บทที่ 9

Real-time Communication

with WebSocket in Spring Boot

หัวข้อที่จะเรียนรู้

- แนวคิด Real-time Communication
- WebSocket vs HTTP
- โปรโตคอล STOMP (Simple Text Oriented Messaging Protocol)
- Spring Boot WebSocket Support
- การพัฒนา Chat Application เบื้องต้น

ทำไมต้อง Real-time?

- HTTP เป็นแบบ **Request–Response** → Client ต้องร้องขอเสมอ
- Real-time ใช้สำหรับ:
  - Chat Application
  - Notification (แจ้งเตือนทันที)
  - Online Collaboration (Docs, Whiteboard)
  - Dashboard Monitoring

HTTP vs WebSocket

HTTP

- Stateless, Request–Response
- Client → Server เท่านั้น

WebSocket

- Full-duplex, Persistent connection
- Client ↔️ Server → สื่อสารสองทางแบบเรียลไทม์

โครงสร้าง WebSocket

```
Client <----- WebSocket -----> Server
```

- Client เปิดการเชื่อมต่อ
- Server เก็บ connection และ push ข้อมูลกลับได้
- ใช้ TCP port เดียว (ปกติ 8080/443)

STOMP Protocol

- Simple Text Oriented Messaging Protocol
- ทำงานบน WebSocket
- ใช้แนวคิด **Publish/Subscribe**

โครงสร้าง:

- **Destination** (เช่น `/topic/messages`)
- **Client Subscribe** → รอรับข้อความ
- **Client Send** → ส่งข้อความ

Spring Boot WebSocket Dependency

Maven:

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-websocket</artifactId>
</dependency>
```

Config WebSocket

```
@Configuration
@EnableWebSocketMessageBroker
public class WebSocketConfig implements WebSocketMessageBrokerConfigurer {
    @Override
    public void configureMessageBroker(MessageBrokerRegistry config) {
        config.enableSimpleBroker("/topic");
        config.setApplicationDestinationPrefixes("/app");
    }

    @Override
    public void registerStompEndpoints(StompEndpointRegistry registry) {
        registry.addEndpoint("/ws").withSockJS();
    }
}
```

Message DTO

```
public class ChatMessage {
    private String sender;
    private String content;
}
```

Controller สำหรับ Chat

```
@Controller
public class ChatController {
    @MessageMapping("/chat.send")
    @SendTo("/topic/messages")
    public ChatMessage send(ChatMessage message) {
        return message;
    }
}
```

Client เชื่อมต่อ (JavaScript)

```
var socket = new SockJS('/ws');
var stompClient = Stomp.over(socket);

stompClient.connect({}, function(frame) {
    stompClient.subscribe('/topic/messages', function(msg) {
        console.log(JSON.parse(msg.body));
    });

    stompClient.send("/app/chat.send", {}, JSON.stringify({
        sender: "Alice", content: "Hello!"
    }));
});
```

Lab

1. ติดตั้ง dependency WebSocket
2. Config WebSocket (`/ws`)
3. เขียน Controller ส่งข้อความ → `/topic/messages`
4. สร้าง Client HTML/JS สำหรับทดสอบ Chat

Assignment

- พัฒนา Chat Room ขนาดเล็ก:
  - `/topic/room1` และ `/topic/room2`
  - ผู้ใช้เลือกห้องที่จะ join ได้
- ข้อกำหนด:
  - เก็บประวัติข้อความล่าสุด (เช่น 10 ข้อความ)
  - Broadcast ให้สมาชิกในห้องเดียวกัน
- ส่งโค้ด + Screenshot การทดสอบ Chat