package com.example.demo.config;

import com.example.demo.dao.WebNoteDAO;

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

import org.springframework.context.ApplicationContext;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.ComponentScan;

import org.springframework.context.annotation.Configuration;

import org.springframework.web.servlet.config.annotation.EnableWebMvc;

import org.springframework.web.servlet.config.annotation.ViewResolverRegistry;

import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

import org.thymeleaf.spring6.SpringTemplateEngine;

import org.thymeleaf.spring6.templateresolver.SpringResourceTemplateResolver;

import org.thymeleaf.spring6.view.ThymeleafViewResolver;

@Configuration

@ComponentScan("com.example.demo") // Сканирование пакетов для компонентов

@EnableWebMvc // Включение поддержки веб MVC

public class NotesConfig implements WebMvcConfigurer {

private final ApplicationContext applicationContext; // Контекст приложения

@Autowired

public NotesConfig(ApplicationContext applicationContext) {

this.applicationContext = applicationContext; // Инициализация контекста приложения

}

@Bean

WebNoteDAO webNoteDAO() {

return new WebNoteDAO(); // Создание объекта WebNoteDAO для взаимодействия с данными

}

}

package com.example.demo.config;

import jakarta.servlet.ServletContext;

import jakarta.servlet.ServletException;

import org.springframework.web.filter.HiddenHttpMethodFilter;

import org.springframework.web.servlet.support.AbstractAnnotationConfigDispatcherServletInitializer;

public class NotesMvcDispatcherServletInitializer extends AbstractAnnotationConfigDispatcherServletInitializer {

// Возвращает конфигурационные классы для корневого контекста

@Override

protected Class<?>[] getRootConfigClasses() {

return null;

}

// Возвращает конфигурационные классы для сервлета

@Override

protected Class<?>[] getServletConfigClasses() {

return new Class[] { NotesConfig.class };

}

// Возвращает URL-мэппинг для сервлета

@Override

protected String[] getServletMappings() {

return new String[] { "/" };

}

// Выполняет настройку сервлета при запуске

public void onStartup(ServletContext aServletContext) throws ServletException {

super.onStartup(aServletContext);

registerHiddenFieldFilter(aServletContext);

}

// Регистрирует фильтр для скрытых HTTP-методов

private void registerHiddenFieldFilter(ServletContext aContext) {

aContext.addFilter("hiddenHttpMethodFilter", new HiddenHttpMethodFilter()).addMappingForUrlPatterns(null, true,

"/\*");

}

}

package com.example.demo.controllers;

import com.example.demo.dao.WebNoteDAO;

import com.example.demo.models.WebNote;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.\*;

@Controller

public class NotesController {

private final WebNoteDAO webNoteDAO;

public NotesController(WebNoteDAO webNoteDAO) {

this.webNoteDAO = webNoteDAO;

}

// Отображение всех заметок на главной странице

@GetMapping("/")

public String index(Model model) {

model.addAttribute("notes", webNoteDAO.index());

return "index";

}

// Отображение конкретной заметки по идентификатору

@GetMapping("/{id}")

public String show(@PathVariable("id") int id, Model model) {

model.addAttribute("note", webNoteDAO.show(id));

return "show";

}

// Создание новой заметки

@PostMapping("/{note}")

public String create(@RequestParam("note") String note, Model model) {

WebNote webNote = new WebNote();

webNote.setNote(note);

return "success Page";

}

// Отображение формы для создания новой заметки

@GetMapping("/new")

public String newNote(Model model) {

model.addAttribute("webNote", new WebNote());

return "new";

}

// Сохранение новой заметки

@PostMapping("/create")

public String create(@ModelAttribute("webNote") WebNote webNote) {

webNoteDAO.save(webNote);

return "redirect:/";

}

// Отображение формы для редактирования заметки

@GetMapping("/{id}/edit")

public String edit(Model model, @PathVariable("id") int id) {

model.addAttribute("webNote", webNoteDAO.show(id));

return "edit";

}

// Обновление существующей заметки

@GetMapping("/{id}/update")

public String update(@ModelAttribute("webNote") WebNote webNote, @PathVariable("id") int id) {

webNoteDAO.update(id, webNote.getNote());

return "redirect:/";

