Quiz Portal----->

pom.xml ------>

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-parent</artifactId>
                <version>2.7.3</version>
                <relativePath /> <!-- lookup parent from repository -->
        </parent>
        <groupId>com.example</groupId>
        <artifactId>OnlineQuiz</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <name>OnlineQuiz</name>
        <description>Demo project for Spring Boot</description>
        <properties>
                <java.version>17</java.version>
        </properties>
        <dependencies>
                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-data-jpa</artifactId>
                </dependency>
                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-web</artifactId>
                </dependency>
                <dependency>
                        <groupId>org.apache.tomcat.embed</groupId>
                        <artifactId>tomcat-embed-jasper</artifactId>
                        <version>9.0.44</version>
                        <scope>provided</scope>
                </dependency>
                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-tomcat</artifactId>
                        <version>2.4.4</version>
                        <scope>provided</scope>
                </dependency>
                <dependency>
                        <groupId>javax.servlet</groupId>
```

```
                        <artifactId>jstl</artifactId>
                        <version>1.2</version>
                </dependency>
                <dependency>
                        <groupId>taglibs</groupId>
                        <artifactId>standard</artifactId>
                        <version>1.1.2</version>
                </dependency>


                <dependency>
                        <groupId>mysql</groupId>
                        <artifactId>mysql-connector-java</artifactId>
                        <scope>runtime</scope>
                </dependency>
                <dependency>
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-test</artifactId>
                        <scope>test</scope>
                </dependency>
        </dependencies>

        <build>
                <plugins>
                        <plugin>
                                <groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-maven-plugin</artifactId>
                        </plugin>
                </plugins>
        </build>

</project>



properties---->


server.port=8083
spring.datasource.url=jdbc:mysql://localhost:3306/quizportal
spring.datasource.username=root
spring.datasource.password=Veda@1202189
spring.jpa.properties.hibernate.show_sql=true
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.hibernate.ddl-auto=update
spring.mvc.view.prefix=/WEB-INF/views/
spring.mvc.view.suffix=.jsp
logging.level.org.hibernate=INFO
logging.level.org.hibernate.SQL=INFO
logging.level.org.hibernate.type.descriptor.sql.BasicBinder=TRACE
```

```
logging.level.org.springframework=INFO
#logging.level.org.apache=ERROR
#logging.level.org.springframework.web=TRACE
```

1.Entities---->

1.1.Admin----.

```
package com.example.quiz.entity;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name="admin")
public class Admin {

	@Id
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	private int id;
	private String name;
	private String email;
	private String password;
	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public String getEmail() {
		return email;
	}
	public void setEmail(String email) {
		this.email = email;
	}
	public String getPassword() {
		return password;
	}
```

```
	public void setPassword(String password) {
		this.password = password;
	}
	@Override
	public String toString() {
		return "admin [id=" + id + ", name=" + name + ", email=" + email +
", password=" + password + "]";
	}
```

1.2.Question----->

```
package com.example.quiz.entity;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

import org.springframework.stereotype.Component;

@Component
@Entity
@Table(name = "question")
public class Question {

	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	private int quesId;
	private String question;
	private String optionA;
	private String optionB;
	private String optionC;
	private int ans;
	private int chose;

	public int getQuesId() {
		return quesId;
	}

	public void setQuesId(int quesId) {
		this.quesId = quesId;
	}

	public String getQuestion() {
		return question;
	}
```

```
	public void setQuestion(String question) {
		this.question = question;
	}

	public String getOptionA() {
		return optionA;
	}

	public void setOptionA(String optionA) {
		this.optionA = optionA;
	}

	public String getOptionB() {
		return optionB;
	}

	public void setOptionB(String optionB) {
		this.optionB = optionB;
	}

	public String getOptionC() {
		return optionC;
	}

	public void setOptionC(String optionC) {
		this.optionC = optionC;
	}

	public int getAns() {
		return ans;
	}

	public void setAns(int ans) {
		this.ans = ans;
	}

	public int getChose() {
		return chose;
	}

	public void setChose(int chose) {
		this.chose = chose;
	}

	@Override
	public String toString() {
		return "Question [quesId=" + quesId + ", question=" + question + ",
optionA=" + optionA + ", optionB=" + optionB
				+ ", optionC=" + optionC + ", ans=" + ans + ",
chose=" + chose + "]";
```

```
	}


}
```

1.3.QuestionForm----->

```
package com.example.quiz.entity;

import java.util.List;

import org.springframework.stereotype.Component;

