**Show View > Other > LogCat (non-duplicated)**

ViewGroup = UI พิเศษ ที่กำหนด Layout โดยเฉพาะ  
Layout จะมีสองประเภทคือ  
 LinearLayout คือ Layout ที่ Widget แต่ละตัวจะเรียงต่อๆกัน เมื่อเราวาง โดยจะมี 2 แนวคือ Vertical (แนวตั้ง) กับ Horizontal (แนวนอน)

Widget ใน LinearLayout จะมี Option: Weight ซึ่งหมายถึง อัตราส่วนที่จะขยายในแนว (ตั้ง นอน) ซึ่งหากมีค่าเท่าๆกัน นั่นหมายความว่า Layout จะขยายเท่ากัน  
 RelativeLayout คือ Layout ที่ Widget สามารถวางได้อย่างอิสระ โดยแต่ละตัว จะมีการอ้างอิงกับตำแหน่งอื่นๆ ของขอบจอบ้าง หรือ ของ Widget อื่นๆบ้าง

Widget ใน RelativeLayout จะมี Option: AlignParent หมายความว่า จะให้ไปชิด มุม/ขอบไหนของหน้าจอ  
 layout\_above หรือ align\_component เป็น Option ที่จะให้เลือกว่า Widget นั้น ๆจะไป ชิด/ติด กับ Widget ใดๆ แทน ขอบ/มุม  
 Android:Gravity คือ จะให้ข้อความ ไปอยู่ส่วนไหน ของกรอบ ของ Relative

TextSize: sp WidgetSize: dp  
Layout: Gravity มันหมายความว่า จะให้ Layout อยู่ตำแหน่งไหน เมื่อเทียบกับ “หน้าจอ”

โดยในการสร้าง UI นั้น จะสามารถสร้าง Layout แบบต่างๆ ได้รวมกัน (แต่ในหน้าจอหลักๆ ก็ต้องมี Layout บอก)

Fragment = <http://www.vogella.com/articles/AndroidFragments/article.html>

**MVC**

public interface OnNumberChangeListener{  
 void onNumberChange(Number number);  
 }

private OnNumberChangeListener onNumberChangeListener;

if(this.onNumberChangeListener!=null){  
 this.onNumberChangeListener.onNumberChange(this);  
}

Interface : ทุก method ต้อง implement ใหม่หมด ดังนั้น เวลาใช้จะต้อง implements จาก class  
abstract : บาง method มันเป็น interface หรือพูดง่ายๆ มันคือ interface ที่ implements ไปแล้วบาง method ดังนั้น ก็ขี้เกียจ implement ต่อ ก็ประกาศมันให้มันเป็น abstract ซึ่ง abstract นี้ ต้องประกาศ extends หากจะใช้งานต่อ

**ListView**

สำหรับ Listview นั้น เราจะต้องทำการ add Adapter ใส่ไปใน List นั้นๆ ซึ่ง Adapter ที่เราจะสร้างนี้ จะต้อง

Extends BaseAdapter implements ListAdapter

**public** View getView(**int** position, View convertView, ViewGroup parent) {  
 getView เป็น method ที่ List จะทำการเรียกทุกครั้ง เมื่อต้องการจะวาด view ใหม่ ใน listview เพราะ listview จะแสดงแค่ 5-8 ตัว บลาๆ ทุกครั้งที่ เรา scroll ไป อันเก่าก็จะถูกเก็บไว้ อันใหม่ก็สร้างใหม่ไป เพราะฉะนั้น Mechanism จะอยู่ตรงนี้

**private** Context mContext;  
 @Override

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

ViewHolder viewHolder = **new** ViewHolder();

// **TODO** Auto-generated method stub

**if**(convertView == **null**){

convertView = LayoutInflater.*from*(mContext).inflate(R.layout.*dotrow*, parent, **false**);

viewHolder.txtCorX = (TextView) convertView.findViewById(R.id.*txtXcor*);

viewHolder.txtCorY = (TextView) convertView.findViewById(R.id.*txtYcor*);

convertView.setTag(viewHolder);

