4134308

การพัฒนาโปรแกรมบนอุปกรณ์เคลื่อนที่ (Mobile Application Development)

ดร.ชูศักดิ์ ยาทองไชย

สาขาวิชาเทคโนโลยีสารสนเทศ มหาวิทยาลัยราชภัฏบุรีรัมย์

1

พื้นฐานเกี่ยวกับอุปกรณ์เคลื่อนที่

- อุปกรณ์เคลื่อนที่ (Mobile Device) คือ อุปกรณ์ประมวลผลแบบพกพา ขนาดเล็ก ส่วนใหญ่สามารถรับข้อมูลผ่านทางจอภาพแบบสัมผัส หรือแป้นพิมพ์ขนาดเล็ก
- อุปกรณ์เคลื่อนที่ ได้แก่
  - โทรศัพท์เคลื่อนที่ สมาร์ตโฟน พีดีเอ (Personal Digital Assistant: PDA) แท็บเลต (Tablet)

2

พื้นฐานเกี่ยวกับอุปกรณ์เคลื่อนที่

- จุดเด่นของอุปกรณ์เคลื่อนที่
  1. ความละเอียดของการแสดงผล
  2. ขนาดเล็ก น้ำหนักเบา พกพาสะดวก
  3. วิธีการนำเข้าข้อมูลทำได้ง่าย หลากหลายรูปแบบ
  4. บอกตำแหน่งทางภูมิศาสตร์ได้
  5. โทรศัพท์ติดต่อสื่อสารได้
  6. ติดต่อสื่อสารผ่านทางบริการรับส่งข้อความได้
  7. ติดต่อสื่อสารผ่านทางบริการมัลติมีเดียได้
  8. ติดตั้ง แอปพลิเคชันสำหรับใช้งานเพิ่มเติมได้
  9. บันทึกเสียง ถ่ายภาพ ถ่ายวีดิโอได้

  ฯลฯ

3

พื้นฐานเกี่ยวกับอุปกรณ์เคลื่อนที่

- แพลตฟอร์มของอุปกรณ์เคลื่อนที่
  1. Android OS
  2. iOS
  3. Microsoft Windows

4

พื้นฐานเกี่ยวกับอุปกรณ์เคลื่อนที่

- Framework สำหรับพัฒนาโปรแกรมบนอุปกรณ์เคลื่อนที่
  - Specific Framework สำหรับพัฒนาแพลตฟอร์มใด ๆ โดยเฉพาะ
    1. Android ใช้ภาษา Kotlin, Java, C++
    2. iOS ใช้ภาษา Swift, Objective-C
    3. Microsoft Windows ใช้ภาษา C#.NET
  - Cross Platform Framework สำหรับพัฒนาให้ใช้ได้หลายแพลตฟอร์ม
    1. Ionic ใช้ภาษาสำหรับพัฒนาแอปพลิเคชันบนเว็บ เช่น HTML5, CSS, JavaScript และ Angular JS Framework ส่วนการควบคุมทรัพยากร เช่น Camera, GPS, และ Audio Recorder จะใช้ Cordova plugins
    2. React Native พัฒนาโดย Facebook ใช้ภาษา Javascript และ React.JS
    3. Flutter พัฒนาโดย Google ใช้ภาษา Dart
    4. Xamarin พัฒนาโดย Microsoft ใช้ภาษา C#

5

การพัฒนาโปรแกรมบน Android

6

แอนดรอยด์ (Android) คืออะไร

แอนดรอยด์ เป็นระบบปฏิบัติการสำหรับอุปกรณ์พกพา เช่น โทรศัพท์มือถือ แท็บเลต ทำงานบนลีนุกซ์เคอร์เนล พัฒนาโดยบริษัท Android Inc. ถูกซื้อโดย Google และเปิดให้นักพัฒนาสามารถแก้ไขโค้ดต่าง ๆ ด้วยภาษาจาวา และควบคุมอุปกรณ์ผ่านทางชุด Java libraries ที่ Google พัฒนาขึ้น

7

Android versions

Name	Version	release date	Security fixes	API level
No official codename	1.0 - 1.1	September 23, 2008	No	1-2
Cupcake	1.5	April 27, 2009	No	3
Donut	1.6	September 15, 2009	No	4
Eclair	2.0 - 2.1	October 26, 2009	No	5-7
Froyo	2.2 - 2.2.3	May 20, 2010	No	8
Gingerbread	2.3 - 2.3.7	December 6, 2010	No	9-10
Honeycomb	3.0 – 3.2.6	February 22, 2011	No	11-13
Ice Cream Sandwich	4.0 – 4.0.4	October 18, 2011	No	14-15
Jelly Bean	4.1 – 4.3.1	July 9, 2012	No	16-18
KitKat	4.4 – 4.4.4	October 31, 2013	No	19-20
Lollipop	5.0 – 5.1.1	November 12, 2014	No	21-22
Marshmallow	6.0 – 6.0.1	October 5, 2015	No	23
Nougat	7.0 – 7.1.2	August 22, 2016	No	24-25
Oreo	8.0 – 8.1	August 21, 2017	Yes	26-27
Pie	9	August 6, 2018	Yes	28
Android 10	10	September 3, 2019	Yes	29
Android 11	11	September 8, 2020	Yes	30

8

สถาปัตยกรรมของระบบปฏิบัติการแอนดรอยด์

- การทำงานของแอนดรอยด์มีพื้นฐานอยู่บนระบบลีนุกซ์เคอร์เนล ซึ่งใช้ Android SDK เป็นเครื่องมือสำหรับการพัฒนาแอปพลิเคชัน และใช้ภาษาจาวาในการพัฒนา
- สถาปัตยกรรมของแอนดรอยด์ ถูกแบ่งออกเป็นลำดับชั้นมี 4 ชั้นหลัก
  1. ชั้นแอปพลิเคชัน (Application) เป็นส่วนแอปพลิเคชันที่พัฒนาขึ้นมาใช้งาน เช่น แอปพลิเคชันรับส่งอีเมล์ SMS ปฏิทิน แผนที่ เว็บบราวเซอร์ รายชื่อผู้ติดต่อ เป็นต้น ซึ่งแอปพลิเคชันจะอยู่ในรูปแบบของไฟล์ .apk โดยทั่วไปแล้วจะอยู่ในไดเร็กทอรี data/app
  2. ชั้นแอปพลิเคชันเฟรมเวิร์ก (Application Framework) จะอนุญาตให้นักพัฒนาสามารถเข้าเรียกใช้งานโดยผ่าน API ซึ่งแอนดรอยด์ได้ออกแบบไว้ เพื่อลดความซ้ำซ้อนในการใช้งานแอปพลิเคชันคอมโพเนนต์
  3. ชั้นไลบรารี (Library) แอนดรอยด์ได้รวบรวมกลุ่มของไลบรารีต่าง ๆ ที่สำคัญและมีความจำเป็นเอาไว้มากมาย เพื่ออำนวยความสะดวกให้กับนักพัฒนาและง่ายต่อการพัฒนาโปรแกรม
  4. ชั้นลีนุกซ์เคอร์เนล (Linux Kernel) ระบบแอนดรอยด์ถูกสร้างบนพื้นฐานของระบบปฏิบัติการลีนุกซ์ ในชั้นนี้จะมีฟังก์ชันการทำงานหลาย ๆ ส่วน แต่โดยส่วนมากแล้วจะเกี่ยวข้องกับฮาร์ดแวร์โดยตรง เช่น การจัดการหน่วยความจำ (Memory Management) การจัดการโพรเซส การเชื่อมต่อเครือข่าย (Networking) เป็นต้น

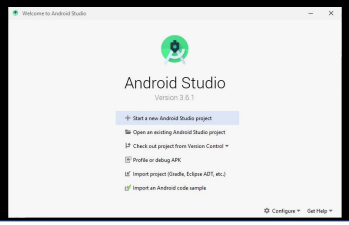
สถาปัตยกรรมของระบบปฏิบัติการแอนดรอยด์

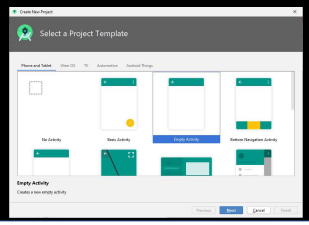
การติดตั้งเครื่องมือที่ใช้ในการพัฒนา

1. ติดตั้ง JDK (Java Development Kit) เพื่อให้เครื่องมีจาวาคอมไพเลอร์ รวมถึงไลบรารีต่าง ๆ ที่จะถูกเรียกใช้โดย Android SDK และแอปพลิเคชันของเรา
2. ติดตั้ง Android Studio ซึ่งรวมเครื่องมือในการพัฒนาแอปพลิเคชันบนระบบปฏิบัติการแอนดรอยด์เอาไว้ด้วยกัน
3. เลือก Android SDK ที่ต้องการพัฒนา
   1. คลิก Tools -> SDK Manager
   2. เลือก Version และ API Level ตามต้องการ
   3. คลิก OK หรือ Apply
4. สร้าง Emulator เพื่อทดสอบแอปพลิเคชันใน Android Studio
   1. คลิก Tools -> AVD Manager -> Create Virtual Device...
   2. เลือกประเภทอุปกรณ์ ขนาดจอภาพ ความละเอียดตามต้องการ
   3. เลือก Version และ API Level ตามต้องการ
   4. ตั้งค่าการทำงานเบื้องต้น
   5. คลิก Finish

การสร้าง Project ใหม่

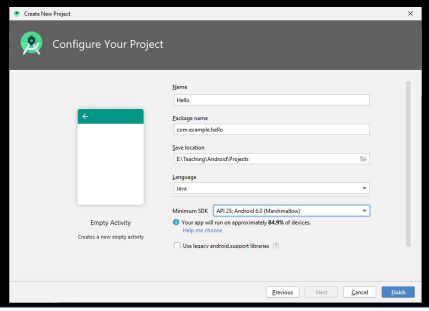
- จะต้องสร้าง 1 Android project ต่อ 1 Application
- ขั้นตอนการสร้าง Project ใหม่
  1. เปิด Android Studio
  2. คลิกที่ +Start a new Android Studio project
  3. เลือก Template ตามต้องการ ในที่นี้ให้เลือก Empty Activity แล้วคลิก Next





การสร้าง Project ใหม่

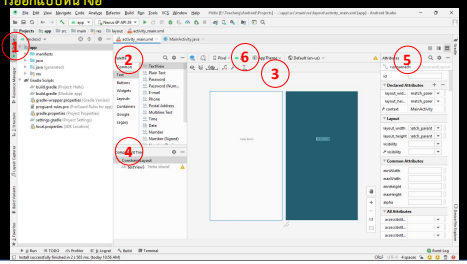
- กำหนดค่า Project โดยระบุ Name, Package name, Save location, Language และ Minimum SDK แล้วคลิก Finish



13

สภาพแวดล้อมของ Android Studio

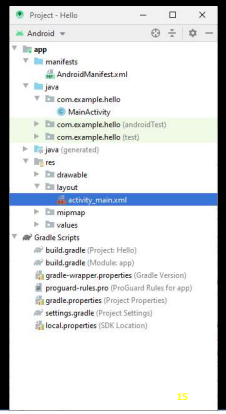
1. แท็บ Project ทำหน้าที่แสดงโครงสร้างรายการไฟล์ทั้งหมดของ Project
2. หน้าต่าง Palette หรือ Toolbox เป็นหน้าต่างเครื่องมือ (View และ Widget) สำหรับใช้ออกแบบส่วนติดต่อกับผู้ใช้
3. ส่วนออกแบบและแสดงผลหน้าจอ มี 3 โหมด คือ Code, Split และ Design (หน้าจอ Design มี 2 ส่วน คือ Design และ Blueprint)
4. หน้าต่าง Component Tree แสดงโครงสร้างของส่วนประกอบต่าง ๆ ที่ได้ออกแบบไว้
5. หน้าต่าง Attributes หรือ Properties สำหรับกำหนดค่า Attribute ของ View และ Widget
6. แถบเครื่องมือควบคุมการออกแบบหน้าจอ



14

โฟลเดอร์ และไฟล์ของ Android Project

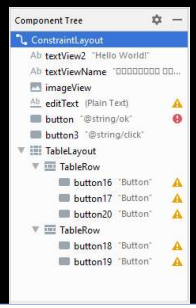
- app/manifests/AndroidManifest.xml เป็นไฟล์ตั้งค่าในการทำงานทั้งหมดของแอปพลิเคชัน
- app/java สำหรับเก็บ java source code ของแต่ละโปรเจกต์
- app/java (generated) เก็บไฟล์ java ที่ถูกสร้างขึ้นมาโดยอัตโนมัติเพื่ออ้างอิงค่าในไฟล์ .xml
- app/res/drawable เก็บไฟล์รูปภาพที่ใช้ในแอปพลิเคชัน
- app/res/layout เก็บไฟล์ที่ใช้ในการออกแบบหน้าจอของแอปพลิเคชัน
- App/res/menu เก็บไฟล์ที่กำหนดเมนูของแอปพลิเคชัน
- app/res/mipmap เก็บไฟล์ไอคอนของแอปพลิเคชัน
- app/res/values เก็บไฟล์ที่เป็นค่าคงที่ เช่น ข้อความ เป็นต้น
- Gradle Scripts เป็นภาษาสคริปต์ที่ใช้ตั้งค่าโปรเจกต์ของแอนดรอยด์ Gradle ของโปรเจกต์ปัจจุบันอยู่ที่ build.gradle (Module: app)

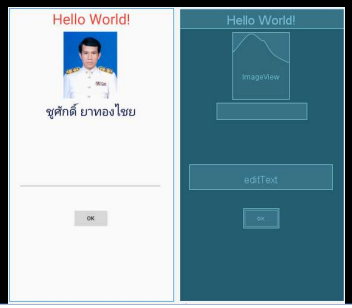


15

การสร้างส่วนติดต่อกับผู้ใช้

- ลาก Widget ประเภท Layout ที่ต้องการมาวางบนหน้าจอ Design หรือ Blueprint (Layout เริ่มต้นจะเป็น ConstraintLayout) จากนั้นกำหนดขนาดของ Layout ตามต้องการ
- ลาก Widget ที่ต้องการมาวางบน Layout ตามตำแหน่งที่ต้องการ หรือวางที่หน้าต่าง Component Tree
- กำหนดค่า Attribute ตามต้องการจากหน้าต่าง Attributes





16

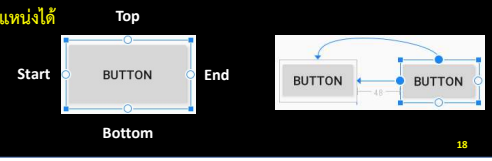
Layout

- เป็นเค้าโครงสำหรับควบคุมการจัดวางตำแหน่ง Widget ที่อยู่ภายใน เพื่อออกแบบส่วนติดต่อกับผู้ใช้ โดยสามารถใช้ Layout วางซ้อนกันภายในได้
- Layout ที่นิยมใช้
  1. Constraint Layout
  2. Linear Layout
  3. Frame Layout
  4. Table Layout

