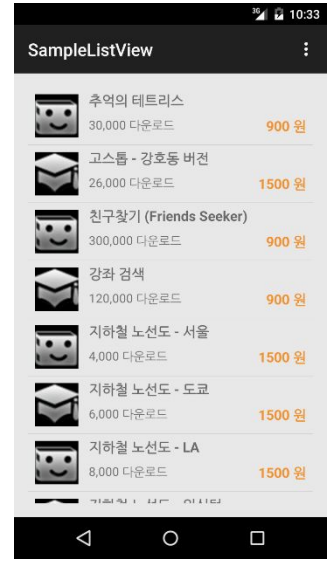
3. 리스트뷰 사용하기

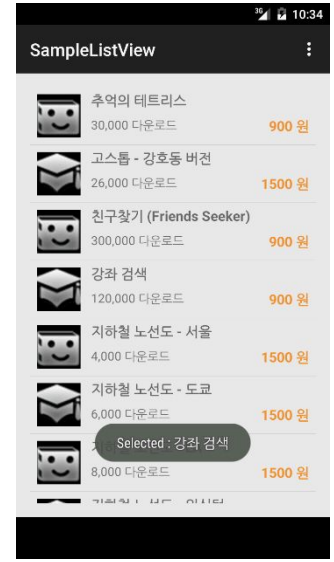
리스트뷰 사용하기 예제

리스트뷰 사용하기 예제

-아이콘이 들어 있는 리스트뷰의 아이템 정의

-리스트뷰의 한 아이템을 선택했을 때 토스트 표시





아이템의 XML 레이아웃

-한 아이템으로 보여질 뷰의 XML 레이아웃 정의

한 아이템의 데이터를 넣을 클래스

-한 아이템의 데이터를 넣을 클래스 정의

어댑터 클래스 정의

-리스트뷰를 보여주는 역할을 하는 어댑터 클래스 정의

리스트뷰를 사용하는 메인 액티비티 코드 작성

-리스트뷰를 사용하는 메인 액티비티 코드 작성

리스트뷰 선택 이벤트 처리 코드 작성

-한 아이템을 선택했을 때의 이벤트 처리

한 아이템으로 보여줄 XML 레이아웃 정의

- 선택위젯에서 각각의 아이템은 동일한 레이아웃을 가진 뷰가 반복적으로 보여짐
- 각각의 아이템을 위한 **XML** 레이아웃이 필요함

[리스트뷰의 한 아이템이 갖는 레이아웃의 예]

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:orientation="horizontal"
  >
```

Continued..

한 아이템으로 보여줄 XML 레이아웃 정의 (계속)

```
<ImageView
  android:id="@+id/iconItem"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:padding="8dip"
  android:layout_gravity="center_vertical"
  />

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

1 왼쪽에 보이는 이미지뷰 정의

Continued..

한 아이템으로 보여줄 XML 레이아웃 정의 (계속)

```
<TextView
android:id="@+id/dataItem01"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:textStyle="bold"
android:textSize="12dp"
android:padding="4dp"
/>

<RelativeLayout
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:padding="4dp"
>
```

2 첫 번째 줄의 텍스트뷰 정의

3 두 번째 줄을 표시할 상대 레이아웃 정의

Continued..

한 아이템으로 보여줄 XML 레이아웃 정의 (계속)

```
<TextView
android:id="@+id/dataItem02"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
/>
<TextView
android:id="@+id/dataItem03"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentRight="true"
android:textColor="#ccf88107"
android:textSize="14dp"
android:textStyle="bold"
android:paddingRight="4dp"
/>
</RelativeLayout>
</LinearLayout>
</LinearLayout>
```

4 텍스트뷰 정의

5 두 번째 줄의 오른쪽에 있는 텍스트뷰 정의

한 아이템으로 보여줄 뷰 정의

- 어댑터의 **getView()** 메소드에서 리턴해 줄 뷰를 별도의 클래스로 정의하여 사용

```
public class IconTextView extends LinearLayout {
…
LayoutInflater inflater = (LayoutInflater)
context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
inflater.inflate(R.layout.listitem, this, true);

mIcon = (ImageView) findViewById(R.id.iconItem);
mIcon.setImageDrawable(aItem.getIcon());


mText01 = (TextView) findViewById(R.id.dataItem01);
mText01.setText(aItem.getData(0));
mText02 = (TextView) findViewById(R.id.dataItem02);
mText02.setText(aItem.getData(1));
mText03 = (TextView) findViewById(R.id.dataItem03);
mText03.setText(aItem.getData(2));
}
```

1 아이템의 모양을 구성한 **XML** 레이아웃을 인플레이터를 이용해 메모리 객체화

2 레이아웃에 정의된 이미지뷰 객체 참조

3 레이아웃에 정의된 텍스트뷰 객체 중에 첫 번째 것 참조

한 아이템으로 보여줄 데이터 클래스 정의

- 각각의 아이템을 위한 데이터도 별도의 클래스로 정의하여 사용

```
public class IconTextItem {

private Drawable mIcon;
private String[] mData;

public IconTextItem(Drawable icon, String[] obj) {
mIcon = icon;
mData = obj;
}
```

1 리스트뷰의 한 아이템에 표시할 데이터를 담고 있을 클래스 정의

2 Drawable 타입의 변수와 문자열 타입의 배열 변수 선언

3 Drawable 객체와 문자열 타입의 배열을 파라미터로 전달받는 생성자

새로운 어댑터 클래스 정의

```
public class IconTextListAdapter extends BaseAdapter {
private Context mContext;

private List<IconTextItem> mItems = new ArrayList<IconTextItem>();

public IconTextListAdapter(Context context) {
mContext = context;
}
…
public int getCount() {
 return mItems.size();
}
…
public Object getItem(int position) {
 return mItems.get(position);
}
…
```

1 **BaseAdapter**를 상속하여 새로운 어댑터 클래스 정의

2 각 아이템의 데이터를 담고 있는 **IconTextItem** 객체를 저장할 **ArrayList** 객체 생성

3 전체 아이템의 개수를 리턴하는 메소드 정의

Continued..

