


SPRAWOZDANIE NR 6			
Nazwa ćwiczenia	Aplikacja webowa		 POLITECHNIKA BYDGOSKA Wydział Telekomunikacji, Informatyki i Elektrotechniki
Przedmiot	Zaawansowane programowanie obiektowe		
Student Grupa Nr indeksu	Paweł Jońca gr 7 122348		
Data ćwiczeń		01.07	Data oddania sprawozdania

Spis treści

Treść zadania	1
Rozwiązanie problemu	1
Kod do NoteController	1
kod Importance	2
Kod do Note.....	3
Kod do NoteRepo	5
kod do NoteService	6
Kod do NoteServiceImp.....	6
Kod do NoteAppApplication.....	7
Wygląd aplikacji w działaniu.....	8
Wnioski.....	9

Treść zadania

Napisać aplikację webową za pomocą framework'u Spring Boot do wyświetlania/dodawania notatek z wybranym poziomem ważności (np. URGENT, STANDARD, OPTIONAL). Formularz do dodawania notatek powinien znajdować się na górze strony, a tuż pod nim lista dodanych już notatek (zgodnie z rysunkiem poniżej). Wartości w polu Importance powinny być ograniczone do wybranego zbioru wartości za pomocą typu Enum

Rozwiązanie problemu

Kod do NoteController

```
package lab6.note_app.controller;

import lab6.note_app.model.Importance;
import lab6.note_app.model.Note;
import lab6.note_app.service.NoteService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
```

```

import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;

/**
 * Kontroler obsługujący żądania związane z notatkami.
 * Zarządza wyświetlaniem notatek, dodawaniem nowych notatek i interakcją z
 * warstwą serwisu.
 */
@Controller
public class NoteController {

    private final NoteService noteService;

    /**
     * Konstruktor do wstrzykiwania zależności NoteService.
     * Spring automatycznie wstrzyknie instancję NoteServiceImpl.
     */
    @Autowired
    public NoteController(NoteService noteService) {
        this.noteService = noteService;
    }

    /**
     * Obsługuje żądania GET dla ścieżki "/list".
     * Wyświetla formularz do dodawania notatek oraz listę istniejących
     * notatek.
     */
    @GetMapping("/list")
    public String showNotesForm(Model model) {
        model.addAttribute("note", new Note()); // Pusty obiekt Note dla
        formularza
        model.addAttribute("notes", noteService.listOfNotes()); // Lista
        wszystkich notatek z serwisu
        model.addAttribute("importanceLevels", Importance.values()); //
        Poziomy ważności dla listy rozwijanej
        return "notes"; // Nazwa szablonu Thymeleaf (notes.html)
    }

    /**
     * Obsługuje żądania POST dla ścieżki "/add" w celu dodawania nowych
     * notatek.
     */
    @PostMapping("/add")
    public String addNote(@ModelAttribute("note") Note note) {
        noteService.addNote(note); // Użycie serwisu do dodania notatki
        return "redirect:/list"; // Przekierowanie z powrotem na stronę
        listy
    }
}

```

kod Importance

```

package lab6.note_app.model;

public enum Importance {

```

```

    URGENT,
    STANDARD,
    OPTIONAL
}

```

Kod do Note

```

package lab6.note_app.model;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.EnumType;
import jakarta.persistence.Enumerated;
import jakarta.persistence.Column;
import jakarta.persistence.PrePersist;
import java.time.LocalDateTime;

/**
 * Reprezentuje pojedynczy wpis notatki w aplikacji.
 * Klasa ta jest Encją, mapowaną na tabelę w bazie danych.
 */
@Entity
public class Note {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id; // Unikalny identyfikator notatki

    private String title; // Tytuł notatki
    private String content; // Treść notatki

    @Enumerated(EnumType.STRING) // Przechowuje nazwę enuma (np. "URGENT")
    jako String w bazie danych
    private Importance importance; // Poziom ważności notatki (URGENT,
    STANDARD, OPTIONAL)

    @Column(name = "timestamp") // Jawne zdefiniowanie nazwy kolumny
    private LocalDateTime timestamp; // Czas dodania notatki

    /**
     * Domyślny konstruktor wymagany przez JPA.
     */
    public Note() {
        // Konstruktor bezargumentowy
    }

    /**
     * Konstruktor do tworzenia nowej instancji Notatki ze wszystkimi
     * właściwościami.
     * Czas dodania jest zazwyczaj ustawiany automatycznie przy utrwalaniu.
     */
    public Note(Long id, String title, String content, Importance
    importance) {