17

Constraint Layout

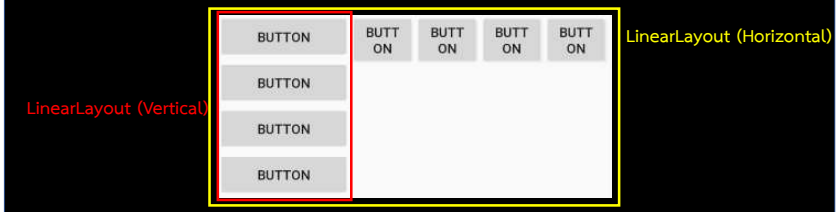
- เป็นค่าเริ่มต้น Layout ของแอนดรอยด์ จัดตำแหน่งของ Widget ง่าย
- ควบคุมการจัดวาง Widget ภายในให้อยู่ในตำแหน่งอ้างอิงตามต้องการ โดยอ้างอิงตำแหน่งกับ Constraint Layout เอง หรือ Widget อื่นที่อยู่ภายในด้วยกัน
- ต้องกำหนดตำแหน่งอ้างอิงอย่างน้อย 2 ตำแหน่ง (แนวตั้ง และแนวนอน) โดยลากจุดอ้างอิงไปยังขอบ Layout หรือจุดอ้างอิงตำแหน่งของ Widget ที่ต้องการ
- แต่ละ Widget จะมีจุดอ้างอิงตำแหน่ง 4 จุด คือ Top, Bottom, Start, End
- เมื่อมีการเปลี่ยนตำแหน่งของ Widget ที่ใช้อ้างอิงตำแหน่ง ตำแหน่งของ Widget ที่อ้างอิงจะเปลี่ยนตาม
- สามารถใช้ Guideline เพื่อจัดตำแหน่งได้



18

Linear Layout

- มีการจัดเรียง Widget ที่อยู่ภายในเป็นแบบเส้นตรงตำลับดับ เช่น เรียงในแบบตามแนวตั้ง (Vertical) หรือเรียงในแบบตามแนวนอน (Horizontal)



19

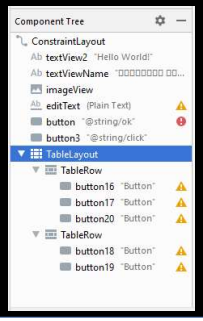
Frame Layout

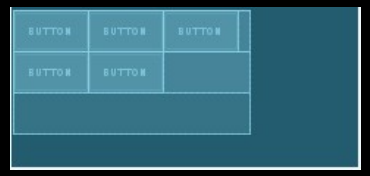
- Widget จะถูกจัดวางซ้อนทับกันเป็นชั้น ๆ ส่วนใหญ่จะใช้จัดวางเพียง Widget เดียว

20

Table Layout

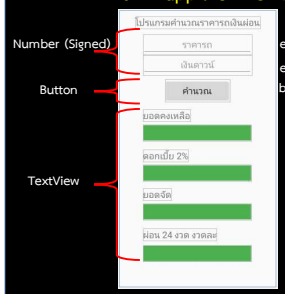
- มีการจัด Widget เป็นแบบตารางโดยที่ Widget แต่ละตัวถูกจัดเป็น 1 คอลัมน์ นอกจากนี้ยังสามารถเพิ่มแถวได้โดยการเพิ่ม TableRow

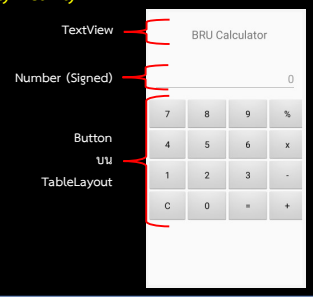




Workshop

- ออกแบบจอภาพ Activity คำนวณราคารถเงินผ่อน (HirePurchase)
- ออกแบบจอภาพ Activity เครื่องคิดเลข (Calculator)
- คลิกขวาที่ app เลือก New->Activity->Empty Activity





กำหนดชื่อใน HirePurchase Activity

id ใน Design	ชื่อ Object ใน java
editPrice	edtPrice
editDown	edtDown
buttonCompute	btnCompute
textViewRemain	textRemain
textViewInterest	textInterest
textViewAmount	textAmount
textViewPerMonth	textPerMonth

เขียนคำสั่งควบคุม HirePurchase Activity

```
public class HirePurchase extends AppCompatActivity {
  EditText edtPrice;
  EditText edtDown;
  TextView textRemain;
  TextView textInterest;
  TextView textAmount;
  TextView textPerMonth;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    ActionBar actionBar = getSupportActionBar();
    actionBar.hide();
    setContentView(R.layout.activity_hire_purchase);
    edtPrice  = findViewById(R.id.editPrice);
    edtDown   = findViewById(R.id.editDown);
    Button btnCompute = findViewById(R.id.buttonCompute);
    textRemain = findViewById(R.id.textViewRemain);
    textInterest = findViewById(R.id.textViewInterest);
    textAmount = findViewById(R.id.textViewAmount);
    textPerMonth = findViewById(R.id.textViewPerMonth);
    if (btnCompute != null) {
      btnCompute.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {
        Integer remain = Integer.parseInt(edtPrice.getText().toString()) - Integer.parseInt(edtDown.getText().toString());
        textRemain.setText(remain.toString());
      }
    });
   }
  }
}
```

พื้นฐานแอปพลิเคชัน

- Android SDK จะรวบรวมโค้ดพร้อมกับไฟล์ข้อมูลและทรัพยากรใด ๆ ลงใน APK ซึ่งเป็นแพ็กเกจ Android มีนามสกุล .apk
- ไฟล์ APK หนึ่งไฟล์จะเก็บเนื้อหาทั้งหมดของแอป Android และเป็นไฟล์ที่อุปกรณ์ในระบบ Android ใช้ติดตั้งแอป
- ระบบปฏิบัติการ Android เป็นระบบ Linux แบบ multiuser ซึ่งแต่ละแอปจะเป็นผู้ใช้ที่แตกต่างกัน
- ค่าเริ่มต้นระบบจะกำหนด ID ผู้ใช้ลินุกซ์ที่ไม่ซ้ำกันให้แต่ละแอป และตั้งค่าสิทธิ์สำหรับไฟล์ทั้งหมดในแอปเพื่อให้เฉพาะ ID ผู้ใช้ที่กำหนดให้กับแอปนั้นเท่านั้นที่สามารถเข้าถึงได้
- แต่ละโปรเซสจะมีเครื่องเสมือน (Virtual Machine: VM) ของตัวเอง ดังนั้นโค้ดของแอปจึงทำงานแยกจากแอปอื่น ๆ
- ค่าเริ่มต้นทุกแอปจะทำงานในโปรเซสลินุกซ์ของตัวเอง ระบบ Android จะเริ่มรันโปรเซสเมื่อต้องเรียกใช้ Component ใด ๆ ของแอป จากนั้นจะปิดโปรเซสเมื่อไม่จำเป็นต้องใช้ หรือเมื่อระบบต้องกู้คืนหน่วยความจำสำหรับแอปอื่น ๆ
- ระบบ Android ใช้หลักการของการให้สิทธิ์แบบน้อยที่สุด ค่าเริ่มต้นแต่ละแอปจะมีสิทธิ์เข้าถึงเฉพาะ Component ที่ต้องใช้ในการทำงานเท่านั้น ทำให้เกิดสภาพแวดล้อมที่ปลอดภัยมากซึ่งแอปไม่สามารถเข้าถึงบางส่วนของระบบที่ไม่ได้รับอนุญาต ซึ่งมีหลายวิธีสำหรับแอปในการแชร์ข้อมูลกับแอปอื่น ๆ และเพื่อให้แอปเข้าถึงบริการของระบบ
- แต่ละแอปสามารถแชร์ ID ผู้ใช้ลินุกซ์เดียวกัน ทำให้สามารถเข้าถึงไฟล์ของกันและกันได้ เพื่อประหยัดทรัพยากร ระบบแอปที่มี ID ผู้ใช้เดียวกันยังสามารถรันในโปรเซสเดียวกันและแชร์ VM เดียวกันได้ โดยแอปจะต้องมีใบรับรอง (Certificate) เดียวกัน
- แอปสามารถขออนุญาตเข้าถึงข้อมูลอุปกรณ์ เช่น ตำแหน่ง กล้อง และการเชื่อมต่อบลูทูธ โดยผู้ใช้ต้องให้สิทธิ์

25

ส่วนประกอบของแอป (App components)

- ส่วนประกอบของแอป มี 4 ประเภท แต่ละประเภทมีจุดประสงค์การใช้และมีวงจรชีวิตแตกต่างกัน
  1. Activities
  2. Services
  3. Broadcast receivers
  4. Content providers

26

Activities

- เป็นส่วนติดต่อกับผู้ใช้ แสดงถึงหนึ่งหน้าจอของแอป เช่น แอปอีเมลอาจมีแต่ละ Activity สำหรับแสดงอีเมลใหม่ เขียนอีเมล และอ่านอีเมล
- Activities จะทำงานร่วมกัน และสอดคล้องกัน แต่จะไม่ขึ้นต่อกัน ดังนั้นแอปอื่นจึงสามารถเรียกใช้ Activities ของแอปอีเมลได้หากแอปอีเมลอนุญาต เช่น แอปกล้องถ่ายรูปสามารถรัน Activity เขียนจดหมาย ในแอปอีเมล เพื่อให้ผู้ใช้แชร์รูปภาพได้
- Activities จะอำนวยความสะดวกในการโต้ตอบระหว่างระบบและแอป ดังนี้
  1. ติดตามสิ่งที่ผู้ใช้ให้ความสำคัญในปัจจุบัน (สิ่งที่อยู่บนหน้าจอ) เพื่อให้แน่ใจว่าระบบยังคงรันโปรเซสของ Activity นั้นอยู่
  2. การรู้ว่า Activity ที่ใช้ก่อนหน้านี้ (Activity ที่หยุดทำงานแล้ว) มีสิ่งที่ผู้ใช้อาจถอยกลับไป จึงให้ความสำคัญกับการรักษาโปรเซสเหล่านั้นไว้
  3. ช่วยให้แอปจัดการกับ Activity ที่ถูกยกเลิก เพื่อให้ผู้ใช้กลับไปทำ Activity นั้นได้ โดยคืนค่าสถานะเดิมก่อนหน้านี้
  4. หาทางให้แอปสร้างและประสานการทำงานของผู้ใช้ระหว่างแอปและระบบ

27

Services

- เป็นแอปที่ทำงานอยู่เบื้องหลัง ซึ่งรันอยู่ในระบบตลอดเวลา
- Services ไม่มีส่วนติดต่อกับผู้ใช้ เช่น service อาจเล่นเพลงขณะที่ผู้ใช้อยู่ในแอปอื่น หรืออาจดึงข้อมูลผ่านเครือข่ายโดยไม่บล็อกการโต้ตอบของผู้ใช้กับ Activity

28

Broadcast receivers

- เป็นส่วนที่ช่วยให้ระบบส่งเหตุการณ์ไปยังแอปที่นอกเหนือจากที่ผู้ใช้ใช้งานปกติ ทำให้แอปตอบสนองต่อ broadcast ของระบบ เช่น แอปสามารถตั้งเวลาปลุกเพื่อโพสต์การแจ้งเตือนผู้ใช้เกี่ยวกับเหตุการณ์ที่กำลังจะเกิดขึ้น การส่งสัญญาณเตือนไปยัง Broadcast receiver ของแอปไม่จำเป็นต้องให้แอปทำงานต่อไปจนกว่านาฬิกาปลุกจะหยุด
- หลาย ๆ broadcast มาจากระบบ เช่น broadcast ที่บอกว่าหน้าจอปิดอยู่ แบตเตอรี่เหลือน้อย หรือมีการจับภาพ
- แอปสามารถเริ่มการทำงานของ broadcast ได้ เช่น แจ้งให้แอปอื่นทราบว่ามีการดาวน์โหลดข้อมูลบางส่วนไปยังอุปกรณ์และพร้อมให้ใช้งานได้
- Broadcast receiver จะไม่แสดงส่วนติดต่อกับผู้ใช้ แต่อาจสร้างการแจ้งเตือนแถบสถานะเพื่อแจ้งเตือนผู้ใช้เมื่อเกิดเหตุการณ์ broadcast
- โดยทั่วไปแล้ว broadcast เป็นเพียงเกตเวย์เชื่อมไปยัง Component อื่น ๆ และมีจุดมุ่งหมายเพื่อทำงานในปริมาณที่น้อยที่สุด เช่น อาจกำหนดเวลาให้ JobService ทำงานบางอย่างตามเหตุการณ์ด้วย JobScheduler

29

Content providers

- เป็นส่วนจัดการชุดข้อมูลแอปที่ใช้ร่วมกัน ซึ่งสามารถจัดเก็บในระบบไฟล์ในฐานข้อมูล SQLite บนเว็บ หรือบนที่เก็บข้อมูลถาวรอื่น ๆ
- แอปอื่น ๆ สามารถสืบค้นหรือแก้ไขข้อมูลผ่านทาง Content provider ได้ หาก Content provider อนุญาต เช่น ระบบ Android มี Content provider ที่จัดการข้อมูลติดต่อของผู้ใช้ คือ ContactsContract.Data เพื่ออ่านและเขียนข้อมูลเกี่ยวกับบุคคลใดบุคคลหนึ่ง
- Content provider เป็นจุดเชื่อมโยงในแอปสำหรับ publish ชื่อรายการข้อมูล โดยการระบุ URI และแอปสามารถตัดสินใจได้ว่าต้องการเชื่อมข้อมูลที่มีอยู่กับเนมสเปซของ URI อย่างไร โดยส่ง URI นั้นให้กับเอนทิตีอื่น ซึ่งสามารถใช้เข้าถึงข้อมูลได้ โดยระบบมีวิธีจัดการแอป ดังนี้
  1. การกำหนด URI ไม่จำเป็นต้องให้แอปทำงาน ดังนั้น URI จะคงอยู่ได้หลังจากออกจากแอปของตนเองแล้ว ระบบต้องตรวจสอบให้แน่ใจว่าแอปที่เป็นเจ้าของยังคงทำงานอยู่เมื่อต้องดึงข้อมูลของแอปจาก URI ที่เกี่ยวข้อง
  2. URI มีรูปแบบการรักษาความปลอดภัย เช่น แอปสามารถใส่ URI สำหรับรูปภาพที่มีในคลิปบอร์ดได้ แต่ล็อคไว้เพื่อไม่ให้แอปอื่นเข้าถึงได้อย่างอิสระ เมื่อแอปที่สองพยายามเข้าถึง URI นั้นบนคลิปบอร์ดระบบจะอนุญาตให้แอปนั้นเข้าถึงข้อมูลผ่าน URI แบบชั่วคราวเท่านั้น
- Content provider ยังมีประโยชน์สำหรับการอ่านและเขียนข้อมูลแบบ private ในแอป และไม่ต้องแชร์