@Component
public class QuestionForm {

	private List<Question> questions;

	public List<Question> getQuestions() {
		return questions;
	}

	public void setQuestions(List<Question> questions) {
		this.questions = questions;
	}
}
```

1.4.QuizPartcipant------>

```
package com.example.quiz.entity;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name="quizparticipants")
public class QuizParticipant {

	@Id
	@GeneratedValue(strategy=GenerationType.IDENTITY)
	private int id;
	private String name;
	public int getId() {
```

```
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	@Override
	public String toString() {
		return "QuizParticipant [id=" + id + ", name=" + name + "]";
	}


}



1.5.Result --->

package com.example.quiz.entity;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

import org.springframework.stereotype.Component;

@Component
@Entity
@Table(name = "results")
public class Result {

	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	private int id;
	private String username;
	private int totalCorrect = 0;

	public Result() {
		super();
	}

	public Result(int id, String username, int totalCorrect) {
		super();
```

```
                this.id = id;
                this.username = username;
                this.totalCorrect = totalCorrect;
        }

        public int getId() {
                return id;
        }

        public void setId(int id) {
                this.id = id;
        }

        public String getUsername() {
                return username;
        }

        public void setUsername(String username) {
                this.username = username;
        }

        public int getTotalCorrect() {
                return totalCorrect;
        }

        public void setTotalCorrect(int totalCorrect) {
                this.totalCorrect = totalCorrect;
        }

}
```

1.6.User------>

```
package com.example.quiz.entity;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table(name="user")
public class User {

        @Id
        @GeneratedValue(strategy=GenerationType.IDENTITY)
```

```
	private int id;
	private String name;
	private String email;
	private String password;
	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public String getEmail() {
		return email;
	}
	public void setEmail(String email) {
		this.email = email;
	}
	public String getPassword() {
		return password;
	}
	public void setPassword(String password) {
		this.password = password;
	}
	@Override
	public String toString() {
		return "User [id=" + id + ", name=" + name + ", email=" + email +
", password=" + password + "]";
	}




}
```

2.Reosotories------>

2.1.AuthUserRepo-------->

```
package com.example.quiz.repositories;
```

```
import java.util.Optional;

import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;

import com.example.quiz.entity.Admin;
import com.example.quiz.entity.User;

@Repository
public interface AuthenticationUserRepository extends CrudRepository<User,
Integer>{

        public Optional<User> findUserByName(String name);

        public Admin save(Admin admin);

}
```

2.2.AuthAdminRepo---->

```
package com.example.quiz.repositories;

import java.util.Optional;

import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;

import com.example.quiz.entity.Admin;

@Repository
public interface AuthenticationAdminRepository extends CrudRepository<Admin,
Integer>{
        public Optional<Admin> findAdminByName(String name);

}
```

2.3.PartcipantRepo----->

```
package com.example.quiz.repositories;

import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;

import com.example.quiz.entity.QuizParticipant;

@Repository
```

```
public interface ParticipantRepository extends CrudRepository<QuizParticipant,
Integer>{

}
```

2.4.QuestionRepo---->

```
package com.example.quiz.repositories;

import java.util.Optional;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.example.quiz.entity.Question;

@Repository
public interface QuestionRepo extends JpaRepository<Question, Integer> {

        Optional<Question> findByQuesId(int quesId);

}
```

2.5.ResultsRepo---->

```
package com.example.quiz.repositories;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.example.quiz.entity.Result;

@Repository
public interface ResultRepo extends JpaRepository<Result, Integer> {

}
```

3.Exceptions---->

```
package com.example.quiz.exceptions;
```

```
public class AdminNotFoundException extends RuntimeException {

	public AdminNotFoundException() {
		super("Invalid admin name or password");
	}

}
```

```
package com.example.quiz.exceptions;

public class UserNotFoundException extends RuntimeException {
	public UserNotFoundException() {
		super("Invalid username or password");
	}
}
```

4.Services---->

4.1.AuthAdminServices----.

```
package com.example.quiz.services;

import java.util.Optional;

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

import com.example.quiz.entity.Admin;
import com.example.quiz.exceptions.AdminNotFoundException;
import com.example.quiz.repositories.AuthenticationAdminRepository;

@Service
public class AuthenticationAdminService {

	@Autowired
	AuthenticationAdminRepository adminAuthRepo;


	public Admin GetAdminByName(String name) {
		Optional<Admin> adminfound = adminAuthRepo.findAdminByName(name);
		if(adminfound.isPresent()) {
			return adminfound.get();
		}
		else {
```

```
                        throw new AdminNotFoundException();
                }
        }



        public Boolean isValidPassword(String cmp, String actual) {
                return ((cmp.equals(actual)) ?  true :  false);
        }

