자바 웹 프로그래밍 최종 점검

최종 점검을 위한 환경 세팅

- https://github.com/slipp/jwp-spring-final 저장소를 자신의 github 계정으로 fork 한다.
- fork한 저장소를 clone한 후 maven 프로젝트로 import 한다.
- src/test/java 소스 폴더의 next.WebServerLauncher를 실행한 후 http://localhost:8080 으로 접속한다.

최종 점검 원칙

- 모든 활동 허용. 예를 들어 교수 또는 동료에게 막히는 부분에 한해 도움 요청 가능
- 가능하면 자신이 모르는 부분을 새롭게 학습한다는 목적을 가지고 진행하면 좋겠다.
- 각 문제를 해결할 때마다 commit log를 남긴다.

1. Tomcat 서버를 시작한 후 http://localhost:8080로 접근하면 DB 테이블이 존재하지 않아 에러가 발생한다. 에러가 발생하는 원인은 jwp.sql 파일이 정상적으로 실행되지 않았기 때문이다. jwp.sql 파일을 읽어 테이블을 생성하고 테스트 데이터를 추가하는 작업은 next.support.DBInitializer가 담당하고 있는데 DBInitializer의 생성자가 실행되지 않고 있다. Spring 프레임워크 기능의 초기화 기능을 활용해 DBInitializer 생성자에 있는 기능이 정상적으로 실행되어 DB 초기화가 가능하도록 한다(힌트: Spring 프레임워크에서 빈 초기화를 위한 기능 활용).

2. 로컬 개발 환경에 Tomcat 서버를 시작하면 Spring 컨테이너(ApplicationContext)가 초기화 과정이 진행된다. 부모 ApplicationContext와 자식 ApplicationContext가 초기화되는 과정에 대해 구체적으로 설명해라. 설명은 clone 한 소스 코드의 README.md 파일에 작성한다.

3. 로컬 개발 환경에 Tomcat 서버를 시작한 후 http://localhost:8080으로 접근하면 질문 목록을 확인할 수 있다. http://localhost:8080으로 접근해서 질문 목록이 보이기까지 흐름에 대해 최대한 구체적으로 설명하라. 설명은 clone 한 소스 코드의 README.md 파일에 작성한다.

4. 이 서비스는 DB에 쿼리를 하는데 성능을 높이기 위해 apache commons의 DBCP2라는 Connection Pool을 사용하고 있다. 그런데 DBCP2보다 성능이 좋은 c3p0 (http://www.mchange.com/projects/c3p0/) 라는 Connection Pool로 변경하려고 한다. pom.xml에 c3p0 라이브러리에 대한 의존 관계를 추가하고 c3p0 Connection Pool을 사용하도록 Spring 설정을 변경한다.

5. 질문하기 과정을 참고해(QuestionController의 form(), save(), form.jsp) 수정하기 기능을 구현한다. 수정하기 링크는 각 게시글의 상세보기 페이지에서 볼 수 있다. 단, 수정 페이지를 구현해보면 질문하기의 form.jsp와 많은 부분이 중복이 발생한다. 따라서 수정하기 기능을 구현하기 위해 새로운 jsp를 추가하지 않고, 같은 form.jsp 파일을 사용해 구현한다.

6. next.controller.qna.QuestionController 클래스는 동시에 2명 이상의 사용자가 접근할 경우 문제가 발생하는 수 있는 코드이다. 문제가 발생하지 않도록 수정하고, 그 이유를 설명하시오. 설명은 clone 한 소스 코드의 README.md 파일에 작성한다.

7. 답변 삭제 기능에 대한 웹 클라이언트 기능은 구현이 되어 있는 상태이다. 단, 클라이언트에 대응하는 서버측 Controller 기능을 구현하지 않았다. Controller 기능을 구현해 답변이 삭제 가능하도록 한다.

- 글쓰이만 답변을 삭제할 수 있다. 답변을 삭제하는 경우 DB에서 완전히 삭제하는 것이 아니라 삭제 상태(deleted)를 false에서 true로 변경해야 한다.
- 답변을 삭제하면 Question 테이블에서 답변 수가 1 감소해야 한다.
- 질문의 답변 목록에서 삭제한 답변이 보이지 않아야 한다.

8. 웹 서비스에서 각 구간별 성능 측정을 하는 것은 상당히 중요하다. 특히 불특정 다수를 대상으로 서비스하기 때문에 이슈 하나로 사용자가 폭증할 수 있기 때문이다. 사용자가 증가하는 시점에 대비하기 위해 Controller, Service, Dao의 모든 메서드 별로 소요되는 시간을 측정하고 싶다. 모든 메서드에 일일이 성능 측정 소스 코드를 추가하는 것은 상당히 짜증나는 작업이다. 이 같은 반복 작업을 Interceptor와 Spring AOP로 나누어 개발한다.

- Controller의 성능 측정은 Interceptor를 활용해 측정한다.
- Service, Dao는 Spring AOP를 활용해 측정한다.

9. 질문 삭제 기능을 구현한다. 질문 삭제에 대한 요구사항은 다음과 같다.

- 질문 데이터를 완전히 삭제하는 것이 아니라 데이터의 상태를 삭제 상태(deleted - boolean type)로 변경한다.
- 로그인 사용자와 질문한 사람이 같은 경우 삭제 가능하다.
- 답변이 없는 경우 삭제가 가능하다.
- 질문자와 답변 글의 모든 답변자 같은 경우 삭제가 가능하다.
- 질문을 삭제할 때 답변 또한 삭제해야 하며, 답변의 삭제 또한 삭제 상태(deleted)를 변경한다.
- 질문자와 답변자가 다른 경우 답변을 삭제할 수 없다.
- 질문 목록과 답변 목록에 삭제된 데이터는 보이지 않아야 한다.
- 삭제한 질문은 상세보기할 수 없어야 한다.

9. 질문 삭제 기능에 대한 추가 요구사항

- 질문 삭제 기능을 최대한 객체지향적으로 구현한다.
- RestAssured 라이브러리를 활용해 삭제 기능에 대한 Acceptance Test를 추가한다.
- Mockito를 활용해 QnaService의 deleteQuestion() 메서드에 대한 단위 테스트를 한다.
- Question, Answer에 대한 단위 테스트를 추가한다.

10. 지금까지 구현한 기능을 개발 서버의 Tomcat 서버에 배포하고 접근 가능하도록 해야 한다. 서버 구성은 nginx + tomcat 구조로 80 port로 접속할 수 있어야 한다. tomcat 서버는 외부에서 접근할 수 없어야 한다. 배포 스크립트를 통해 배포를 자동화해야 한다.

최종 점검 결과 제출 방법

- javajigi@connect.or.kr 메일로 다음 내용을 공유한다.

제출할 내용

- fork한 자신의 저장소 url
- 배포한 서버의 ip
- 접속할 서버의 id/pwd. 비밀번호는 자신이 사용하지 않는 비밀번호로 수정 후 공유
- 배포 서버의 tomcat 디렉토리, 배포 쉘 스크립트 위치