**SOURCE CODE**

**CODE:**

**Entity:**

**User**

package com.example.model;

import java.time.LocalDateTime;

import java.util.HashMap;

import java.util.Map;

import javax.persistence.CollectionTable;

import javax.persistence.Column;

import javax.persistence.Convert;

import javax.persistence.ElementCollection;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.MapKeyJoinColumn;

import javax.persistence.NamedNativeQuery;

import javax.persistence.Table;

import javax.validation.constraints.Email;

import javax.validation.constraints.NotBlank;

import com.example.model.Question;

import com.example.model.User;

import com.example.model.Role;

import com.example.model.convertor.RoleConventer;

import lombok.Getter;

import lombok.Setter;

@NamedNativeQuery(

name = "User.countCorrectAnswers",

query = "SELECT Count(correct\_answer) "

+ " FROM users LEFT JOIN users\_answers on users.user\_id = users\_answers.user\_id"

+ " LEFT JOIN questions on users\_answers.question\_id = questions.question\_id"

+ " WHERE username = :username AND answer = correct\_answer")

@Entity

@Table(name="users")

@Getter@Setter

public class User {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

@Column(name = "user\_id")

private Long id;

@NotBlank

@Column(unique = true, nullable = false)

private String username;

@Column(nullable = false)

private String password;

@NotBlank

@Email

@Column(unique = true, nullable = false)

private String email;

private String firstName;

private String lastName;

private LocalDateTime lastActive;

@Convert(converter = RoleConventer.class)

@Column(length = 4, nullable = false)

private Role role;

@ElementCollection(fetch = FetchType.LAZY)

@CollectionTable(name = "users\_answers", joinColumns = @JoinColumn(name = "user\_id"))

@MapKeyJoinColumn(name = "question\_id")

@Column(name = "answer")

private Map<Question, String> answers = new HashMap<>();

public void storeAnsweredQuestion(Question question, String answer) {

answers.put(question, answer);

}

@Override

public String toString() {

return username;

}

@Override

public boolean equals(Object other) {

if ( ! (other instanceof User))

return false;

User otherU = (User) other;

return this.username.equals(otherU.username);

}

@Override

public int hashCode() {

return this.username.hashCode();

}

}

Question:

package com.example.model;

import java.util.ArrayList;

import java.util.HashSet;

import java.util.List;

import javax.persistence.\*;

import javax.validation.constraints.NotBlank;

import javax.validation.constraints.NotEmpty;

import org.hibernate.validator.constraints.UniqueElements;

import org.springframework.data.annotation.CreatedBy;

import org.springframework.data.jpa.domain.support.AuditingEntityListener;

import lombok.AccessLevel;

import lombok.Getter;

import lombok.Setter;

import com.example.model.Question;

import com.example.model.User;

@Entity @EntityListeners(AuditingEntityListener.class)

@Table(name="questions")

@Getter @Setter

public class Question {

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

@Column(name = "question\_id")

private long id;

@CreatedBy

@ManyToOne(targetEntity = User.class, optional = false)

@JoinColumn(name="created\_by\_user", referencedColumnName = "user\_id", nullable = false)

private User createdBy;

/\* validation \*/

@NotBlank

//

@Column(nullable = false)

private String questionText;

@Column(nullable = false)

private String correctAnswer;

/\* validation \*/

@NotEmpty()

@UniqueElements(message = "Each answer has to be different.")

//

@ElementCollection(fetch = FetchType.LAZY)

@CollectionTable(name = "answers", joinColumns = @JoinColumn(name = "question\_id", referencedColumnName = "question\_id"))

@OrderColumn(name = "ordinal", columnDefinition = "tinyint")

@Column(name = "answer", nullable = false)

private List<@NotBlank String> answers = new ArrayList<String>(); // Includes the correctAnswer.

@Transient

private int selectedAnswerIndex; // Used in a form when user create a question.

@Transient

@Getter(value = AccessLevel.NONE)

@Setter(value = AccessLevel.NONE)

private int hash; // Default to 0

@Transient

@Getter(value = AccessLevel.NONE)

@Setter(value = AccessLevel.NONE)

private boolean hashIsZero; // Default to false;

public Question() {

}

@Override

public String toString() {

return questionText;

}

@Override

public boolean equals(Object obj) {

if (this == obj)

return true;

if ( ! (obj instanceof Question))

return false;

Question otherQ = (Question) obj;

if ( ! this.questionText.equals(otherQ.questionText))

return false;

/\* Check if answers are the same - irrelevant of order. \*/

return new HashSet<>(this.answers).equals(new HashSet<>(otherQ.answers));

}

@Override

public int hashCode() {

int h = hash;

if (h == 0 && !hashIsZero) {

h = questionText.hashCode();

for (String ans : answers) {

h += ans.hashCode();

}

if (h == 0) {

hashIsZero = true;

} else {

hash = h;

}

}

return h;

}

}

Role:

**package** com.example.model;

**import** com.example.model.Role;

**public** **enum** Role {

***USER***("Basic", "user"),

***CONTRIBUTOR***("Contributor", "cntr"),

***MODERATOR***("Moderator", "mod"),

***ADMINISTRATOR***("Administrator", "admn");

**public** **final** String displayValue;

**public** **final** String shortName; // arbitrary max lenght = 4

Role(String display, String shortName) {

**this**.displayValue = display;

**this**.shortName = shortName;

}

**public** **static** Role fromShortName(String shortName ) {

**switch** (shortName) {

**case** "user":

**return** ***USER***;

**case** "cntr":

**return** ***CONTRIBUTOR***;

**case** "mod":

**return** ***MODERATOR***;

**case** "admn":

**return** ***ADMINISTRATOR***;

**default**:

**throw** **new** IllegalArgumentException("Short[" + shortName + "] not supported.");

}

}

}

Role convertor

package com.example.model.convertor;

import javax.persistence.AttributeConverter;

import com.example.model.Role;

public class RoleConventer implements AttributeConverter<Role, String> {

@Override

public String convertToDatabaseColumn(Role role) {

return role.shortName;

}

@Override

public Role convertToEntityAttribute(String dbData) {

return Role.fromShortName(dbData);

}

}

**DTO:**

Question:

package com.example.dto;

import java.util.List;

import com.example.dto.QuestionDTO.CreatedBy;

public interface QuestionDTO {

Long getId();

CreatedBy getCreatedBy();

String getQuestionText();

String getCorrectAnswer();

List<String> getAnswers();

public interface CreatedBy {

String getUsername();

}

}

User: package com.example.dto;

import java.time.LocalDateTime;

import java.util.Arrays;

import lombok.Value;

import com.example.model.Role;

@Value

public class UserDTO {

String username;

String email;

String firstName;

String lastName;

Role role;

LocalDateTime lastActive;

public UserDTO(String username, String email, String firstName, String lastName, Role role, LocalDateTime lastActive) {

this.username = username;

this.email = redact(email);

this.firstName = firstName;

this.lastName = lastName;

this.role = role;

this.lastActive = lastActive;

}

private String redact(String emailFull) {

StringBuilder sb = new StringBuilder();

String[] parts = emailFull.split("@");

char[] chars = new char[parts[0].length()];

