บทที่ 9

การสื่อสารกับโลกภายนอก - REST APIs

อ.ดร.พงษ์ดนัย จิตตวิสุทธิกุล

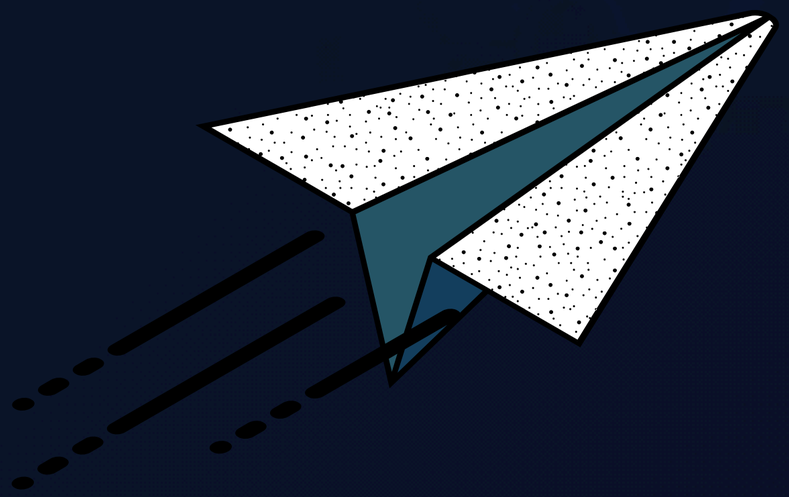
Topic

5 หัวข้อหลัก:

- ความรู้เบื้องต้นเกี่ยวกับ API และสถาปัตยกรรม REST
- กลไกการสื่อสาร: HTTP Methods และรูปแบบข้อมูล JSON
- การเรียกใช้ REST API ใน Android ด้วยไลบรารี Retrofit
- การจัดการ Asynchronous Operations ด้วย Kotlin Coroutines
- ปฏิบัติการในห้องเรียน: สร้างแอพพลิเคชัน แสดงรายการข้อมูลจาก Public API

Objective behavior

3 วัตถุประสงค์เชิงพฤติกรรม:



อธิบายหลักการพื้นฐาน 3 ประการ (Client-Server, Stateless, Cacheable) และเลือกใช้ HTTP Method ได้ถูกต้อง

ระบุส่วนประกอบหลักที่จำเป็นในการสร้าง Network Request (API Interface, Instance, Model, Scope) ได้

ตั้งค่าโปรเจกต์ด้วย Version Catalogs เพื่อดึงข้อมูล JSON จาก Public API มาแสดงใน RecyclerView ได้สำเร็จ

API คืออะไร? (เปรียบเทียบกับร้านอาหาร)

API (Application Programming Interface) คือ "ตัวกลาง" หรือ "ข้อตกลง" ที่ให้ซอฟต์แวร์ 2 ตัวคุยกันได้

 **ลูกค้า (Client) = แอปของเรา**

มีความต้องการ (หิว) ต้องการสั่งข้อมูล (อาหาร)

 **ห้องครัว (Server) = ฐานข้อมูลบนเน็ต**

มีข้อมูลและวัตถุดิบทั้งหมด แต่ลูกค้าเข้าไปหยิบเองไม่ได้

 **พนักงานเสิร์ฟ (API) = ตัวกลาง**

รับคำสั่งจากเราไปบอกครัว และนำผลลัพธ์ (Response) มาเสิร์ฟ

สถาปัตยกรรม REST

REST (Representational State Transfer) ไม่ใช่ตัวโปรแกรม แต่เป็น "สไตล์การออกแบบ API" ที่มีกฎระเบียบชัดเจน

1. Client-Server

แบ่งแยกหน้าที่ชัดเจนระหว่างผู้ขอบริการ (แอป) และผู้ให้บริการ (เซิร์ฟเวอร์) ทำให้พัฒนาได้อย่างอิสระ