30

การทำงานของระบบ Android

- ลักษณะเฉพาะของการออกแบบระบบ Android คือ แอปใด ๆ สามารถเรียกใช้ Component ของแอปอื่นได้ เช่น หากต้องการให้ผู้ใช้ถ่ายภาพด้วยกล้องของอุปกรณ์ อาจมีแอปอื่นที่ทำอยู่แล้ว เราสามารถใช้แอปนี้ได้เลยแทนการพัฒนา Activity เพื่อถ่ายภาพใหม่ โดยไม่ต้องรวมหรือเชื่อมโยงไปยังแอปกล้องถ่ายรูป แต่สามารถเรียกใช้ Activity ในแอปกล้องถ่ายรูปได้ เมื่อทำงานเสร็จรูปภาพจะถูกส่งกลับไปยังแอปของเรา โดยผู้ใช้จะรู้สึกว่ากล้องเป็นส่วนหนึ่งของแอปเรา
- เมื่อระบบเริ่มรัน Component หากโปรเซสยังไม่ได้ทำงานจะมีการสร้างอินสแตนซ์ของคลาสที่จำเป็นสำหรับ Component นั้น เช่น หากแอปเราเรียกใช้ Activity ในแอปกล้องถ่ายรูป Activity นั้นจะทำงานในโปรเซสที่เป็นของแอปกล้องถ่ายรูปไม่ใช่ในโปรเซสของแอปเรา จึงไม่เหมือนกับแอปในระบบอื่น
- แอป Android จะไม่มีจุดเริ่มต้นเพียงจุดเดียว คือ ไม่มีฟังก์ชัน main ()
- เนื่องจากระบบรันแต่ละแอปโดยแยกโปรเซสกัน และจำกัดสิทธิ์ของไฟล์ที่จะเข้าถึงโดยแอปอื่น แอปของเราจึงไม่สามารถเปิดใช้งาน Component จากแอปอื่นได้ แต่ระบบ Android สามารถเปิดใช้งาน Component ในแอปอื่นให้ส่งข้อความไปยังระบบที่ต้องการเรียกใช้ Component เฉพาะ จากนั้นระบบจะเปิดใช้งาน Component นั้นให้

31

การเรียกใช้ Component

- Activities, Services, และ Broadcast receivers จะถูกเรียกใช้ผ่าน message แบบ asynchronous เรียกว่า intent
- intent จะถูกสร้างโดย Intent อ็อบเจกต์ ซึ่งจะกำหนด message เพื่อเปิดใช้งาน Component
- intent ของ Activities และ Services จะต้องกำหนดการกระทำ เช่น ดู หรือส่งข้อมูล และอาจระบุ URI ของข้อมูลที่จะดำเนินการ รวมถึงสิ่งอื่น ๆ ที่ Component ที่กำลังเริ่มรันอาจจำเป็นต้องรู้ เช่น intent อาจร้องขอให้ Activity แสดงรูปภาพหรือเปิดหน้าเว็บเพจ ในบางกรณีสามารถเรียกใช้ Activity เพื่อรับผลลัพธ์ซึ่ง Activity จะส่งคืนผลลัพธ์ใน intent ด้วย
- intent ของ Broadcast receivers จะกำหนดแค่ broadcast เท่านั้น เช่น broadcast เพื่อระบุว่าแบตเตอรี่ของอุปกรณ์เหลือน้อย จะมีเฉพาะข้อความที่ระบุว่า แบตเตอรี่เหลือน้อย
- Content providers จะไม่ถูกเรียกใช้โดย intents แต่จะถูกเรียกใช้งานเมื่อกำหนด target โดยส่ง request จาก ContentResolver

32

การเรียกใช้ Component

- เมธอดสำหรับเรียกใช้ intent แต่ละประเภท ได้แก่
  - การเรียกใช้ หรือส่งบางอย่างให้ Activity ทำได้โดยส่งผ่าน Intent โดยเรียกเมธอด startActivity() หรือ startActivityForResult () เมื่อต้องการให้ส่งคืนผลลัพธ์
  - ใน Android 5.0 (API level 21) ขึ้นไป สามารถใช้คลาส JobScheduler เพื่อกำหนดเวลาดำเนินการ สามารถผูกกับบริการได้โดยเรียก Intent เมธอด bindService ()
  - สามารถเรียกใช้ broadcast ได้โดยเรียก Intent เมธอด เช่น sendBroadcast(), sendOrperedBroadcast() หรือ sendStickyBroadcast()
  - สามารถสอบถามไปยัง Content provider ได้ โดยเรียกใช้เมธอด query() บน ContentResolver

33

การทำงานกับ Activities

- คลาส Activity เป็นองค์ประกอบที่สำคัญของแอปแอนดรอยด์
- หลักการทำงานของ Activity คือ launched and put together ซึ่งเป็นพื้นฐานของแพลตฟอร์มแอปพลิเคชันโมเดล
- การเขียนโปรแกรมทั่วไปเริ่มด้วยเมธอด main() แต่ระบบแอนดรอยด์จะเริ่มต้นโค้ดในอินสแตนซ์ของ Activity โดยเรียกใช้ callback method เฉพาะที่สอดคล้องกับขั้นตอนเฉพาะของวงจรชีวิตของ Activity
- การใช้งานแอปบนอุปกรณ์เคลื่อนที่แตกต่างจากเดสก์ท็อปตรงที่การโต้ตอบของผู้ใช้กับแอปไม่ได้เริ่มต้นในที่เดียวกันเสมอไป แต่ผู้ใช้มักจะเริ่มต้นโดยไม่ได้ตั้งใจ เช่น หากเปิดแอปอีเมลจากหน้าจอหลัก อาจเห็นรายการอีเมล ในทางตรงกันข้ามหากใช้แอปโซเชียลมีเดียที่เปิดแอปอีเมลขึ้นมา อาจไปที่หน้าจอของแอปอีเมลโดยตรงเพื่อเขียนอีเมล
- ในช่วงเวลาหนึ่ง ๆ ในการทำงานของแอปจะมีเพียง Activity เดียวเท่านั้นที่พร้อมมีปฏิสัมพันธ์กับผู้ใช้ นั่นคือเมื่อมี Activity ใด ๆ เริ่มทำงานขึ้นมาใหม่ จะทำให้ Activity เดิมที่กำลังทำงานอยู่เปลี่ยนสถานะ (State) ของตัวเองไป ซึ่งสถานะหลัก ๆ ของ Activity ได้แก่ Create, Start, Resume, Pause, Stop, และ Destroy เพื่อควบคุมการทำงานของ Activity ให้เหมาะสมกับสถานะที่เปลี่ยนไป เราต้องเขียนคำสั่งไว้ภายใน LifeCycle Callback Method ต่าง ๆ เพื่อควบคุมการทำงานของแต่ละสถานะนั้น ๆ เช่น onCreate() เป็น Method ที่จะถูกทำงานเมื่อ Activity เปลี่ยนสถานะเป็น Create, onStart() เป็น Method ที่จะถูกทำงานเมื่อ Activity เปลี่ยนสถานะมาเป็น Start เป็นต้น

34

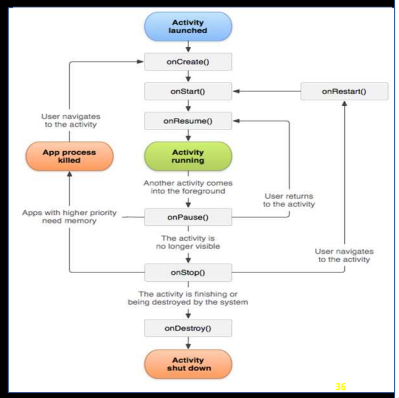
การทำงานกับ Activities

- ขณะที่ผู้ใช้เปิดใช้งานแอปแล้วออกจากแอปแล้วก็กลับเข้ามาในแอปอีกครั้ง Activity จะมีการย้ายสถานะ (state) ต่าง ๆ เกิดขึ้นในวงจรชีวิตของ Activity เช่น เมื่อ Activity เริ่มทำงานครั้งแรก ก็จะแสดงขึ้นมาอยู่ด้านบนสุดของระบบ และรอรับการทำงานจากผู้ใช้ ในระหว่างนี้ระบบจะมีการเรียกใช้งาน callback method ต่าง ๆ (method ที่ถูกเรียกใช้งานอัตโนมัติ) ใน Activity นั้น
- ถ้าผู้ใช้ใช้งานใด ๆ ที่เป็นการเรียก Activity อื่นขึ้นมา หรือสลับไปใช้งานแอปอื่น ระบบจะเรียก callback method อีกอันขึ้นมาและซ่อนแอปไว้ด้านหลัง คือไม่แสดง Activity แต่ instance และ state ยังทำงานอยู่
- ภายใน callback method เราสามารถกำหนดการทำงานใน Activity เมื่อผู้ใช้ออกจากแอป และกลับเข้ามาใช้งานแอปใหม่อีกครั้งได้ เช่น ถ้าเป็นแอป streaming video อาจจะสั่งให้หยุด video ชั่วคราว และปิดการเชื่อมต่อ network ไว้ก่อนเมื่อผู้ใช้สลับไปใช้แอปอื่น และทันทีที่ผู้ใช้กลับมาใช้งานแอปต่อ ก็ให้เชื่อมต่อกับ network และอนุญาตให้ผู้ใช้กลับไปเล่น videoใ นตำแหน่งที่ค้างต่อไปทันทีได้โดยไม่ต้องเริ่มต้นใหม่ เป็นต้น
- ในขณะที่ Activity ทำงาน ระบบจะเรียกใช้ callback method ตามลำดับในลักษณะที่คล้ายกับการก่อพีระมิด เช่น เมื่อระบบสร้าง instance ของ Activity ขึ้นมาใหม่ callback method ก็จะขยับ state ขึ้นมาด้านบนพีระมิดคือจุดที่ Activity กำลังทำงานแสดงอยู่ด้านหน้าสุดและผู้ใช้กำลังใช้งานอยู่ และเมื่อผู้ใช้กำลังจะออกจาก Activity ระบบจะก็เรียกใช้ method อื่น ซึ่งก็ทำให้ state ถอยกลับไปอยู่ด้านล่างของพีระมิดตามลำดับเพื่อค่อย ๆ ถอน Activity ออกไป ในบางกรณี Activity จะย้ายลงมาอยู่บางจุดและรอจังหวะที่จะถูกเรียกกลับขึ้นมาด้านบนอีก (กรณีเช่นเมื่อผู้ใช้สลับไปใช้งานแอปอื่น แล้วกลับมาใช้งาน)

35

การทำงานกับ Activities

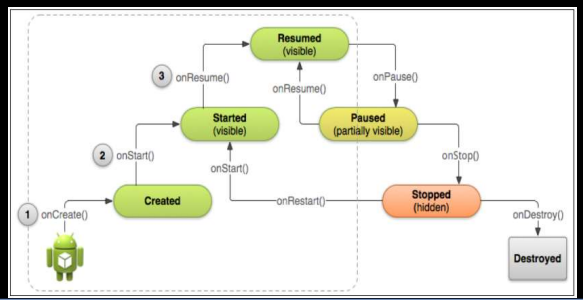
- Activity life cycle ประกอบไปด้วย 6 state ซึ่งจะมี callback method ที่จะถูกเรียกใช้ในระหว่างที่มีการเปลี่ยน state ดังนี้
  1. onCreate() ถูกเรียกใช้เมื่อเริ่มสร้าง Activity
  2. onStart() ถูกเรียกใช้เมื่อ Activity ปรากฏต่อ user
  3. onResume() ถูกเรียกใช้เมื่อ Activity มีปฏิสัมพันธ์ กับ user (สถานะนี้ application ถูกใช้งานแล้ว)
  4. onPause() ถูกเรียกใช้เมื่อ Activity ไม่สามารถมีปฏิสัมพันธ์กับ user หรือไม่ได้ execute code ใด ๆ ซึ่งเกิดจาก Activity ก่อนหน้าโดนเรียกกลับมาใช้งาน (resume)
  5. onStop() ถูกเรียกใช้เมื่อ Activity ไม่ปรากฏต่อ user
  6. onRestart() ถูกเรียกใช้เมื่อ Activity ถูก restart
  7. onDestroy() ถูกเรียกใช้เมื่อ Activity ถูกทำลายโดย system



36

การทำงานกับ Activities

- โครงสร้างของวงจรชีวิต Activity จะมี 3 เมธอดหลักที่ระบบเรียกใช้อัตโนมัติตามลำดับเมื่อมีการสร้างอินสแตนซ์ของ Activity ขึ้นมาใหม่ ได้แก่ onCreate(), onStart() และ onResume() เมื่อลำดับการทำงานเสร็จสิ้นลง Activity จะมาอยู่ที่สถานะ Resumed ซึ่งผู้ใช้ใช้งานอยู่จนกว่าจะเปลี่ยนไปเรียกใช้งาน Activity อื่นขึ้นมา



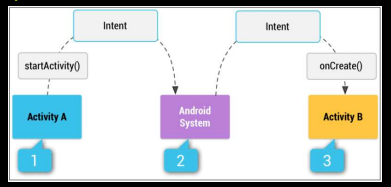
ตัวอย่างโค้ดแสดงลำดับการทำงานของ State ใน LogCat

```
@Override
protected void onCreate(Bundle
savedInstanceState) {
  super.onCreate(savedInstanceState);
  ActionBar actionBar = getSupportActionBar();
  actionBar.hide();
  setContentView(R.layout.activity_main);
  Log.d("Android: ","onCreate() method");
}
@Override
protected void onStart() {
  super.onStart();
  Log.d("Android: ","onStart() method");
}
@Override
protected void onResume() {
  super.onResume();
  Log.d("Android: ","onResume() method");
}
```

```
@Override
protected void onPause() {
  super.onPause();
  Log.d("Android: ","onPause() method");
}
@Override
protected void onStop() {
  super.onStop();
  Log.d("Android: ","onStop() method");
}
@Override
protected void onRestart() {
  super.onRestart();
  Log.d("Android: ","onRetart() method");
}
@Override
protected void onDestroy() {
  super.onDestroy();
  Log.d("Android: ","onDestroy() method");
}
```


Intents

- อินเทนต์ (Intents) เป็นหนึ่งในอ็อบเจกต์แอนดรอยด์ มีลักษณะเป็น message ให้คอมโพเนนต์ต่าง ๆ สามารถเรียกใช้ฟังก์ชันจาก component ตัวอื่นได้ เช่น Activity สามารถส่ง intent ไปยังระบบของแอนดรอยด์เพื่อให้ Activity อื่นทำงาน



- อินเทนต์ทำให้คอมโพเนนต์ต่าง ๆ ที่เชื่อมต่อกันแบบหลวม ๆ ให้ประสานงานกันได้ ทั้งยังสามารถใช้เป็นตัวส่งสัญญาณไปยังระบบแอนดรอยด์ เพื่อบอกว่าเกิดเหตุการณ์ใด ๆ ขึ้น เพื่อให้คอมโพเนนต์อื่นที่เกี่ยวข้องกับตัวเหตุการณ์นี้ได้ถูกกระตุ้น
- อินเทนต์สามารถเก็บข้อมูลได้ โดยข้อมูลเหล่านี้สามารถนำส่งให้คอมโพเนนต์อื่น เช่น แอปสามารถเรียกคอมโพเนนต์ของเบราว์เซอร์ได้ โดยข้อมูลที่ส่งให้ก็คือ URL ไปยังคอมโพเนนต์เบราว์เซอร์

Intent filter

- Intent Filter เป็น instance ของคลาส IntentFilter ใช้สำหรับ Launch คอมโพเนนต์ที่ต้องการขึ้นมาในระบบ
- โดยทั่วไปเราจะไม่ได้สร้าง Intent Filters โดยใช้ Java Code ส่วนใหญ่แล้วจะประกาศ Intent Filter ไว้ใน Application Manifest File (AndroidManifest.xml) โดยประกาศไว้ภายใน element ที่มีชื่อว่า <intent-filter>
- หากต้องการ launch คอมโพเนนต์ใดให้ย้าย <intent-filter> ไปอยู่ในแท็กของคอมโพเนนต์นั้น

```
<activity android:name=".Calculator" />
<activity android:name=".HirePurchase" />
<activity android:name=".MainActivity">
   <intent-filter>
     <action android:name="android.intent.action.MAIN" />
     <category  android:name="android.intent.category.LAUNCHER" />
   </intent-filter>
</activity>
```

