Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №2
із дисципліни «*Розроблення веб-застосунків за допомогою Spring Framework*»
Тема: «Створення та зв’язування компонентів»
Варіант-11

Виконали:
Студенти групи ІА-24
Коханчук Михайло Миколайович
Котярчук Максим Сергійович

Перевірив:
Нікітін Валерій Андрійович

Київ-2024

Мета: У межах даної лабораторної роботи передбачено додавання датчика світла до віртуального пристрою IoT.

Тема:

11	**Тема:** Сайт подачі петицій **Сутності:** Петиція, голоси за петицію **Актори:** Користувач **Сценарії використання:** Користувач: Створення/Видалення/Перегляд Петиції, Генерація URL для переходу на голосування за петицію, голосування за петицію.

Завдання:

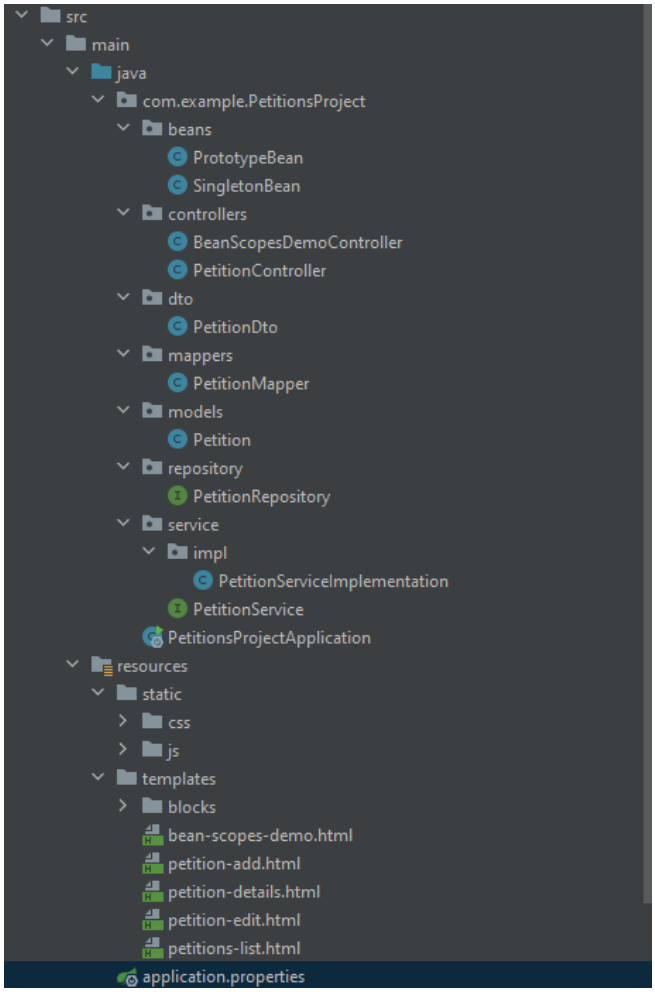
1. Отримайте у викладача номер варіанта для своєї бригади.
2. Створіть проект, який буде включати наступні модулі:
   - Spring Web;
   - Thymeleaf.
3. Компоненти рівня доступу до даних (@Repository) реалізуйте у вигляді заглушки (Stub або Fake). Повноцінна реалізація даного рівня передбачається у наступних лабораторних.
4. Рівень представлення реалізуйте у вигляді контролера (@Controller) та шаблонів (Thymeleaf) з мінімальною функціональністю, достатньою для демонстрації компонентів бізнес-логіки. Повноцінна реалізація даного рівня передбачається у наступних лабораторних.
5. На рівні сервісів (@Service) реалізуйте компоненти бізнес-логіки. При цьому в лабораторній роботі мають бути продемонстровані наступні можливості Spring Framework:
   - анотація @Component (@Repository, @Service, @Controller);
   - анотація @Bean;
   - створення бінів типу singleton та prototype;
   - ін’єкція залежностей через конструктор, сетер, та напряму у поле біна;