}
```

4.2.AuthUserServices------>

```
package com.example.quiz.services;

import java.util.Optional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.example.quiz.entity.User;
import com.example.quiz.exceptions.UserNotFoundException;
import com.example.quiz.repositories.AuthenticationUserRepository;

@Service
public class AuthenticationUserService {
        @Autowired
        AuthenticationUserRepository authRepo;

        public User GetUserByName(String name) {
                Optional<User> userfound = authRepo.findUserByName(name);
                if (userfound.isPresent()) {
                        return userfound.get();
                } else {
                        throw new UserNotFoundException();

                }
        }
        public Boolean isValidPassword(String cmp, String actual) {
                return ((cmp.equals(actual)) ? true : false);
        }

}
```

4.3.QuizServices----->

```
package com.example.quiz.services;

import java.util.ArrayList;
import java.util.List;
import java.util.Random;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Sort;
import org.springframework.stereotype.Service;

import com.example.quiz.entity.Question;
import com.example.quiz.entity.QuestionForm;
import com.example.quiz.entity.Result;
import com.example.quiz.repositories.ResultRepo;
import com.example.quiz.repositories.QuestionRepo;

@Service
public class QuizService {

	@Autowired
	Question question;
	@Autowired
	QuestionForm qForm;
	@Autowired
	QuestionRepo qRepo;
	@Autowired
	Result result;
	@Autowired
	ResultRepo rRepo;

	public QuestionForm getQuestions() {
		List<Question> allQues = qRepo.findAll();
		List<Question> qList = new ArrayList<Question>();

		Random random = new Random();

		for(int i=0; i<5; i++) {
			int rand = random.nextInt(allQues.size());
			qList.add(allQues.get(rand));
			allQues.remove(rand);
		}

		qForm.setQuestions(qList);

		return qForm;
	}
```

```
	public int getResult(QuestionForm qForm) {
		int correct = 0;

		for(Question q: qForm.getQuestions())
			if(q.getAns() == q.getChose())
				correct++;

		return correct;
	}

	public void saveScore(Result result) {
		Result saveResult = new Result();
		saveResult.setUsername(result.getUsername());
		saveResult.setTotalCorrect(result.getTotalCorrect());
		rRepo.save(saveResult);
	}

	public List<Result> getTopScore() {
		List<Result> sList = rRepo.findAll(Sort.by(Sort.Direction.DESC,
"totalCorrect"));

		return sList;
	}
}
```

5.Controller------>

5.1.Main Controller------>

```
package com.example.quiz.controllers;

import java.util.List;
import java.util.Optional;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.ui.Model;
import org.springframework.ui.ModelMap;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
```

```
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

import com.example.quiz.entity.Admin;
import com.example.quiz.entity.Question;
import com.example.quiz.entity.QuestionForm;
import com.example.quiz.entity.Result;
import com.example.quiz.entity.User;
import com.example.quiz.repositories.AuthenticationAdminRepository;
import com.example.quiz.repositories.AuthenticationUserRepository;
import com.example.quiz.repositories.ParticipantRepository;
import com.example.quiz.repositories.QuestionRepo;
import com.example.quiz.services.AuthenticationAdminService;
import com.example.quiz.services.AuthenticationUserService;
import com.example.quiz.services.QuizService;

@RestController
public class MainController {
        Logger logger = LoggerFactory.getLogger(MainController.class);

        @Autowired
        AuthenticationUserService userAuthService;

        @Autowired
        AuthenticationAdminService adminAuthService;

        @Autowired
        AuthenticationUserRepository userAuthRepo;

        @Autowired
        AuthenticationAdminRepository adminAuthRepo;

        @Autowired
        ParticipantRepository participantRepo;

        @Autowired
        QuestionRepo quizRepo;

        @Autowired
        Result result;

        @Autowired
        QuizService qService;

        Boolean submitted = false;

    @GetMapping("/home")
```

```
public ModelAndView home() {
    ModelAndView mv = new ModelAndView();
    mv.setViewName("home");
    return mv;
}

    @GetMapping("/admin")
    public ModelAndView admin() {
            ModelAndView mv = new ModelAndView();
    mv.setViewName("admin");
            return mv;
    }

    @GetMapping("/participant")
    public ModelAndView participant() {
            ModelAndView mv = new ModelAndView();
    mv.setViewName("participant");
            return mv;
    }

    @RequestMapping("/loginUser")
    public ModelAndView loginParticipant() {
            ModelAndView mv = new ModelAndView();
    mv.setViewName("loginUser");
            return mv;
    }