Arrays.fill(chars, '\*');

sb.append(chars);

switch (chars.length) {

case 0:

case 1: break;

case 2: {sb.setCharAt(0, parts[0].charAt(0)); break;}

default: {

int end = parts[0].length()-1;

sb.setCharAt(0, parts[0].charAt(0));

sb.setCharAt(end, parts[0].charAt(end));

break;

}

}

sb.append('@').append(parts[1]);

return sb.toString();

}

}

**SERVICE:**

QuestionService:

package com.example.service;

import java.util.ArrayList;

import java.util.List;

import java.util.Set;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.data.domain.Page;

import org.springframework.data.domain.PageRequest;

import org.springframework.data.domain.Pageable;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.example.dto.QuestionDTO;

import com.example.model.Question;

import com.example.repository.QuestionRepository;

@Service

@Transactional

public class QuestionService {

@Autowired

private QuestionRepository questionRepo;

public Set<QuestionDTO> listAll() {

return questionRepo.findBy();

}

public Set<QuestionDTO> listAllByUser(String username) {

return questionRepo.findByCreatedByUsername(username);

}

public int getNumberOfQuestions() {

return (int) questionRepo.count();

}

public QuestionDTO getQuestion(long questionId) {

return questionRepo.getById(questionId);

}

public Question getQuestionEntity(long id) {

return questionRepo.findById(id).orElseThrow();

}

public boolean exist(Question question) {

Set<Question> fromDB = questionRepo.findByQuestionText(question.getQuestionText());

return fromDB.contains(question);

}

public QuestionDTO getQuestionByIndex(int qIndex) {

Pageable pageRequest = PageRequest.of(qIndex, 1);

Page<QuestionDTO> page = questionRepo.findByOrderById(pageRequest);

if ( ! page.hasContent())

throw new IndexOutOfBoundsException();

return page.getContent().get(0);

}

public Question save(Question q) {

List<String> ans = q.getAnswers();

int markedAsCorrect = q.getSelectedAnswerIndex();

q.setCorrectAnswer(ans.get(markedAsCorrect));

return questionRepo.save(q);

}

public void delete(Long id) {

questionRepo.deleteById(id);

}

public List<String> getQuestionTexts() {

return questionRepo.findAllQuestionTexts();

}

}

QuizService:

package com.example.service;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import com.example.model.Question;

import com.example.model.User;

@Service

public class QuizService {

@Autowired

private QuestionService questionService;

@Autowired

private UserService userService;

public void storeUsersAnswer(String username, long questionId, String answer) {

Question question = questionService.getQuestionEntity(questionId);

User user = userService.getUser(username);

user.storeAnsweredQuestion(question, answer);

userService.update(user);

}

public int getUserScore(String username) {

return userService.getScore(username);

}

public void removeQuestion(long id) {

userService.removeFromUsersAnswers(id);

questionService.delete(id);

}

}

UserService:

package com.example.service;

import java.util.ArrayList;

import java.util.List;

import java.util.Set;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.example.dto.QuestionDTO;

import com.example.dto.UserDTO;

import com.example.model.User;

import com.example.repository.UserRepository;

@Service

@Transactional

public class UserService {

@Autowired

private UserRepository userRepo;

@Autowired

private BCryptPasswordEncoder bCryptPasswordEncoder;

public boolean usernameIsAvaible(String username) {

return ( ! userRepo.existsUserByUsername(username));

}

public boolean emailIsAvaible(String email) {

return ( ! userRepo.existsUserByEmail(email));

}

public User getUser(String username) {

return userRepo.findByUsername(username).orElseThrow();

}

public User register(User user) {

user.setPassword(bCryptPasswordEncoder.encode(user.getPassword()));

return userRepo.save(user);

}

public User update(User user) {

return userRepo.save(user);

}

public Set<UserDTO> getUsersInfo() {

return userRepo.getUsersInfo();

}

public void deleteUser(User user) {

userRepo.delete(user);

}

public Set<String> getUsernamesThatAnswered(QuestionDTO q, boolean correctly ) {

if (correctly)

return userRepo.getUsernamesThatAnsweredWith(q.getId(), q.getCorrectAnswer());

else

return userRepo.getUsernamesThatAnswered(q.getId());

}

public Set<String> getUsernamesThatAnsweredEveryQ(boolean correctly) {

if (correctly)

return userRepo.getUsernamesThatCorrectlyAnsweredEveryQuestion();

else

return userRepo.getUsernamesThatAnsweredEveryQuestion();

}

public int getScore(String username) {

return userRepo.countCorrectAnswers(username);

}

public void removeFromUsersAnswers(long questionId) {

userRepo.removeFromUsersAnswers(questionId);

}

public Float getAnsweredPercentage(long questionId) {

List<Integer> ansCount = userRepo.answersChosenCount(questionId);

int totalAnswers = 0;

for (Integer i : ansCount) {

totalAnswers += i;

}

int timesSkipped = userRepo.skippedCount(questionId);

if (totalAnswers + timesSkipped != 0)

return ((float) totalAnswers) / (totalAnswers + timesSkipped);

else

return -1.0f;

}

public List<Float> getAnswersDistribution(long questionId) {

List<Integer> ansCount = userRepo.answersChosenCount(questionId);

int totalAnswers = 0;

for (Integer i : ansCount) {

totalAnswers += i;

}

List<Float> ansDistribution = new ArrayList<>();

for (Integer i : ansCount) {

if (totalAnswers == 0) {

ansDistribution.add(0.0f);

continue;

}

ansDistribution.add( ((float) i )/ totalAnswers);

}

return ansDistribution;

}

}

**REPOSITORY:**

EntityAudtityrepo:

package com.example.repository;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.data.domain.AuditorAware;

import org.springframework.data.jpa.repository.config.EnableJpaAuditing;

import org.springframework.security.core.context.SecurityContextHolder;

import com.example.model.User;

@Configuration

@EnableJpaAuditing

public class EntityAuditingConfig {

@Autowired

private UserRepository userRepository;

@Bean

public AuditorAware<User> auditorAware() {

AuditorAware<User> currentAuditor = () -> {String signedInUser = SecurityContextHolder.getContext().getAuthentication().getName();

return userRepository.findByUsername(signedInUser);};

return currentAuditor;

}

}

QuestionRepo:

Package com.example.repository;

import java.util.List;

import java.util.Set;

import org.springframework.data.domain.Page;

import org.springframework.data.domain.Pageable;

import org.springframework.data.jpa.repository.Query;

import org.springframework.data.repository.PagingAndSortingRepository;

import com.example.dto.QuestionDTO;

import com.example.model.Question;

public interface QuestionRepository extends PagingAndSortingRepository<Question, Long> {

public QuestionDTO getById(Long questionId);

public Set<QuestionDTO> findBy();

public Set<QuestionDTO> findByCreatedByUsername(String username);

public Set<Question> findByQuestionText(String questionText);

public Page<QuestionDTO> findByOrderById(Pageable pageRequest);

@Query("SELECT q.questionText FROM Question q ORDER BY q.id")

public List<String> findAllQuestionTexts();