Порядок виконання лабораторної роботи:

Перелік залежностей встановлених при створенні проекту:

```
<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-thymeleaf</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>nz.net.ultraq.thymeleaf</groupId>
        <artifactId>thymeleaf-layout-dialect</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-devtools</artifactId>
        <scope>runtime</scope>
        <optional>true</optional>
    </dependency>
    <dependency>
        <groupId>org.postgresql</groupId>
        <artifactId>postgresql</artifactId>
        <scope>runtime</scope>
    </dependency>
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <optional>true</optional>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-validation</artifactId>
    </dependency>
</dependencies>

<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
            <configuration>
                <excludes>
                    <exclude>
                        <groupId>org.projectlombok</groupId>
                        <artifactId>lombok</artifactId>
                    </exclude>
                </excludes>
            </configuration>
        </plugin>
    </plugins>
</build>
```

Загальна структура проекту:

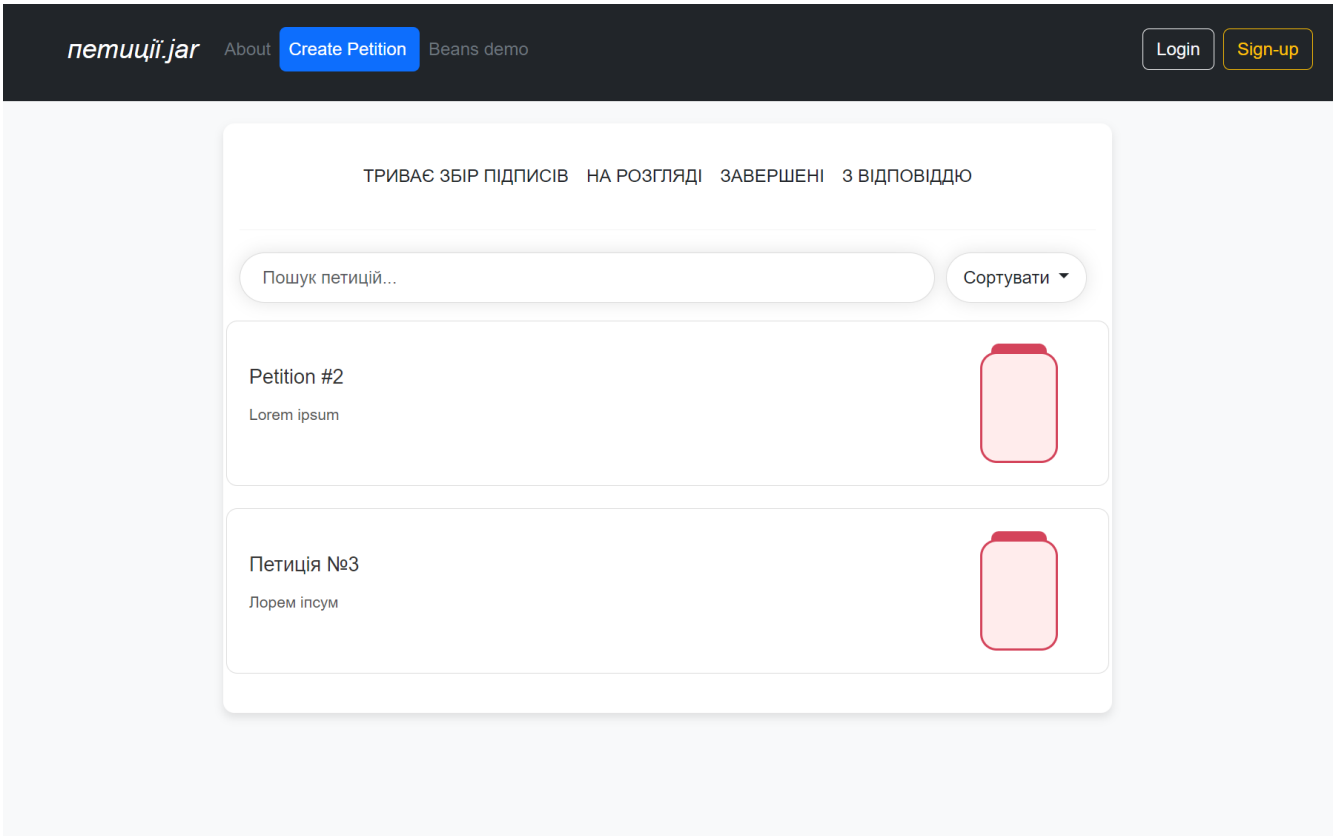


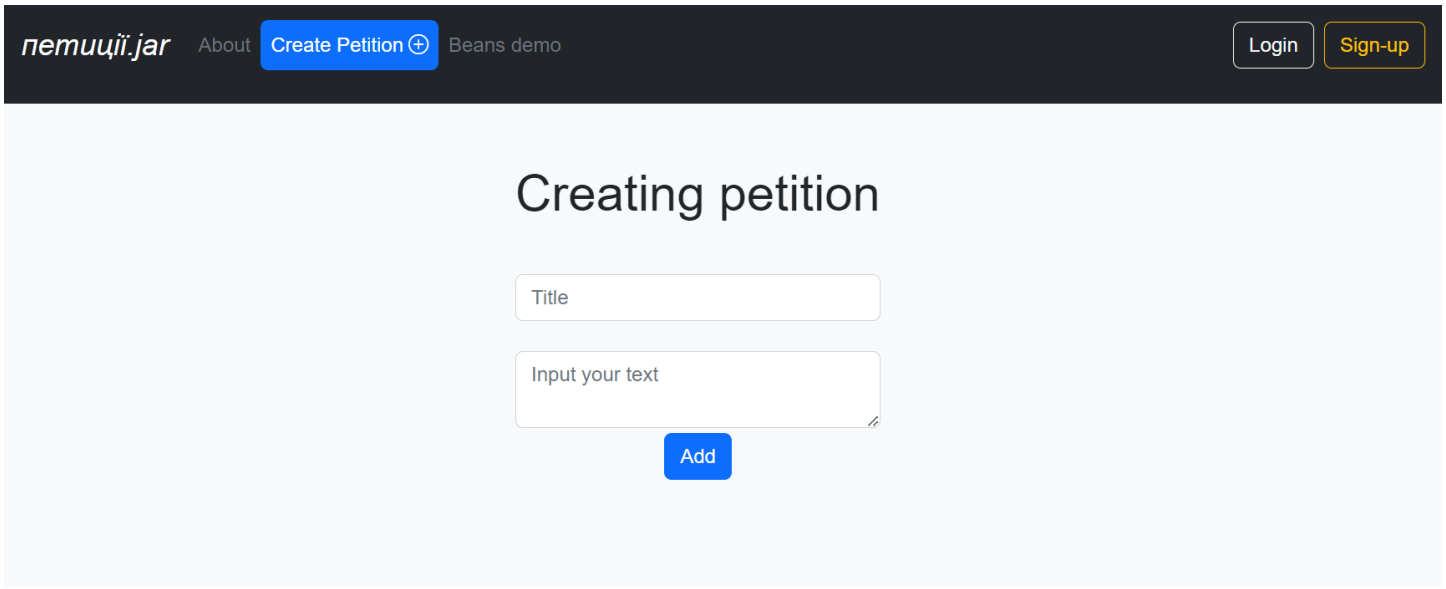
Repository:

