เอกสารประกอบการสอน Workshop “การพัฒนา MVC Web App – Student Check-in by Geolocation (HTML5 + JS + Spring Boot)”

รายละเอียดกรณีศึกษา User Story “เช็คชื่อเข้าชั้นเรียน Attendance Check-in via Geolocation”

ตัวอย่าง User Story Details

User Story: เช็คชื่อเข้าชั้นเรียน (**Attendance Check-in via Geolocation)**

As a student ในฐานะนักศึกษา
I want to check in to my class using my current geolocation,
ฉันต้องการเช็คชื่อเข้าห้องเรียนโดยใช้จุดพิกัดละติจูด ลองจิจูดปัจจุบันขณะเช็คชื่อ
So that the system can confirm that I am physically present within the designated classroom area.
เพื่อที่จะให้ระบบตรวจเช็คจุดพิกัดขณะเช็คชื่อเข้าเรียนว่าฉันอยู่ภายในพื้นที่ของห้องเรียนจริงขณะเช็คชื่อเข้าเรียน

สถาปัตยกรรม MVC (Model-View-Controller) สำหรับการพัฒนา Web Application

การพัฒนา Web Application นิยมใช้โครงสร้างสถาปัตยกรรมแยกชั้น (Layer Architecture) โดยแยก Components ออกเป็น Layer ความรับผิดชอบของคอมโพเน้นต์แต่ละ Layerจะมีความแตกต่างอย่างชัดเจน และ สามารถปฏิสัมพันธ์กับคอมโพเน้นต์ที่อยู่ชั้นติดกัน Layers ได้แก่

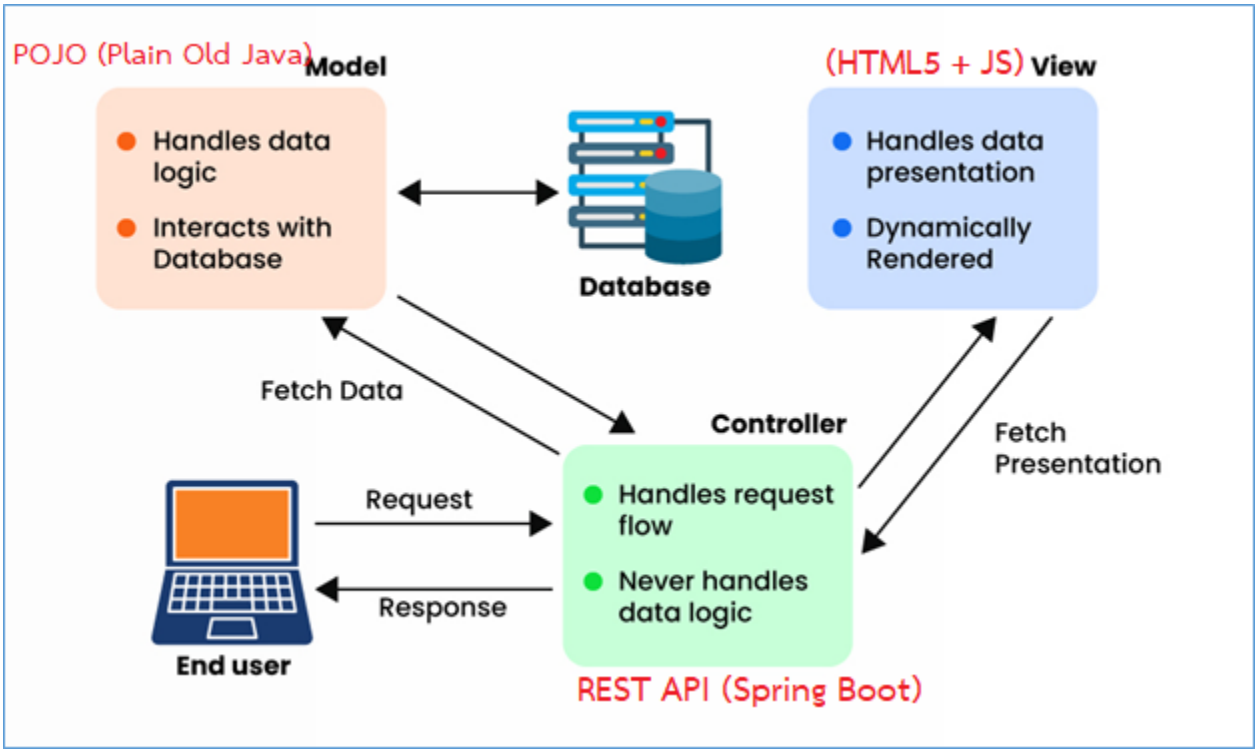
- **View** หน้าที่
  - ❖ แสดง User Interface ให้ผู้ใช้โต้ตอบ เพื่อรับข้อมูลจากผู้ใช้
  - ❖ นำข้อมูลอินพุตจาก UI และ Intent จาก User มาสร้าง Message Payload ส่งไปเป็น HTTP Request ส่งไปยัง API หลังบ้านเพื่อประมวลผลต่อ
  - ❖ นำข้อมูลผลลัพธ์ที่ได้จากการประมวลผของระบบมาแสดงให้ผู้ใช้ทำงานต่อ (Dynamically Rendering Contents)
- **Controller** หน้าที่
  - ❖ รับ HTTP request จาก View ตรวจสอบความถูกต้องของ input (validation) □ เรียกใช้ Service เพื่อประมวลผข้อมูลที่ได้รับ □ ส่งผลลัพธ์การประมวลผลออกเป็น Response กลับไปให้ View
  - ❖ ทำหน้าที่เป็น “ตัวกลาง” ระหว่าง View กับ Model ทำหน้าที่ปรับปรุงค่าของข้อมูลที่เก็บใน model เพื่อให้ view ปรับเปลี่ยนการแสดงผลข้อมูลตาม Model ที่เปลี่ยนไป
  - ❖ ทำหน้าที่ร้อยเรียงลำดับขั้นตอนการทำงานให้ถูกต้องตามที่ควร (Handles Request Flow)

