บทที่ 8

การจัดเก็บข้อมูลในเครื่องด้วย SQLite และ Room

อ.ดร.พงษ์ดนัย จิตตวิสุทธิกุล

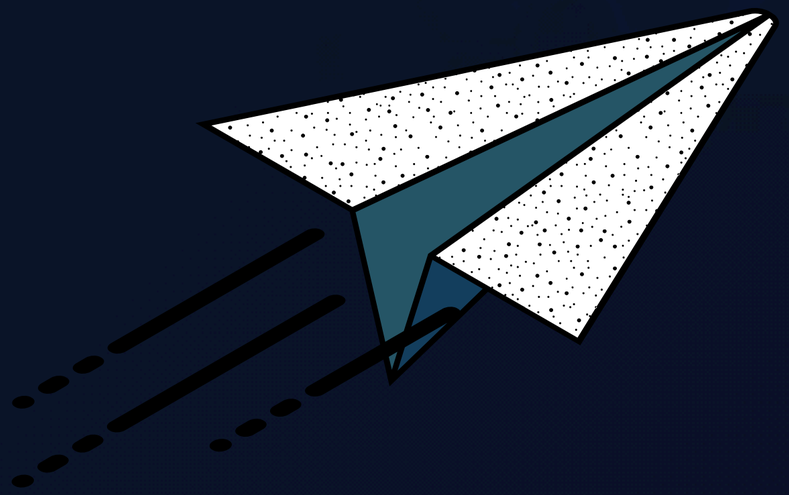
Topic

3 หัวข้อหลัก:

- ความรู้เบื้องต้นเกี่ยวกับ SQLite และ Room
- สาธิตการใช้งาน Room
- ปฏิบัติการ: สร้างแอป To-Do List ด้วย Room

Objective behavior

3 วัตถุประสงค์เชิงพฤติกรรม:



อธิบายแนวคิดและระบุหน้าที่ของส่วนประกอบหลักใน Room Persistence Library ได้

ประยุกต์ใช้ Annotation และไวยากรณ์ของ Room ในการสร้าง Entity, DAO และ Database ได้อย่างถูกต้อง

อธิบายสถาปัตยกรรมภาพรวมของแอปพลิเคชันและขั้นตอนการตั้งค่าโปรเจกต์เพื่อใช้งาน Room ได้

จาก "ความจำสั้น" สู่ "ความจำระยะยาว"

ข้อจำกัดของ State

ในบทที่ 7 เราใช้ State เพื่อให้หน้าจอจดจำข้อมูล แต่มันคือ "ความจำระยะสั้น" ที่เก็บไว้ใน RAM เมื่อปิดแอพหรือรีสตาร์ทเครื่อง ข้อมูลจะหายไปทั้งหมด

การจัดเก็บข้อมูลถาวร

บทนี้เราจะมอบ **"ความจำระยะยาว"** ให้แอพ โดยเรียนรู้วิธีการบันทึกข้อมูลลงในหน่วยความจำของโทรศัพท์ เพื่อให้ข้อมูลยังคงอยู่แม้ปิดแอพไปแล้ว

SQLite: "สมุดบันทึก" ประจำโทรศัพท์

SQLite คืออะไร?

ระบบฐานข้อมูลเชิงสัมพันธ์: เก็บข้อมูลเป็น "ตาราง" (แถวและคอลัมน์) คล้าย Excel

ไม่ต้องใช้เซิร์ฟเวอร์: ทำงานภายในแอพ ไม่ต้องต่อเน็ต

ไฟล์เดียวจบ: ทุกตารางและข้อมูลถูกเก็บในไฟล์เดียว จัดการง่าย

ปัญหาของการเขียนแบบดั้งเดิม

เขียน SQL เองทั้งหมด: เสี่ยงต่อการพิมพ์ผิด

ไม่มีการตรวจจับตอนคอมไพล์: พิมพ์ชื่อตารางผิด โปรแกรมยังรันผ่าน แต่จะไปแครชตอนใช้งานจริง

โค้ดเยิ่นเย้อ (Boilerplate): ต้องเขียนโค้ดแปลงข้อมูลกลับไปมาซับซ้อน

Room: "ปากกาอัจฉริยะ"

เพื่อแก้ปัญหาของ SQLite Google จึงสร้าง **Room Persistence Library** ขึ้นมา

Room ไม่ใช่ฐานข้อมูลใหม่

มันคือ Abstraction Layer (ชั้นนามธรรม) ที่ครอบทับ SQLite ไว้อีกที

เปรียบเทียบให้เห็นภาพ

ถ้า SQLite คือ "สมุดบันทึก" ปกติ... Room คือ **"ปากกาอัจฉริยะ และชุดแม่แบบ"** ที่ช่วยให้เราเขียนลงสมุดได้ง่ายขึ้น เป็นระเบียบ และปลอดภัยจากข้อผิดพลาด

ประโยชน์หลัก

ตรวจสอบคำสั่ง SQL ให้ตั้งแต่ตอนเขียนโค้ด

3 ส่วนประกอบหลักของ Room

1. Entity

"แม่แบบของหน้ากระดาษ"

คือ Data Class ที่กำหนดโครงสร้างของ "ตาราง" แต่ละตัวแปร (Property) ในคลาสจะถูกแปลงเป็น "คอลัมน์" ใช้ Annotation @Entity

2. DAO

"ชุดคำสั่งของปากกา"

Data Access Object เป็น Interface ที่กำหนดวิธีการจัดการข้อมูล (เพิ่ม, ลบ, ค้นหา) โดยใช้ @Insert, @Query Room จะสร้างโค้ด SQL ให้เอง