}

// Удаление существующей заметки

@GetMapping("/delete/{id}")

public String delete(@PathVariable("id") int id) {

webNoteDAO.delete(id);

return "redirect:/";

}

}

package com.example.demo.dao;

import com.example.demo.models.WebNote;

import java.util.ArrayList;

import java.util.List;

public class WebNoteDAO {

private static int NOTES\_COUNT;

private final List<WebNote> notes;

public WebNoteDAO() {

// Конструктор класса, инициализирующий список заметок и добавляющий пять заметок по умолчанию

notes = new ArrayList<>();

notes.add(new WebNote(++NOTES\_COUNT, "note1"));

notes.add(new WebNote(++NOTES\_COUNT, "note2"));

notes.add(new WebNote(++NOTES\_COUNT, "note3"));

notes.add(new WebNote(++NOTES\_COUNT, "note4"));

notes.add(new WebNote(++NOTES\_COUNT, "note5"));

}

public List<WebNote> index() {

// Метод возвращает список всех заметок

return notes;

}

public WebNote show(int id) {

// Метод позволяет найти заметку по заданному ID

return notes.stream().filter(n -> n.getId() == id).findAny().orElse(null);

}

public void save(WebNote webNote) {

// Метод для добавления новой заметки в список

notes.add(webNote);

}

public void update(int id, String note) {

// Метод для обновления текста заметки по её ID

WebNote toUpdateNote = show(id);

toUpdateNote.setNote(note);

}

public void delete(int id) {

// Метод удаляет заметку по её ID

notes.removeIf(n -> n.getId() == id);

}

}

<!DOCTYPE html>

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

<head>

<meta charset="UTF-8">

<title>Редактирование заметки</title>

</head>

<body>

<form th:method="POST" th:action="@{'/update/' + ${webNote.getId()}}" th:object="${webNote}">

<!-- Форма для обновления заметки, отправляющая данные методом POST -->

<label for="note">Введите заметку:</label>

<input type="text" th:field="\*{note}" id="note"/>

<!-- Поле ввода текста для обновления заметки -->

<input type="submit" value="Обновить заметку"/>

<!-- Кнопка отправки формы для обновления заметки -->

</form>

</body>

</html>

<!DOCTYPE html>

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

<head>

<meta charset = "UTF-8">

<title>Список заметок</title>

</head>

<body>

<div th:each = "note : ${notes}"> <!-- Цикл для перебора заметок из списка -->

<a th:href="@{/{id} (id=${note.getId()})" <!-- Создание ссылки с идентификатором заметки -->

th:text="${note.getNote()}">содержание</a> <!-- Отображение текста заметки внутри ссылки -->

</div>

<br/>

<hr/>

<a href="/new">Новая заметка</a> <!-- Ссылка для создания новой заметки -->

</body>

</html>

<!DOCTYPE html>

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

<head>

<meta charset="UTF-8">

<title>Создание новой заметки</title>

</head>

<body>

<form method = "POST" th:action="@{/create}" th:object="${webNote}">

<!-- Форма для создания новой заметки, отправляет данные методом POST на /create с объектом webNote -->

<label for="id"> Введите ID: </label>

<input type = "number" th:field="\*{id}" id="id"/>

<!-- Поле для ввода ID, связанное с полем id объекта webNote -->

<br>

<br>

<label for="note"> Введите заметку: </label>

<input type = "text" th:field="\*{note}" id="note"/>

<!-- Поле для ввода заметки, связанное с полем note объекта webNote -->

<input type="submit" value="Добавить заметку"/>

<!-- Кнопка для отправки формы и добавления новой заметки -->

</form>

</body>

</html>

<!DOCTYPE html>

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

<head>

<meta charset="UTF-8">

<title>Заметка</title>

</head>

<body>

<p th:text="${note.getNote()}">Содержание</p> <!-- Отображает содержание заметки -->

<p th:text="${note.getId()}">Идентификатор</p> <!-- Отображает идентификатор заметки -->

<a th:href="@{/{id}/редактировать(id=${note.getId()})}">Изменить</a> <!-- Создает ссылку для редактирования заметки -->

<form th:method="GET" th:action="@{/delete/{id} (id=${note.getId()})}">

<input type="submit" value="Удалить"/> <!-- Форма для удаления заметки -->

</form>

</body>

</html>