หมายเหตุ Controller จะมีการพัฒนาโค้ดเพื่อเปิดเป็น REST API ให้ View เรียกใช้งาน

- **Model** หน้าที่
  - ❖ กำหนดโครงสร้างของข้อมูลที่ใช้แลกเปลี่ยนระหว่างคอมโพเน้นต์ต่างๆ (Defines the structure of the data)
  - ❖ ควบคุมการประมวลผลข้อมูลให้เป็นไปตาม Logic ที่ถูกต้อง (Handle Data Logic) เช่น สร้างรายการข้อมูลการเช็คชื่อเข้าเรียน ข้อมูลต้องมีเงื่อนไขอย่างไรจึงจะถูกต้อง

ก่อนที่จะบันทึกลงในระบบ การจะฝากเงินเข้าบัญชี Logic การฝากเงินเพิ่มเข้าบัญชีต้องเป็นอย่างไร?

- ติดต่อกับฐานข้อมูลเพื่อช่วยจัดการข้อมูลให้ระบบ ได้แก่ Create สร้างข้อมูลรายการใหม่ Retrieve ค้นคืนข้อมูล Update ปรับปรุงข้อมูล Delete ลบข้อมูลในฐานข้อมูล (Interacts with Database)



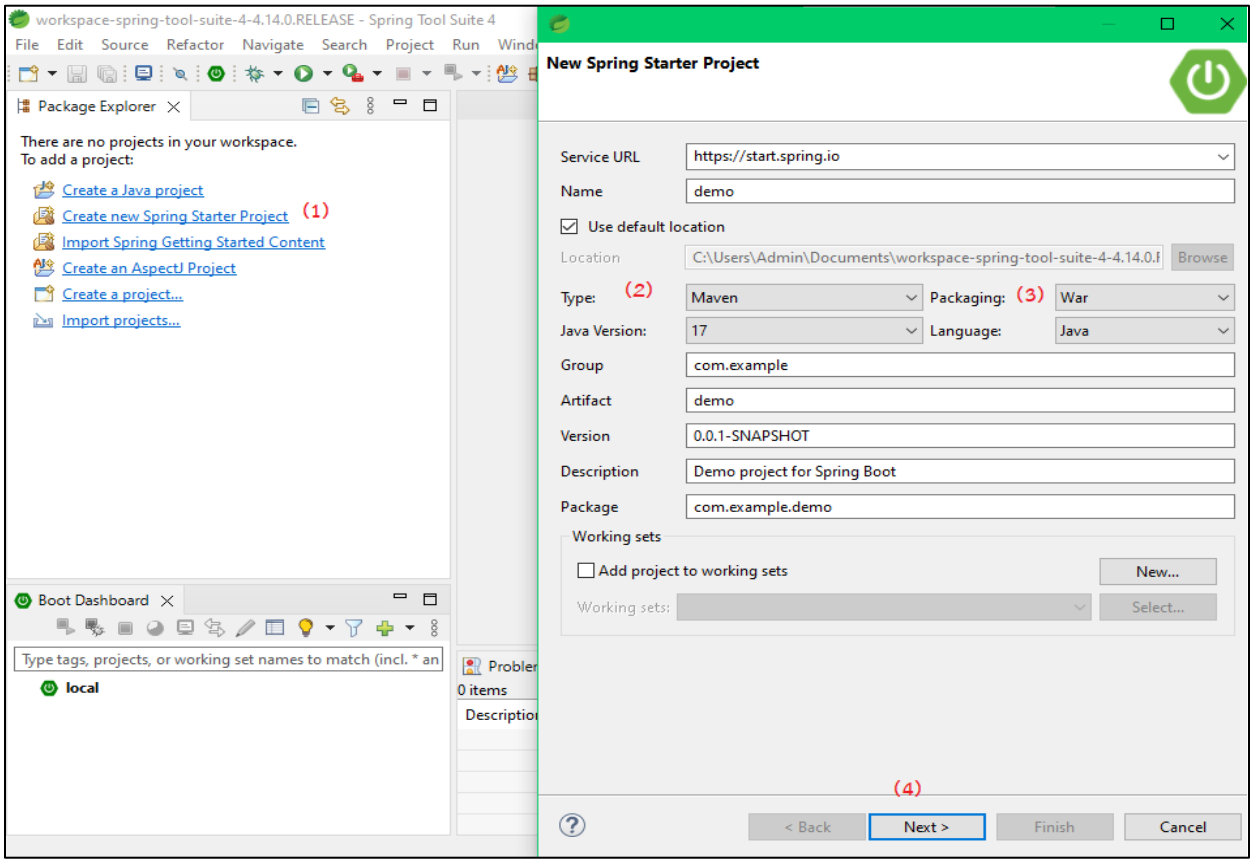
รูปที่ 1 สถาปัตยกรรม MVC สำหรับ Web Application