    @RequestMapping("/loginAdmin")
    public ModelAndView loginAdmin() {
            ModelAndView mv = new ModelAndView();
    mv.setViewName("loginAdmin");
            return mv;
    }

    @RequestMapping("/createQuiz")
    public ModelAndView createQuiz() {
            ModelAndView mv = new ModelAndView();
    mv.setViewName("createquiz");
            return mv;
    }

    @RequestMapping("/access")
    public ModelAndView accessPage() {
            ModelAndView mv = new ModelAndView();
    mv.setViewName("access");
            return mv;
    }

    @RequestMapping("/updateQuiz")
    public ModelAndView updateQuiz() {
```

```
        ModelAndView mv = new ModelAndView();
mv.setViewName("updatequiz");
        return mv;
}

@RequestMapping("/deleteQuiz")
public ModelAndView deleteQuiz() {
        ModelAndView mv = new ModelAndView();
mv.setViewName("deletequiz");
        return mv;
}

@RequestMapping("/update")
public ModelAndView updateAdminPassword() {
        ModelAndView mv = new ModelAndView();
mv.setViewName("update");
        return mv;
}

@RequestMapping("/showParticipants")
public ModelAndView showUsers() {
        ModelAndView mv = new ModelAndView();
mv.setViewName("participantList");
        return mv;
}


@RequestMapping("/viewParticipantForm")
public ModelAndView createQuizParticipant() {
        ModelAndView mv = new ModelAndView();
mv.setViewName("quizparticipant");
        return mv;
}

@RequestMapping("/quiz")
public ModelAndView quiz() {
        ModelAndView mv = new ModelAndView();
mv.setViewName("quiz");
        return mv;
}

@RequestMapping("/score")
public ModelAndView Score() {
        ModelAndView mv = new ModelAndView();
mv.setViewName("scoreboard");
        return mv;
}

@RequestMapping("/submit")
public ModelAndView submit() {
```

```
                ModelAndView mv = new ModelAndView();
        mv.setViewName("result");
                return mv;
        }

        @RequestMapping("/validParticipant")
        public ModelAndView validParticipant() {
                ModelAndView mv = new ModelAndView();
        mv.setViewName("validParticipant");
                return mv;
        }

        @RequestMapping(value="/viewParticipantForm", method=RequestMethod.POST)
        public ModelAndView authenticateUser(@RequestParam("name") String name,
@RequestParam("password") String pswd) {
                ModelAndView userLoginSuccess = new ModelAndView();
                userLoginSuccess.setViewName("userLoginSuccess");

                ModelAndView userNotFound = new ModelAndView();
                userNotFound.setViewName("userNotFound");
                User user = userAuthService.GetUserByName(name);
                if(userAuthService.isValidPassword(pswd, user.getPassword()))
                        {
                                return userLoginSuccess;
                        }
                else {
                        return userNotFound;
                }

        }
        @RequestMapping(value="/viewAccess", method=RequestMethod.POST)
        public ModelAndView authenticateAdmin(@RequestParam("name") String name,
@RequestParam("password") String pswd) {
                ModelAndView access = new ModelAndView();
                access.setViewName("access");

                ModelAndView adminNotFound = new ModelAndView();
                adminNotFound.setViewName("adminNotFound");
                Admin admin = adminAuthService.GetAdminByName(name);
                logger.info(admin.getName() + " attempted to login with " +
admin.getPassword());
                if(adminAuthService.isValidPassword(pswd, admin.getPassword())) {
                        return access;
                }
                else {
                        return adminNotFound;
                }

        }
```

```
        @RequestMapping(value="/createUser", method = RequestMethod.POST)
        public ModelAndView createParticipantAcc(@ModelAttribute("user") User user,
ModelMap model) {
                ModelAndView loginUser = new ModelAndView();
                loginUser.setViewName("loginUser");

                userAuthRepo.save(user);
                return loginUser;
        }

        @RequestMapping(value="/createAdmin", method = RequestMethod.POST)
        public ModelAndView createAdminAcc(@ModelAttribute("admin") Admin admin,
ModelMap model) {
                ModelAndView loginAdmin = new ModelAndView();
                loginAdmin.setViewName("loginAdmin");
                adminAuthRepo.save(admin);
                return loginAdmin;
        }


        @RequestMapping(value="/update", method = RequestMethod.POST)
        public ModelAndView updateAdmin(@ModelAttribute("admin") Admin uiadmin,
ModelMap model) {
                ModelAndView userNotFound = new ModelAndView();
                userNotFound.setViewName("userNotFound");