}

UserRepo:

package com.example.repository;

import java.util.List;

import java.util.Optional;

import java.util.Set;

import org.springframework.data.jpa.repository.Modifying;

import org.springframework.data.jpa.repository.Query;

import org.springframework.data.repository.PagingAndSortingRepository;

import com.example.dto.UserDTO;

import com.example.model.User;

public interface UserRepository extends PagingAndSortingRepository<User, Long> {

public Boolean existsUserByUsername(String username);

public boolean existsUserByEmail(String email);

public Optional<User> findByUsername(String username);

@Query("SELECT NEW ujfaA.springQuiz.dto.UserDTO(u.username, u.email, u.firstName, u.lastName, u.role, u.lastActive)"

+ " FROM User u")

public Set<UserDTO> getUsersInfo();

public int countCorrectAnswers(String username);

@Query("SELECT u.username"

+ " FROM User u, IN(u.answers) ans"

+ " WHERE KEY(ans).id = :questionId AND ans <> '' ")

public Set<String> getUsernamesThatAnswered(long questionId);

@Query("SELECT u.username"

+ " FROM User u, IN(u.answers) ans"

+ " WHERE KEY(ans).id = :questionId AND ans = :answer")

public Set<String> getUsernamesThatAnsweredWith(long questionId, String answer);

@Query(nativeQuery = true,

value = "SELECT users.username"

+ " FROM users RIGHT JOIN ("

+ " SELECT user\_id, count(\*) AS questionsAnswered"

+ " FROM users\_answers"

+ " WHERE answer <> '' "

+ " GROUP BY user\_id ) AS t1"

+ " ON users.user\_id = t1.user\_id"

+ " WHERE questionsAnswered = (SELECT count(\*) FROM questions)")

public Set<String> getUsernamesThatAnsweredEveryQuestion();

@Query(nativeQuery = true,

value = "SELECT users.username"

+ " FROM users RIGHT JOIN ("

+ " SELECT user\_id, count(\*) AS questionsAnsweredCorrectly"

+ " FROM users\_answers NATURAL JOIN questions"

+ " WHERE answer = correct\_answer"

+ " GROUP BY user\_id ) AS t1"

+ " ON users.user\_id = t1.user\_id"

+ " WHERE questionsAnsweredCorrectly = (SELECT count(\*) FROM questions)")

public Set<String> getUsernamesThatCorrectlyAnsweredEveryQuestion();

@Modifying

@Query(nativeQuery = true,

value = "DELETE FROM users\_answers WHERE question\_id = ?1")

public void removeFromUsersAnswers(long questionId);

@Query("SELECT count (answer)"

+ " FROM User u, In(u.answers) ans"

+ " WHERE KEY(ans).id = :questionId AND ans = '' ")

public int skippedCount(long questionId);

@Query(nativeQuery = true,

value = "SELECT count(users\_answers.answer)"

+ " FROM answers left join users\_answers on answers.answer = users\_answers.answer AND answers.question\_id = users\_answers.question\_id"

+ " WHERE answers.question\_id = :questionId"

+ " GROUP BY answers.answer"

+ " ORDER BY ordinal")

public List<Integer> answersChosenCount(long questionId);

}

**CONTROLLER:**

QuizAdminController:

package com.example.controller;

import java.security.Principal;

import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.security.core.Authentication;

import org.springframework.security.core.authority.SimpleGrantedAuthority;

import org.springframework.stereotype.Controller;

import org.springframework.ui.ModelMap;

import org.springframework.validation.BindingResult;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.ModelAttribute;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.server.ResponseStatusException;

import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import com.example.dto.QuestionDTO;

import com.example.model.Question;

import com.example.service.QuestionService;

import com.example.service.QuizService;

import com.example.service.UserService;

@Controller

public class QuizAdministratorController {

@Autowired

private QuestionService questionService;

@Autowired

private UserService userService;

@Autowired

private QuizService quizService;

@GetMapping("/questions")

public String getQuestions(ModelMap model) {

model.addAttribute("questions", questionService.listAll());

return "questions";

}

@GetMapping("/questions/byMe")

public String getQuestionsByUser(Principal principal, ModelMap model) {

String currentUsername = principal.getName();

model.addAttribute("questions", questionService.listAllByUser(currentUsername));

return "questions";

}

@GetMapping("/questions/new")

public String newQuestion(@ModelAttribute Question question,

@RequestParam(name = "numberOfAnswers", defaultValue = "3") int numberOfAnswers,

ModelMap model) {

model.addAttribute("numberOfAnswers", numberOfAnswers);

model.addAttribute("MAX\_ANSWERS", 5);

return "newQuestion";

}

@PostMapping("/questions/new")

public String addQuestion(@Valid Question question, BindingResult bindingResult, ModelMap model, RedirectAttributes redirectAttrs) {

if (bindingResult.hasErrors()) {

model.addAttribute("numberOfAnswers", question.getAnswers().size());

model.addAttribute("MAX\_ANSWERS", 5);

return "newQuestion";

}

if (questionService.exist(question)) {

redirectAttrs.addAttribute("numberOfAnswers", question.getAnswers().size());

redirectAttrs.addFlashAttribute("message", "A question like this already exist.");

redirectAttrs.addFlashAttribute(question);

return"redirect:/questions/new";

}

try {

questionService.save(question);

} catch (Exception e) {

redirectAttrs.addAttribute("numberOfAnswers", question.getAnswers().size());

redirectAttrs.addFlashAttribute("message", "There was an error while adding the question.");

redirectAttrs.addFlashAttribute(question);

return"redirect:/questions/new";

}

return "redirect:/questions/byMe";

}

@GetMapping("/questions/{qId:[0-9]+}")

public String getQuestionStats(@PathVariable(name = "qId") long qId, Authentication auth, RedirectAttributes redirectAttrs, ModelMap model) {

QuestionDTO question = questionService.getQuestion(qId);

if (question == null)

throw new ResponseStatusException(HttpStatus.NOT\_FOUND, "Question not found.");

if (auth.getAuthorities().contains(new SimpleGrantedAuthority("ROLE\_CONTRIBUTOR")) ) {

String author = question.getCreatedBy().getUsername();

if ( ! auth.getName().equals(author)) {

redirectAttrs.addFlashAttribute("message","You don't have the required permission to see the question's stats.");

return "redirect:/questions/byMe";

}

}

model.addAttribute("question", question);

model.addAttribute("answeredPercentage", userService.getAnsweredPercentage(qId));

model.addAttribute("answersDistribution", userService.getAnswersDistribution(qId));

return "questionStats";

}

@PostMapping("/questions/delete")

public String removeQuestion(@RequestParam Long qId, Authentication auth, RedirectAttributes redirectAttrs) {

String redirectModifier = "";

if (auth.getAuthorities().contains(new SimpleGrantedAuthority("ROLE\_CONTRIBUTOR")) ) {

QuestionDTO q = questionService.getQuestion(qId);

String author = q.getCreatedBy().getUsername();

if ( ! auth.getName().equals(author)) {

redirectAttrs.addFlashAttribute("message", "You don't have the required permission to delete the question.");

return "redirect:/questions/byMe";

}

redirectModifier = "/byMe";