ใน Workshop นี้ พัฒนา View ด้วย HTML5 สำหรับสร้าง UI ให้ User ใช้ในการใส่ข้อมูลอินพุต และการแสดงผลลัพธ์ ส่วน JS ใช้สำหรับสร้าง HTTP Request และ ส่งต่อเรียกไปยัง API หลังบ้าน และ รอผลลัพธ์นำกลับมา Render ผลในหน้า Web ต่อไป

ขั้นตอนการสร้าง Spring Boot App ที่ประกอบด้วย Frontend และ Backend

ขั้นที่ 1 สร้าง New Spring Starter Project ดังรูปที่ 2

ให้ระบุประเภท เป็น Maven Project (1) และ Packaging เป็น War เพื่อใช้ Maven ในการ Build Code ให้เป็นไฟล์ประเภท War ซึ่งจะเป็น Shippable Product ที่พร้อมติดตั้งบน Web Server (Tomcat) กด Next เพื่อทำในขั้นตอนต่อไปคือการกำหนด Dependencies หรือ Libraries ที่จำเป็นสำหรับ Coding



รูปที่ 2 หน้าจอเริ่มต้นสร้าง Spring Boot Starter Project

ขั้นที่ 2 กำหนด Dependencies ที่สำคัญสำหรับพัฒนา Web Application ได้แก่ Spring boot starter web, Spring boot starter validation, spring-boot-devtools, Lombok ดังรูปที่ 5 หากพบปัญหา Import Lombok แล้วเกิด Error ใน pom.xml ให้แก้Ffpdesof Version ของ Lombok ตามโค้ดด้านล่าง

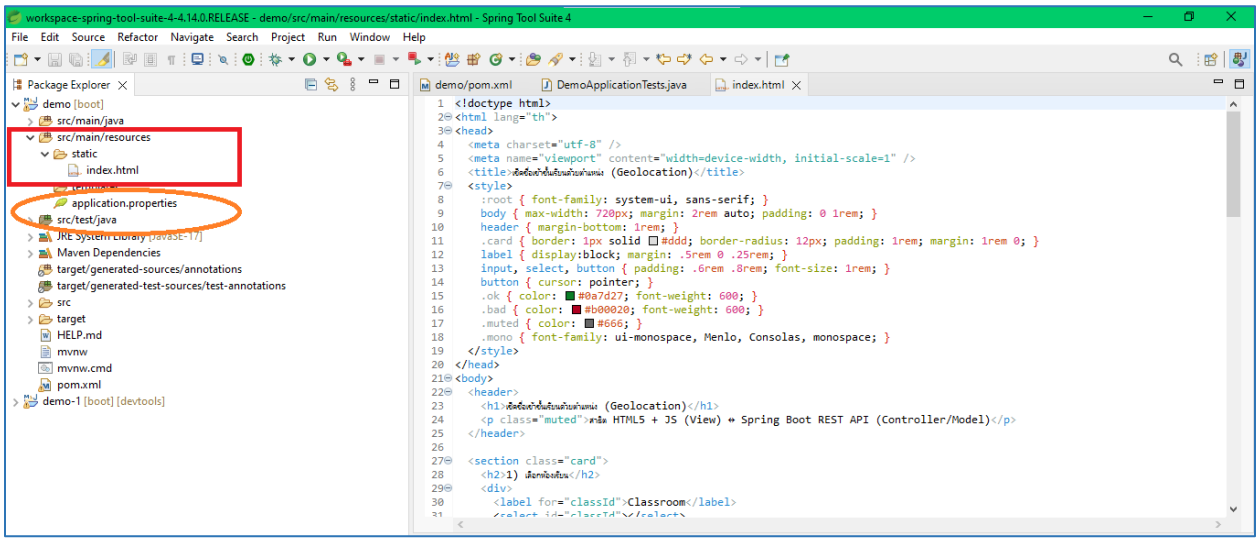
```
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
```

```
    <version>1.18.34</version>

    <scope>provided</scope>

</dependency>
```