3. Database

"สมุดบันทึกทั้งเล่ม"

เป็น Abstract Class ศูนย์กลางที่เชื่อมโยง Entity และ DAO เข้าด้วยกัน ใช้ Annotation @Database เพื่อบอกว่ามีตารางอะไรบ้าง

ปฏิบัติการ: สร้างแอพ To-Do List ด้วย Room

ประยุกต์ใช้ Jetpack Compose + State + Room
เพื่อสร้างแอพบันทึกงานที่ข้อมูลไม่สูญหาย

ขั้นตอนที่ 1: การตั้งค่า (Dependencies)

เหมือนการเข้าครัว เราต้องเพิ่มเครื่องมือลงในแคตตาล็อก libs.versions.toml ก่อน

- [versions] และ [plugins] เพิ่มเวอร์ชันของ room, ksp (Kotlin Symbol Processing) และตั้งค่า Plugin com.google.devtools.ksp
- [libraries] เพิ่มไลบรารี room-runtime, room-compiler, room-ktx และ lifecycle-viewmodel-compose
- build.gradle.kts (Module :app) นำ Plugin KSP มาใช้งาน และใส่ implementation(...) เข้าไปในส่วน dependencies จากนั้นกด Sync Now

ขั้นตอนที่ 2 & 3: สร้าง Entity และ DAO

1. สร้าง Entity (Task.kt)

กำหนดโครงสร้างตารางข้อมูลงาน

```
@Entity(tableName = "tasks_table") data class
Task( @PrimaryKey(autoGenerate = true) val id:
Int = 0, val title: String )
```

2. สร้าง DAO (TaskDao.kt)

กำหนดคำสั่งที่ใช้ติดต่อฐานข้อมูล

```
@Dao interface TaskDao { @Insert suspend fun
insertTask(task: Task) @Query("SELECT * FROM
tasks_table ... ") fun getAllTasks(): Flow> }
```

*Flow ช่วยให้ UI อัปเดตอัตโนมัติเมื่อข้อมูลเปลี่ยน

ขั้นตอนที่ 4 & 5: Database และ ViewModel

ประกอบ "สมุดบันทึก"

สร้างคลาสศูนย์กลางสืบทอดจาก RoomDatabase ระบุ Entity ที่ใช้งาน และสร้าง companion object (Singleton) เพื่อให้มีฐานข้อมูลแค่ก้อนเดียวในแอพ

```
@Database(entities = [Task::class], version = 1)
```

สร้าง "ผู้จัดการ" (ViewModel)

สร้าง TaskViewModel เป็นตัวกลางระหว่าง UI (หน้าจอ) และ DAO (ฐานข้อมูล) ป้องกันไม่ให้ UI เรียกใช้ฐานข้อมูลโดยตรง ทำให้โค้ดสะอาดและจัดการง่าย

ผลลัพธ์: To-Do App

ใน MainActivity.kt เราสร้าง UI ด้วย Compose และเชื่อมกับ ViewModel

ลำดับการทำงาน (Data Flow)

1.UI: ผู้ใช้พิมพ์ชื่องานและกด Add

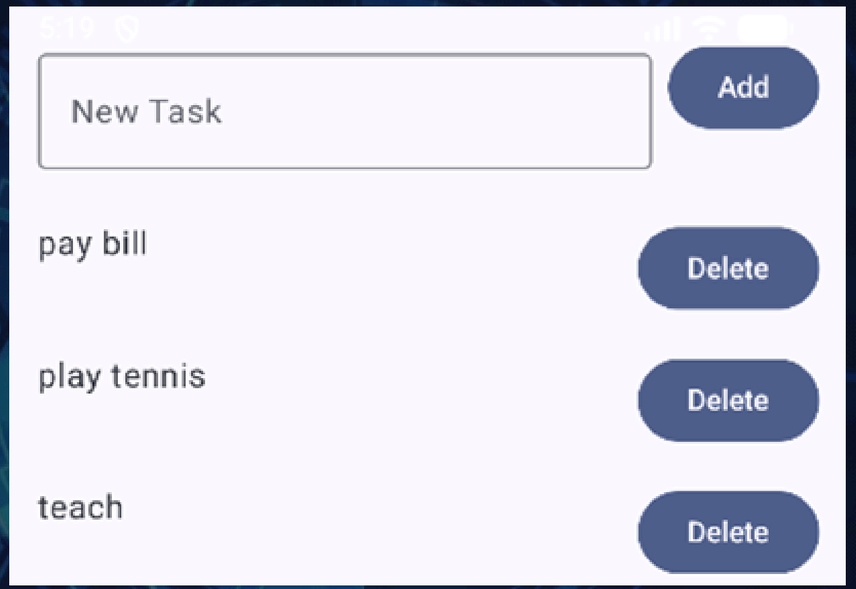
2.ViewModel: รับคำสั่ง และตรวจสอบข้อมูล

3.DAO: สั่ง เพื่อเขียนลงเครื่อง

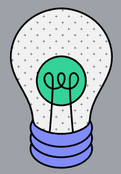
4.Database: Room แปลงข้อมูลลงตาราง

5.Flow: ส่งข้อมูลชุดใหม่กลับมา อัปเดต UI ทันที!

ทดสอบปิดแอพแล้วเปิดใหม่ ข้อมูลจะไม่หายไป!



Questions & Discussion



วันนี้เราได้เรียนรู้การให้ "ความทรงจำระยะยาว"
กับแอพด้วย SQLite และ Room
ทำความเข้าใจ Entity, DAO, Database
และสถาปัตยกรรม ViewModel