```

```

        this.id = id;
        this.title = title;
        this.content = content;
        this.importance = importance;
        // timestamp zostanie ustawiony przez @PrePersist
    }

    /**
     * Ustawia znacznik czasu automatycznie przed utrwaleniem encji.
     */
    @PrePersist
    protected void onCreate() {
        this.timestamp = LocalDateTime.now();
    }

    // --- Metody Getter ---

    /**
     * Pobiera ID notatki.
     */
    public Long getId() {
        return id;
    }

    /**
     * Pobiera tytuł notatki.
     */
    public String getTitle() {
        return title;
    }

    /**
     * Pobiera treść notatki.
     */
    public String getContent() {
        return content;
    }

    /**
     * Pobiera poziom ważności notatki.
     */
    public Importance getImportance() {
        return importance;
    }

    /**
     * Pobiera znacznik czasu notatki.
     */
    public LocalDateTime getTimestamp() {
        return timestamp;
    }

    // --- Metody Setter ---

    /**
     * Ustawia ID notatki.
     */
    public void setId(Long id) {
        this.id = id;
    }

```

```

/**
 * Ustawia tytuł notatki.
 */
public void setTitle(String title) {
    this.title = title;
}

/**
 * Ustawia treść notatki.
 */
public void setContent(String content) {
    this.content = content;
}

/**
 * Ustawia poziom ważności notatki.
 */
public void setImportance(Importance importance) {
    this.importance = importance;
}

/**
 * Ustawia znacznik czasu notatki.
 */
public void setTimestamp(LocalDateTime timestamp) {
    this.timestamp = timestamp;
}

/**
 * Nadpisuje metodę toString dla lepszego logowania/debugowania.
 */
@Override
public String toString() {
    return "Note{" +
        "id=" + id +
        ", title='" + title + '\'' +
        ", content='" + content + '\'' +
        ", importance=" + importance +
        ", timestamp=" + timestamp +
        '\'';
}
}

```

Kod do NoteRepo

```

package lab6.note_app.repository;

import lab6.note_app.model.Note;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public interface NoteRepo extends JpaRepository<Note, Long> {
    /**
     * Pobiera wszystkie notatki posortowane malejąco według znacznika
     czasu.
     * Sygnatura tej metody jest automatycznie implementowana przez Spring

```

```
Data JPA.
    */
    List<Note> findByOrderByTimestampDesc();
}
```

kod do NoteService

```
package lab6.note_app.service;

import lab6.note_app.model.Note;
import java.util.List;

/**
 * Interfejs dla warstwy usług Notatek.
 * Definiuje operacje logiki biznesowej dla notatek.
 */
public interface NoteService {
    /**
     * Pobiera listę wszystkich notatek.
     */
    List<Note> listOfNotes();

    /**
     * Dodaje nową notatkę do systemu.
     */
    void addNote(Note note);
}
```

Kod do NoteServiceImpl

```
package lab6.note_app.service;

import lab6.note_app.model.Note;
import lab6.note_app.repository.NoteRepo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

/**
 * Implementacja interfejsu NoteService.
 * Zawiera logikę biznesową do zarządzania notatkami.
 * Komunikuje się z NoteRepo w celu interakcji z bazą danych.
 */
@Service
public class NoteServiceImpl implements NoteService {

    private final NoteRepo noteRepo;

    /**
     * Konstruktor do wstrzykiwania zależności NoteRepo.
     * Spring automatycznie wstrzyknie instancję NoteRepo.
     */
    @Autowired
    public NoteServiceImpl(NoteRepo noteRepo) {
        this.noteRepo = noteRepo;
    }

    /**
     * Pobiera listę notatek, posortowanych malejąco według znacznika

```

```

czasu.
    * Wykorzystuje niestandardową metodę zdefiniowaną w NoteRepo.
    */
    @Override
    public List<Note> listOfNotes() {
        return noteRepo.findByOrderByTimestampDesc();
    }

    /**
     * Dodaje nową notatkę do bazy danych.
     */
    @Override
    public void addNote(Note note) {
        noteRepo.save(note);
    }
}