ชนิดของ Intents

1. Explicit Intents จะระบุชื่อของคอมโพเนนท์ด้วยการใช้คลาสเป็นไอดี มักนำไปใช้ภายในแอปเดียวกันที่มีหลายคลาสที่ควบคุมโดยผู้พัฒนาโปรแกรม ดังตัวอย่าง เป็นการส่งอินเทนต์ไปยังระบบแอนดรอยด์ โดยระบุคลาส ActivityTwo เป็นผู้รับ จากนั้นจะเริ่มการทำงานของคลาส ActivityTwo

```
Intent i = new Intent(this, ActivityTwo.class);
startActivity(i);
```

2. Implicit Intents จะไม่ระบุคอมโพเนนต์ แต่จะระบุ action ที่ต้องการกระทำและอาจรวมถึง URI ที่จะนำไปใช้กับ action เหล่านั้น ดังตัวอย่าง จะบอกระบบแอนดรอยด์เพื่อต้องการดูเว็บเพจ ซึ่งโดยทั่วไประบบจะเรียกเว็บเบราว์เซอร์ที่เข้ากับอินเทนต์นี้ขึ้นมา ซึ่งคอมโพเนนต์อื่นก็สามารถที่จะเข้ากับอินเทนต์นี้ได้เช่นกัน

```
Intent i = new Intent(Intent.ACTION_VIEW, Uri.parse("https://www.google.ac.th"));
```

เมื่ออินเทนต์นี้ถูกส่งให้กับระบบแอนดรอยด์ ระบบจะค้นหาทุกคอมโพเนนต์ที่ถูก register เข้ากับเหตุการณ์และชนิดของ datatype นั้น ๆ ถ้ามีเพียงคอมโพเนนต์เดียวที่ระบบค้นพบ ระบบจะเริ่มการทำงานคอมโพเนนต์นั้นทันที ถ้ามีหลายตัวถูกค้นพบ จะมีไดอะล็อกขึ้นมาเพื่อให้เลือกว่าจะใช้คอมโพนเนนต์ตัวใด

41

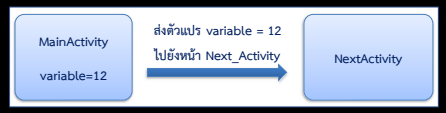
Workshop

- สร้าง Button Widget ใน activity_main ขึ้นมา 2 ปุ่ม กำหนด text เป็น HirePurchase และ Calculator ตามลำดับ
- เขียนคำสั่งเมื่อคลิกปุ่ม HirePurchase ให้เรียกใช้ Activity HirePurchase และเมื่อคลิกปุ่ม Calculator ให้เรียกใช้ Activity Calculator ตามลำดับ

42

การส่งข้อมูลระหว่าง Activity

- การส่งผ่านข้อมูลระหว่าง Activity คือ การส่งผ่านข้อมูลเพื่อเรียก Intents หรือเรียกหน้าใหม่โดยอาจจะส่งข้อมูลไปพร้อมกับการเรียกใช้ Activity โดยมี 4 แบบ
  1. ส่งผ่านข้อมูลโดยใช้ putExtra() , getExtras()
  2. ส่งผ่านข้อมูลโดยสร้าง bundle ไว้ก่อน แล้วส่งตัว bundle
  3. ส่งผ่านข้อมูล และรับค่าส่งกลับ
  4. ส่งผ่านข้อมูลไปยังแอปอื่น



43

การส่งผ่านข้อมูลโดยใช้ putExtra(), getExtras()

- คอมโพเนนต์ที่สร้างอินเทนต์ สามารถเพิ่มข้อมูลเข้าไปโดยการเรียกเมธอด putExtra( ) โดย Extra คือ คีย์ (key) และค่าที่จะส่ง (value) โดยที่คีย์นั้นมักจะเป็นข้อมูลชนิดสตริง ส่วนค่าที่จะส่งนั้นเราสามารถแทนด้วยตัวแปรได้

Code ฝั่ง Activity ที่ส่ง

```
Intent i = new Intent(this, NextActivity.class);
i.putExtra("id", id);
startActivity(i);
```

Code ฝั่ง Activity ที่รับ

```
Bundle b = getIntent().getExtras();
int id = b.getInt("id");
```

44

การส่งผ่านข้อมูลโดยใช้ putExtra(), getExtras()

Code ฝั่ง Activity ที่ส่ง

```
btnGo = findViewById(R.id.buttonGo);
btnGo.setOnClickListener(new OnClickListener() {
  public void onClick(View v) {
    Intent i = new Intent(getApplicationContext(), NextActivity.class);
    i.putExtra("id", "001");
    startActivity(i);
}});
```

Code ฝั่ง Activity ที่รับ

```
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.Activity_Next);
  TextView tv = findViewById(R.id.textViewShow);
  Bundle b = getIntent().getExtras();
  String id = b.getString("id");
  tv.setText(id);
}
```

45

การส่งผ่านข้อมูลโดยสร้าง bundle ไว้ก่อน แล้วส่งตัว bundle

Code ฝั่ง Activity ที่ส่ง

```
btnGo = findViewById(R.id.buttonGo);
btnGo.setOnClickListener(new OnClickListener() {
  public void onClick(View v) {
    Intent i = new Intent(getApplicationContext(), NextActivity.class);
    Bundle b = new Bundle();
    b.putString("id", "001");
    i.putExtra(b);
    startActivity(i);}});
```

Code ฝั่ง Activity ที่รับ

```
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.Activity_Next);
  TextView tv = findViewById(R.id.textViewShow);
  Bundle b = getIntent().getExtras();
  String id = b.getString("id");
  tv.setText(id);
}
```

46

Workshop

- แก้ไข activity_main ให้ส่งค่าราคารถ 100000 และเงินดาวน์ 20000 ไปที่ HirePurchase
- แก้ไข HirePurchase ให้รับค่าจาก activity_main นำไปกำหนดเป็นค่าเริ่มต้นให้กับราคารถ และเงินดาวน์

47

การส่งผ่านข้อมูล และรับค่าส่งกลับ

- เมธอด startActivityForResult(Intent, int) ใช้สำหรับส่งอินเทนต์ และรอรับค่าที่ถูกส่งกลับมาจาก Activity ที่ถูกเรียก โดยพารามิเตอร์ int คือ ค่าสำหรับอ้างอิงเมื่อข้อมูลถูกส่งกลับมา

```
Intent i = new Intent(getApplicationContext(), SecondActivity.class);
startActivityForResult(i, 0);
```

- ต้อง override เมธอด onActivityResult(int, int, Intent) โดยพารามิเตอร์ int แรก คือ ค่าอ้างอิงที่ส่งไปตอนเรียก int ตัวที่สอง คือ ค่าผลลัพธ์ที่ส่งกลับมาจาก Activity ที่ถูกเรียก และ Intent คือ ข้อมูลที่ถูกส่งกลับมา ซึ่งสามารถเรียกใช้ข้อมูลโดยผ่านเมธอด get...Extra(type)

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
  super.onActivityResult(requestCode, resultCode, data);
  TextView tv = findViewById(R.id.txtReturn);
  if (resultCode == RESULT_OK) {
    tv.setText(data.getStringExtra("RESULT_RETURN"));
  }}
```

- Activity ที่ถูกเรียก หากต้องการส่งข้อมูลกลับ ต้องสร้าง Intent ขึ้นมา แล้วส่งกลับผ่าน method setResult(int, Intent) โดยพารามิเตอร์ int คือ ค่าผลลัพธ์ที่ต้องการส่งกลับ และ Intent คือ ข้อมูลที่ต้องการส่งกลับ จากนั้นเรียก method finish() เพื่อสิ้นสุดการทำงานของ Activity

```
Intent i = new Intent();
i.putExtra("RESULT_RETURN", txtReturn.getText().toString());
setResult(RESULT_OK, i);
finish();
```

48

Workshop

- สร้าง TextView Widget ค่าผ่อนรถต่อเดือน ใน activity_main
- แก้ไขคำสั่งใน activity_main ให้ส่งค่าไปที่ HirePurchase และรับค่าผ่อนรถต่อเดือนกลับมาแสดงใน TextView ที่สร้างขึ้น

การส่งผ่านข้อมูลไปยังแอปอื่น

- การใช้ Intent เรียกแอปอื่น

```
Intent intent = new Intent();
intent.setAction(Intent.ACTION_MAIN);
intent.addCategory(Intent.CATEGORY_APP_MAPS);//พิมพ์ Intent.ระบุแอปที่ต้องการ
startActivity(intent);
```

- การใช้ Intent เรียกเว็บเบราว์เซอร์

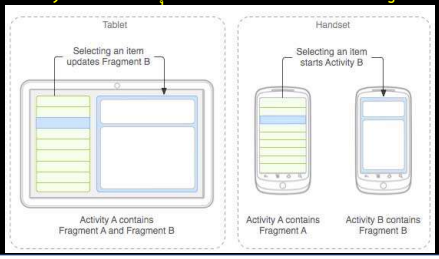
```
Intent goWebSite = new Intent(Intent.ACTION_VIEW);
goWebSite.setData(Uri.parse("https://google.co.th"));
startActivity(goWebSite);
```

เปิด androidManifest.xml แล้วเพิ่ม permission ให้แอปเราสามารถใช้ Internet ได้

```
<uses-permission android:name="android.permission.INTERNET"/>
```

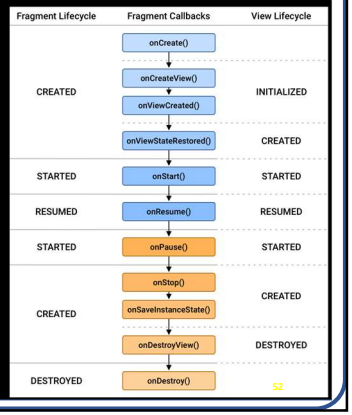

Fragment

- ใช้สำหรับแอปที่ต้องการแสดงผลหลายหน้าจอ โดยขณะใดขณะหนึ่งจะแสดงผลได้หลาย Fragment
- Fragment มีองค์ประกอบบางส่วนและการทำงานคล้าย Activity โดยมีไฟล์ที่เป็นองค์ประกอบเหมือนกัน คือ xml และ java จับคู่กัน สำหรับเป็นส่วนแสดงผล และส่วนควบคุม
- การสร้าง Fragment จะคล้ายกับ Activity แต่ Fragment จะไม่สามารถแสดงผลได้ด้วยตนเอง เมื่อสร้างเสร็จจะต้องนำไปวางบน Activity ซึ่งจะใช้ควบคุมการสับเปลี่ยนแสดงผลของ Fragment



Fragment lifecycle

- Fragment lifecycle ประกอบด้วย 5 state ซึ่งจะมี callback method ที่จะถูกเรียกใช้ในระหว่างที่มีการเปลี่ยน state ดังนี้
  1. onAttach() เมื่อ Fragment ถูกเรียกครั้งแรกจาก Activity
  2. onCreate() เมื่อ Fragment ถูกสร้างขึ้น
  3. onCreateView() เมื่อวิวบนเลย์เอาต์ของ Fragment ถูกสร้างขึ้น
  4. onViewCreated() เมื่อวิวบน Fragment ถูกสร้างจนเสร็จสมบูรณ์
  5. onViewStateRestored() เมื่อวิวถูกย้ายไปอยู่ในstate CREATED
  6. onStart() เมื่อ Fragment กำลังจะปรากฏบนหน้าจอ
  7. onResume() เมื่อ Fragment ปรากฏบนหน้าจอ (Active)
  8. onPause() เมื่อ Fragment ไม่ Active แต่ยังปรากฏบนหน้าจอ เนื่องจากมีอ็อบเจกต์อื่นปรากฏขึ้นมาเหนือ Fragment
  9. onStop() เมื่อ Fragment ไม่ปรากฏบนจอ
  10. onDestroyView() เมื่อวิวบน Fragment ถูกทำลาย
  11. onDestroy() เมื่อ Fragment ถูกทำลาย

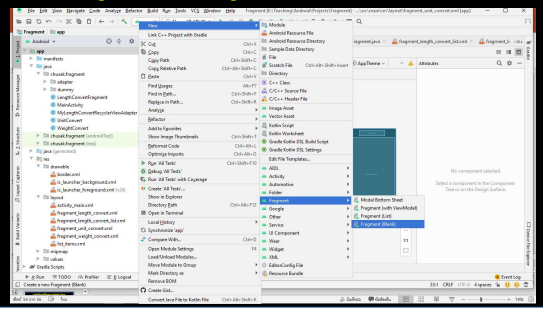


Fragment Transaction

- มีหน้าที่จัดการ Fragment เพื่อให้ไปแสดงผลใน Layout บน Activity ตามต้องการ ทั้งการเพิ่ม ลบออกจาก Layout ช่วยให้เราสามารถเปลี่ยน Fragment สลับไปมาได้
- เมธอดสำหรับจัดการ Fragment
  1. add() ทำหน้าที่เพิ่ม Fragment ลงไปทำให้สามารถซ้อนทับกันได้
  2. replace() เป็นการแทนที่ Fragment ตัวใหม่ทับตัวเก่าที่มีอยู่ก่อนหน้านี้
  3. addToBackStack() จะเก็บ Fragment ไว้ในสแตก เมื่อต้องการย้อนกลับก็จะเรียก Fragment ก่อนหน้ามาแสดงผลโดยไม่ต้องสร้างใหม่ ส่วนมากนิยมใช้ replace() คู่กับ addToBackStack() เพราะแอปส่วนมากจะเป็นแบบซ้อนกันไปเรื่อย ๆ เวลากด Back กลับมาก็จะดึงจากสแตกมาใช้งานต่อได้ทันที

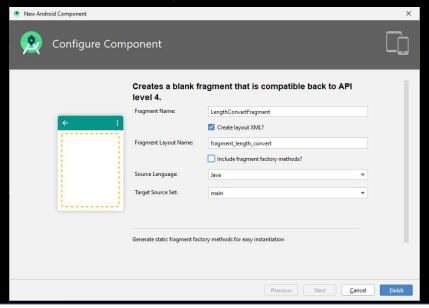
การสร้าง Fragment

1. คลิกขวาที่ app ในหน้าต่าง Project
2. เลือก New->Fragment->Fragment (Blank)



การสร้าง Fragment

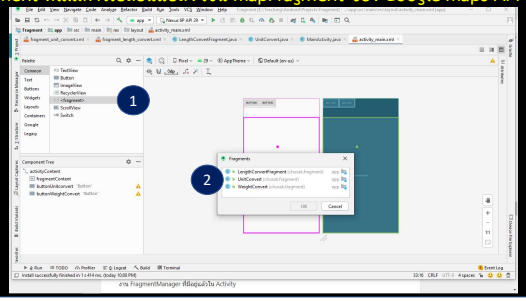
3. ใส่ชื่อ Fragment และ Uncheck Include fragment factory method?
4. คลิก Finish
5. ออกแบบหน้าจอ Fragment ได้เช่นเดียวกับ Activity



การวาง Fragment ลงใน Activity กรณีที่มีเพียง Fragment เดียว

1. ลาก <> <fragment> ในหน้าต่าง palette ไปวางใน Layout หรือ ViewGroup ที่อยู่ในไฟล์ Layout ที่ต้องการ
2. เลือก Fragment ที่ต้องการวางตามต้องการ

** เหมาะกับ Fragment ที่ไม่มีการเปลี่ยนแปลง เช่น MapFragment ของ Google Maps API



การวาง Fragment ลงใน Activity กรณีที่มีการสลับ Fragment

1. เลือก ...Activity.java ที่ต้องการวาง Fragment ลงไป
2. ต้องกำหนด id ให้กับ Layout ที่ต้องการวาง (Fragment ต้องถูกวางลงบน Layout หรือ ViewGroup เท่านั้น)
3. พิมพ์คำสั่งสำหรับวาง Fragment ในเมธอดตามต้องการ