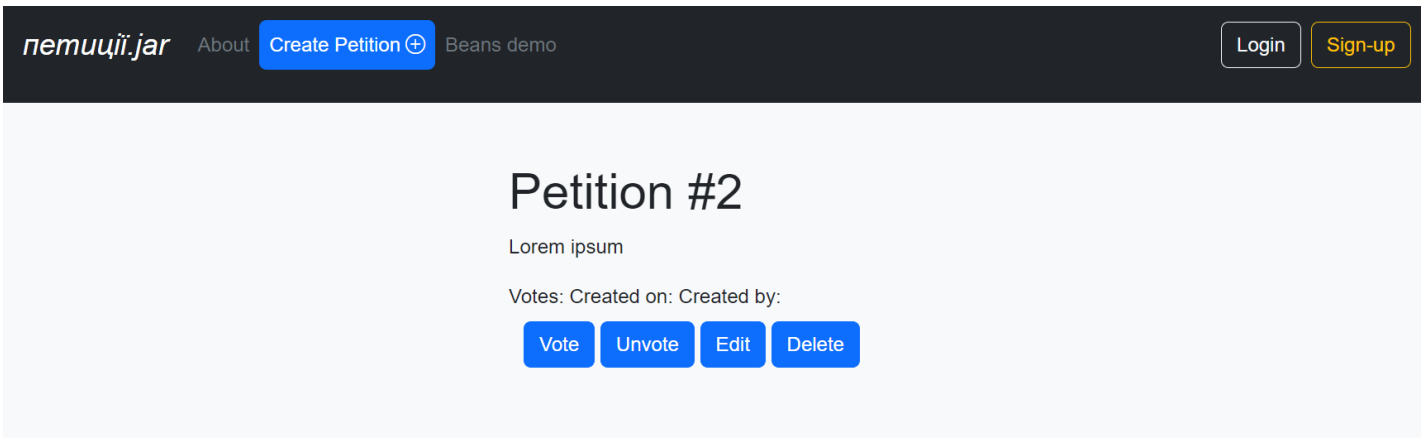
```java
4 usages
public interface PetitionRepository extends JpaRepository<Petition, Long> {
    no usages
    Optional<Petition> findByTitle(String title);
    1 usage
    @Query("SELECT p from Petition p WHERE p.title LIKE CONCAT('%', :query, '%')")
    List<Petition> searchPetitions(String query);

}
```

Рівень представлення:

```java
@Controller
public class PetitionController {
    8 usages
    private PetitionService petitionService;
    // Ін'єкція через конструктор
    no usages
    @Autowired
    public PetitionController(PetitionService petitionService) { this.petitionService = petitionService; }

    no usages
    @GetMapping("/petitions")
    public String listPetitions(Model model){
        List<PetitionDto> petitions = petitionService.findAllPetitions();
        model.addAttribute( attributeName: "petitions", petitions);
        return "petitions-list";
    }

    no usages
    @GetMapping("/petitions/{petitionId}")
    public String getPetition(@PathVariable("petitionId") Long petitionId, Model model) {
        PetitionDto petition = petitionService.findByPetitionId(petitionId);
        model.addAttribute( attributeName: "petition", petition);
        return "petition-details";
    }

    no usages
    @GetMapping("/petitions/add")
    public String addPetition(Model model) {
        Petition petition = new Petition();
        model.addAttribute( attributeName: "petition", petition);
        return "petition-add";
    }
```


петиції.jar
About
Create Petition
Beans demo
Login
Sign-up
ТРИВАЄ ЗБІР ПІДПИСІВ
НА РОЗГЛЯДІ
ЗАВЕРШЕНІ
З ВІДПОВІДДЮ
Пошук петицій...
Сортувати
Petition #2
Lorem ipsum
Петиція №3
Лорем іпсум


петиції.jar
About
Create Petition
Beans demo
Login
Sign-up
Creating petition
Title
Input your text
Add


петиції.jar
About
Create Petition
Beans demo
Login
Sign-up
Petition #2
Lorem ipsum
Votes: Created on: Created by:
Vote
Unvote
Edit
Delete