}

String message;

try {

quizService.removeQuestion(qId);

message = "The question has been removed.";

} catch (Exception e) {

message = "There had been an error while trying to remove the question.";

}

redirectAttrs.addFlashAttribute("message", message);

return "redirect:/questions" + redirectModifier;

}

@GetMapping("/users")

public String getUsersInfo(ModelMap model) {

model.addAttribute("users", userService.getUsersInfo());

return "usersInfo";

}

@GetMapping("/users/usersEng")

public String getUserEngagement(@RequestParam(name = "q", defaultValue = "-1") int qIndex,

@RequestParam(name = "correctly", defaultValue = "false") boolean correctly,

ModelMap model) {

model.addAttribute("texts", questionService.getQuestionTexts());

model.addAttribute("selectedIndex", qIndex);

model.addAttribute("checked", correctly);

if (qIndex == -1) {

model.addAttribute("usernames", userService.getUsernamesThatAnsweredEveryQ(correctly));

}

else {

QuestionDTO q = questionService.getQuestionByIndex(qIndex);

model.addAttribute("usernames", userService.getUsernamesThatAnswered(q, correctly));

}

return"usersEngagement";

}

}

QuizController:

package com.example.controller;

import java.security.Principal;

import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.security.core.Authentication;

import org.springframework.security.core.authority.SimpleGrantedAuthority;

import org.springframework.stereotype.Controller;

import org.springframework.ui.ModelMap;

import org.springframework.validation.BindingResult;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.ModelAttribute;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.server.ResponseStatusException;

import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import com.example.dto.QuestionDTO;

import com.example.model.Question;

import com.example.service.QuestionService;

import com.example.service.QuizService;

import com.example.service.UserService;

@Controller

public class QuizAdministratorController {

@Autowired

private QuestionService questionService;

@Autowired

private UserService userService;

@Autowired

private QuizService quizService;

@GetMapping("/questions")

public String getQuestions(ModelMap model) {

model.addAttribute("questions", questionService.listAll());

return "questions";

}

@GetMapping("/questions/byMe")

public String getQuestionsByUser(Principal principal, ModelMap model) {

String currentUsername = principal.getName();

model.addAttribute("questions", questionService.listAllByUser(currentUsername));

return "questions";

}

@GetMapping("/questions/new")

public String newQuestion(@ModelAttribute Question question,

@RequestParam(name = "numberOfAnswers", defaultValue = "3") int numberOfAnswers,

ModelMap model) {

model.addAttribute("numberOfAnswers", numberOfAnswers);

model.addAttribute("MAX\_ANSWERS", 5);

return "newQuestion";

}

@PostMapping("/questions/new")

public String addQuestion(@Valid Question question, BindingResult bindingResult, ModelMap model, RedirectAttributes redirectAttrs) {

if (bindingResult.hasErrors()) {

model.addAttribute("numberOfAnswers", question.getAnswers().size());

model.addAttribute("MAX\_ANSWERS", 5);

return "newQuestion";

}

if (questionService.exist(question)) {

redirectAttrs.addAttribute("numberOfAnswers", question.getAnswers().size());

redirectAttrs.addFlashAttribute("message", "A question like this already exist.");

redirectAttrs.addFlashAttribute(question);

return"redirect:/questions/new";

}

try {

questionService.save(question);

} catch (Exception e) {

redirectAttrs.addAttribute("numberOfAnswers", question.getAnswers().size());

redirectAttrs.addFlashAttribute("message", "There was an error while adding the question.");

redirectAttrs.addFlashAttribute(question);

return"redirect:/questions/new";

}

return "redirect:/questions/byMe";

}

@GetMapping("/questions/{qId:[0-9]+}")

public String getQuestionStats(@PathVariable(name = "qId") long qId, Authentication auth, RedirectAttributes redirectAttrs, ModelMap model) {

QuestionDTO question = questionService.getQuestion(qId);

if (question == null)

throw new ResponseStatusException(HttpStatus.NOT\_FOUND, "Question not found.");

if (auth.getAuthorities().contains(new SimpleGrantedAuthority("ROLE\_CONTRIBUTOR")) ) {

String author = question.getCreatedBy().getUsername();

if ( ! auth.getName().equals(author)) {

redirectAttrs.addFlashAttribute("message","You don't have the required permission to see the question's stats.");

return "redirect:/questions/byMe";

}

}

model.addAttribute("question", question);

model.addAttribute("answeredPercentage", userService.getAnsweredPercentage(qId));

model.addAttribute("answersDistribution", userService.getAnswersDistribution(qId));

return "questionStats";

}

@PostMapping("/questions/delete")

public String removeQuestion(@RequestParam Long qId, Authentication auth, RedirectAttributes redirectAttrs) {

String redirectModifier = "";

if (auth.getAuthorities().contains(new SimpleGrantedAuthority("ROLE\_CONTRIBUTOR")) ) {

QuestionDTO q = questionService.getQuestion(qId);

String author = q.getCreatedBy().getUsername();

if ( ! auth.getName().equals(author)) {

redirectAttrs.addFlashAttribute("message", "You don't have the required permission to delete the question.");

return "redirect:/questions/byMe";

}

redirectModifier = "/byMe";

}

String message;

try {

quizService.removeQuestion(qId);

message = "The question has been removed.";

} catch (Exception e) {

message = "There had been an error while trying to remove the question.";

}

redirectAttrs.addFlashAttribute("message", message);

return "redirect:/questions" + redirectModifier;

}

@GetMapping("/users")

public String getUsersInfo(ModelMap model) {

model.addAttribute("users", userService.getUsersInfo());

return "usersInfo";

}

@GetMapping("/users/usersEng")

public String getUserEngagement(@RequestParam(name = "q", defaultValue = "-1") int qIndex,

@RequestParam(name = "correctly", defaultValue = "false") boolean correctly,

ModelMap model) {

model.addAttribute("texts", questionService.getQuestionTexts());

model.addAttribute("selectedIndex", qIndex);

model.addAttribute("checked", correctly);

if (qIndex == -1) {

model.addAttribute("usernames", userService.getUsernamesThatAnsweredEveryQ(correctly));

}

else {

QuestionDTO q = questionService.getQuestionByIndex(qIndex);

model.addAttribute("usernames", userService.getUsernamesThatAnswered(q, correctly));

}

return"usersEngagement";

}

}

SecurityController:

package com.example.controller;

import java.security.Principal;

import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.security.core.Authentication;

import org.springframework.security.core.authority.SimpleGrantedAuthority;

import org.springframework.stereotype.Controller;

import org.springframework.ui.ModelMap;

import org.springframework.validation.BindingResult;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.ModelAttribute;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.server.ResponseStatusException;

import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import com.example.dto.QuestionDTO;

import com.example.model.Question;

import com.example.service.QuestionService;

import com.example.service.QuizService;

import com.example.service.UserService;

@Controller

public class QuizAdministratorController {

@Autowired

private QuestionService questionService;

@Autowired

private UserService userService;

@Autowired

private QuizService quizService;

@GetMapping("/questions")

public String getQuestions(ModelMap model) {

model.addAttribute("questions", questionService.listAll());

return "questions";

