NAGELLA PRAJITHRAJ

Day 5: Spring MVC - Administration Portal for Transit Management

Task 1: Utilize Spring MVC to Create an Admin Portal for Transit Officials to Manage Routes and Schedules

Set Up Spring MVC Configuration:
Create a Spring MVC configuration class to enable MVC and set up view resolvers.
java
Copy code

```java
@Configuration
@EnableWebMvc
@ComponentScan(basePackages = "com.example.transitapp")
public class WebConfig implements WebMvcConfigurer {

    @Bean
    public InternalResourceViewResolver viewResolver() {
        InternalResourceViewResolver resolver = new
InternalResourceViewResolver();
        resolver.setPrefix("/WEB-INF/views/");
        resolver.setSuffix(".html");
        return resolver;
    }
}
```

1.

Create a Controller for Admin Operations:
Implement a controller to handle admin actions for managing routes and schedules.
java
Copy code

```java
@Controller
@RequestMapping("/admin")
public class AdminController {

    @Autowired
    private RouteService routeService;

    @GetMapping("/dashboard")
    public String dashboard(Model model) {
        model.addAttribute("routes", routeService.getAllRoutes());
        return "admin/dashboard";
    }

    @GetMapping("/route/add")
    public String addRouteForm(Model model) {
        model.addAttribute("route", new Route());
        return "admin/addRoute";
    }

    @PostMapping("/route/add")
    public String addRoute(@ModelAttribute Route route) {
        routeService.addRoute(route);
        return "redirect:/admin/dashboard";
    }

    // Additional methods for updating and deleting routes
}
```

2.

Implement the RouteService:
Create a service class to manage routes.
java
Copy code

```
@Service
public class RouteService {

    private List<Route> routes = new ArrayList<>();

    public List<Route> getAllRoutes() {
        return routes;
    }

    public void addRoute(Route route) {
        routes.add(route);
    }

    // Additional methods for updating and deleting routes
}
```

3.

Define the Route Model:
Create a simple model class for routes.
java
Copy code
```
public class Route {
    private String name;
    private String startLocation;
    private String endLocation;

    // Getters and setters
}
```

4.

Task 2: Integrate Thymeleaf with Spring MVC for Real-Time Updates and Schedule Changes
Add Thymeleaf Dependencies:
Include Thymeleaf in your `pom.xml`.
xml
Copy code
```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

1.

Configure Thymeleaf View Resolver:
Configure Thymeleaf as the view resolver.
java
Copy code
```
@Configuration
public class ThymeleafConfig {

    @Bean
    public SpringTemplateEngine templateEngine() {
        SpringTemplateEngine templateEngine = new
SpringTemplateEngine();
        templateEngine.setTemplateResolver(templateResolver());
        templateEngine.setEnableSpringELCompiler(true);
        return templateEngine;
    }

    @Bean
    public SpringResourceTemplateResolver templateResolver() {
        SpringResourceTemplateResolver templateResolver = new
SpringResourceTemplateResolver();
        templateResolver.setPrefix("classpath:/templates/");
        templateResolver.setSuffix(".html");
        templateResolver.setTemplateMode(TemplateMode.HTML);
        templateResolver.setCacheable(true);
        return templateResolver;
    }
}
```

2.

Create Thymeleaf Templates:
Develop Thymeleaf templates for the admin portal.
`dashboard.html`
html
Copy code

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Admin Dashboard</title>
</head>
<body>
    <h1>Admin Dashboard</h1>
    <a href="/admin/route/add">Add Route</a>
    <table>
        <tr>
            <th>Name</th>
            <th>Start Location</th>
            <th>End Location</th>
        </tr>
        <tr th:each="route : ${routes}">
            <td th:text="${route.name}"></td>
            <td th:text="${route.startLocation}"></td>
            <td th:text="${route.endLocation}"></td>
        </tr>
    </table>
</body>
</html>
addRoute.html
```
html
Copy code
```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Add Route</title>
</head>
<body>
    <h1>Add Route</h1>
    <form th:action="@{/admin/route/add}" th:object="${route}"
method="post">
        <label>Name: <input type="text"
th:field="*{name}"/></label><br/>
        <label>Start Location: <input type="text"
th:field="*{startLocation}"/></label><br/>
        <label>End Location: <input type="text"
th:field="*{endLocation}"/></label><br/>
        <button type="submit">Add Route</button>
    </form>
</body>
</html>
```

3.

Task 3: Develop Form Handling in Spring MVC for Incident Reporting and User Feedback
Create Models for Incident Reports and Feedback:
Define the models for incident reports and user feedback.
java
Copy code
```
public class IncidentReport {
    private String description;
    private String reporter;
    private Date date;

    // Getters and setters
}

public class UserFeedback {
    private String username;
    private String feedback;
    private Date date;

    // Getters and setters
}
```

1.

Add Form Handlers in the Controller:
Implement form handling methods in the controller.
java
Copy code

```
@Controller
@RequestMapping("/admin")
public class AdminController {

    // ... previous methods

    @GetMapping("/incident/report")
    public String reportIncidentForm(Model model) {
        model.addAttribute("incidentReport", new IncidentReport());
        return "admin/reportIncident";
    }

    @PostMapping("/incident/report")
    public String reportIncident(@ModelAttribute IncidentReport
incidentReport) {
        // Save incident report
        return "redirect:/admin/dashboard";
    }

    @GetMapping("/feedback")
    public String feedbackForm(Model model) {
        model.addAttribute("userFeedback", new UserFeedback());
        return "admin/feedback";
    }

    @PostMapping("/feedback")
    public String submitFeedback(@ModelAttribute UserFeedback
userFeedback) {
        // Save user feedback
        return "redirect:/admin/dashboard";
    }
}
```

2.

Create Thymeleaf Templates for Forms:
Develop Thymeleaf templates for the incident report and feedback forms.
`reportIncident.html`
html
Copy code

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Report Incident</title>
</head>
<body>
    <h1>Report Incident</h1>
    <form th:action="@{/admin/incident/report}"
th:object="${incidentReport}" method="post">
        <label>Description: <input type="text"
th:field="*{description}"/></label><br/>
        <label>Reporter: <input type="text"
th:field="*{reporter}"/></label><br/>
        <label>Date: <input type="date"
th:field="*{date}"/></label><br/>
        <button type="submit">Submit Report</button>
    </form>
</body>
</html>
```

feedback.html

html

Copy code

```html
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>User Feedback</title>
</head>
<body>
    <h1>User Feedback</h1>
    <form th:action="@{/admin/feedback}" th:object="${userFeedback}"
method="post">
        <label>Username: <input type="text"
th:field="*{username}"/></label><br/>
        <label>Feedback: <textarea
th:field="*{feedback}"></textarea></label><br/>
        <label>Date: <input type="date"
th:field="*{date}"/></label><br/>
        <button type="submit">Submit Feedback</button>
    </form>
</body>
</html>
```

3.

This setup provides a robust foundation for an admin portal using Spring MVC and Thymeleaf, enabling transit officials to manage routes and schedules, handle incident reports, and gather user feedback. Adjust the code and configurations as needed to fit your specific application requirements.

4o