Рівень сервісів:

```java
6 usages  1 implementation
public interface PetitionService {
    1 usage  1 implementation
    List<PetitionDto> findAllPetitions();
    1 usage  1 implementation
    Petition savePetition(PetitionDto petition);
    2 usages  1 implementation
    PetitionDto findByPetitionId(long petitionId);
    1 usage  1 implementation
    void deletePetition(Long petId);
    1 usage  1 implementation
    void updatePetition(PetitionDto petition);
    1 usage  1 implementation
    List<PetitionDto> searchPetitions(String query);
}
```

```java
1 usage
@Service
public class PetitionServiceImplementation implements PetitionService {
    7 usages
    private PetitionRepository petitionRepository;
    no usages
    @Autowired // Ін'єкція через сеттер
    public void setPetitionRepository(PetitionRepository petitionRepository) {
        this.petitionRepository = petitionRepository;
    }

    1 usage
    @Override
    public List<PetitionDto> findAllPetitions() {
        List<Petition> petitions = petitionRepository.findAll();
        return petitions.stream().map((petition) -> mapToPetitionDto(petition)).collect(Collectors.toList());
    }

    1 usage
    @Override
    public Petition savePetition(PetitionDto petitionDto) {
        Petition petition = mapToPetition(petitionDto);
        return petitionRepository.save(petition);
    }

    2 usages
    @Override
    public PetitionDto findByPetitionId(long petitionId) {
        Petition petition = petitionRepository.findById(petitionId).get();
        return mapToPetitionDto(petition);
    }
```

- анотація @Component (@Repository, @Service, @Controller):

```
@Controller
public class PetitionController {
```

```
@Service
public class PetitionServiceImplementation implements PetitionService {
```

```
    7 usages
@Repository
public interface PetitionRepository extends JpaRepository<Petition, Long> {
```

```
@Component
@Scope(value = ConfigurableBeanFactory.SCOPE_PROTOTYPE)
public class PrototypeBean {
}
```

- анотація @Bean:

```
@Bean
@Scope("prototype")
public PrototypeBean prototypeBean() {
    return new PrototypeBean();
}
```

- створення бінів типу singleton та prototype:

```
@SpringBootApplication
public class PetitionsProjectApplication {
    no usages
    public static void main(String[] args) { SpringApplication.run(PetitionsProjectApplication.class, args); }
    no usages
    @Bean
    @Scope("prototype")
    public PrototypeBean prototypeBean() {
        return new PrototypeBean();
    }
}
```

```
6 usages
public class PrototypeBean {
}
```

```
@Component
@Scope(value = ConfigurableBeanFactory.SCOPE_SINGLETON)
public class SingletonBean {
}
```

Spring Bean Scopes Demonstration

Singleton Scope

The singleton bean will retain the same instance across the entire Spring context.

Bean1 HashCode: **2087696217**

Bean2 HashCode: **2087696217**

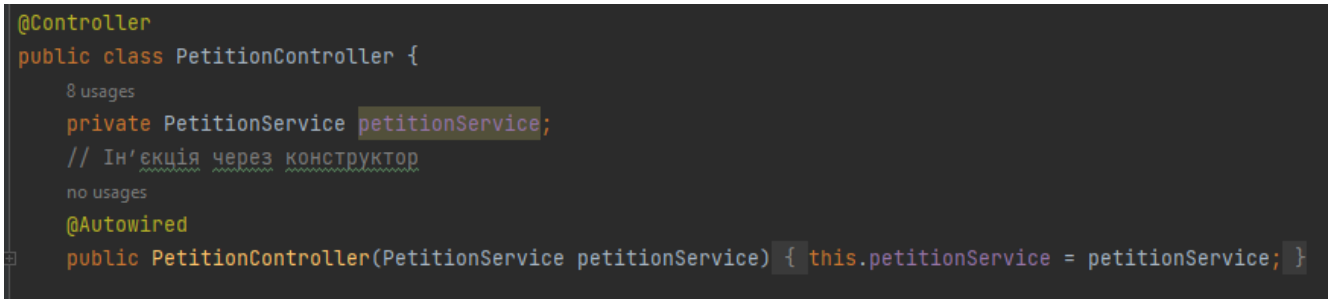
Prototype Scope

Each time a prototype bean is requested, a new instance will be created.