}

@GetMapping("/questions/byMe")

public String getQuestionsByUser(Principal principal, ModelMap model) {

String currentUsername = principal.getName();

model.addAttribute("questions", questionService.listAllByUser(currentUsername));

return "questions";

}

@GetMapping("/questions/new")

public String newQuestion(@ModelAttribute Question question,

@RequestParam(name = "numberOfAnswers", defaultValue = "3") int numberOfAnswers,

ModelMap model) {

model.addAttribute("numberOfAnswers", numberOfAnswers);

model.addAttribute("MAX\_ANSWERS", 5);

return "newQuestion";

}

@PostMapping("/questions/new")

public String addQuestion(@Valid Question question, BindingResult bindingResult, ModelMap model, RedirectAttributes redirectAttrs) {

if (bindingResult.hasErrors()) {

model.addAttribute("numberOfAnswers", question.getAnswers().size());

model.addAttribute("MAX\_ANSWERS", 5);

return "newQuestion";

}

if (questionService.exist(question)) {

redirectAttrs.addAttribute("numberOfAnswers", question.getAnswers().size());

redirectAttrs.addFlashAttribute("message", "A question like this already exist.");

redirectAttrs.addFlashAttribute(question);

return"redirect:/questions/new";

}

try {

questionService.save(question);

} catch (Exception e) {

redirectAttrs.addAttribute("numberOfAnswers", question.getAnswers().size());

redirectAttrs.addFlashAttribute("message", "There was an error while adding the question.");

redirectAttrs.addFlashAttribute(question);

return"redirect:/questions/new";

}

return "redirect:/questions/byMe";

}

@GetMapping("/questions/{qId:[0-9]+}")

public String getQuestionStats(@PathVariable(name = "qId") long qId, Authentication auth, RedirectAttributes redirectAttrs, ModelMap model) {

QuestionDTO question = questionService.getQuestion(qId);

if (question == null)

throw new ResponseStatusException(HttpStatus.NOT\_FOUND, "Question not found.");

if (auth.getAuthorities().contains(new SimpleGrantedAuthority("ROLE\_CONTRIBUTOR")) ) {

String author = question.getCreatedBy().getUsername();

if ( ! auth.getName().equals(author)) {

redirectAttrs.addFlashAttribute("message","You don't have the required permission to see the question's stats.");

return "redirect:/questions/byMe";

}

}

model.addAttribute("question", question);

model.addAttribute("answeredPercentage", userService.getAnsweredPercentage(qId));

model.addAttribute("answersDistribution", userService.getAnswersDistribution(qId));

return "questionStats";

}

@PostMapping("/questions/delete")

public String removeQuestion(@RequestParam Long qId, Authentication auth, RedirectAttributes redirectAttrs) {

String redirectModifier = "";

if (auth.getAuthorities().contains(new SimpleGrantedAuthority("ROLE\_CONTRIBUTOR")) ) {

QuestionDTO q = questionService.getQuestion(qId);

String author = q.getCreatedBy().getUsername();

if ( ! auth.getName().equals(author)) {

redirectAttrs.addFlashAttribute("message", "You don't have the required permission to delete the question.");

return "redirect:/questions/byMe";

}

redirectModifier = "/byMe";

}

String message;

try {

quizService.removeQuestion(qId);

message = "The question has been removed.";

} catch (Exception e) {

message = "There had been an error while trying to remove the question.";

}

redirectAttrs.addFlashAttribute("message", message);

return "redirect:/questions" + redirectModifier;

}

@GetMapping("/users")

public String getUsersInfo(ModelMap model) {

model.addAttribute("users", userService.getUsersInfo());

return "usersInfo";

}

@GetMapping("/users/usersEng")

public String getUserEngagement(@RequestParam(name = "q", defaultValue = "-1") int qIndex,

@RequestParam(name = "correctly", defaultValue = "false") boolean correctly,

ModelMap model) {

model.addAttribute("texts", questionService.getQuestionTexts());

model.addAttribute("selectedIndex", qIndex);

model.addAttribute("checked", correctly);

if (qIndex == -1) {

model.addAttribute("usernames", userService.getUsernamesThatAnsweredEveryQ(correctly));

}

else {

QuestionDTO q = questionService.getQuestionByIndex(qIndex);

model.addAttribute("usernames", userService.getUsernamesThatAnswered(q, correctly));

}

return"usersEngagement";

}

}

**Index.html**

<!DOCTYPE html>

<html xmlns:th=*"http://www.thymeleaf.org"*

xmlns:sec=*"http://www.thymeleaf.org/thymeleaf-extras-springsecurity5"*>

<head>

<meta charset=*"UTF-8"*>

<link rel=*"stylesheet"* href=*"https://www.w3schools.com/w3css/4/w3.css"*>

<title> Spring Quiz - Home</title>

</head>

<body>

<header

th:replace=*"~{fragments/HeaderAndFooter :: header}"*>

</header>

<nav

th:replace=*"~{fragments/Navigation :: topBar}"* >

</nav>

<div class=*"w3-container w3-padding-large"* align=*"center"*>

<h2 th:text = *"${'Welcome ' + session.userFirstName}"*> </h2>

</div>

<div class=*"w3-container w3-padding-large"* align=*"center"*>

<a th:href=*"@{/quiz/start}"* class=*"w3-btn w3-large w3-green"*>Start Quiz</a>

</div>

<footer

th:replace=*"~{fragments/HeaderAndFooter :: footer}"*>

</footer>

</body>

</html>

**Login.html**

<!DOCTYPE html>

<html xmlns:th=*"http://www.thymeleaf.org"*>

<head>

<meta charset=*"UTF-8"*>

<link rel=*"stylesheet"* href=*"https://www.w3schools.com/w3css/4/w3.css"*>

<title>Spring Quiz - Login</title>

</head>

<body>

<header

th:replace=*"~{fragments/HeaderAndFooter :: header}"*>

</header>

<nav

th:replace=*"~{fragments/Navigation :: topBar}"* >

</nav>

<div class=*"w3-container"* align=*"center"*>

<h2 th:text=*"Welcome"*></h2>

</div>

<div class=*"w3-panel"* align=*"center"*>

<h4 align=*"center"*>Please log in:</h4>

<form class =*"w3-container w3-card w3-light-grey"* style="width:*50%*;"

th:action=*"@{/login}"* method=*"post"*>

<p align=*"left"*>

<label th:for=*"username"*>Username:</label>

<input class=*"w3-input w3-border"*

type=*"text"* name=*"username"* id=*"username"* required=*"required"* autofocus=*"autofocus"* />