```
if (savedInstanceState == null) { //ตรวจสอบการเก็บค่าตัวแปรในกรณีที่เคยกำหนดไว้ก่อนหน้านี้
    getSupportFragmentManager().beginTransaction()
    .replace(R.id.fragmentContent, new LengthConvertFragment())
    .addToBackStack(null) //กำหนดให้เก็บ Fragment ไว้ใน BackStack สำหรับกดปุ่มย้อนกลับ หรือ
                          //เรียกเมธอด popBackStack() ถ้าไม่ต้องการย้อนกลับก็ไม่ต้องใส่
    .commit();
}
```

** fragmentContent คือ id ของ Layout หรือ ViewGroup ที่ต้องการวาง Fragment

** LengthConvertFragment คือ java Class ของ Fragment

** หากใช้ .add แทน .replace แล้ว Fragment จะถูกวางซ้อนทับกันเป็นชั้น ๆ

57

ตัวอย่างการสลับ Fragment

```
protected void onCreate(final Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  ActionBar actionBar = getSupportActionBar();
  actionBar.hide();
  setContentView(R.layout.activity_main);
  Button lengthConvert = findViewById(R.id.buttonLengthconvert);
  Button weightConvert = findViewById(R.id.buttonWeightConvert);
  if (lengthConvert != null) {
    lengthConvert.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        if (savedInstanceState == null) {
          getSupportFragmentManager().beginTransaction()
          .replace(R.id.fragmentContent, new LengthConvert())
          .addToBackStack(null)
          .commit();
        }
      }
    });
  }
```

```
if (weightConvert != null) {
  weightConvert.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
      if (savedInstanceState == null) {
        getSupportFragmentManager().beginTransaction()
        .replace(R.id.fragmentContent, new WeightConvert())
        .addToBackStack(null)
        .commit();
      }
    }
  });
}
}
```

58

Workshop

1. สร้าง Project ใหม่ ชื่อ FragmentExample
2. สร้าง Fragment ขึ้นมา 3 Fragment ดังนี้
   1. Fragment สำหรับแปลงค่าหน่วยความยาว (LenghtConvertFragment)
   2. Fragment สำหรับแปลงค่าหน่วยน้ำหนัก (WeightConvertFragment)
   3. Fragment สำหรับแสดงแผนที่ (MapFragment)
3. สร้างปุ่ม 3 ปุ่มใน activity_main.xml เพื่อคลิกเปิดแต่ละ Fragment
4. สร้าง FrameLayout ใน activity_main.xml สำหรับวาง Fragment

59

การส่ง-รับข้อมูลระหว่าง Activity กับ Fragment (วิธีที่ 1)

- Activity จะส่งข้อมูลเป็นแบบ Bundle ไปให้ Fragment ตั้งแต่ตอนสร้าง Fragment ขึ้นมา
  1. สร้างออบเจกต์ของคลาส Bundle และเรียกใช้เมธอด putXxx() เพื่อส่งค่า Key และ Value ตามต้องการ
  2. ส่ง Bundle ออบเจกต์ผ่านเมธอด setArguments(bundle) ของ Fragment
  3. Fragment จะรับค่าผ่านเมธอด this.getArguments().getXxx("key"); โดย XXX ได้แก่ String, Int, Double, Boolean

Code ฝั่ง Activity ที่ส่ง

```
Bundle bundle = new Bundle();
bundle.putString("key", "value" );
FlagmentClass flagmentObj = new FlagmentClass()
flagmentObj.setArguments(bundle);
getSupportFragmentManager().beginTransaction().replace(R.id.fragment_content, flagmentObj).commit();
```

** fragment_content คือ Layout ที่ต้องการวาง Fragment บน Activity

** FlagmentClass คือ Java Class ของ Fragment

Code ฝั่ง Fragment ที่รับ

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
     String myValue = this.getArguments().getString("key"); //สามารถใส่ code รับค่าในเมธอดใดก็ได้
     ...
}
```

60

การส่ง-รับข้อมูลระหว่าง Activity กับ Fragment (วิธีที่ 2)

1. ฝั่ง Activity ให้สร้างเมธอดสำหรับส่งข้อมูลเพื่อให้ Fragment เรียกรับค่าผ่านเมธอดจากออบเจกต์ของ Activity
2. ฝั่ง Fragment สร้างออบเจกต์ของ Activity ที่จะรับค่าผ่านเมธอด โดยใส่ code ลงในเมธอดที่ต้องการ

Code ฝั่ง Activity ที่ส่ง

```
public String getData() {
  return "3000";
}
```

Code ฝั่ง Fragment ที่รับ

```
MainActivity activity = (MainActivity) getActivity();
editTextUnit.setText(activity.getData());
```

61

การส่ง-รับข้อมูลระหว่าง Fragment กับ Activity

1. สร้าง Interface และกำหนดเมธอดแบบ abstract สำหรับการเชื่อมโยงข้อมูล

```
public interface FragmentToActivity {    void communicate(String comm);    }
```

2. Code ฝั่ง Fragment ที่ส่ง ให้ประกาศแอตทริบิวต์ของคลาสเป็นชนิด Interface แล้ว override เมธอด onAttach() ตามตัวอย่าง เมื่อต้องการส่งค่าให้เรียกใช้เมธอดของ Interface ในเมธอดที่ต้องการ

```
private FragmentToActivity mCallback; //ประกาศเป็นแอตทริบิวต์ของคลาส เพื่อให้เรียกใช้ในเมธอดต่าง ๆ ได้
@Override
public void onAttach(Context context) {
  super.onAttach(context);
  try {
     mCallback = (FragmentToActivity) context;
     mCallback.communicate("data");  //สามารถใส่ในเมธอดใดก็ได้เมื่อต้องการส่งข้อมูล
  } catch (ClassCastException e) {
     throw new ClassCastException(context.toString() + " must implement FragmentToActivity");
  }
}
```

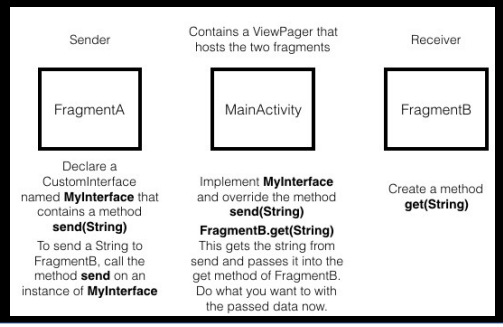
3. Code ฝั่ง Activity ที่รับให้ implements Interface ที่สร้างในขั้นตอนที่ 1 และเขียนทับเมธอดของ Interface แล้วเรียกใช้ค่าอาร์กิวเมนต์ที่ส่งมาจาก Fragment

```
public void communicate(String comm) {
   unitConvert.setText(comm.toString());
}
```

62

การรับ-ส่งข้อมูลระหว่าง Fragment กับ Fragment

- ทำได้โดยส่งผ่าน Activity
- Fragment ที่ต้องการส่งข้อมูลให้ส่งข้อมูลไปที่ Activity แล้วให้ Activity ส่งต่อไปยัง Fragment อื่น



63

Dialog Fragment

- ใช้สำหรับแสดงหน้าต่างแจ้งเตือน เพื่อแจ้งให้ผู้ใช้ดำเนินการบางอย่าง
- การสร้าง Dialog Fragment
  1. สร้างคลาสใหม่ สืบทอดจากคลาส DialogFragment และกำหนดคอนสตรักเตอร์ว่าง
  2. กำหนด static เมธอด สำหรับกำหนด instance ของตัวมันเอง โดยรับข้อมูลจากพารามิเตอร์เป็นค่าที่ต้องการแสดง
  3. Overide เมธอด onCreateDialog()

64

สร้างคลาสแสดง Dialog (MessageDialog.java)

```
public class MessageDialog extends DialogFragment {
    public MessageDialog() {
        // Required empty public constructor
    }
    public static MessageDialog newInstance(String title, String msg) {
        MessageDialog msgDlg = new MessageDialog();
        Bundle args = new Bundle();
        args.putString("title", title);
        args.putString("message", msg);
        msgDlg.setArguments(args);
        return msgDlg;
    }
    @Override public Dialog onCreateDialog(Bundle savedInstanceState) {
        String title = getArguments().getString("title");
        String message = getArguments().getString("message");
        AlertDialog.Builder builder = new AlertDialog.Builder(getActivity())
                .setTitle(title)
                .setMessage(message)
                .setPositiveButton("OK", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                    }
                });
        return builder.create();
    }
}
```


การเรียกใช้คลาส MessageDialog

1. ประกาศ และสร้างออบเจกต์ พร้อมกำหนดค่า ชื่อหน้าต่าง(title) และข้อความแจ้งเตือน (message) ของหน้าต่างแจ้งเตือน

```
MessageDialog alert = MessageDialog.newInstance("แจ้งเตือน", "ตัวเลขต้องมากกว่า 0");
```

2. เรียกใช้เมธอด show() ผ่านออปเจกต์ alert

```
alert.show(getActivity().getSupportFragmentManager(), null);
```


Confirm Dialog Fragment

- ใช้สำหรับแสดงหน้าต่างแจ้งเตือน เพื่อแจ้งให้ผู้ใช้ดำเนินการบางอย่าง แล้วให้ผลลัพธ์กลับมา เช่น แจ้งผู้ใช้ให้เลือกตอบ ใช่/ไม่ใช่ ตกลง/ยกเลิก
- คล้ายกับ Dialog โดยจะมีปุ่มกดให้เลือก 2 ปุ่ม คือ Negative (ซ้าย) และ Positive (ขวา) เมื่อผู้ใช้คลิกปุ่มก็จะส่งผลการคลิกกลับ

สร้างคลาสแสดง Confirm Dialog (ConfirmDialog.java)

```
public class ConfirmDialog extends DialogFragment {
  enum Button { Negative, Positive } //ใช้เก็บผลลัพธ์การกดปุ่ม
  public ConfirmDialog() {}
  public static ConfirmDialog newInstance
   (String msg, String negText, String posText) {
   ConfirmDialog cfDlg = new ConfirmDialog();
   Bundle args = new Bundle();
   args.putString("message", msg);
   args.putString("negText", negText);
   args.putString("posText", posText);
   cfDlg.setArguments(args);
   return cfDlg;
  }
  @Override public Dialog onCreateDialog(Bundle savedInstanceState) {
   String message = getArguments().getString("message");
   String negText = getArguments().getString("negText");
   String posText = getArguments().getString("posText");
   AlertDialog.Builder builder = new AlertDialog.Builder(getActivity())
     .setMessage(message)
     .setNegativeButton(negText, new DialogInterface.OnClickListener() {
      @Override
      public void onClick(DialogInterface dialog, int which) {
        mListener.onFinishDialog(ConfirmDialog.Button.Negative);
      }
     })
     .setPositiveButton(posText, new DialogInterface.OnClickListener() {
      @Override
      public void onClick(DialogInterface dialog, int which) {
        mListener.onFinishDialog(ConfirmDialog.Button.Positive);
      }
     });
     return builder.create();
  }
  interface OnFinishDialogListener {
     void onFinishDialog(ConfirmDialog.Button button);
  }
  private  OnFinishDialogListener mListener;
  public  void setOnFinishDialogListener(OnFinishDialogListener listener)
{
     mListener = listener;
  }
}
```

การเรียกใช้คลาส ConfirmDialog

1. ประกาศ และสร้างออบเจกต์ พร้อมกำหนดค่าข้อความ (message) ข้อความ Negative และข้อความ Positive

```
ConfirmDialog cfDlg = ConfirmDialog.newInstance("ต้องการออกจากโปรแกรมหรือไม่", "ไม่ใช่","ใช่");
```

2. เรียกใช้เมธอด show() ผ่านออปเจกต์ cfDlg

```
cfDlg.show(getSupportFragmentManager(),null);
```

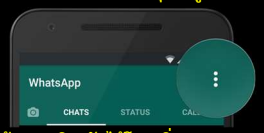
3. ตรวจสอบการกดปุ่ม และกำหนดการทำงาน

```
cfDlg.setOnFinishDialogListener(new ConfirmDialog.OnFinishDialogListener() {
  @Override
  public void onFinishDialog(ConfirmDialog.Button button) {
    if (button == ConfirmDialog.Button.Positive) {
      finishAffinity(); //ปิดทุก Activity ที่ค้างในสแตก
      System.exit(0); //ออกจากโปรแกรม
    }
  }
});
```

69

การสร้างเมนูบน Action Bar

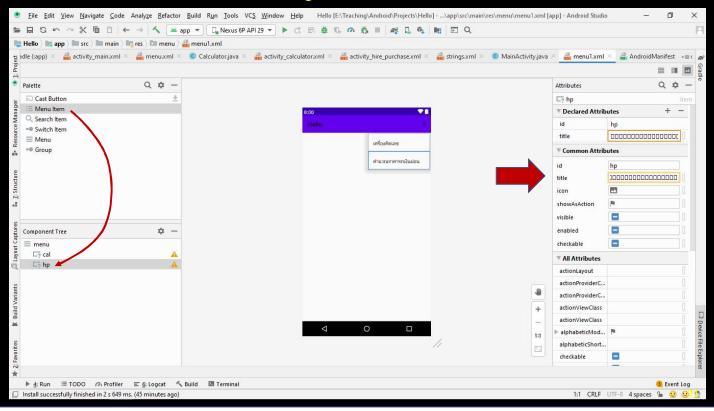
- ตั้งแต่แอนดรอยด์ 4.2 ขึ้นไป จะมีการกำหนดปุ่มเมนูไว้บน Action Bar ด้านขวามือ



- เราสามารถกำหนดเมนูให้แอปพลิเคชันได้โดยเพิ่ม resource file ใน app/res/menu โดย
  1. คลิกขวาที่ ▽app ▽res ▷menu เลือก New -> Menu Resource File
  2. กำหนดชื่อไฟล์ แล้วคลิก OK
  3. ดับเบิลคลิกเปิดไฟล์เมนู เลือกโหมด Design
  4. ลาก Menu Item ในหน้าต่าง Palette ไปวางในหน้าต่าง Component tree ภายใต้ menu
  5. กำหนด id, title, และ icon ในหน้าต่าง Attributes