2. Stateless

"ไร้สถานะ" เซิร์ฟเวอร์จะไม่จำคำขอก่อนหน้า ทุกคำขอต้องส่งข้อมูลไปให้ครบถ้วนเสมอ ทำให้ระบบขยายตัว (Scale) ได้ง่าย

3. Cacheable

สามารถจัดเก็บข้อมูลสำรอง (Cache) ไว้ใช้ซ้ำได้ เพื่อลดภาระเซิร์ฟเวอร์และทำให้แอปโหลดเร็วขึ้น

กลไกการสื่อสาร: HTTP Methods (CRUD)

เราใช้มาตรฐาน HTTP ในการบอกพนักงานเสิร์ฟ (API) ว่าเราต้องการ "ทำอะไร" กับข้อมูล

GET (Read)

ขอดูเมนู: ใช้สำหรับ "ดึงข้อมูล" จากเซิร์ฟเวอร์ โดยไม่ทำให้ข้อมูลบนเซิร์ฟเวอร์เปลี่ยนแปลง

POST (Create)

สั่งอาหาร: ใช้สำหรับ "สร้างข้อมูลใหม่" ส่งข้อมูลไปยังเซิร์ฟเวอร์ (เช่น สมัครสมาชิกใหม่)

DELETE (Delete)

ยกเลิกออเดอร์: ใช้สำหรับ "ลบข้อมูล" ออกจากระบบ

PUT / PATCH (Update)

เปลี่ยนออเดอร์: ใช้สำหรับ "แก้ไข/อัปเดตข้อมูล" (PUT = เปลี่ยนทั้งชุด, PATCH = เปลี่ยนบางส่วน)

รูปแบบข้อมูล: JSON

JavaScript Object Notation

อาหารที่เสิร์ฟกลับมาจะอยู่ในรูปแบบ JSON ซึ่งเป็นมาตรฐานสากล เพราะน้ำหนักเบาและคนอ่านเข้าใจได้ง่าย

- **Object:** อยู่ในปีกกา { } ทำงานแบบ Key/Value
- **Array:** อยู่ในวงเล็บเหลี่ยม [ ] สำหรับข้อมูลที่เป็นรายการ (List)
- **Primitive:** รองรับ String, Number, Boolean, Null


```json
{
  "page": 1,
  "results": [
    {
      "id": 533535,
      "title": "Deadpool & Wolverine",
      "vote_average": 8.5
    },
    {
      "id": 718821,
      "title": "Twisters",
      "vote_average": 7.9
    }
  ]
}
```

เครื่องมือคู่ใจ: ไลบรารี Retrofit

Retrofit คือ ไลบรารีที่พัฒนาโดยบริษัท Square ที่ช่วยเปลี่ยนการต่อ API ที่ซับซ้อนให้กลายเป็นเรื่องง่าย

Type-safe HTTP Client เชื่อมต่อเครือข่ายอย่างปลอดภัย ป้องกันปัญหาการแปลงชนิดข้อมูลผิดพลาด

เปลี่ยน API ให้เป็น Kotlin Interface เราแค่เขียน Interface และแปะ Annotation (เช่น @GET) Retrofit จะสร้างโค้ดเชื่อมต่อให้เองทั้งหมด

ทำงานร่วมกับ Converter ได้ สามารถติดตั้ง "นักแปลภาษา" (เช่น Kotlinx Serialization หรือ Gson) เพื่อแปลง JSON เป็น Data Class ของแอปได้อัตโนมัติ

การจัดการงานด้วย Coroutines (Asynchronous)

ทำไมต้อง Asynchronous?

การโหลดข้อมูลจากเน็ตใช้เวลา ถ้ารอใน Main Thread (พนักงานต้อนรับ) แอปจะค้าง หรือเกิดอาการ ANR (App Not Responding)

Kotlin Coroutines

โซลูชันของ Google สำหรับจัดการงานเบื้องหลัง เปรียบเสมือน "เธรดน้ำหนักเบา"

- **suspend fun:** ฟังก์ชันที่หยุดพักได้เพื่อรอข้อมูล โดยไม่บล็อกเธรดหลัก
- **viewModelScope:** จัดการขอบเขตการทำงาน ถ้าย้ายหน้าจอจะยกเลิกงานอัตโนมัติ ป้องกัน Memory Leak

ปฏิบัติการ: MovieDB Demo

มาสร้างแอปดึงข้อมูลภาพยนตร์ยอดฮิตของจริงกัน!