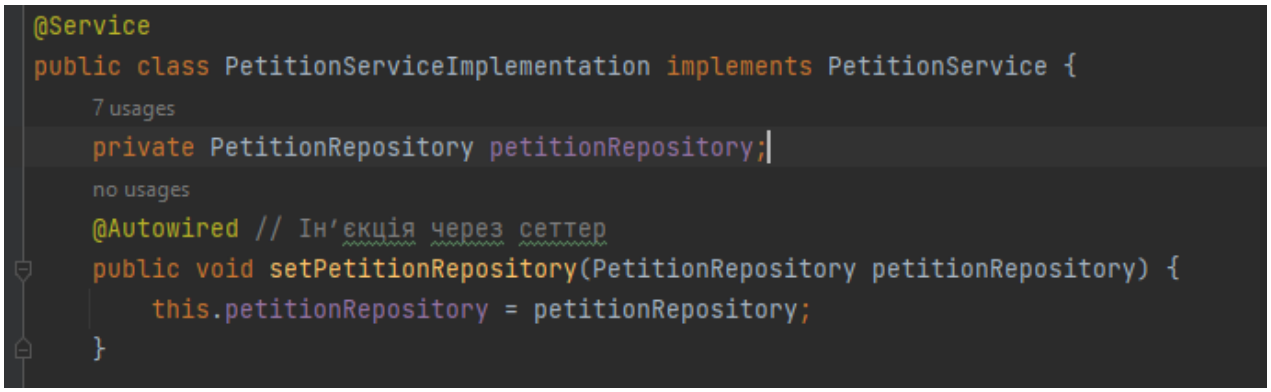
Bean HashCode3: **1470335882**

Bean HashCode4: **1291738134**

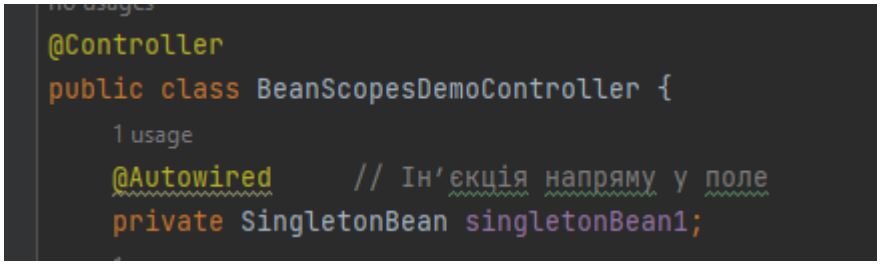
- ін’єкція залежностей через конструктор, сетер, та напряму у поле біна:



```
@Controller
public class PetitionController {
    8 usages
    private PetitionService petitionService;
    // Ін'єкція через конструктор
    no usages
    @Autowired
    public PetitionController(PetitionService petitionService) { this.petitionService = petitionService; }
```



```
@Service
public class PetitionServiceImplementation implements PetitionService {
    7 usages
    private PetitionRepository petitionRepository;
    no usages
    @Autowired // Ін'єкція через сеттер
    public void setPetitionRepository(PetitionRepository petitionRepository) {
        this.petitionRepository = petitionRepository;
    }
```



```
@Controller
public class BeanScopesDemoController {
    1 usage
    @Autowired      // Ін'єкція напряму у поле
    private SingletonBean singletonBean1;
```

GitHub: https://github.com/mykh3398/springLabs/tree/main/lab2

Висновок: в ході даної роботи я дізнався більше про основні анотації Spring, Java Beans; повторив принципи SOLID.