完整技術開發筆記

一、資料庫相關

1. SQL JOIN

- **LEFT JOIN vs INNER JOIN**
  - LEFT JOIN：
    - 訂單查詢時建議使用
    - 即使商品下架，仍能顯示訂單明細相關資訊
    - 保持歷史訂單資料完整性
  - INNER JOIN：
    - 商品下架後，對應的 order_item_table 會消失
    - 查詢結果可能顯示 NULL
    - 導致歷史訂單明細不完整

2. ACID 特性

1. **原子性 (Atomicity)**
   - 要麼全部成功，要麼全部失敗
   - 使用 2 Phase Commitment
   - 先寫入日誌檔，再寫入資料庫

2. **一致性 (Consistency)**
   - 資料需符合資料庫名稱和型別
   - CRUD 資料須符合定義規則
   - 例如：NOT NULL 限制

3. **隔離性 (Isolation)**
   - 使用 Lock 機制
   - 資料獨佔直到 commit 或 rollback
   - 防止更新遺失（Lost Update）

4. **持久性 (Durability)**

- 提交的資料永久保存
- 正式寫入磁碟
- 交易日誌特性：
  - 日誌檔案非永久保存
  - 日誌循環：1-3 循環到 1
  - 日誌延續時間由 DBA 決定

3. Redis

- 適用於聊天室設計
- 可作為非同步資料庫使用

4. JDBC

- Java 資料庫驅動程式
- 負責與資料庫溝通

二、Java 開發

1. 資料結構（Java Collections）

- **List:** 有序可重複
- **Set:** 無序不可重複
- **Map:** 鍵值對（key-value）
- **TreeMap:** 可排序的 Map

2. ListMap 與 MapList

- **ListMap:**
  - 適用情境：一筆訂單包含多筆明細
  - 結構：`List<OrderItem> orderItemList`
  - OrderItem 包含多個屬性（如 productId, quantity, amount, orderId）

- **MapList:**
  - 適用於需要分群或分類的資料
  - 例如：依日期、標籤分類
  - 適合快速查找分類資料

3. 物件導向程式設計（OOP）

1. **封裝**

- 參數方法包在 class 內
- private 私有化
- public getter/setter 公開化

2. **繼承**
   - extends：實作繼承，限單一父類
   - implements：介面繼承，定義行為規範

3. **多型**
   - 需要有繼承關係
   - 方法參數型別一致但實作邏輯不同
   - 父類別宣告 = new 子類別（父面子身）

4. Servlet & JSP

- **Servlet 特性:**
  - Call by value：基本型別（int, long, String）
  - Call by reference：物件（request, response）
  - StringBuffer：多執行緒環境
  - StringBuilder：單執行緒
  - 生命週期：
    1. 創建 servlet 類並繼承 httpServlet
    2. 覆寫 dopost/doget
    3. 配置 web.xml
    4. 編譯打包
    5. 部署及訪問
  - doGet：參數置於 Request header，不可加密
  - doPost：參數置於 Request body，可加密

- **JSP 特性:**
  - 可寫動態網頁
  - 流程：創建新檔 → 撰寫.jsp → 執行
  - Tomcat 自動編譯成 Java Servlet
  - MVC 架構全在後台，無前後分離
  - 作用域：
    - page：頁面局部變數

    - request：傳遞數據參數
    - session：存取用戶資料
    - application：存取全局配置

5. Web 應用程式條件

1. 必須有 WEB-INF 目錄
2. 必須有 web.xml 檔案

6. Filter 過濾器

- 類似 AOP 功能
- 用於編碼轉換
- 進行權限檢查

三、框架應用

1. Spring Boot

- **Enterprise Bean:**
    - 用於分散式程式
    - EJB 解決 Java 分散式問題
    - 使用 HTTPS 跨機器溝通
    - XML 用於資料傳遞

- **IOC（控制反轉）：**
    - 降低耦合度
    - 物件託管給 Spring IOC 容器
    - 減少 new 物件問題
    - 使用單例模式（Singleton）
    - Constructor 宣告為 private
    - 物件以 static 方式建立

- **AOP（切面導向程式設計）：**
    - 可在方法執行前後執行
    - 例如：@Transactional 註解

2. MyBatis

- **特性：**
  - 支援動態生成 SQL
  - 提供豐富標籤
  - 可自定義 SQL 查詢
  - SQL 可寫在 DAO 或 XML 檔案中
- **對比其他框架：**
  - JPA：不支援複雜 SQL，適合簡單邏輯
  - Hibernate：需大量調整映射
  - JDBC：只能用 Map，不支援物件
- **MyBatis Plus:**
  - 提供基本 CRUD SQL
  - 支援複合主鍵處理

四、前端開發

1. MVVM 框架

- **Model (M):**
  - 業務邏輯
  - 變數 I/O
  - Pinia 狀態管理
  - Script setup method
- **View (V):**
  - Template 中的 HTML 內容
- **ViewModel (VM):**
  - 使用 {{}} 取代 ID
  - 支援非同步數值變更
  - 雙向綁定（v-model）
  - 負責 VNode 生成和管理
  - 處理虛擬 DOM 樹

2. Vue.js 功能

- **watch vs computed vs method:**
  - watch：無快取，適合監聽觸發事件
  - computed：有快取，適合計算加總
  - method：每次呼叫都執行一次

3. 前端技術

- **Ajax vs Submit:**
  - Ajax：非同步請求，頁面不重整
  - Submit：表單提交後頁面重整，資料消失

- **Bootstrap：**
  - 快速開發框架
  - 支援 RWD 響應式設計
  - 使用 SCSS 提高 CSS 開發效率

五、開發工具

1. 偵錯工具

- **瀏覽器：** F12 + console.log
- **後端：** Logger + debug

2. Maven

- 套件管理功能
- 版本一致性控制
- 使用 Nexus Maven 確保安全
- 常用指令：
  - mvn clean install
  - clean：清理 target
  - install：將 jar 檔放入 Maven 倉庫

3. Node.js

- JavaScript 執行環境
- 可執行類似 Java 的功能
- 支援檔案存取、報表列印

4. 日誌工具

- **SLF4J 日誌等級：**
  - trace > debug > info > warn > error
  - 依需求選擇適當等級

5. 版本控制

- **Pull Request (PR):**
  - 用於 Code Review
  - 確保程式碼品質

六、系統概念

1. 同步與非同步

- **同步：**
  - 依序執行
  - 需等待前一步驟完成
- **非同步：**
  - 無需等待每步驟完成
  - 可同時處理多個任務
  - 不依賴前一步驟結果

2. 作用域生命週期

- **application：** 應用程式啟動至結束（佔用最多記憶體）
- **session：** 登入至登出或 30 分鐘超時
- **request：** Servlet 間通訊
- **redirect：** 經由瀏覽器間接通訊