서버프로그래밍 Spring을 이용한 식당 주문 시스템

2023078086 최은재

1. 프로젝트 개요

이번 프로젝트는 Spring MVC 기반의 간단한 식당 주문 관리 시스템입니다.
메뉴 등록, 테이블 관리, 주문 생성 및 통계 조회 기능을 제공합니다.
과제 요구사항에 따라 Web Application 형태로 구현되었으며, 데이터는 로컬에 JSON 파일 형태로 저장됩니다.

2. 실행 방법

1) 실행 환경

Ubuntu-based container (학교 서버 기본 환경)

Java 11.0.29

Maven 3.6.3

Jetty Server 사용

2) 실행 명령

① 프로젝트 압축 해제

② ./run.sh 실행: run.sh 파일에 실행권한을 부여하였으므로 project4 폴더로 이동 후 바로 실행하실 수 있습니다.

③ 접속 주소: http://10.198.138.212:3010/admin/status.html

3) run.sh 기능

Maven 빌드 mvn clean package

실행 환견 메모리 최적화 (MAVEN_OPTS 설정)

Jetty 서버 실행

3. 성능 개선

프로젝트 평가 기준에 "다중 사용자 접근 환경에서도 안정적으로 동작할 것"이 존재했기에 대규모 요청 처리와 안정적 스레드 관리에 용이한 Tomcat 서버를 이용하려고 했습니다. 그러나 실제로 학교 서버 컨테이너에서 Tomcat을 적용하는 과정에서 여러 제약과 문제가 발생했습니다.

1) 권한 문제로 인해 WAR 배포 불가
/usr/share/tomcat9/webapps/ 경로에서 root 권한이 있어 WAR 배포가 불가능했습니다.

2) Tomcat7 Maven Plugin은 Java 11과의 호환성 문제로 반복적인 오류가 발생하였습니다.

3) ab 테스트 시 비정상 종료
Tomcat 문제를 해결하기 위해 시도한 다양한 설정(스레드 축소, 서버.xml 최적화, JVM 조정)에도 불구하고 컨테이너 환경 자체가 Tomcat에 적합하지 않다고 판단했습니다.

이후 Jetty Maven Plugin 기반으로 전환하였습니다. Jetty는 상대적으로 가벼운 스레드풀 구현(QueuedThreadPool)을 사용하여, Tomcat보다 컨테이너의 스레드/메모리 제한을 덜 받습니다. 또한 Jetty Maven Plugin은 WAR 배포 없이 바로 실행되므로, 학교 서버의 권한 제약과 구식 Tomcat 환경의 영향을 받지 않습니다. 여전히 Jetty에서도 스레드 관련 경고가 뜨지만, 서버가 중단되지 않고 안정적으로 요청을 처리했다는 점에서 최종적으로 Jetty를 선택했습니다.

4. 성능 개선 요소

1) 서버 스레드 ·메모리 최적화
컨테이너 메모리가 제한되어 있기 때문에 큰 스레드풀은 오히려 위험할 수 있습니다. 따라서 JVM 스택 사이즈를 축소시키고 Jetty Maven Plugin로 서버를 실행시킵니다. Jetty Maven Plugin은 Tomcat 대비 가벼우며, thread-per-request 방식이 단순해 컨테이너 환경에 안정적입니다.

2) 파일 기반 스토리지 최적화
JSON 캐시 사용: 메모리 캐싱 후 필요한 시점에만 파일을 다시 씁니다. 이를 통해 디스크 I/O를 감소시키고, 고부하 시 안정성이 향상됩니다.

3) 불필요한 스캔 비활성화

```
<reload>manual</reload>
<scanIntervalSeconds>0</scanIntervalSeconds>
```

JSP/Sevlet 스캔을 최소화하여 서버 부팅 시간이 크게 줄어들고, 동시 접속자 증가 시 CPU 낭비를 방지합니다.