</p>

<p align=*"left"*>

<label for=*"password"*>Password:</label>

<input class=*"w3-input w3-border"*

type=*"password"* name=*"password"* id=*"password"* required=*"required"* />

</p>

<p>

<input class=*"w3-btn w3-block w3-green w3-m"*

type=*"submit"* value=*"Login"*>

</p>

</form>

</div>

<div align=*"center"*>

<p th:text=*"${message}"*></p>

</div>

<div align=*"center"*>

<p> If you don't have an account click

<a th:href=*"@{/registration}"*> Register new account</a>

</p>

</div>

<footer

th:replace=*"~{fragments/HeaderAndFooter :: footer}"*>

</footer>

</body>

</html>

**Registration.html**

<!DOCTYPE html>

<html xmlns:th=*"http://www.thymeleaf.org"*>

<head>

<meta charset=*"UTF-8"*>

<link rel=*"stylesheet"* href=*"https://www.w3schools.com/w3css/4/w3.css"*>

<title>Spring Quiz - registration</title>

</head>

<body>

<header

th:replace=*"~{fragments/HeaderAndFooter :: header}"*>

</header>

<nav

th:replace=*"~{fragments/Navigation :: topBar}"* >

</nav>

<div class=*"w3-panel"* align=*"center"*>

<h4 align=*"center"*>Register new account</h4>

<form class=*"w3-container w3-card w3-light-grey"* style="width:*50%*;"

action=*"#"* th:action=*"@{/registration}"* th:object=*"${user}"* method=*"post"*>

<p align=*"left"*>

<label th:for=*"firstName"*>First Name:</label>

<input class=*"w3-input w3-border"*

th:field=*"\*{firstName}"* autofocus />

<label class=*"w3-text-red"*

th:errors=*"\*{firstName}"*> input error </label>

</p>

<p align=*"left"*>

<label th:for=*"lastName"*>Last Name:</label>

<input class=*"w3-input w3-border"*

th:field=*"\*{lastName}"* />

<label class=*"w3-text-red"*

th:errors=*"\*{lastName}"*> input error </label>

</p>

<p class=*"w3-tooltip"* align=*"left"*>

<label th:for=*"email"*>Email: \*</label>

<span class=*"w3-text w3-tag w3-animate-opacity"*> As this is demo app, any email in the correct format is OK.</span>

<input class=*"w3-input w3-border"*

type=*"email"* th:field=*"\*{email}"* required />

</p>

<p align=*"left"*>

<label th:for=*"username"*>Username: \*</label>

<input class=*"w3-input w3-border"*

th:field=*"\*{username}"* required />

<label class=*"w3-text-red"*

th:errors=*"\*{username}"*> username error </label>

</p>

<p align=*"left"*>

<label th:for=*"password"*>Password: \*</label>

<input class=*"w3-input w3-border"*

type=*"password"* th:field=*"\*{password}"* required />

<label class=*"w3-text-red"*

th:errors=*"\*{password}"*> password error </label>

<p align=*"left"*>

<span class=*"w3-button w3-hover-none w3-text-grey w3-right w3-small"*

style="margin-bottom: *8px*;"

onclick="document.getElementById('id01').style.display='block'">More

about the account types</span>

<label th:for=*"role"*>Account type:</label>

<select class=*"w3-select w3-border"* th:field=*"\*{role}"*>

<option th:each=*"r : ${T(ujfaA.springQuiz.model.Role).values()}"*

class=*"w3-border-white w3-hover-border-green"*

th:value=*"${r}"* th:text=*"${r.displayValue}"*></option>

</select>

</p>

<p>

<input class=*"w3-btn w3-block w3-green w3-m"*

type=*"submit"* value=*"Register"* />

</p>

</form>

</div>

<!-- modal -->

<div id=*"id01"* class=*"w3-modal"*>

<div class=*"w3-modal-content"*>

<header class=*"w3-container w3-green"*>

<span onclick="document.getElementById('id01').style.display='none'"

class=*"w3-button w3-display-topright"*>&times;</span>

<h3>Account types:</h3>

</header>

<div class=*"w3-container"*>

<p>With <b>Basic</b> account you will be able to take part in the quiz - answer questions and see a score at the end.</p>

<p>With <b>Contributor's</b> account you will be able to add take part in the quiz, but also to add and remove questions that you added.</p>

<p>With <b>Moderator's</b> account in addition to the above, you will be able to see and delete questions added by anyone. You will also have access to some stats about the quiz.</p>

<p>With <b>Administrator's</b> account you will have full access. <b>(recommended)</b></p>

</div>

<footer class=*"w3-container w3-green w3-center"*>

<span onclick="document.getElementById('id01').style.display='none'"

class=*"w3-button w3-hover-none w3-text-white"* style="text-decoration: *underline*;">close</span>

</footer>

</div>

</div>

<div align=*"center"*>

<p th:text=*"${message}"*></p>

</div>

<footer th:replace=*"~{fragments/HeaderAndFooter :: footer}"*> </footer>

</body>

</html>

**Userinfo.html**

<!DOCTYPE html>

<html xmlns:th=*"http://www.thymeleaf.org"*

xmlns:sec=*"http://www.thymeleaf.org/thymeleaf-extras-springsecurity5"*>

<head>

<meta charset=*"UTF-8"*>

<link rel=*"stylesheet"* href=*"https://www.w3schools.com/w3css/4/w3.css"*>

<title> Spring Quiz - Users info</title>

</head>

<body>

<header

th:replace=*"~{fragments/HeaderAndFooter :: header}"*>

</header>

<nav

th:replace=*"~{fragments/Navigation :: topBar}"* >

</nav>

<div class=*"w3-container w3-center w3-padding-large"*>

<h2> Users info </h2>

</div>

<div class=*"w3-container w3-center"*>

<table class =*"w3-table w3-bordered w3-centered"*>

<thead>

<tr>

<th>Username</th>

<th>First name</th>

<th>Last name</th>

<th>Email</th>

<th>Account</th>

<th>Last logged in</th>

</tr>

</thead>

<tbody>

<tr th:each=*"u : ${users}"* >

<td th:text=*"${u.username}"*></td>

<td th:text=*"${u.firstName}"*></td>

<td th:text=*"${u.lastName}"*></td>

<td th:text=*"${u.email}"*></td>

<td th:text=*"${u.role.displayValue}"*></td>

<td th:text=*"${u.lastActive}"*></td>

</tr>

</tbody>

</table>

</div>

<footer

th:replace=*"~{fragments/HeaderAndFooter :: footer}"*>

</footer>

</body>

</html>

**Userengagement.html**

<!DOCTYPE html>

<html xmlns:th=*"http://www.thymeleaf.org"*

xmlns:sec=*"http://www.thymeleaf.org/thymeleaf-extras-springsecurity5"*>

<head>

<meta charset=*"UTF-8"*>

<link rel=*"stylesheet"* href=*"https://www.w3schools.com/w3css/4/w3.css"*>

<title> Spring Quiz - Administrator - Questions - User engagement</title>

</head>

<body>

<header

th:replace=*"~{fragments/HeaderAndFooter :: header}"*>

</header>

<nav

th:replace=*"~{fragments/Navigation :: topBar}"* >

</nav>

<div class=*"w3-container w3-center w3-padding-large"*>

<h2>User engagement</h2></div>

<form class=*" w3-container w3-center"*

th:action=*"@{/users/usersEng}"* method=*"get"*>

<label class=*"w3-medium"* th:for=*"select"*>Users that answered question: </label>

<select class=*"w3-select w3-center"* style="width: *auto*;" name=*"q"* th:id=*"select"*>