70

การสร้างเมนูบน Action Bar



การสร้างเมนูบน Action Bar

- เขียนคำสั่งในไฟล์ java

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
  getMenuInflater().inflate(R.menu.menu, menu); //กำหนดให้แสดงปุ่มเมนู
  return super.onCreateOptionsMenu(menu);       //R.menu.menu คือชื่อไฟล์ menu.xml
}
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
//if (item.toString().equals("Calculator")) { //ตรวจสอบการกดเมนูโดยใช้ title
  if (item.getItemId() == R.id.cal) {         //ตรวจสอบการกดเมนูโดยใช้ id
    Intent calculator = new Intent(MainActivity.this, Calculator.class);
    startActivity(calculator);
  } else if (item.toString().equals("HirePurchase")) {
    Intent hp = new Intent(MainActivity.this, HirePurchase.class);
    hp.putExtra("price", 100000);
    hp.putExtra("down", 20000);
    startActivityForResult(hp,0);
  }
  return super.onOptionsItemSelected(item);
}
```

72

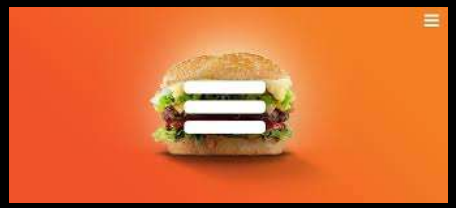
ระบบ Dependencies

- เป็นช่องทางการเพิ่ม Widget พิเศษต่าง ๆ เข้ามาในโปรเจกต์ มีวิธีเพิ่ม 3 วิธี
  1. Widget ที่มีลูกศรลงต่อท้ายในหน้าต่าง Palette สามารถใช้ได้เลย
  2. คลิกเมนู File->Project Structure...->Dependencies คลิกปุ่ม + และเลือกรายการ Library Dependency แล้วเลือกรายการตามต้องการ
  3. เปิดไฟล์ build.gradle (Module: app) แล้วพิมพ์รายการ Dependency เพิ่มตามต้องการ

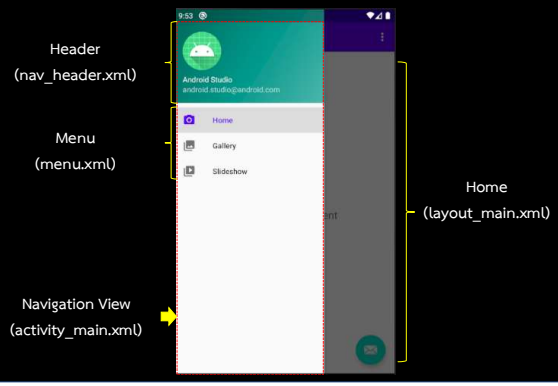
```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])

    implementation 'androidx.appcompat:appcompat:1.2.0'
    implementation 'androidx.legacy:legacy-support-v4:1.0.0'
    implementation 'com.google.android.material:material:1.2.1'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test.ext:junit:1.1.2'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'
}
```


เมนูแบบ Navigation Drawer

- รูปแบบเมนูที่นิยมใช้ในแอป เรียกว่า Navigation Drawer โดยจะมีปุ่มไว้ให้ผู้ใช้กดเรียกว่า ปุ่ม Hamburger ซึ่งเมื่อผู้ใช้กดก็จะมีเมนู Navigation Drawer ปรากฏขึ้นมาให้เลือก

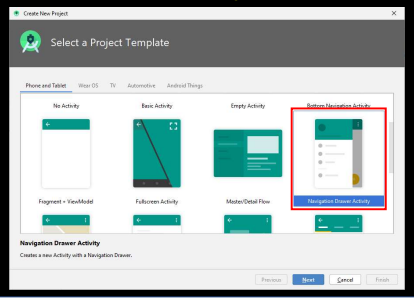


ส่วนประกอบของเมนูแบบ Navigation Drawer



การสร้างเมนูแบบ Navigation Drawer จากเทมเพลต

1. เลือก File->New->New Project...(กรณีสร้างโปรเจกต์ใหม่) หรือคลิกขวาที่ app เลือก New->Activity->Navigation Drawer Activity (กรณีต้องการเพิ่มเมนูภายหลัง) เลือกเทมเพลต Navigation Drawer Activity จากนั้นไปแก้ไขเมนูจากเทมเพลต ตามต้องการ



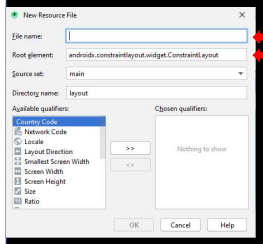
การสร้างเมนูแบบ Navigation Drawer ด้วยตนเอง

1. แก้ไข Layout ในไฟล์ activity_main.xml โดยเปลี่ยนเป็น DrawerLayout หรือกรณีสร้าง Layout ใหม่ ให้เลือก DrawerLayout

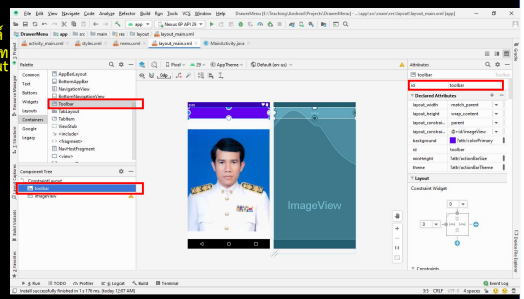
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.drawerlayout.widget.DrawerLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:id="@+id/drawer_layout"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context=".MainActivity">
```


การสร้างเมนูแบบ Navigation Drawer ด้วยตนเอง

2. กรณีต้องการสร้างหน้าแรก (Home) ให้เพิ่ม Layout ใหม่ โดยคลิกขวาที่ app->res->layout เลือก New->Layout Resource File (ตกแต่ง UI ตามต้องการ) และใส่ Container->Toolbar

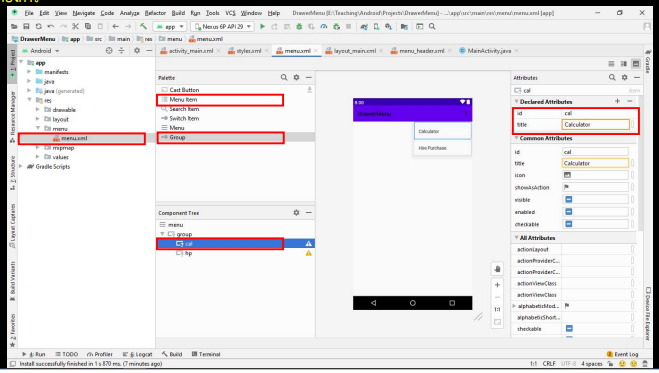


ชื่อไฟล์
ประเภท Layout



การสร้างเมนูแบบ Navigation Drawer ด้วยตนเอง

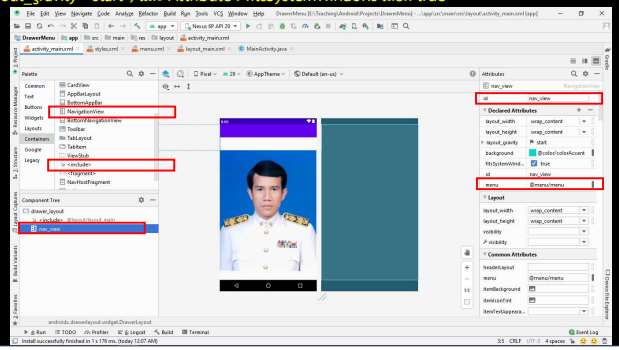
3. เพิ่มไฟล์เมนูโดยคลิกขวาที่ res เลือก New->Resource File ในช่อง Resource Type ให้เลือก Menu ตอนสร้างเมนูให้ใส่ Group ก่อน และกำหนดเมนูตามต้องการ



การสร้างเมนูแบบ Navigation Drawer ด้วยตนเอง

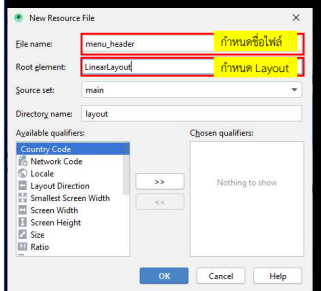
4. ใส่ layout_main ลงในไฟล์ activity_main.xml โดยเพิ่ม Container-> <include> แล้วเลือก Attribute->layout เป็น layout_main

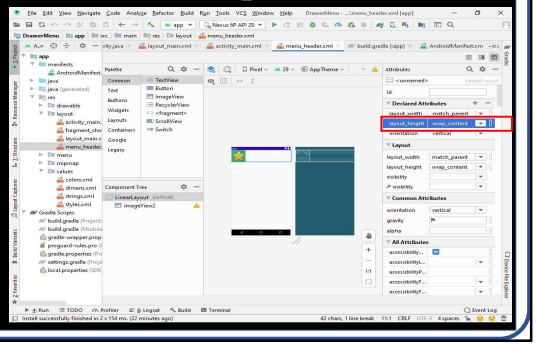
5. เพิ่ม Container->Navigation View ลงในไฟล์ activity_main.xml และกำหนด Attribute->id และ Attribute->menu เป็นไฟล์เมนูที่สร้างไว้ Attribute->layout_gravity เลือก Start (ถ้าไม่มีให้พิมพ์เพิ่มลงใน xml ในส่วน NavigationView เป็น android:layout_gravity="start") และ Attribute->fitsSystemWindows เลือก true



การสร้างเมนูแบบ Navigation Drawer ด้วยตนเอง

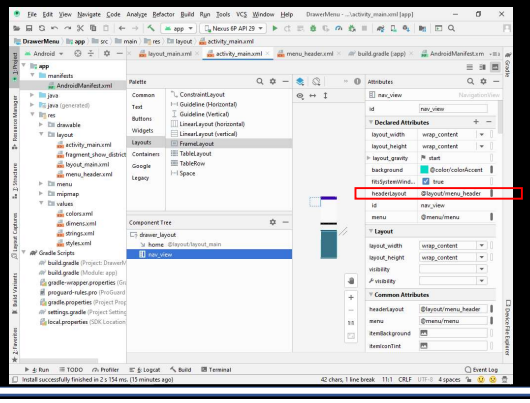
6. สร้าง Layout สำหรับ Header ของเมนู (ไม่สร้างก็ได้ เมนูที่แสดงก็จะมีเฉพาะรายการเมนู) โดยคลิกขวาที่ res เลือก New->Layout Resource File กำหนดชื่อไฟล์ เลือกประเภทของ Layout แล้วคลิก OK
7. ตกแต่งหน้าจอเช่น กำหนดสี ใส่ภาพ Logo หรือข้อความตามต้องการ (layout_height ของ Layout หลัก ควรกำหนดเป็น wrap_contents





การสร้างเมนูแบบ Navigation Drawer ด้วยตนเอง

8. ที่ activity_main.xml เลือก NavigationView เลือก Attributes->headerLayout เป็นไฟล์ Header ที่สร้าง



การสร้างเมนูแบบ Navigation Drawer ด้วยตนเอง

9. แก้ไขไฟล์ (ถ้าไม่มีให้สร้างใหม่) res->values->styles.xml กำหนดให้ซ่อน ActionBar

```
<resources>

    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>

</resources>
```


การสร้างเมนูแบบ Navigation Drawer ด้วยตนเอง

10. เขียนคำสั่งควบคุมใน MainActivity.java

```
public class MainActivity extends AppCompatActivity {
  private DrawerLayout drawerLayout;
  private ActionBarDrawerToggle toggle;
  private NavigationView navigationView;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    drawerLayout = findViewById(R.id.drawer_layout);
    Toolbar toolbar = findViewById(R.id.toolbar);
    setSupportActionBar(toolbar); //เปลี่ยนจาก ActionBar เป็น Toolbar
    ActionBar actionBar = getSupportActionBar();
    actionBar.setTitle("");        //แก้ชื่อโปรเจกต์บน ActionBar
    actionBar.setDisplayHomeAsUpEnabled(true); //แสดงปุ่ม Hamberger
    actionBar.setHomeAsUpIndicator(R.drawable.ic_menu_black_24dp); //กำหนดรูปให้ปุ่ม
    navigationView = findViewById(R.id.nav_view);
    navigationView.setNavigationItemSelectedListener(this); //ตรวจจับการเลือกเมนู
  }
```

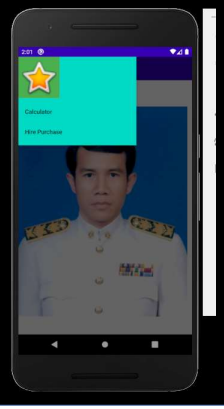
การสร้างเมนูแบบ Navigation Drawer ด้วยตนเอง

10. เขียนคำสั่งควบคุมใน MainActivity.java

```
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) { //ตรวจการคลิกปุ่ม Hamberger
  if (item != null && item.getItemId() == android.R.id.home) {
    if (drawerLayout.isDrawerVisible(Gravity.LEFT)) {
      drawerLayout.closeDrawer(Gravity.LEFT);
    } else {
      drawerLayout.openDrawer(Gravity.LEFT);
    }
  }
  return super.onOptionsItemSelected(item);
}
@Override
public boolean onNavigationItemSelected(@NonNull MenuItem item) {//ตรวจการเลือกเมนู
  switch (item.getItemId()) {
    case R.id.cal:
      Log.d("item: " , item.getTitle().toString());
      break;
    case R.id.hp:
      Log.d("item: " , item.getTitle().toString());
  }
  drawerLayout.closeDrawers();  //ปิดเมนู
  return false;
}
```

85

การสร้างเมนูแบบ Navigation Drawer ด้วยตนเอง



86

รายงาน

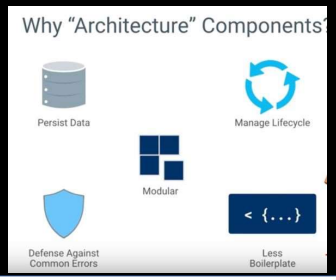
- แบ่งกลุ่มทำรายงาน กลุ่มละ 3 คน นำเสนอและสาธิตให้สมาชิกในห้องทำตาม ส่งเป็นไฟล์ Powerpoint ดังนี้
  1. Spinner นำเสนอวันที่ 26 ม.ค. 64
  2. CardView นำเสนอวันที่ 2 ก.พ. 64
  3. RecyclerView นำเสนอวันที่ 9 ก.พ. 64

87

การเขียนโปรแกรมติดต่อกับฐานข้อมูลบนแอนดรอยด์

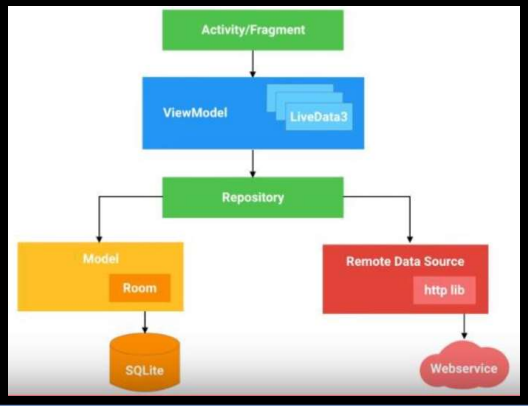
- Android Architecture Compoments
  - Architecture Components คือกลุ่ม Library ของ Google ซึ่งประกอบไปด้วย Room, LiveData, Lifecycle และ ViewModel ซึ่ง Library เหล่านี้จะช่วยจัดการเรื่อง Lifecycle และการเชื่อมต่อกับ Database ช่วยแยก app ให้เป็นสัดส่วน ง่ายต่อการทดสอบ ช่วยจัดการ error ที่เกิดขึ้น และช่วยลดคำสั่งที่ต้องเขียนซ้ำ ๆ ที่เรียกว่า boilerplate code

https://medium.com/@leelorz6/รู้จัก-Architecture-Component-ของใหม่จาก-Google-ที่จะมาช่วยจัดการ-project-ของคุณ- 85e48a6b8ff8



88

Android Architecture Components



89

การเชื่อมต่อกับฐานข้อมูลโดยใช้ Room

- Room เป็น Object Mapping Library สำหรับ SQLite ช่วยลด boilerplate code เวลาติดต่อกับ Database
- สมมติว่าต้องการ Table ที่ชื่อว่า Trail ที่ประกอบด้วย id, name, kilometers และ difficulty เราสามารถสร้างไฟล์ POJO (Plain Old Java Object) ขึ้นมา
- เราสามารถใช้ Room มาช่วยจัดการโดยใส่ @ annotation เป็น @Entity ด้านบนของ class POJO เพื่อบอก Room ว่าจะใช้ class นี้เป็น table และถ้าต้องการบอกว่า field ไหน เป็น primary key ก็สามารถใส่ @PrimaryKey หน้า field นั้น
- ต้องสร้างไฟล์ DAO (Data Access Object) เพื่อใช้จัดการเกี่ยวกับการติดต่อ Database แต่ละ annotation จะแสดงถึงการกระทำกับ Database ได้แก่ @Insert (ใส่ข้อมูลลงไปใน Database), @Update (อัพเดทข้อมูลใน Database), @Delete (ลบข้อมูลใน Database),@Query จะเป็นการใส่ query ที่อยากเขียนขึ้นมาเอง