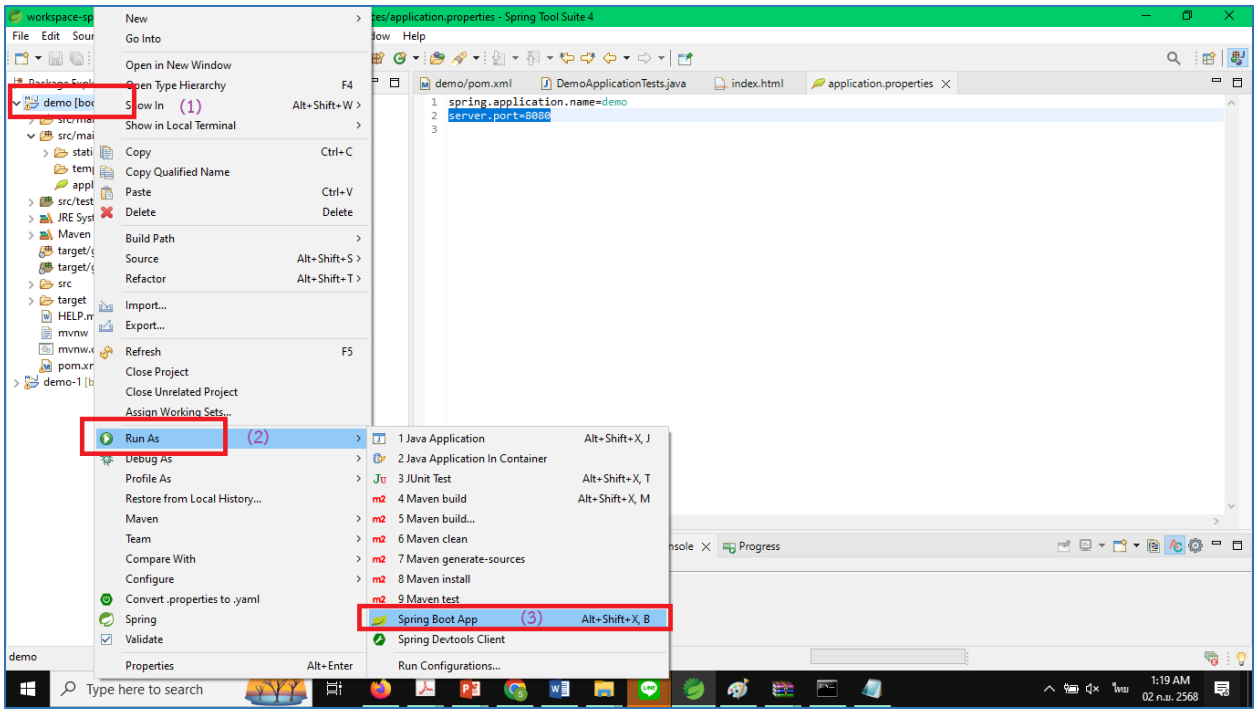
ขั้นตอนที่ 3 เพิ่มไฟล์ index.html ในโฟล์เดอร์ src/main/resources/static ดังรูปที่ 3 พร้อมทั้งกำหนด port ของ spring boot app ให้รันที่ port 8080 กำหนดในไฟล์ application.properties ภายใต้โฟล์เดอร์ src/main/resources โดยพิมพ์ค่า configuration ดังนี้ “`server.port=8080`”