}**else**{

viewHolder = (ViewHolder) convertView.getTag();

}

//Update new Row

Dot dot = (Dot) getItem(position);

viewHolder.txtCorX.setText(dot.getCorX() + "");

viewHolder.txtCorY.setText(dot.getCorY() + "");

**return** convertView;

}

Method: getView คือ method ที่ถูกเรียกใช้ทุกๆครั้ง เมื่อมีการสร้าง LIST ใหม่ (ย้ำน๊ะว่า List) แล้วจะมีการ return convertView เข้าไป หมายความว่า ให้สร้าง view อะไรเข้าไปใน LISTVIEW นั้น

จาก method นี้ จะได้ว่า  
convertView จะมีค่า != null หากว่าเป็น view เก่า (เดิม) ซึ่ง method .getTag จะรับ List เดิม!!! มา recycle มาใช้ โดยมันจะบอกว่า ตัวเก่าที่รับมาคือตัวที่เท่าไหร่ (ก็ได้รับค่า **int** position เข้ามา)   
แต่ถ้าหากว่า convertView = null เราก็ทำการสร้าง convertView ใหม่ (คือตัว dotrow) แล้วให้ viewHolder ชี้ Link ไปใน dotrow ของ convertView (ทำการชี้ไปตลอด) จากนั้น ก็ setTag ไป

ในส่วน Update เนื่องจาก เรายัด viewHolder เข้าไปใน converView แล้ว คำสั่ง getTag คือ ดึงตัวที่ถูกยัดไปใน view ออกมา) เราก็เปลี่ยนออกมาง่ายๆ จากนั้นก็ทำการ return ปกติ

LayoutInflater.*from*(mContext).inflate(R.layout.*dotrow*, parent, **false**);

Inflat คือการเปลี่ยน Layout ในรูป xml ให้กลายเป็น view  
การอยากเข้าถึง res ให้เข้าถึงผ่าน context แล้ว Activity เป็น subclass ของ context  
ซึ่ง Adapter จะเข้าถึง res ผ่าน MainActivity (Context)

@Override

**public** **void** onDotChange(Dots dots) {

// **TODO** Auto-generated method stub

mAdapter.notifyDataSetChanged();

}

อย่าลืม ให้ notifyDataSetChanged ไปใน mAdapter ด้วย เพื่อให้ mAdapter สั่ง Update ListView (จริงๆมันก็คือการใช้คำสั่ง getView นั่นแหละ แต่ว่า convertView จะเท่ากับ null เพราะต้องเป็นตัวใหม่)

mListView = (ListView) findViewById(R.id.*listView1*);

mAdapter = **new** DotsListAdapter(**this**) {

@Override

**public** Object getItem(**int** position) {

**return** mDots.getPosition(position);

}

@Override

**public** **int** getCount() {

**return** mDots.getSize();

}

};

mListView.setAdapter(mAdapter);

อย่าลืม setAdapter ด้วย เมื่อสร้าง Adapter เสร็จ เพราะว่าจริงๆแล้ว Adapter มันก็เหมือน EventHandle นั่นละ

Preference คือ Persistance-Data อย่างง่าย เป็น file-base ซึ่งเป็นค่าคงที่ ซึ่งส่วนมากอาจจะเป็น Current-User Data ไรงี้ โดยจะเก็บรูปแบบใน key-Value ลองหา keyword SharedPreference ดู

Main Android Component

Activity : ส่วนติดต่อกับผู้ใช้ (View & Control ใน MVC) หรือ User Interface   
Service : Activity ไม่มี UI (Background Process) แตกต่างกับ Tread  
ContentProvider : Database สาธารณะ (แต่ในความเป็นจริง ไม่จำเป็นต้อง share DB เสมอไป ซึ่งมันมีประโยชน์หลายอย่าง)  
BroadcastReceiver : Receiver รับ Event บางอย่างของ Android เช่น คนโทรเข้า (เขียนง่ายๆ)