```

Kod do NoteAppApplication

```

package lab6.note_app;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.autoconfigure.domain.EntityScan;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.data.jpa.repository.config.EnableJpaRepositories;

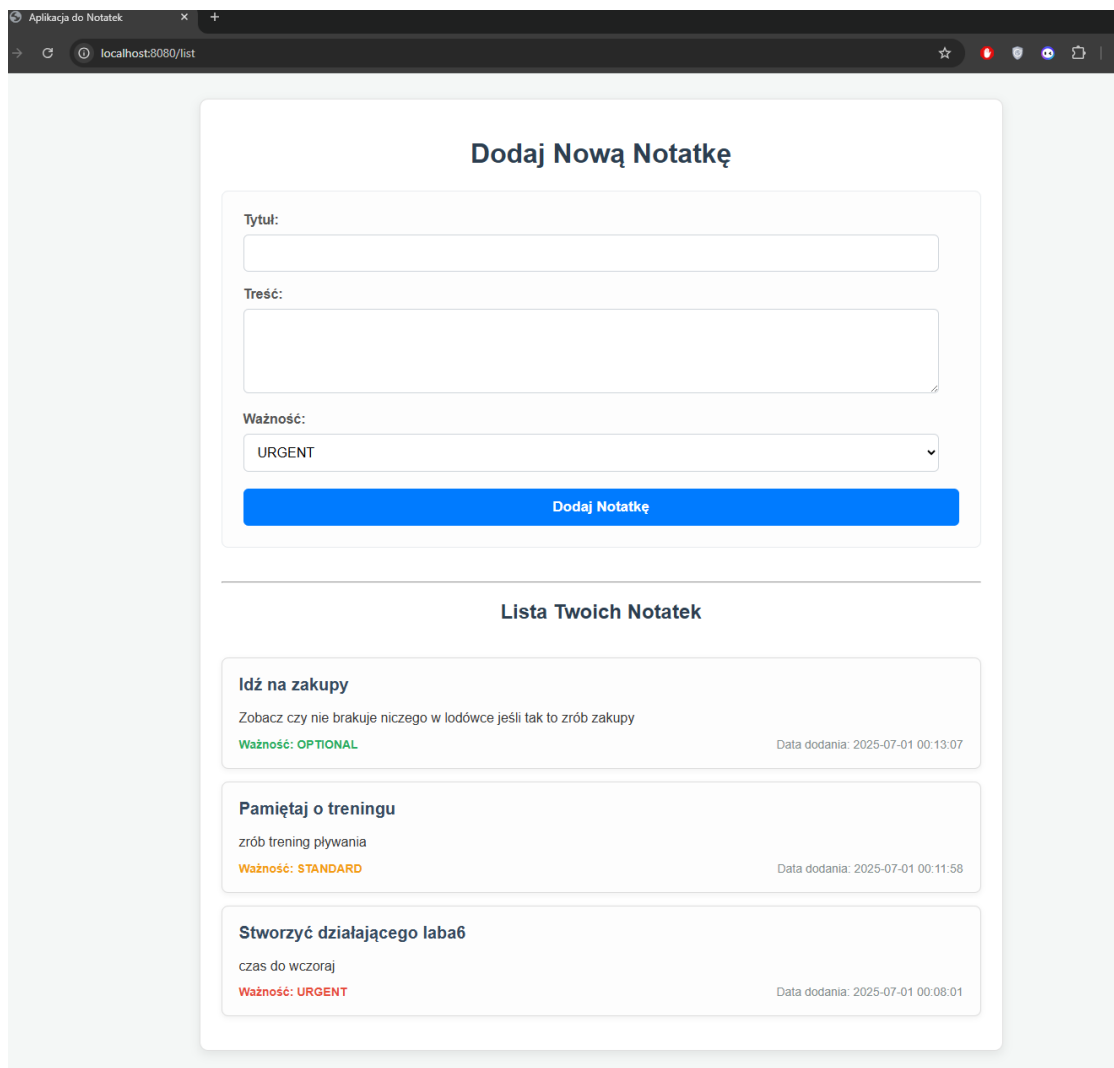
/**
 * Główna klasa aplikacji Spring Boot.
 * Konfiguruje skanowanie komponentów, repozytoria JPA i skanowanie encji.
 * Zapewnia odnalezienie wszystkich komponentów w pakiecie bazowym
 * 'lab6.note_app' i jego podpakietach.
 */
@SpringBootApplication
// Jawnie skanuje pakiet model w poszukiwaniu encji JPA
@EntityScan("lab6.note_app.model")
// Jawnie włącza repozytoria JPA w pakiecie repository
@EnableJpaRepositories("lab6.note_app.repository")
public class NoteAppApplication {

    public static void main(String[] args) {
        SpringApplication.run(NoteAppApplication.class, args);
    }
}