Plain Old Java Object (POJO)

```
public class Trail {
    public String id;
    public String name;
    public double kilometers;
    public int difficulty;
}
```

Room Plain Old Java Object

```
@Entity
public class Trail {
    public @PrimaryKey String id;
    public String name;
    public double kilometers;
    public int difficulty;
}
```

TrailDao.java

```
@Dao
public interface TrailDao {
    // Create, read, update, delete examples
    @Insert(onConflict = IGNORE)
    void insertTrail(Trail trail);

    @Query("SELECT * FROM Trail")
    public List<Trail> findAllTrails();

    @Update(onConflict = REPLACE)
    void updateTrail(Trail trail);

    @Query("DELETE FROM Trail")
    void deleteAll();
}
```

90

Room

- Room จะรู้วิธีจัดการกับ POJO เช่น เวลา Insert ข้อมูล Room จะรู้ Type ของ POJO (ในที่นี้คือ Trail) ถ้าสั่ง select * from ก็จะได้ข้อมูลกลับคืนมาเป็น List (ในที่นี้คือ List ของ Trail) ซึ่งทั้งหมดนี้ Room จัดการให้ทั้งหมด
- ถ้า Select field ที่ไม่มีใน Database Room จะรับรู้ได้ และแสดง error ออกมา

TrailDao.java

```
// Create example
@Insert(onConflict = IGNORE)
void insertTrail(Trail trail);

// Read example
@Query("SELECT * FROM Trail")
public List<Trail> findAllTrails();
```

Checks SQLite at compile time

```
@Query("SELECT kilometerz FROM Trail")
public List<Double> findKilometerzForAllTrails();
```

There is a problem with the query: [SQLITE_ERROR] SQL error or missing da
Not sure how to convert a Cursor to this method's return type

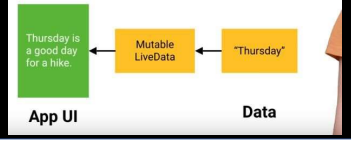
91

LiveData

- ช่วยรับรู้ถึงการเปลี่ยนแปลงของข้อมูล และ Lifecycle
- LiveData จะสังเกตุการเปลี่ยนแปลงของข้อมูล เมื่อข้อมูลมีการเปลี่ยนแปลง เราสามารถนำข้อมูลที่เปลี่ยนแปลงนั้นไปแสดงผลที่ UI (User Interface) ได้
- เราสามารถใช้ class MutableLiveData (LiveData ที่สามารถเซตค่าได้) เพื่อสังเกตการณ์เปลี่ยนแปลงของข้อมูลได้ เมื่อเซตค่าใหม่ให้ตัวแปร dayOfWeek ซึ่งเป็นตัวแปรประเภท MutableLiveData ก็จะอัพเดทค่าให้ UI โดยอัตโนมัติ

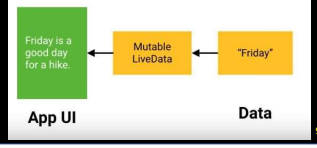
Using MutableLiveData

```
MutableLiveData<String> dayOfWeek = new MutableLiveData<>();
dayOfWeek.observe(this, data -> {
    mTextView.setText(dayOfWeek.getValue() +
                " is a good day for a hike.");});
```



Using MutableLiveData

```
// Elsewhere in the code!
dayOfWeek.setValue("Friday");
```



92

LiveData

- เราสามารถเปลี่ยน DAO เป็น LiveData ได้
- เมื่อมีการอัปเดตค่า Trail จะสามารถอัปเดต UI ได้อัตโนมัติ
- LiveData ยังรับรู้ถึงการเปลี่ยนแปลงของ Lifecycle ด้วย Lifecycle Aware Component โดยจะรู้ว่า UI อยู่บนหน้าจอหรือไม่ หรือโดนทำลายไปแล้ว ซึ่ง LiveData จะไม่ส่งข้อมูลให้กับหน้า UI ที่ไม่ได้ active อยู่

TrailDao.java

```
@Dao
public interface TrailDao {
  // Create, read, update, delete examples
  @Insert(onConflict = IGNORE)
  void insertTrail(Trail trail);

  @Query("SELECT * FROM Trail")
  public LiveData<List<Trail>> findAllTrails();

  @Update(onConflict = REPLACE)
  void updateTrail(Trail trail);

  @Query("DELETE FROM Trail")
  void deleteAll();
```

Using LiveData and Room

```
trailsLiveData.observe(this, trails -> {
  // Update UI, in this case a RecyclerView
  mTrailsRecyclerAdapter.replaceItems(trails);
  mTrailsRecyclerAdapter.notifyDataSetChanged();
});
```

Lifecycle Aware Component?
- On screen
- Off screen
- Destroyed

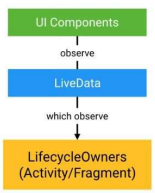
Lifecycle

- Component ที่ทำหน้าที่จัดการ Lifecycle มี 2 ประเภท
  1. Lifecycle Owners หมายถึง object ที่มี Lifecycle ได้แก่ Activity และ Fragment
  2. Lifecycle Observers หมายถึง object ที่คอยสังเกต Lifecycle Owners และจะถูกแจ้งเมื่อมีการเปลี่ยนแปลง Lifecycle จาก Lifecycle Owners ตัวอย่างของ Lifecycle Observers ได้แก่ LiveData
- LiveData นั้น implement interface LifecycleObserver ไว้ จึงสามารถรับรู้การเปลี่ยนแปลง LifeCycle จาก method startup() ซึ่งรับรู้ OnStart() ของ Lifecycle Owner และ cleanup() ซึ่งรับรู้ OnStop() ของ Lifecycle Owner
- Observing Flow จะใช้ UI คอยสังเกต LiveData ซึ่ง LiveData ก็จะคอยสังเกต Lifecycle Owners (Activity/Fragment) อีกที

LiveData.java

```
abstract public class LiveData<T> implements LifecycleObserver {
  @OnLifecycleEvent(Lifecycle.Event.ON_START)
  void startup() {...}

  @OnLifecycleEvent(Lifecycle.Event.ON_STOP)
  void cleanup() {...}
}
```

Observing Flow



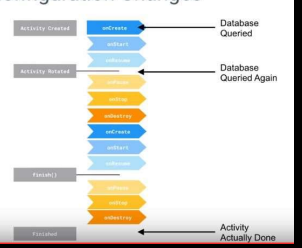
Lifecycle

- เราสามารถสร้าง Lifecycle Observer ขึ้นเองแบบ LiveData ได้
- เวลาที่มี Configuration Change (เช่นตอนหมุนจอ) LiveData จะถูกเรียกใหม่ทุกรอบ ซึ่งทำให้เกิดการ query database ซ้ำ ๆ ซึ่งแก้ไขได้โดยใช้ ViewModel

Your Library

```
MyLibraryClass implements LifecycleObserver {
  @OnLifecycleEvent(Lifecycle.Event.ON_START)
  void startup() {...}

  @OnLifecycleEvent(Lifecycle.Event.ON_STOP)
  void cleanup() {...}
}
```

Configuration Changes



ViewModel

- ViewModel ทำหน้าที่จัดการ Model หรือข้อมูลโดยเฉพาะ เมื่อมี Configuration Change ทำให้ข้อมูลที่เก็บไว้ใน ViewModel ไม่หายไปและสามารถเรียกข้อมูลที่เก็บไว้มาใช้ได้เมื่อต้องการ โดยไม่ต้อง query หรือดึงจาก api อีกรอบ
- ViewModel สร้างโดยไป extend class AndroidViewModel และใส่ส่วนการดึงข้อมูลจาก database ลงไป (LiveData ที่ feed ข้อมูลจาก Database)
- เมื่อ Activity หรือ Fragment ถูกสร้างขึ้นใหม่จาก Configuration Change เราสามารถดึงข้อมูลจาก ViewModel ได้
- ครั้งแรกที่เรียก ViewModel จาก Activity นั้น ViewModel จะถูกสร้างขึ้น เมื่อเกิด Configuration Change จะมีการเรียก ViewModel อีกรอบ ซึ่ง ViewModel จะไม่ได้ถูกสร้างใหม่ แต่จะเรียก ViewModel ตัวเดิม ทำให้ได้ข้อมูลที่เก็บไว้มา โดยไม่ต้อง query ข้อมูลจาก Database อีกรอบ

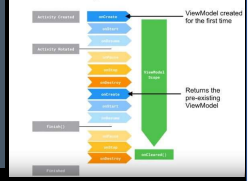
TrailListViewModel.java

```
public class TrailListViewModel extends AndroidViewModel {
  private AppDatabase mDatabase;
  private LiveData<List<Trail>> trails;

  public TrailListViewModel(Application application) {
    super(application);
    // AppDatabase is a Room database singleton
    // Check the guide for more information
    mDatabase = AppDatabase.getDb(getApplication());
    trails = mDatabase.trailModel().findAllTrails();
  }
  // Getters and setters
}
```

RecommendedTrailsActivity.java

```
// In onCreate
trailListViewModel = ViewModelProviders.of(this)
.get(TrailListViewModel.class);

// Code to set up the RecyclerView omitted
trailListViewModel.getTrails().observe(this, trails -> {
  mTrailsRecyclerAdapter.replaceItems(trails);
  mTrailsRecyclerAdapter.notifyDataSetChanged();
});
```

Rotate Away!



Workshop

1. สร้างโปรเจกต์ใหม่แบบ Basic Activity ตั้งชื่อว่า BasicRoom
2. เพิ่ม Dependencies
   1. ตัว Runtime คือ Androidx.room.room-runtime
   2. กลุ่มคลาสของ Android Architecture Components คือ android.room:room-compiler
3. สร้างคลาสสำหรับเก็บข้อมูลลูกค้า (POJO) ไว้ในแพ็กเกจย่อย db
   1. สร้างแพ็กเกจ db โดยคลิกขวาที่ java->com.xxx.basicroom เลือก New->Package กำหนดชื่อ db
   2. คลิกขวาที่ db เลือก New->Java Class กำหนดชื่อ Customer คลิก OK
4. สร้างส่วนจัดการ หรือเข้าถึงข้อมูลลูกค้า (DAO)
   1. คลิกขวาที่ db เลือก New->Java Class กำหนดชื่อ CustomerDAO เปลี่ยน Kind: เป็น Interface คลิก OK
5. สร้างฐานข้อมูลใน SQLite
   1. สร้างคลาสเพื่อทำหน้าที่สร้างฐานข้อมูล โดยคลิกขวาที่ db เลือก New->Java Class กำหนดชื่อ AppDatabase เปลี่ยน Modifiers: เป็น Abstract คลิก OK
6. สร้างส่วนติดต่อกับผู้ใช้สำหรับแสดงข้อมูล
   1. เปิด content_main.xml ลาก RecyclerView ไปวางบน Layout กำหนด id เป็น rvCustomerLists กำหนดจุดยึดทั้ง 4 ด้าน เลือก layout_width และ layout_height เป็น match_constraint
   2. สร้าง Layout สำหรับแสดงข้อมูลแต่ละแถวใน RecyclerView โดยคลิกขวาที่ Layout->New->Layout Resource File กำหนดชื่อเป็น rv_item_customer เปลี่ยน Root element: เป็น LinearLayout แบบ Vertical เลือก layout_width เป็น match_parent และ layout_height เป็น wrap_content
      1. ลาก CardView ไปวางบน Layout กำหนด id เป็น cvCustomer เลือก layout_width เป็น match_parent และ layout_height เป็น wrap_content
      2. ลาก LinearLayout แบบ Horizontal ไปวางบน CardView cvCustomer เลือก layout_width เป็น match_parent และ layout_height เป็น wrap_content
      1. ลาก TextView ไปวางบน LinearLayout กำหนด id เป็น tvFullName, layout_height เป็น wrap_content
      2. ลาก TextView ไปวางบน LinearLayout กำหนด id เป็น tvAddress, layout_height เป็น wrap_content

97

Workshop

7. สร้างตัวกลางดึงข้อมูลจากฐานข้อมูล SQLite ไปแสดงผลที่ UI
   1. สร้างแพ็กเกจ adapter โดยคลิกขวาที่ java->com.xxx.basicroom เลือก New->Package กำหนดชื่อ adapter
   2. คลิกขวาที่ adapter เลือก New->Java Class กำหนดชื่อ adt_rv_customer.java คลิก OK
8. สร้างส่วนติดต่อกับผู้ใช้สำหรับรับข้อมูล
   1. สร้าง Activity สำหรับรับข้อมูล โดยคลิกขวาที่ app>New-> Activity ->Empty Activity กำหนดชื่อเป็น CreateCustomer
   2. เปิด Layout activity_create_customer.xml และสร้างส่วนรับข้อมูล โดย
      1. ลาก Widget EditText มาวางใน Layout กำหนด id เป็น edtFullName
      2. ลาก Widget EditText มาวางใน Layout กำหนด id เป็น edtAddress
      3. ลาก Widget Button มาวางใน Layout กำหนด id เป็น cmdSave
      4. ลาก Widget Button มาวางใน Layout กำหนด id เป็น cmdClear

98

Customer.java

```
package chusak.basicroom.db;

import androidx.room.Entity;
import androidx.room.PrimaryKey;

@Entity
public class Customer {
    @PrimaryKey(autoGenerate = true)
    private int id;
    private String FullName;
    private String Address;
    public int getId(){
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
```

```
public String getFullName() {
        return FullName;
    }

    public void setFullName(String fullName) {
        FullName = fullName;
    }

    public String getAddress() {
        return Address;
    }

    public void setAddress(String address) {
        Address = address;
    }
}
```

99

CustomerDAO.java

```
package chusak.basicroom.db;

import androidx.room.Dao;
import androidx.room.Insert;
import androidx.room.OnConflictStrategy;
import androidx.room.Query;

import java.util.List;

@Dao
public interface CustomerDAO {
    @Query("SELECT * FROM Customer")
    List<Customer> getAllCustomer();
    @Insert(onConflict = OnConflictStrategy.REPLACE)
    void create(Customer customer);
}
```

100

AppDatabase.java

```
package chusak.basicroom.db;
import android.content.Context;
import androidx.room.Database;
import androidx.room.Room;
import androidx.room.RoomDatabase;
@Database(entities = {Customer.class}, version = 1)
public abstract class AppDatabase extends RoomDatabase {
    private static final String DATABASE_NAME = "myshop";
    private static AppDatabase db;
    public abstract CustomerDAO customerDAO();
    public static AppDatabase getInstance(Context context) {
        if (db==null) {
            db = Room.databaseBuilder(context.getApplicationContext(),AppDatabase.class, DATABASE_NAME)
                    .allowMainThreadQueries()
                    .build();
        }
        return db;
    }
    public static void desInstance() {
        db = null;
    }
}
```