เป้าหมายของแล็บ

- สมัครขอรับ API Key จาก The Movie Database (TMDb)
- ตั้งค่า Dependencies อย่างถูกต้อง (Version Catalogs)
- ดึงข้อมูลหนังผ่าน Retrofit & Coroutines
- นำข้อมูลมาแสดงผลในหน้าจอด้วย **RecyclerView**

ขั้นตอนที่ 1: การตั้งค่าโปรเจกต์ (Setup)

1. ขอ Internet Permission

ในไฟล์ AndroidManifest.xml

```
"android.permission.INTERNET" />
```

2. ตั้งค่า libs.versions.toml

เพิ่มไลบรารีที่จำเป็นทั้งหมดลงในแคตตาล็อก

- ✓ Retrofit (สำหรับการสื่อสาร)
- ✓ Kotlinx Serialization (ตัวแปลง JSON)
- ✓ Coroutines (จัดการ Asynchronous)
- ✓ RecyclerView (สร้างรายการ List)

```
// ตัวอย่างการเพิ่มใน build.gradle.kts (app)
dependencies {
implementation(libs.androidx.recyclerview)
implementation(libs.retrofit)
implementation(libs.retrofit.converter.kotlinx)
implementation(libs.kotlinx.serialization.json)
implementation(libs.kotlinx.coroutines.android)
implementation(libs.androidx.lifecycle.viewmodel) }
```

ขั้นตอนที่ 2 & 3: Model และ API Interface

Data Model (Kotlin Class)

สร้างโครงสร้างรองรับ JSON ที่ได้กลับมา

```kotlin
@Serializable
data class Movie(
  @SerialName("id") val id: Int,
  @SerialName("title") val title: String,
  @SerialName("overview") val overview: String
)

@Serializable
data class PopularMoviesResponse(
  @SerialName("results") val movies: List<Movie>
)
```

API Interface & Retrofit

กำหนดจุด Endpoint (เมนูอาหาร) ให้ Retrofit

```kotlin
interface TMDbApi {
  @GET("movie/popular")
  suspend fun getPopularMovies(
    @Query("api_key") apiKey: String,
    @Query("page") page: Int
  ): PopularMoviesResponse
}

// ใน RetrofitInstance ...
.addConverterFactory(json.asConverterFactory( ... ))
```

ขั้นตอนที่ 4 & 5: ViewModel และ UI

ViewModel (คนจัดการงาน)

ดึงข้อมูลใน Background Thread

```
fun fetchPopularMovies() {
  viewModelScope.launch {
    try {
      val response = api.getPopularMovies("KEY",
1)
      _movies.postValue(response.movies)
    } catch (e: Exception) {
      _errorMessage.postValue(e.message)
    }
  }
}
```

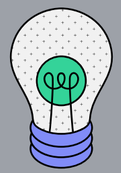
UI (MainActivity & RecyclerView)

คอยสังเกต (Observe) ข้อมูลเพื่ออัปเดตหน้าจอ

```
viewModel.movies.observe(this) { movies →
  // ข้อมูลหนังเปลี่ยน อัปเดต Adapter
  moviesAdapter.updateMovies(movies)
}

viewModel.fetchPopularMovies() // สั่งเริ่มดึงข้อมูล
```

Questions & Discussion



วันนี้เราได้เรียนรู้การทำให้แอปคุยกับ
เซิร์ฟเวอร์ด้วย REST API
ผ่านการใช้งาน Retrofit และจัดการงาน
เบื้องหลังด้วย Coroutines อย่างมืออาชีพ