Java Web Programming 상태 점검

실습 시작 과정

- 자신의 계정으로 fork한 jwp-basic 프로젝트의 브랜치를 step7-self-check 으로 이동한다.
- src/test/java 디렉토리의 next.WebServerLauncher의 main 메서드를 실행한 후 http://localhost:8080으로 접근한다.

실습 결과물 제출 방법

- 요구사항을 완료하면 자신이 작업한 결과물을 github에 올린다.
- **각 요구사항에 대해 자신이 명확하게 알고 있는지의 여부와 자신의 상태를 기록해 제출한다.**

사전 정보

실습으로 진행할 웹 애플리케이션은 질문과 답변을 주고 받을 수 있는 Q&A 서비스이다. 이 서비스의 테이블 구조는 src/main/resources 디렉토리의 jwp.sql 파일에서 확인할 수 있다. 개발 편의성을 위해 8개의 질문 데이터를 추가해 놓은 상태이다.

MVC 프레임워크 사용 방법은 다음과 같다.

- 서버가 시작할 때 DB 초기화는 next.support.context.ContextLoaderListener 클래스의 contextInitialized(), Mapping 초기화는 core.mvc.DispatcherServlet 클래스의 init()가 담당한다.
- 모든 Controller는 AbstractController를 상속하고, core.mvc.Controller 인터페이스의 execute() 메써드를 구현해야 한다.
- URL과 Controller의 Mapping은 core.mvc.RequestMapping에서 담당한다.
- Controller에서 페이지 이동을 forward 방식인 경우 JstlView를 생성할 때 “list.jsp”와 같이 JSP이름을 직접 입력, redirect 방식인 경우 “redirect:/users”와 같이 접두어로 “redirect:”를 사용한다.
- Controller에서 JSON 형태로 API를 제공해야 하는 경우 execute() 메써드에서 jsonView() 메서드를 활용해 JsonView를 생성할 수 있다.
- 입력 페이지에 접근하는 것과 같이 특별한 로직이 필요 없는 경우 ForwardController를 활용해 new ForwardController("form.jsp")와 같이 mapping하는 것이 가능하다.

요구사항

1. 로컬 개발 환경에 Tomcat 서버를 시작하면 Servlet Container의 초기화 과정이 진행된다. 현재 소스 코드에서 초기화되는 과정에 대해 설명해라(WebServerLauncher의 시작 과정이 아니라 clone한 프로젝트의 초기화 과정이다.). 설명은 clone 한 소스 코드의 README.md 파일에 작성한다. README.md 파일은 마크다운 문법을 따른다. 마크다운 문법은 http://daringfireball.net/projects/markdown/syntax 에서 참조 가능하다.(힌트 : DB 초기화를 담당하는 ContextLoaderListener 클래스와 Mapping 초기화를 담당하는 DispatcherServlet 클래스부터 분석을 시작한다.)

2. 로컬 개발 환경에 Tomcat 서버를 시작한 후 http://localhost:8080으로 접근하면 질문 목록

을 확인할 수 있다. http://localhost:8080으로 접근해서 질문 목록이 보이기까지 소스 코드의 호출 순서 및 흐름을 설명하라. 설명은 clone 한 소스 코드의 README.md 파일에 작성한다. README.md 파일은 마크다운 문법을 따른다. 마크다운 문법은 http://daringfireball.net/projects/markdown/syntax 에서 참조 가능하다.

3. 질문 목록은 정상적으로 동작하지만 질문하기 기능은 정상적으로 동작하지 않는다. 질문하기 기능을 구현한다. 질문 추가 로직은 QuestionDao 클래스의 insert method 활용 가능하다. HttpServletRequest에서 값을 추출할 때는 ServletRequestUtils 클래스를 활용 가능하다. 질문하기를 성공한 후 질문 목록 페이지("/")로 이동해야 한다.

4. 로그인하지 않은 사용자도 질문하기가 가능하다. 로그인한 사용자만 질문이 가능하도록 수정한다. 또한 질문할 때 글쓴이를 입력하지 않고 로그인한 사용자 정보를 가져와 글쓴이 이름으로 등록한다.(힌트 : session.getAttribute("user")와 같이 Session에서 로그인 정보를 가져올 수 있다.)

5. 질문 목록에서 제목을 클릭하면 상세보기 화면으로 이동한다. 상세보기 화면에서 답변 목록이 정적인 HTML로 구현되어 있다. 답변 목록을 정적인 HTML이 아니라 데이터베이스에 저장되어 있는 답변을 출력하도록 구현한다. 단, <%%>와 같이 스크립틀릿을 사용하지 않고 JSTL과 EL(expression language)만으로 구현해야 한다.