101

adt_rv_customer.java

```
package chusak.basicroom.adapter;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;
import java.util.List;
import chusak.basicroom.R;
import chusak.basicroom.db.Customer;
public class adt_rv_customer extends
RecyclerView.Adapter<adt_rv_customer.ViewHolder> {
    private List<Customer> customers;
    public adt_rv_customer(List<Customer> c) {
        this.customers = c;
    }
    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent,
int viewType) {
        View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.rv_item_cust
omer, parent, false);
        return new ViewHolder(view);
    }
```

```
@Override
public void onBindViewHolder(@NonNull ViewHolder holder, int
position) {
  holder.tvFullName.setText(customers.get(position).getFullName());
  holder.tvAddress.setText(customers.get(position).getAddress());
}
@Override
public int getItemCount() {
  return customers.size();
}

public class ViewHolder extends RecyclerView.ViewHolder {
  public TextView tvFullName;
  public TextView tvAddress;
  public ViewHolder(View itemView) {
    super(itemView);
    tvFullName = itemView.findViewById(R.id.tvFullName);
    tvAddress = itemView.findViewById(R.id.tvAddress);
  }
 }
}
```

102

CreateCustomer.java

```
package chusak.basicroom;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import chusak.basicroom.db.AppDatabase;
import chusak.basicroom.db.Customer;
public class CreateCustomer extends AppCompatActivity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_create_customer);
    final EditText edtFullName = findViewById(R.id.edtFullName);
    final EditText edtAddress = findViewById(R.id.edtAddress);
    Button cmdSave = findViewById(R.id.cmdSave);
    cmdSave.setOnClickListener(new View.OnClickListener() {
    @Override
```

```
public void onClick(View v) {
    String strFullName = edtFullName.getText().toString().trim();
    String strAddress = edtAddress.getText().toString().trim();
    if (TextUtils.isEmpty(strFullName)) return;
    if (TextUtils.isEmpty(strAddress)) return;
    Customer c = new Customer();
    c.setFullName(strFullName);
    c.setAddress(strAddress);
    AppDatabase.getInstance(CreateCustomer.this).customerDAO().create(c);
    edtFullName.getText().clear();
    edtAddress.requestFocus();
    startActivity(new Intent(CreateCustomer.this, MainActivity.class));
 }});
Button cmdClear = findViewById(R.id.cmdClear);
cmdClear.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
  edtFullName.setText("");
  edtAddress.setText("");
}});}
@Override
protected void onDestroy() {
  AppDatabase.desInstance();
  super.onDestroy();
 }
}
```

103

MainActivity.java

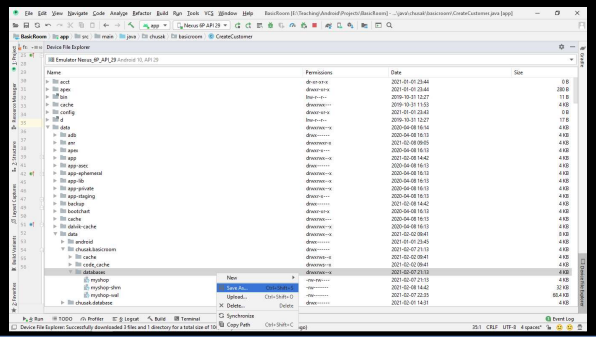
```
public class MainActivity extends AppCompatActivity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Toolbar toolbar = findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);
    FloatingActionButton fab = findViewById(R.id.fab);
    RecyclerView rv =
findViewById(R.id.rvCustomerLists);
    RecyclerView.Adapter adt = new
adt_rv_customer(getCustomerLists());
    rv.setLayoutManager(new
LinearLayoutManager(MainActivity.this));
    rv.setAdapter(adt);
    fab.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {
        startActivity(new Intent(MainActivity.this,
CreateCustomer.class));
      }});}
```

```
private List<Customer> getCustomerLists() {
  return AppDatabase.getInstance(this).customerDAO().
getAllCustomer();
}
@Override
public boolean onCreateOptionsMenu(Menu menu) {
  getMenuInflater().inflate(R.menu.menu_main, menu);
  return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
  int id = item.getItemId();
  if (id == R.id.action_create) {
    return true;
  }
  return super.onOptionsItemSelected(item);
}
@Override
protected void onDestroy() {
  AppDatabase.desInstance();
  super.onDestroy();
}
}
```

104

การนำฐานข้อมูล SQLite ออกจาก Emulator

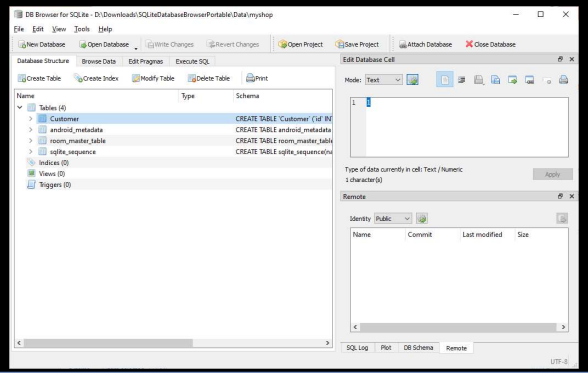
1. เปิดหน้าต่าง Device File Explorer โดยคลิกที่มุมขวาล่างของหน้าต่าง Android Studio
2. คลิกขวาที่ data->data->databases เลือก Save As...



105

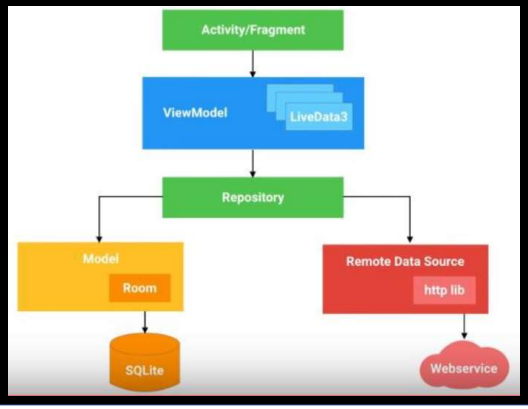
เครื่องมือจัดการฐานข้อมูล SQLite

- SQLite Browser ดาวน์โหลดที่ https://sqlitebrowser.org/dl/



106

Android Architecture Components



107

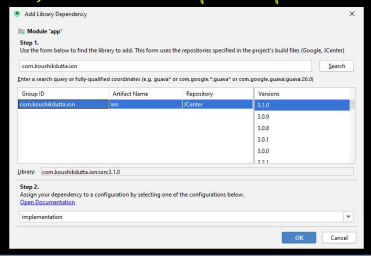
Remote Data Source Tools

- Retrofit
- Ion (koushikdutta)
- OkHttp
- Volley

108

การใช้ Ion (koushikdutta)

1. เพิ่ม Dependency Ion (koushikdutta) เข้ามาในโปรเจกต์
   1. คลิก Project Structure
   2. คลิก Add Library Dependency
   3. พิมพ์ com.koushikdutta.ion แล้วคลิกปุ่ม Search
   4. จะปรากฏชื่อ Dependency ขึ้นมา ให้เลือกเวอร์ชันล่าสุด แล้วคลิกปุ่ม OK



การใช้ Ion (koushikdutta) เพื่อเรียกใช้ข้อมูลผ่านอินเทอร์เน็ต

1. เพิ่ม uses-permission ในไฟล์ AndroidManifest.xml เพื่อให้แอปพลิเคชันเชื่อมต่อกับอินเทอร์เน็ตได้

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.somsakelect.sqltesting">
    <uses-permission android:name="android.permission.INTERNET"/>
    <application
        ...
</manifest>
```

2. เพิ่ม compileOptions ในไฟล์ build.gradle (Module: app)

```
android {
    compileSdkVersion 29
    buildToolsVersion "29.0.3"
    compileOptions {
      sourceCompatibility JavaVersion.VERSION_1_8
      targetCompatibility JavaVersion.VERSION_1_8
    }
    ...
```


ตัวอย่างการใช้ Ion (koushikdutta) เพื่อแสดงไฟล์ภาพจากอินเทอร์เน็ต

```
ImageView imageView = findViewById(R.id.imageView);
Ion.getDefault(this).getConscryptMiddleware().enable(false);
Ion.with(this)
        .load("http://khmersilk.bru.ac.th/images/4.1.jpg") //โหลดภาพจากเว็บไซต์
        .withBitmap()
        .centerCrop()
        .intoImageView(imageView); //นำภาพที่โหลดไปแสดงใน imageView
```


ตัวอย่างการใช้ Ion (koushikdutta) เพื่อแสดงข้อมูลจากเว็บเซิร์ฟเวอร์

1. เขียนสคริปต์ไว้ที่ฝั่งเซิร์ฟเวอร์เพื่อดึงข้อมูลจากฐานข้อมูล และส่งกลับไปยังแอปพลิเคชันในรูปแบบที่ต้องการ

selectprovince.php

```
<?php
  require("include/connect.php");
  $conn = new mysqli($servername, $username, $password, $dbname);
  if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
  }
  $sql = "SELECT pid AS id,thai AS name FROM province ORDER BY name";
  $conn->set_charset("utf8");
  $result = $conn->query($sql);
  $itms=array();  //กำหนดอาร์เรย์เพื่อเก็บข้อมูลจากตาราง
  while($row = $result->fetch_assoc()) {
    $itms[]=$row;  //อ่านข้อมูลแต่ละแถวใส่อาร์เรย์
  }
  $conn->close();
  print json_encode($itms);  //ส่งข้อมูลกลับไปยังแอปพลิเคชันที่เรียกในรูปแบบ JSON Array
?>
```

ตัวอย่างการใช้ Ion (koushikdutta) เพื่อแสดงข้อมูลจากเว็บเซิร์ฟเวอร์

2. เขียนคำสั่งเรียกใช้สคริปต์เพื่อดึงข้อมูลมาแสดงในแอปพลิเคชัน

```
Spinner province = findViewById(R.id.spinnerProvince); //กำหนด Spinner เพื่อแสดงข้อมูล
Ion.getDefault(this).getConscryptMiddleware().enable(false);
Ion.with(this)
  .load("http://safeagro.bru.ac.th/selectprovince.php") //เรียกสคริปต์เพื่อดึงข้อมูลจากเซิร์ฟเวอร์
  .asJsonArray() //กำหนดรูปแบบข้อมูลที่รับมาเป็นแบบ JSON Array
  .setCallback(new FutureCallback<JsonArray>() {
      @Override
      public void onCompleted(Exception e, JsonArray result) {
        ArrayList<String> provinceList = new ArrayList<>(); //กำหนด ArrayList สำหรับเก็บข้อมูล
        for (int i=0; i<result.size(); i++) { //อ่านข้อมูลจาก JSON Array ใส่ลงใน ArrayList
           provinceList.add(result.get(i).getAsJsonObject().get("thai").getAsString());
        }
        province.setAdapter(new ArrayAdapter<String>
           (MainActivity.this,R.layout.support_simple_spinner_dropdown_item,provinceList));
      } //ใส่ ArrayList ลงใน Spinner
  });
```

113

ตัวอย่างการใช้ Ion (koushikdutta) เพื่อเพิ่มข้อมูลในเว็บเซิร์ฟเวอร์

1. เขียนสคริปต์ไว้ที่ฝั่งเซิร์ฟเวอร์เพื่อรับข้อมูลจากแอปพลิเคชัน และเพิ่มลงในฐานข้อมูล

addUser.php

```
<?php
  require("include/connect.php");
  $conn = new mysqli($servername, $username, $password, $dbname);
  if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
  }
  $name = $_POST['name'];  //รับข้อมูลจากแอปพลิเคชันที่ส่งมาแบบ POST
  $pass = $_POST['password'];
  $sql = "INSERT INTO users (id, name, password) VALUES (NULL, '$name', '$pass')";
  if ($conn->query($sql) === TRUE) { //รันคำสั่งเพิ่มข้อมูล
    echo "New record created successfully"; //ส่งผลลัพธ์กลับแบบ String
  } else {
    echo "Error: " . $sql . "<br>" . $conn->error; //ส่งผลลัพธ์กลับแบบ String
  }
  $conn->close();
?>
```

114

ตัวอย่างการใช้ Ion (koushikdutta) เพื่อเพิ่มข้อมูลในเว็บเซิร์ฟเวอร์

2. เขียนคำสั่งเรียกใช้สคริปต์เพื่อส่งข้อมูลไปยังเว็บเซิร์ฟเวอร์

```
Ion.with(MainActivity.this)
  .load("http://10.133.0.164/android/addUser.php") //เรียกใช้สคริปต์บนเว็บเซิร์ฟเวอร์
  .setBodyParameter("name", name) //ส่งค่าให้ name ไปยังเว็บเซิร์ฟเวอร์แบบ POST กำหนดชื่อ name
  .setBodyParameter("password", pass)
  .asString() //รับค่าส่งกลับจากเว็บเซิร์ฟเวอร์เป็น String
  .setCallback(new FutureCallback<String>() {
     @Override
     public void onCompleted(Exception e, String result) {
       if(result!=null){
         Toast.makeText(MainActivity.this, result, Toast.LENGTH_LONG).show(); //แสดงค่าที่รับมา
                                                                              //จากเว็บเซิร์ฟเวอร์
       } else {                                                              //แบบ String
         Toast.makeText(MainActivity.this, "Error: "+e, Toast.LENGTH_LONG).show();
       }
     }
  });
```

115

ตัวอย่างการใช้ Ion (koushikdutta) เพื่อลบข้อมูลในเว็บเซิร์ฟเวอร์

1. เขียนสคริปต์ไว้ที่ฝั่งเซิร์ฟเวอร์เพื่อรับข้อมูลจากแอปพลิเคชัน และลบในฐานข้อมูล

deleteUser.php

```
<?php
  require("include/connect.php");
  $conn = new mysqli($servername, $username, $password, $dbname);
  if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
  }
  $userId = $_POST['userId']; //รับข้อมูลจากแอปพลิเคชันที่ส่งมาแบบ POST
  $sql = "DELETE FROM users WHERE id='$userId'";
  if ($conn->query($sql) === TRUE) {//รันคำสั่งลบข้อมูล
      echo "Delete successfully"; //ส่งผลลัพธ์กลับแบบ String
  } else {
      echo "Error: " . $sql . "<br>" . $conn->error; //ส่งผลลัพธ์กลับแบบ String
  }
  $conn->close();
?>
```

116

ตัวอย่างการใช้ Ion (koushikdutta) เพื่อลบข้อมูลในเว็บเซิร์ฟเวอร์

2. เขียนคำสั่งเรียกใช้สคริปต์เพื่อส่งข้อมูลไปยังเว็บเซิร์ฟเวอร์

```
Ion.with(MainActivity.this)
  .load("http://10.133.0.164/android/deleteUser.php")  //เรียกใช้สคริปต์บนเว็บเซิร์ฟเวอร์
  .setBodyParameter("userID", id) //ส่งค่าใน id ไปยังเว็บเซิร์ฟเวอร์แบบ POST กำหนดชื่อ userID
  .asString() //รับค่าส่งกลับจากเว็บเซิร์ฟเวอร์เป็น String
  .setCallback(new FutureCallback<String>() {
     @Override
     public void onCompleted(Exception e, String result) {
        if(result!=null){
          Toast.makeText(MainActivity.this, result, Toast.LENGTH_LONG).show(); //แสดงค่าที่รับมา
                                                                                //จากเว็บเซิร์ฟเวอร์
        } else {                                                                //แบบ String
          Toast.makeText(MainActivity.this, "Error: "+e, Toast.LENGTH_LONG).show();
        }
     }
  });
```

117

แหล่งข้อมูลเพิ่มเติมเกี่ยวกับ Ion (koushikdutta)

- http://www.somsakelect.com/2019/03/18/android-mysql
- https://github.com/koush/ion

118