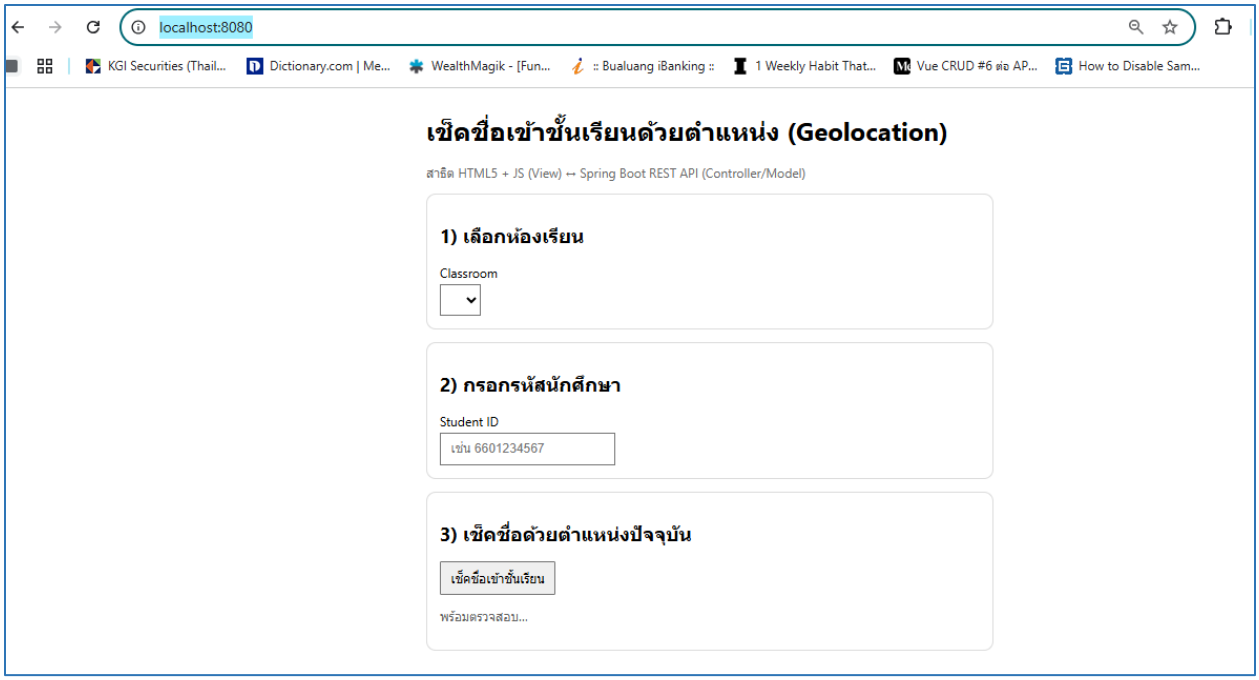
รูปที่ 3 เพิ่มไฟล์ html ที่เป็นหน้า main page ในโฟล์เดอร์ static

ขั้นที่ 4 ทดสอบ Build -> Deploy Spring Boot App โดยคลิกเม้าส์ขวาบน Project Demo > เลือกเมนู Run > เลือกเมนูย่อย Spring Boot App ดัง**รูปที่ 4** ซึ่งจะได้ผ

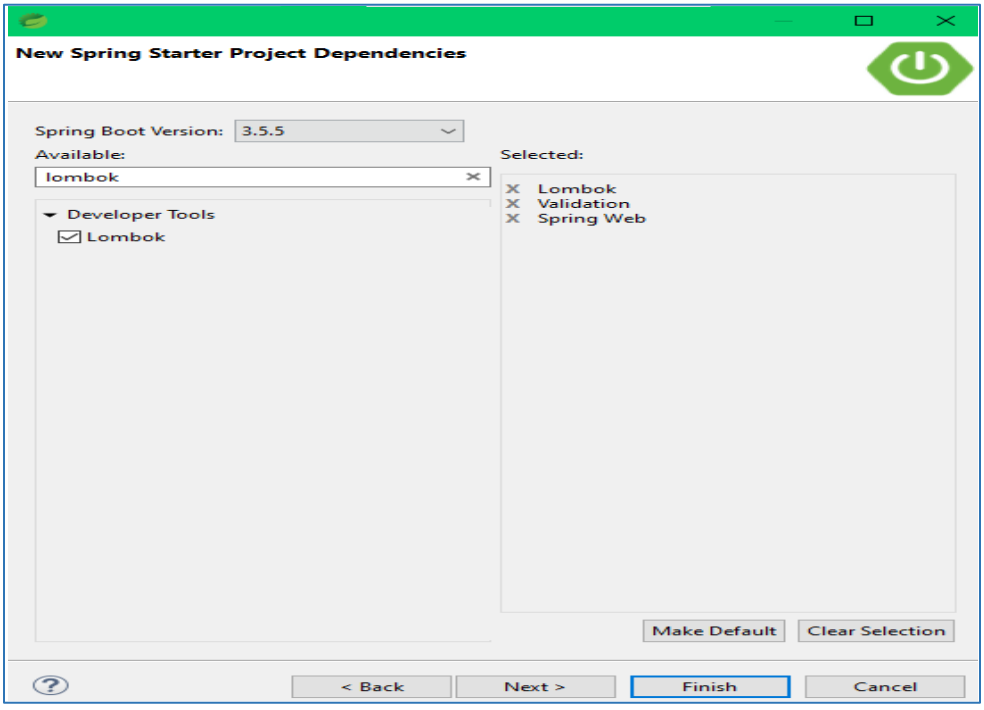
ขั้นที่ 5 ให้เปิด Browser พิมพ์ URL: http://localhost:8080/ เพื่อเรียกหน้า index.html ผลลัพธ์แสดงดังรูปที่ 6