새로운 어댑터 클래스 정의 (계속)

```
public View getView(int position, View convertView, ViewGroup parent) {
IconTextView itemView;
if (convertView == null) {
itemView = new IconTextView(mContext);
} else {
itemView = (IconTextView) convertView;
}
itemView.setIcon(mItems.get(position).getIcon());
itemView.setText(0, mItems.get(position).getData(0));
itemView.setText(1, mItems.get(position).getData(1));
itemView.setText(2, mItems.get(position).getData(2));
return itemView;
}
}
```

4 아이템에 표시할 뷰 리턴하는 메소드 정의

getView() 메소드

[Reference]

public View getView (int position, View convertView, ViewGroup parent)

- 첫 번째 파라미터 - 아이템의 인덱스를 의미하는 것으로 리스트뷰에서 보일 아이템의 위치 정보라 할 수 있음.
  0부터 시작하여 아이템의 개수만큼 파라미터로 전달됨
- 두 번째 파라미터 - 현재 인덱스에 해당하는 뷰 객체를 의미하는데 안드로이드에서는 선택 위젯이 데이터가 많아 스크롤될 때 뷰를 재활용하는 메커니즘을 가지고 있어 한 번 만들어진 뷰가 화면 상에 그대로 다시 보일 수 있도록 되어 있음. 따라서 이 뷰가 널값이 아니면 재활용 가능함
- 세 번째 파라미터 - 부모 컨테이너 객체임

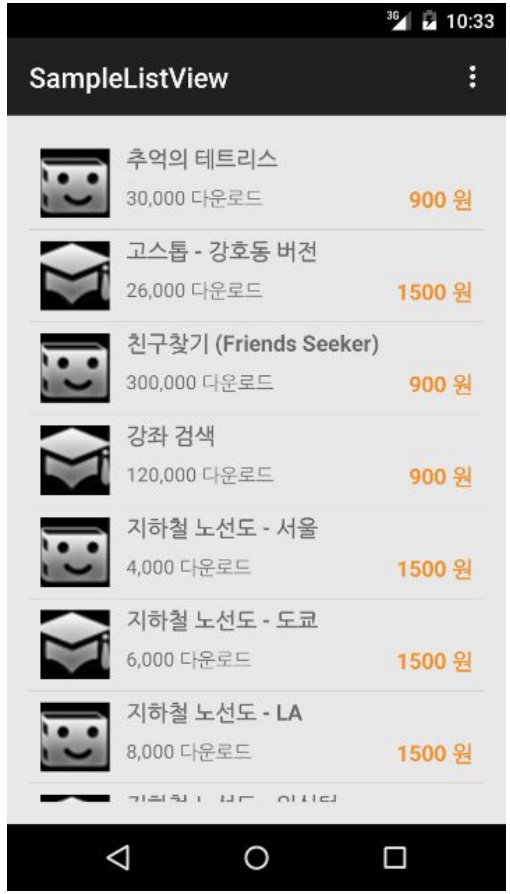
메인 액티비티 코드 만들기

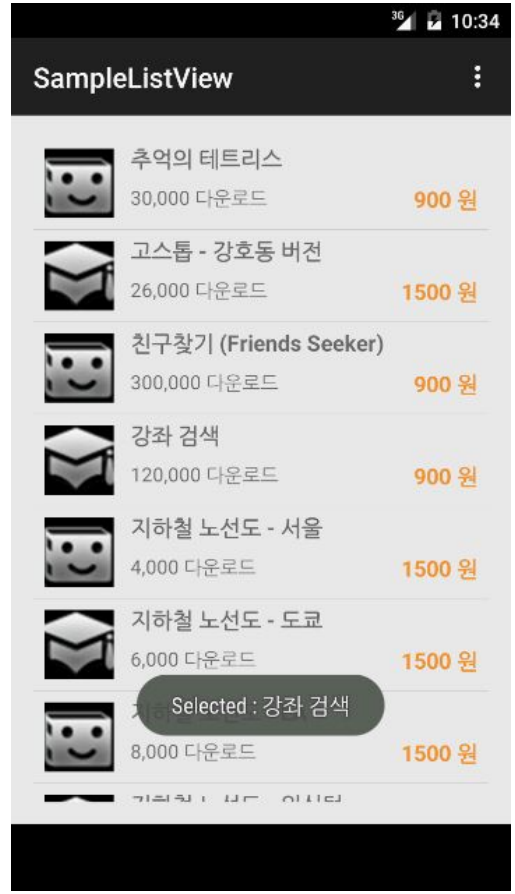
```
list.setAdapter(adapter);

list.setOnItemClickListener(new OnItemClickListener() {
public void onItemClick(AdapterView parent,View v,int position, long id) {
IconTextItem curItem = (IconTextItem) adapter.getItem(position);
String[] curData = curItem.getData();
Toast.makeText(getApplicationContext(), "Selected : " + curData[0], Toast.LENGTH.SHORT).show();
}
});
```

리스트뷰에 어댑터 객체 설정

아이템을 클릭했을 때 토스트 메시지를 보여주도록 리스너 설정

실행 화면





3. 리스트뷰 사용하기

[References]

• 기본 서적

2016, 정재곤, “Do it! 안드로이드 앱 프로그래밍(개정3판)”, 이지스퍼블리싱(주)

• Android Website

http://www.android.com/

• Google Developer’s Conference

http://code.google.com/events/io/

• Android SDK Documentation