**Android Option Menu**

อยู่กับตัว Activity โดย Override Method ใน Activity ชื่อว่า OnCreateOptionMenu กับ OnOptionItemSelected (แบบง่าย)

@Override

**public** **boolean** onOptionsItemSelected(MenuItem item) {

**switch**(item.getItemId()){

**case** *1001*:

mDots.clearDot();

**return** **true**; //การ return false มันจะไปหาตัวอื่นที่ดักจับอยู่

}

**return** **super**.onOptionsItemSelected(item);

}

@Override

**public** **boolean** onCreateOptionsMenu(Menu menu) {

menu.add(Menu.*NONE*, *1001*, Menu.*NONE*, getString(R.string.*clear*));

**return** **true**;

}

**Android Context Menu**

registerForContextMenu(mListView);เขียนนี่เข้าไปในส่วนของ onCreate Method ก่อน เพื่อให้ Widget นั้น มีคำสั่งเมื่อทำการ LongClick

@Override

**public** **boolean** onContextItemSelected(MenuItem item) {

AdapterContextMenuInfo mInfo = (AdapterContextMenuInfo) item.getMenuInfo();

**switch**(item.getItemId()){

**case** *MENU\_DELETE\_ITEM*:

mDots.delete(mInfo.position);

**return** **true**;

}

**return** **super**.onContextItemSelected(item);

}

@Override

**public** **void** onCreateContextMenu(ContextMenu menu, View v,

ContextMenuInfo menuInfo) {

menu.add(Menu.*NONE*, *MENU\_DELETE\_ITEM*, Menu.*NONE*, R.string.*delete*);

**super**.onCreateContextMenu(menu, v, menuInfo);

}

**Dialog**

**private** **void** confirmDelete(**final** **int** position) {

**new** AlertDialog.Builder(**this**).setTitle("Confirm Delete?")

.setMessage("Are you sure to delete?")

.setPositiveButton("Yes", **new** OnClickListener() {

@Override

**public** **void** onClick(DialogInterface dialog, **int** which) {

// **TODO** Auto-generated method stub

mDots.delete(position);

}

})

.setNegativeButton("No", **null**).show();

}

**private** **void** EditBox(**final** **int** position){

View view = getLayoutInflater().inflate(R.layout.*editdialog*, **null**);

**final** EditText nullcoorX = (EditText) view.findViewById(R.id.*editX*);

**final** EditText nullcoorY = (EditText) view.findViewById(R.id.*editY*);

nullcoorX.setText(mDots.getPosition(position).getCorX() + "");

nullcoorY.setText(mDots.getPosition(position).getCorY() + "");

**new** AlertDialog.Builder(**this**).setTitle("Confirm Edit")

.setView(view)

.setPositiveButton("Yes", **new** OnClickListener() {

@Override

**public** **void** onClick(DialogInterface dialog, **int** which) {

**int** corX = Integer.*parseInt*(nullcoorX.getText().toString());

**int** corY = Integer.*parseInt*(nullcoorY.getText().toString());

mDots.edit(position,corX,corY);

}

})

.setNegativeButton("No", **null**).show();

}

ป.ล. กำหนดให้ layout.editdialog เป็น dialog ที่มี edittext เป็นแบบ number และมี 2 ตัว

**Activity Insert  
ป.ล. ไป** mListView.setOnItemClickListener(**this**); **ก่อน**

@Override

**public** **void** onItemClick(AdapterView<?> adapterView, View view, **int** position, **long** id) {

// start new activity

Intent intend = **new** Intent(**this**, SecondActivity.**class**);

intend.putExtra(Const.*CONSTX*, mDots.getPosition(position).getCorX());

intend.putExtra(Const.*CONSTY*, mDots.getPosition(position).getCorY());