<option th:each=*"question, loop : ${texts}"*

th:text=*"${question}"* th:value=*"${loop.index}"*

th:selected=*"${loop.index EQ selectedIndex}"* />

<option th:text=*"'every question'"* th:value=*"-1"* th:selected=*"${selectedIndex == -1}"* />

</select>

<p>

<label th:for=*"cbox"*> Show only users that answered correctly</label>

<input class=*"w3-check"*

type=*"checkbox"* name=*"correctly"* th:id=*"cbox"* th:checked=*"${checked}"*/></p>

<input class=*"w3-btn w3-green w3-small"* type=*"submit"* value=*"Show"*/>

</form>

<div class=*" w3-container w3-center w3-mmargin-top"*>

<p th:each=*"username : ${usernames}"*

th:text =*"${username}"*> username </p>

<p th:if=*"${#sets.isEmpty(usernames)}"*

th:text=*"${'None of the users answered'*

*+ (selectedIndex == -1 ? ' every' : ' the') + ' question'*

*+ (checked ? ' correctly' : '') + '.'}"*> username</p>

</div>

<footer

th:replace=*"~{fragments/HeaderAndFooter :: footer}"*>

</footer>

</body>

</html>

**Question.html**

<!DOCTYPE html>

<html xmlns:th=*"http://www.thymeleaf.org"*

xmlns:sec=*"http://www.thymeleaf.org/thymeleaf-extras-springsecurity5"*>

<head>

<meta charset=*"UTF-8"*>

<link rel=*"stylesheet"* href=*"https://www.w3schools.com/w3css/4/w3.css"*>

<link rel=*"stylesheet"* href=*"https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css"*>

<title> Spring Quiz - Administrator - Questions</title>

</head>

<body>

<header

th:replace=*"~{fragments/HeaderAndFooter :: header}"*>

</header>

<nav

th:replace=*"~{fragments/Navigation :: topBar}"* >

</nav>

<div class=*"w3-container w3-center w3-padding-large"*>

<h2 th:text=*"'Questions' + (${param.byMe}? ' added by me' : '')"*>Questions</h2>

</div>

<div class=*"w3-container w3-center w3-padding-large"*>

<a th:href=*"@{/questions/new}"* class=*"w3-btn w3-large w3-green"*> Add new </a>

</div>

<div align=*"center"*>

<p th:text=*"${message}"*></p>

</div>

<div align=*"center"*>

<div class=*"w3-container w3-border w3-border-white w3-hover-border-green w3-section "* align=*"justify"* style="width:*70%*;"

th:each=*"question : ${questions}"*>

<div class=*"w3-bar"*>

<h5 class=*"w3-bar-item"*> <b th:text=*"${question.questionText}"*>Question text</b></h5>

<form class=*"w3-bar-item w3-right"*

sec:authorize = *"hasRole('CONTRIBUTOR')"*

th:action=*"@{/questions/delete}"* method=*"post"*>

<input type=*"hidden"* name=*"qId"* th:value=*"${question.id}"*/>

<input type=*"hidden"* name=*"createdBy"* th:value=*"${question.createdBy.username}"*/>

<input class=*"w3-button w3-small"*

type=*"submit"* value=*"delete"* />

</form>

<div class=*"w3-bar-item w3-right"*>

<a class=*" w3-button w3-small"*

th:href=*"@{/questions/{id}(id=${question.id})}"*>

<i class=*"fa fa-info-circle w3-large"* aria-hidden=*"true"*></i>

info </a>

</div>

</div>

<ul class=*"w3-ul"*>

<li class=*"w3-border-bottom"*

th:each=*"answer,iterStat : ${question.answers}"*>

<span th:text=*"${iterStat.count +'. ' + answer + ' '}"*>Answer</span>

<span class=*"fa fa-check w3-large w3-green"*

th:if=*"${answer EQ question.correctAnswer}"*>

</span>

</li>

</ul>

</div>

</div>

<footer

th:replace=*"~{fragments/HeaderAndFooter :: footer}"*>

</footer>

</body>

</html>

**NewQuestion.html**

<!DOCTYPE html>

<html xmlns:th=*"http://www.thymeleaf.org"*

xmlns:sec=*"http://www.thymeleaf.org/thymeleaf-extras-springsecurity5"*>

<head>

<meta charset=*"UTF-8"*>

<link rel=*"stylesheet"* href=*"https://www.w3schools.com/w3css/4/w3.css"*>

<title> Spring Quiz - Questions - New</title>

</head>

<body>

<header

th:replace=*"~{fragments/HeaderAndFooter :: header}"*>

</header>

<nav

th:replace=*"~{fragments/Navigation :: topBar}"* >

</nav>

<div class=*"w3-panel w3-padding w3-center"*>

<h4 align=*"center"*>Create new question</h4>

<form th:action=*"@{/questions/new}"* method=*"get"*>

<p th:text= *"'Number of answers: '"*> </p>

<select name=*"numberOfAnswers"*>

<option th:each=*"n : ${#numbers.sequence(2, MAX\_ANSWERS)}"*

th:value=*"${n}"* th:text=*"${n}"* th:selected=*"${n EQ numberOfAnswers}"*></option>

</select>

<input type=*"submit"* value=*"Refresh"* />

</form>

</div>

<div class=*"w3-panel"* align=*"center"*>

<form class=*"w3-container w3-card w3-light-grey"* style="width:*80%*;"

action=*"#"* th:action=*"@{/questions/new}"* th:object=*"${question}"* method=*"post"*>

<p align=*"left"*>

<label th:for=*"questionText"*>Question:</label>

<label class=*"w3-text-red"*

th:errors=*"\*{questionText}"*> QuestionText error </label>

<input class=*"w3-input w3-border"*

type=*"text"* th:field=*"\*{questionText}"*

required autofocus />

</p>

<div class=*"w3-quarter w3-right"* align=*"center"*>

<label th:text=*"'Correct answer:'"*> Correct answer </label>

</div>

<div th:each=*"ansCount : ${#numbers.sequence(1, numberOfAnswers)}"*>

<div align=*"left"*>

<label th:for=*"${#ids.next('answers')}"*

th:text=*"${'Answer ' + ansCount + ':'}"*>Answer:

</label>

<!-- Using expression pre-processing, so that "index" variable is resolved before (at a "first pass")

and the Spring EL engine receives a numeric literal when it executes the whole expression: answers[index]

http://forum.thymeleaf.org/How-to-render-Spring-fields-in-a-loop-tp4025542p4025556.html-->

<label class=*"w3-text-red"*

th:errors=*"\*{answers[\_\_${ansCountStat.index}\_\_]}"*> Answer error </label>

</div>

<div class=*"w3-row"*>

<div class=*"w3-threequarter"*>

<input class=*" w3-input w3-border"*

type=*"text"* name=*"answers"* th:value=*"${ ansCount <= #lists.size(question.answers) ? question.answers[ansCountStat.index] : ''}"*

th:id=*"${#ids.seq('answers')}"*

required />

</div>

<div class=*"w3-quarter"*>

<input class=*"w3-radio"*

type=*"radio"* th:field=*"\*{selectedAnswerIndex}"* th:value=*"${ansCountStat.index}"*/>