รูปที่ 4 วิธีการรัน Web App ที่พัฒนาด้วย Spring Boot



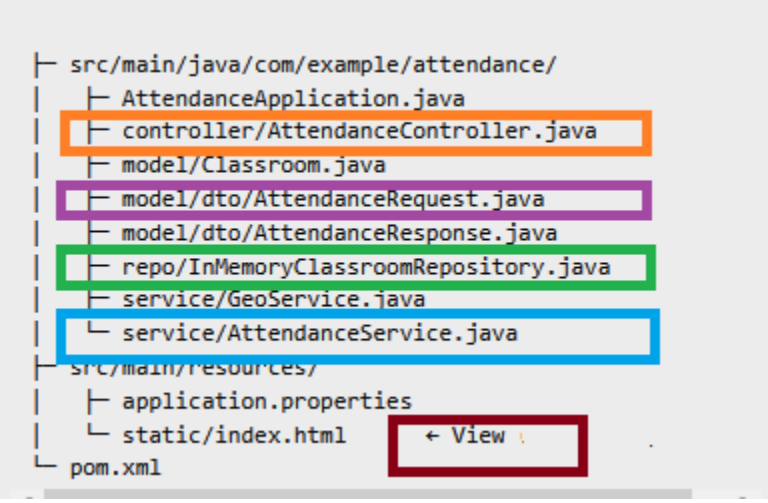
รูปที่ 5 หน้า index.html ที่เป็นหน้าหลักเปิดด้วย Browser และ URL ที่ set ไว้



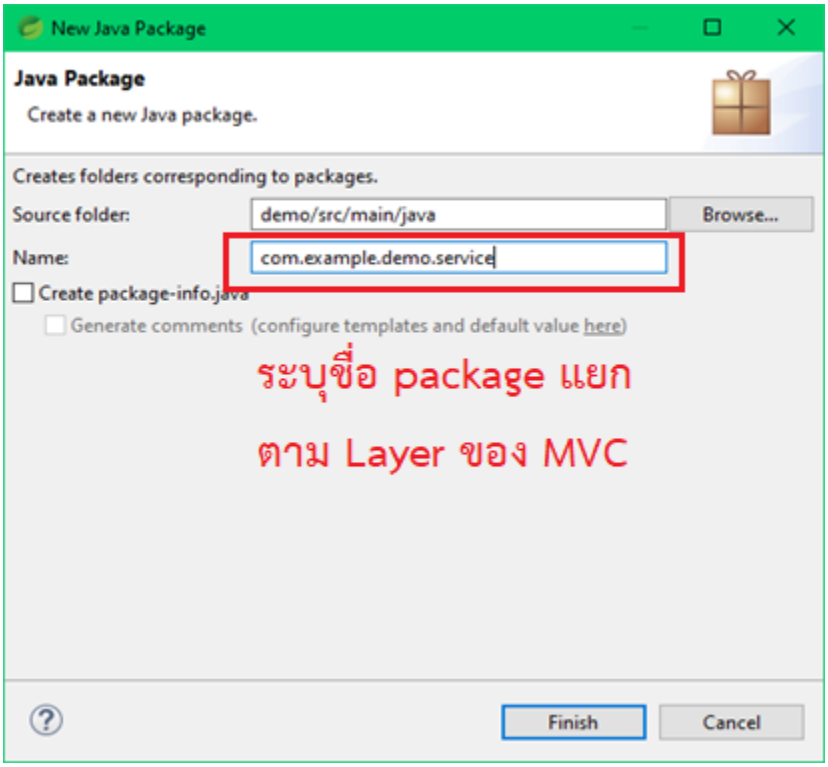
รูปที่ 6 หน้าระบุ Dependenciesสำหรับ Project