//startActivity(intend);

startActivityForResult(intend, 0);

}

@Override

**protected** **void** onActivityResult(**int** requestCode, **int** resultCode, Intent data) {

//super.onActivityResult(requestCode, resultCode, data);

**if**(requestCode==0 && resultCode==0){

Toast.*makeText*(**this**,data.getExtras().getInt(Const.*CONSTX*) + " " + data.getExtras().getInt(Const.*CONSTY*), 0).show();

}

}

@Override

**protected** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*bignum*);

TextView CoorXTxt = (TextView) findViewById(R.id.*txtBigX*);

TextView CoorYTxt = (TextView) findViewById(R.id.*txtBigY*);

Intent intent = getIntent();

**if**(intent != **null** && intent.getExtras() != **null**){

**int** corX = intent.getExtras().getInt("coorX");

**int** corY = intent.getExtras().getInt("coorY");

CoorXTxt.setText(corX + "");

CoorYTxt.setText(corY + "");

}

setResult(0, intent); //

}

ป.ล. **implements** Parcelable หากต้องการให้ส่ง Object อื่นๆไปด้วย

<activity

android:name=*"SecondActivity"*

android:label=*"Second Activity"*

>

</activity>

ต้องสร้าง Activity ใหม่ ที่ AndroidMagnifiest ด้วย

คำสั่ง finish() ใช้ในการปิด Activity

**PhoneGPS**

@Override

**protected** **void** onResume() {

**super**.onResume();

mLocationManager.requestLocationUpdates(LocationManager.*GPS\_PROVIDER*, 1500, 1, mLocate);

}

@Override

**protected** **void** onPause() {

**super**.onPause();

mLocationManager.removeUpdates(mLocate);

}

@Override

**public** **void** onLocationChanged(Location location) {

**this**.location = location;

NotifyGPSChange();

}

เราจะดูค่า la,longtitude จาก location นี่ละ

**Drawing (Custom View)**

สร้าง View พิเศษ

1. Extend view ไป แล้ว override ให้หมด  
   **public** **class** DotsView **extends** View {
2. สร้าง Observer ที่สามารถตรวจดูได้ว่า มีจุดกี่จุด มีจุดอะไรบ้าง  
    **public** **interface** DotsViewDataSource{

Dot getItem(**int** Position);

**int** getCount();

}

**private** DotsViewDataSource dataSource;

**private** Paint paint;

**public** **void** setDataSource(DotsViewDataSource dataSource) {

**this**.dataSource = dataSource;

}

1. สร้าง Observer สำหรับคำสั่ง onTouch  
    **public** **interface** DotsTouchListener{

**void** onDotsTouch(DotsView dotsView, **float** coorX, **float** coorY);

}

DotsTouchListener onDotsTouchListener;

**public** **void** setOnDotsTouchListener(DotsTouchListener onDotsTouchListener) {

**this**.onDotsTouchListener = onDotsTouchListener;

}

1. คำสั่ง onDraw...  
    @Override

**protected** **void** onDraw(Canvas canvas) {

canvas.drawColor(Color.*WHITE*);

**if**(dataSource != **null**){

**for**(**int** position=0; position<dataSource.getCount(); position++){

Dot dot = dataSource.getItem(position);

paint.setColor(Color.*BLUE*);

canvas.drawCircle(dot.getCorX(), dot.getCorY(), 10, paint);

}

}

}

1. ที่ Activity คำสั่ง invalid เอาไว้สำหรับเรียกใช้คำสั่ง onDraw ใหม่ (วาดใหม่ทุกครั้ง ของเดิมลบทิ้ง)  
    @Override

**public** **void** onDotChange(Dots dots) {

// **TODO** Auto-generated method stub

mAdapter.notifyDataSetChanged();

mDotsView.invalidate();