6. 자바 기반으로 웹 프로그래밍을 할 경우 한글이 깨진다. 한글이 깨지는 문제를 해결하기 위해 ServletFilter를 활요해 문제를 해결할 수 있다. core.web.filter.CharacterEncodingFilter에 어노테이션 설정을 통해 한글 문제를 해결한다.(힌트 : WebFilter annotation)

7. next.web.qna package의 ShowController는 멀티 쓰레드 상황에서 문제가 발생할 가능성이 있는 코드이다. 멀티 쓰레드 상황에서 문제가 발생하지 않도록 수정한다. 멀티 쓰레드에서 문제가 되는 이유를 README.md 파일에 작성한다.

8. 상세보기 화면에서 답변하기 기능은 정상적으로 동작한다. 단, 답변을 추가할 경우 댓글의 수가 증가하지 않는다. 답변을 추가하는 시점에 질문(QUESTIONS 테이블)의 댓글 수(countOfAnswer)도 1 증가해야 한다. 데이터베이스 접근 로직은 직접 구현해야 한다. **(선택) AJAX로 댓글을 추가하는 경우 화면의 답변 수도 증가해야 한다. 자바스크립트 구현을 통해 답변 수도 증가하도록 구현한다.**

9. 이 Q&A 서비스는 모바일에서도 서비스할 계획이라 API를 추가해야 한다. API는 JSON 형식으로 제공할 계획이다. /api/qna/list URL로 접근했을 때 질문 목록을 JSON 데이터로 조회할 수 있도록 구현한다.

10. 상세보기 화면의 답변 목록에서 답변을 삭제해야 한다. 답변 삭제 또한 화면을 깜빡이지 않고 구현이 가능하도록 AJAX로 구현한다. js/scripts.js 파일의 답변 추가 로직을 참고해 답변 삭제 로직을 구현한다.

11. 질문 수정이 가능해야 한다. 질문 수정은 글쓴이와 로그인 사용자가 같은 경우에만 수정이 가능하다.

12. Controller에서 접근하는 QuestionDao와 AnswerDao, DAO에서 데이터베이스 접근 로직을 구현할 때 사용하는 JdbcTemplate은 인스턴스를 여러 개 생성할 필요없다. 인스턴스를 하나만 생성하도록 구현한다.(힌트 싱글톤 패턴)

13. 질문 삭제 기능을 구현한다. 질문 삭제가 가능한 경우는 다음과 같다. “답변이 없는 경우 삭제가 가능하다. 질문자와 답변자가 같은 경우 삭제가 가능하다. 질문자와 답변자가 다른 경우 답변을 삭제할 수 없다”. 이 질문 삭제 기능은 일반 PC와 모바일 모두를 지원하려고 한다. 삭제를 완료한 후 일반 PC의 웹 브라우저는 JspView를 활용해 목록 페이지(“redirect:/”)로 이동하고, 모바일은 JsonView를 활용해 응답 결과를 JSON으로 전송하려고 한다. 이를 지원하려면 두 개의 Controller가 필요하다. 각 Controller를 구현해보면 많은 중복이 발생한다. 각 Controller에서 발생하는 중복을 제거한다. 상속 또는 새로운 클래스를 만들어 위임할 수 있다(composition, 조합이라고 한다.). 어느 방식으로 중복을 제거하는 것이 좋을지에 대해서도 고민해 본다.

14. **(선택)13번 문제를 구현할 때 단위 테스트가 가능하도록 구현한다. Dao를 사용하는 모든 Controller 클래스는 데이터베이스가 설치되어 있어야 하며, 테이블까지 생성되어 있는 상태에서만 테스트가 가능하다. 데이터베이스가 존재하지 않는 상태에서도 위 로직을 단위 테스트하고 싶다.(힌트 : Dependency Injection, Map을 활용한 메모리 DB, Mockito 테스트 프레임워크)**

15. **(선택) RequestMapping 코드를 보면 컨트롤러가 추가될 때마다 요청 URL과 컨트롤러를 추가해야 하는 불편함이 있다. 서블릿과 같이 애노테이션을 활용해 설정을 추가하고 서버가 시작할 때 자동으로 매핑이 되도록 개선해 본다. 이 문제는 쉽지 않은 문제이다. 이 단계에서는 어떻게 개선하는 것이 좋겠다는 설계만 해도 충분하다.(힌트 : @Controller 애노테이션을 추가하고, 자바 리플렉션을 활용.)**

모든 요구사항을 완료했으면 github에 commit & push한다.