```

Wygląd aplikacji w działaniu

Tak się prezentuje aplikacja webowa



The screenshot shows a web browser window with the address bar displaying 'localhost:3080/list'. The application has a dark theme. The main content area is divided into two sections. The top section, titled 'Dodaj Nową Notatkę', contains a form with three input fields: 'Tytuł:', 'Treść:', and 'Ważność:'. The 'Ważność:' field is a dropdown menu currently set to 'URGENT'. Below the form is a blue button labeled 'Dodaj Notatkę'. The bottom section, titled 'Lista Twoich Notatek', displays a list of three notes. Each note card shows a title, a description, a priority level (e.g., 'OPTIONAL', 'STANDARD', 'URGENT'), and a timestamp.

Dodaj Nową Notatkę

Tytuł:

Treść:

Ważność:

URGENT

Dodaj Notatkę

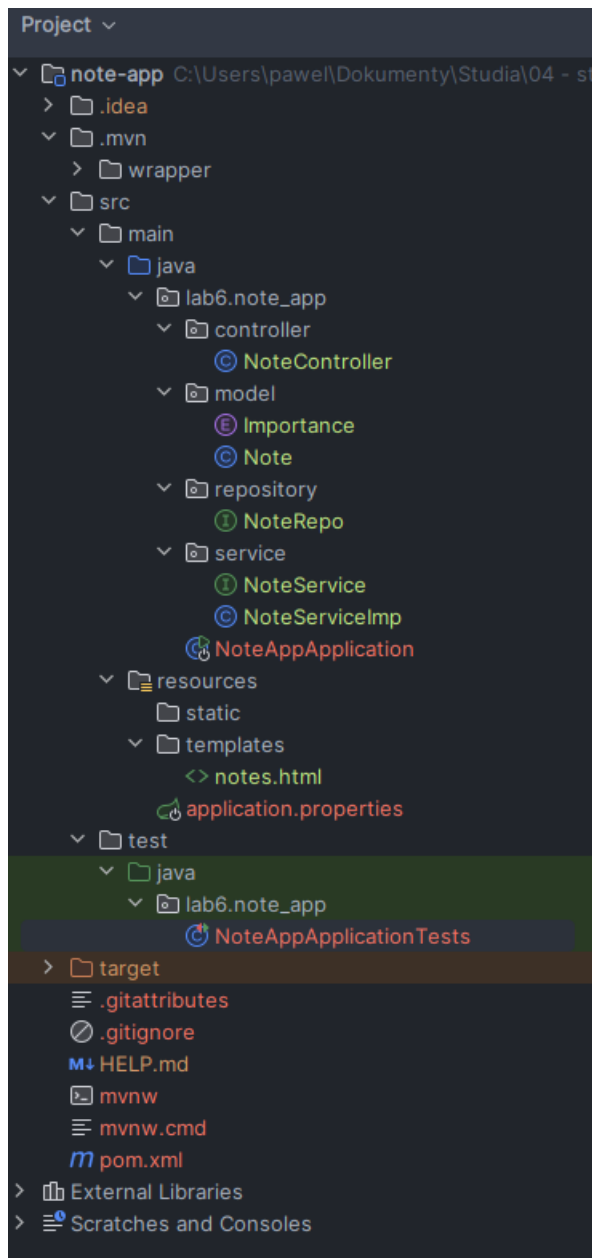
Lista Twoich Notatek

Idź na zakupy
Zobacz czy nie brakuje niczego w lodówce jeśli tak to zrób zakupy
Ważność: **OPTIONAL**
Data dodania: 2025-07-01 00:13:07

Pamiętaj o treningu
zrób trening pływania
Ważność: **STANDARD**
Data dodania: 2025-07-01 00:11:58

Stworzyć działającego labaa6
czas do wczoraj
Ważność: **URGENT**
Data dodania: 2025-07-01 00:08:01

Wygląd struktury projektu



Wnioski

Zadanie pozwoliło mi zrozumieć, jak zbudować pełną aplikację webową w Spring Boot do zarządzania notatkami, z formularzem i listą, używając enumów do ważności. Nauczyłem się, jak ważne są poprawne pakiety, gettery i settery oraz jak Thymeleaf łączy się z klasami Java, zwłaszcza z enumami. Pomimo kilku błędów, które udało się rozwiązać, aplikacja działa, co pokazuje, jak wszystkie warstwy Spring Boot współpracują ze sobą.