}

1. คำสั่ง onTouch (Override) สำหรับ setOnTouchListener

@Override

**public** **void** onDotsTouch(DotsView dotsView, **float** coorX, **float** coorY) {

mDots.addDot(**new** Dot((**int**) coorX,(**int**) coorY));

}

1. คำสั่งสำหรับตรวจดู View ว่า การ Touch ใน View เป็นแบบไหน แล้วต้องทำอะไร ในที่นี้ ต้องการให้มันไปเรียก Listener

@Override

**public** **boolean** onTouchEvent(MotionEvent event) {

**switch**(event.getActionMasked()){

**case** MotionEvent.*ACTION\_DOWN*:

**if**(**this**.onDotsTouchListener!=**null**){

**this**.onDotsTouchListener.onDotsTouch(**this**, event.getX(), event.getY());

}

}

**return** **super**.onTouchEvent(event);

}

1. จะได้ CustomView ให้ไปวาด ใน Layout หากสร้าง View เสร็จ (ก่อน 5,6 ) ซึ่งก่อน 5 ก็ทำการ set Widget ซะ
2. View สามารถดูความกว้าง ยาวได้ จาก mDotsView.getWidth()

**Savestate**

หากเปลี่ยน orientation แล้ว จะทำให้ข้อมูลหายไป เพราะว่า android จะมีการ Destroy Activity แล้วสร้างใหม่ ดังนั้น วิธีแก้คือ

1. เขียนใน Magnifest ใน Activity (แต่มันคลุมไม่หมดน๊ะ) android:configChanges=*"orientation|keyboardHidden"*
2. สังเกตว่า คำสั่ง onCreate ของ View มี SaveState อยู่ ดังนั้ Override ไป ดังนี้

**if**(savedInstanceState != **null**){

mRPSmodel.SetScore(savedInstanceState.getIntArray("Score"));

txtComp.setText(savedInstanceState.getString("CS"));

txtYour.setText(savedInstanceState.getString("YS"));

}

} //From onCreate

@Override

**protected** **void** onSaveInstanceState(Bundle outState) {

// **TODO** Auto-generated method stub

**super**.onSaveInstanceState(outState);

outState.putIntArray("Score",mRPSmodel.getScoreSet());

outState.putString("CS",(String) txtComp.getText());

outState.putString("YS",(String) txtYour.getText());

}

**Intent : ส่ง Message ข้าม Component**

ใน Activity เรียกว่า intent ที่มีผู้รับแน่นอน : Android มี intent ที่ผู้รับไม่แน่นอนได้ (แล้วใครจะรับละ) <implicit intent> ดูตัวอย่างใน Magnifest ซึ่งมี 3 ส่วนคือ Action, Catagoly, Data

Intent intent = **new** Intent();

intent.setAction(Intent.*ACTION\_VIEW*);

intent.setData(Uri.*parse*("http://www.facebook.com"));

startActivity(intent); //เปิด URL หากเปิด file ก็ file:// ก็ได้

Intent.ACTION จะมีหลายตัว ให้ดูใน Doc เอา

**Database Connection: SQLite**

Class Database  
 1) Cursor : ไว้ loop บน Query (Lazy Loading ถ้าไม่ access ข้อมูล ณ จุดนั้น มันจะไม่ถูกโหลดมาจริงๆ)  
 2) Database Util : จัดการ Escape String (สำหรับ encode)  
 3) อื่นๆ พวก Database Helper

Table  
 \_ID -> Primary Key (เป็น field นี้)

SQLite Open Helper (extends SQLite Open Helper)  
 onOpen : เรียกเมื่อมีระบบสั่ง open Database ครั้งแรก (call, get, database table first time)  
 onCreate : ตอนสร้าง   
 onUpgrade : ตอนทำ Migration

เพื่อความเป็นระเบียบ เราจะแยก onCreate, onUpgrade ให้เป็น static method ในแต่ละ table (ย้ำ table) แล้วก็เรียก static method ใน class ที่ implement มา