<u>ขั้นที่ 6</u> สร้าง Package เพื่อแยกโค้ดของ Components แต่ละ Layer ดัง**รูปที่ 7** ประกอบด้วย Packages “controller”, “model” (ทำหน้าที่ Data Transfer Object (DTO) กำหนดโครงสร้างข้อมูลและจัดการ Logic

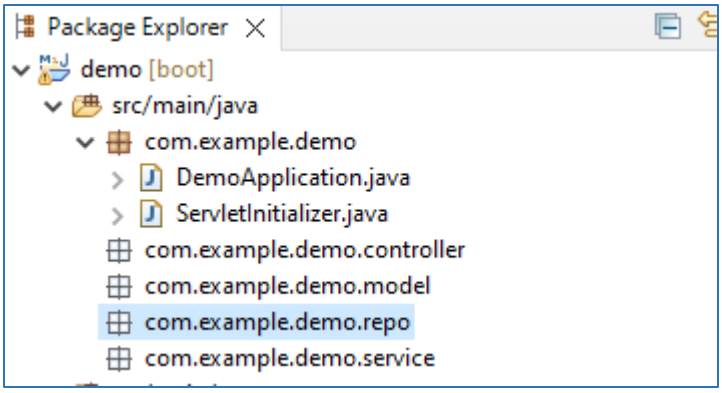
ที่เกี่ยวกับข้อมูล (Business Logic)), “repo” (แยก repo มาจาก model เพื่อใช้ในการจัดการ Data ในฐานข้อมูลเรียกอีกชื่อว่า Data Access Object (DAO)), “service”แยกจาก Controller เพื่อทำหน้าที่ประมวลผลตามลำดับขั้นตอนทางธุรกิจหรือ Business Flow ดูตัวอย่างหน้าต่างการสร้าง Package “service” จาก**รูปที่ 8**



รูปที่ 7 แยก Packages เพื่อเก็บ Components แต่ละ Layer ของ MVC



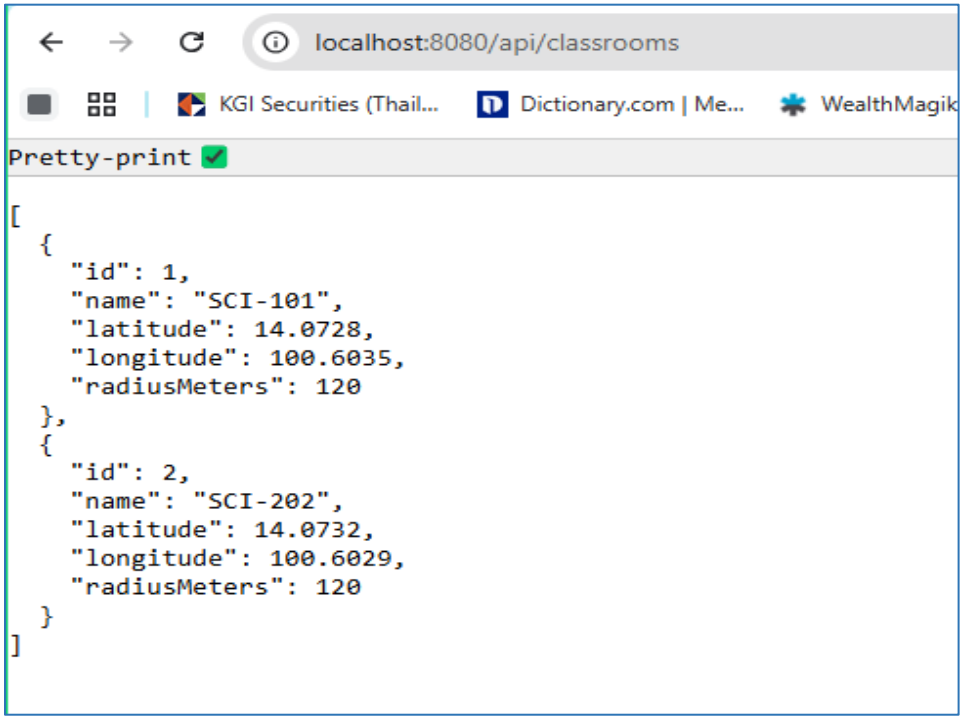
รูปที่ 8 หน้าจอตัวอย่างการสร้าง Packages แยกตาม Layers ของ MVC