                ModelAndView updateSuccess = new ModelAndView();
                updateSuccess.setViewName("updateSuccess");

                Optional<Admin> optProduct =
adminAuthRepo.findById(uiadmin.getId());
                if (optProduct.isEmpty()) {
                        return userNotFound;
                }
                Admin dbAdmin = optProduct.get();
                dbAdmin.setName(uiadmin.getName());
                dbAdmin.setPassword(uiadmin.getPassword());
                adminAuthRepo.save(dbAdmin);
                model.addAttribute("users", adminAuthRepo.findAll());
                return updateSuccess;
        }

        @RequestMapping(value="/createQuiz", method=RequestMethod.POST)
        public ModelAndView createNewQuiz(@ModelAttribute("question") Question
question, ModelMap model) {
                ModelAndView quizCreateSuccess = new ModelAndView();
                quizCreateSuccess.setViewName("quizCreateSuccess");

                quizRepo.save(question);
                return quizCreateSuccess;
```

```
	}

	@RequestMapping(value="/updateQuiz", method=RequestMethod.POST)
	public ModelAndView updateQuiz(@ModelAttribute("question") Question
question, ModelMap model) {
		ModelAndView quizNotFound = new ModelAndView();
		quizNotFound.setViewName("quizNotFound");

		ModelAndView quizUpdateSuccess = new ModelAndView();
		quizUpdateSuccess.setViewName("quizUpdateSuccess");

		Optional<Question> optQuestion =
quizRepo.findById(question.getQuesId());
		if(optQuestion.isEmpty()) {
			return quizNotFound;
		}
		Question dbQuiz = optQuestion.get();
		dbQuiz.setOptionA(question.getOptionA());
		dbQuiz.setOptionB(question.getOptionB());
		dbQuiz.setOptionC(question.getOptionC());
		dbQuiz.setAns(question.getAns());
		dbQuiz.setChose(question.getChose());
		quizRepo.save(dbQuiz);
		model.addAttribute("quizzes", quizRepo.findAll());
		return quizUpdateSuccess;
	}

	@RequestMapping(value="/deleteQuiz", method=RequestMethod.POST)
	public ModelAndView deleteQuiz(@ModelAttribute("question") Question
question, ModelMap model) {
		ModelAndView quizNotFound = new ModelAndView();
		quizNotFound.setViewName("quizNotFound");

		ModelAndView quizDeleteSuccess = new ModelAndView();
		quizDeleteSuccess.setViewName("quizDeleteSuccess");

		Optional<Question> optQuestion =
quizRepo.findById(question.getQuesId());
		if(optQuestion.isEmpty()) {
			return quizNotFound;
		}
		Question dbQuiz = optQuestion.get();
		dbQuiz.setQuesId(question.getQuesId());
		quizRepo.delete(dbQuiz);
		return quizDeleteSuccess;
	}



	@RequestMapping(value="/createQuizParticipant", method=RequestMethod.POST)
```

```
        public ModelAndView quiz(@RequestParam String name, Model m,
RedirectAttributes ra, @ModelAttribute("question") Question question) {
                ModelAndView quizpage = new ModelAndView();
                quizpage.setViewName("quiz");

                ModelAndView validParticipant = new ModelAndView();
                validParticipant.setViewName("validParticipant");

                if(name.isBlank()) {
                        ra.addFlashAttribute("warning", "You must enter your
name");
                        return validParticipant;
                }

                submitted = false;
                result.setUsername(name);

                QuestionForm qForm = qService.getQuestions();
                m.addAttribute("qForm", qForm);

                return quizpage;
        }

          @PostMapping("/submit")
            public ModelAndView submit(@ModelAttribute QuestionForm qForm, Model m)
{
                ModelAndView mv = new ModelAndView();
                mv.setViewName("result");
                if(!submitted) {
                    result.setTotalCorrect(qService.getResult(qForm));
                    qService.saveScore(result);
                    submitted = true;
                }

                return mv;
            }

        @GetMapping("/score")
        public ModelAndView score(Model m) {
                ModelAndView mv = new ModelAndView();
                mv.setViewName("scoreboard");
                List<Result> sList = qService.getTopScore();
                m.addAttribute("sList", sList);

                return mv;
        }

}
```

Home.Jsp--->

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Online Quiz</title>
</head>
<body>
        <h1 align="center">Welcome to Online Quiz Portal</h1>
        <ul>
            <li><a href="participant">Participant</a></li><br>
            <li><a href="admin">Admin </a></li>
        </ul>
    </form>
</body>
</html>
```