**public** **static** **final** String *TABLE\_NAME* = "dot";

**public** **static** **class** DotColumns **implements** BaseColumns {

**public** **static** **final** String *X* = "x";

**public** **static** **final** String *Y* = "y";

}

ทำเพราะในบางครั้ง เราต้องกลับมาดูว่า filed ใน database จริงๆคืออะไร เพื่อจะ query มันออกมา ดังนั้น เพื่อไม่เป็นการเสียเวลา :3

**public** **static** **void** onCreate(SQLiteDatabase db) {

StringBuilder sb = **new** StringBuilder();

sb.append("CREATE TABLE " + DotTable.*TABLE\_NAME* + " (");

sb.append(BaseColumns.*\_ID* + " INTEGER PRIMARY KEY, ");

sb.append(DotColumns.*X* + " INTEGER, ");

sb.append(DotColumns.*Y* + " INTEGER");

sb.append(");");

db.execSQL(sb.toString());

}

BaseColumn ไม่มีอะไรเลย : มีแค่ attribute ที่เป็น \_ID แค่นั้นเอง (จำเป็นในการทำ content provider)

**private** DotOpenHelper openHelper;

**private** SQLiteDatabase db;

**public** Dots(Context context){

mContext = context;

openHelper = **new** DotOpenHelper(mContext);

db = openHelper.getWritableDatabase();

}

อย่าลืม import Android.Content.Context ก่อน

**public** **void** addDot(Dot dot){

DotList.add(dot);

ContentValues Values = **new** ContentValues();

Values.put(DotTable.DotColumns.*X*, dot.getCorX());

Values.put(DotTable.DotColumns.*Y*, dot.getCorY());

db.insert(DotTable.*TABLE\_NAME*, **null**, Values);

NotifyDotChange();

}

**public** **void** addDot(Dot dot){

DotList.add(dot);

ContentValues Values = **new** ContentValues();

Values.put(DotTable.DotColumns.*X*, dot.getCorX());

Values.put(DotTable.DotColumns.*Y*, dot.getCorY());

db.insert(DotTable.*TABLE\_NAME*, **null**, Values);

NotifyDotChange();

}

**public** **void** clearDot(){

DotList.clear();

db.delete(DotTable.*TABLE\_NAME*, **null**, **null**);

NotifyDotChange();

}

**public** **int** getSize(){

//return DotList.size();

//db.query(table, columns, selection, selectionArgs, groupBy, having, orderBy)

Cursor cursor = db.query(DotTable.*TABLE\_NAME*, **null**, **null**, **null**, **null**, **null**, **null**);

**int** Size = cursor.getCount();

cursor.close();

**return** Size;

}

**public** Dot getPosition(**int** index){

//return this.DotList.get(index);

Cursor cursor = db.query(DotTable.*TABLE\_NAME*, **null**, **null**, **null**, **null**, **null**, **null**);

cursor.moveToPosition(index);

**int** corX = cursor.getInt(cursor.getColumnIndex(DotTable.DotColumns.*X*));

**int** corY = cursor.getInt(cursor.getColumnIndex(DotTable.DotColumns.*Y*));

**return** **new** Dot(corX,corY);

}

**Database Connect : Content Provider (เรียกใช้ DB <SQLite> ผ่านนี่)**

Main Action: Insert Update Delete Query

สิ่งที่ต้องรู้ URI (เก็บที่ URI ไหน เก็บ Doc ดูน๊ะ) ซึ่งมีของระบบหลายตัว มีการ getType(Uri) สำหรับเช็คว่าเป็น Item เดี่ยว หรือ ListItem

**Java Shortcut**

**Ctrl + 1 : เอาไว้สำหรับแก้ตัวแดง หรือ ตัวช่วยในด้านอื่นๆ (Extract Constant)  
Ctrl + Space : เอาไว้สำหรับ Auto-Complete  
Source > Generate [Getter Setter | Override]  
Reflector > Extract…**