รูปที่ 9 ผลลัพธ์หลังแยก Packages

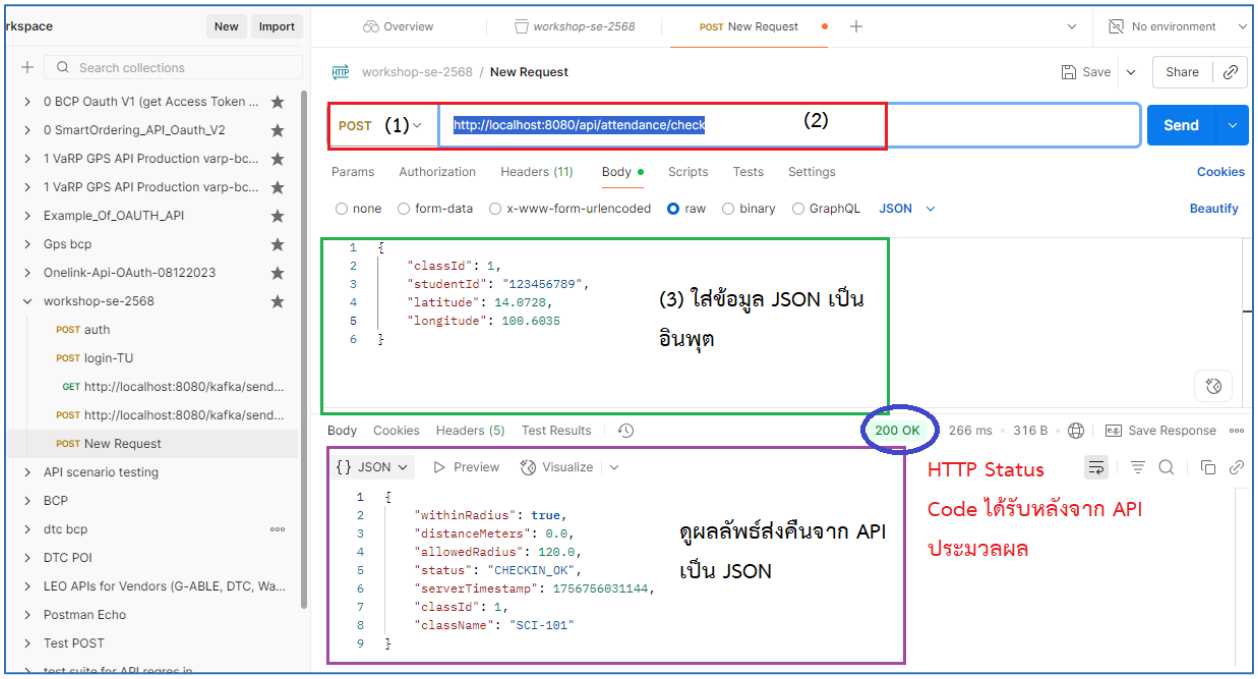
ขั้นที่ 7 เริ่มเขียนโค้ดคอมโพเน้นต์ประเภท Controller ที่ เปิดให้บริการในรูปแบบ REST API (เรียกใช้ผ่านโพรโตคอล http/http ส่งรับข้อมูลในรูปแบบ JSON เรียกเมท็อด GET, POST, PUT, DELETE) เพื่อเปิด API ให้ View สามารถรียกใช้ได้ผ่าน Browser เรียก URL: http://localhost:8080/api/classrooms

ผลการเรียกเมท็อด GET ผ่าน REST API เพื่อดึงรายละเอียด classrooms ทั้งหมดมา ได้ผลดัง**รูปที่ 10**



รูปที่ 10 ตัวอย่างกรเรียกใช้ REST API ผ่าน Controller

<u>ขั้นที่ 8</u>: เรียกใช้ API ด้วยเมท็อด POST เพื่อ check ชื่อเข้าชั้นเรียน จำเป็นต้องใช้เครื่องมือ POSTMAN ในการส่ง HTTP Request ไปยัง URL: http://localhost:8080/api/attendance/check พร้อมทั้งส่งข้อมูล Payload Message ไปเป็นอินพุตในรูปแบบ JSON ดัง**รูปที่ 11**



รูปที่ 11 ทดสอบเรียก REST API (POST Method) โดยใช้ POSTMAN

REST API ของมหาวิทยาลัยธรรมศาสตร์สำหรับ เช็ค Login

URL สำหรับเข้าไปศึกษาและเรียกใชเ API ของม.ธ. คือ https://restapi.tu.ac.th/home/