</div>

</div>

</div>

<label class=*"w3-text-red"* th:errors=*"\*{answers}"*> Answers error </label>

<p>

<input class=*"w3-btn w3-block w3-green w3-m"*

type=*"submit"* value=*"Add question"* />

</p>

</form>

</div>

<p align=*"center"* th:text=*"${message}"*></p>

<footer

th:replace=*"~{fragments/HeaderAndFooter :: footer}"*>

</footer>

</body>

</html>

**QuestionStatus.html**

<!DOCTYPE html>

<html xmlns:th=*"http://www.thymeleaf.org"*>

<head>

<meta charset=*"UTF-8"*>

<link rel=*"stylesheet"* href=*"https://www.w3schools.com/w3css/4/w3.css"*>

<link rel=*"stylesheet"* href=*"https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css"*>

<title> Spring Quiz - Administrator - Question Stats</title>

</head>

<body>

<header

th:replace=*"~{fragments/HeaderAndFooter :: header}"*>

</header>

<nav

th:replace=*"~{fragments/Navigation :: topBar}"* >

</nav>

<div class=*"w3-container w3-center w3-padding-large"*>

<h2 th:text=*"'Questions' + (${param.byMe}? ' added by me' : '')"*>Question Stats</h2>

</div>

<div align=*"center"*>

<p th:text=*"${message}"*></p>

</div>

<div align=*"center"*>

<div class=*"w3-container w3-border w3-border-white w3-hover-border-green w3-section "* align=*"justify"* style="width:*70%*;">

<div class=*"w3-bar"*>

<h5 class=*"w3-bar-item"*> <b th:text=*"${question.questionText}"*>Question text</b></h5>

</div>

<ul class=*"w3-ul"*>

<li class=*"w3-border-bottom"*

th:each=*"answer,iterStat : ${question.answers}"*>

<span th:text=*"${iterStat.count +'. ' + answer + ' '}"*>Answer</span>

<span class=*"fa fa-check w3-large w3-green"*

th:if=*"${answer EQ question.correctAnswer}"*>

</span>

</li>

</ul>

</div>

<p th:text=*" 'The question was created by ' + ${question.createdBy}"* > The question was created by: username </p>

<p th:text=*"${((answeredPercentage >= 0.0)? #numbers.formatPercent(answeredPercentage,1,1) + ' answered this question and ' + #numbers.formatPercent(1.0 - answeredPercentage,1,1) + ' skipped the questions.'*

*: 'None of the user encountered this question in the quiz yet.') }"* > </p>

<div th:unless=*"${#aggregates.sum(answersDistribution) == 0.0}"*>

<p> Of those who answered: </p>

<p th:each=*"a,iteration : ${answersDistribution}"*

th:text=*"${iteration.count + '. answer has been chosen by ' + #numbers.formatPercent(a,1,1) + ' of participants.'}"*>

has been chosen by</p>

</div>

</div>

<footer

th:replace=*"~{fragments/HeaderAndFooter :: footer}"*>

</footer>

</body>

</html>

**showQ uestion.html**

<!DOCTYPE html>

<html xmlns:th=*"http://www.thymeleaf.org"*

xmlns:sec=*"http://www.thymeleaf.org/thymeleaf-extras-springsecurity5"*>

<head>

<meta charset=*"UTF-8"*>

<link rel=*"stylesheet"* href=*"https://www.w3schools.com/w3css/4/w3.css"*>

<title> Spring Quiz - Questions</title>

</head>

<body>

<header

th:replace=*"~{fragments/HeaderAndFooter :: header}"*>

</header>

<nav th:replace=*"~{fragments/Navigation :: topBar}"*></nav>

<div class=*"w3-container w3-padding w3-center"*>

<h4 align=*"center"*

th:text=*"${'Question ' + (qIndex + 1) + '/' + numberOfQuestions}"*>

Question</h4>

<div align=*"center"*>

<div class=*"w3-card"* style="width: *80%*;" align=*"center"*>

<form class=*"w3-container w3-center w3-light-grey"*

th:action=*"@{/quiz/show}"* method=*"post"*>

<input type=*"hidden"* name=*"qId"* th:value=*"${question.id}"* /> <input

type=*"hidden"* name=*"q"* th:value=*"${qIndex}"* />

<p align=*"left"*>

<b th:text=*"${'&emsp;' + question.questionText}"*>Question Text</b>

</p>

<p align=*"left"* th:each=*"answer,loop : ${question.answers}"*>

<input class=*"w3-radio"* type=*"radio"* th:name=*"selectedAnswer"*

th:value=*"${answer}"* th:id=*"${'ans' + loop.index}"*> <label

th:for=*"${'ans' + loop.index}"* th:text=*"${answer}"*>

Answer </label> </p>

<p><input class=*"w3-btn w3-block w3-green w3-m"* type=*"submit"*

value=*"Submit answer"* /> </p>

</form>

<form class=*" w3-center w3-padding w3-light-grey"*

th:action=*"@{/quiz/skip}"* method=*"post"*>

<input type=*"hidden"* name=*"qId"* th:value=*"${question.id}"* /> <input

type=*"hidden"* name=*"q"* th:value=*"${qIndex}"* /> <input

class=*"w3-btn w3-block w3-grey w3-m w3-small"* type=*"submit"*

value=*"Skip this question"* />

</form>

</div>

</div>

</div>

<div align=*"center"*>

<p th:text=*"${message}"*></p>

</div>

<footer

th:replace=*"~{fragments/HeaderAndFooter :: footer}"*>

</footer>

</body>

</html>

**Completion.html**

<!DOCTYPE html>

<html xmlns:th=*"http://www.thymeleaf.org"*

xmlns:sec=*"http://www.thymeleaf.org/thymeleaf-extras-springsecurity5"*>

<head>

<meta charset=*"UTF-8"*>

<link rel=*"stylesheet"* href=*"https://www.w3schools.com/w3css/4/w3.css"*>

<title> Spring Quiz - Questions</title>

</head>

<body>

<header

th:replace=*"~{fragments/HeaderAndFooter :: header}"*>

</header>

<nav

th:replace=*"~{fragments/Navigation :: topBar}"* >

</nav>

<h2 align=*"center"*> You completed the Quiz </h2>

<p align=*"center"* th:text=*"${'Your score is: ' + score}"*> Your score is: /score/ </p>

<footer

th:replace=*"~{fragments/HeaderAndFooter :: footer}"*>

</footer>

</body>

</html>

**Error.html**

<!DOCTYPE html>

<html xmlns:th=*"http://www.thymeleaf.org"*>

<head>

<meta charset=*"UTF-8"*>

<link rel=*"stylesheet"* href=*"https://www.w3schools.com/w3css/4/w3.css"*>

<title> Spring Quiz - Error </title>

</head>

<body>

<header

th:replace=*"~{fragments/HeaderAndFooter :: header}"*>

</header>

<h1>:-(</h1>

<h2>Something went wrong and we couldn't complete your request.</h2>

<h5 th:text=*"${errorMessage}"*></h5>

<a href=*"/"*>Go back to home page</a>

<footer

th:replace=*"~{fragments/HeaderAndFooter :: footer}"*>

</footer>

</body>

</html>