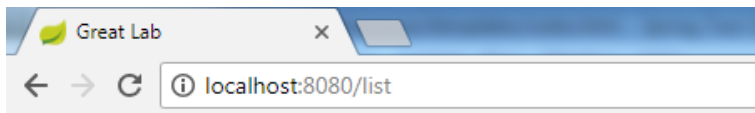


## ĆWICZENIE: Aplikacja webowa

(4 godziny zajęciowe)

1. 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*.



### Add your Note:

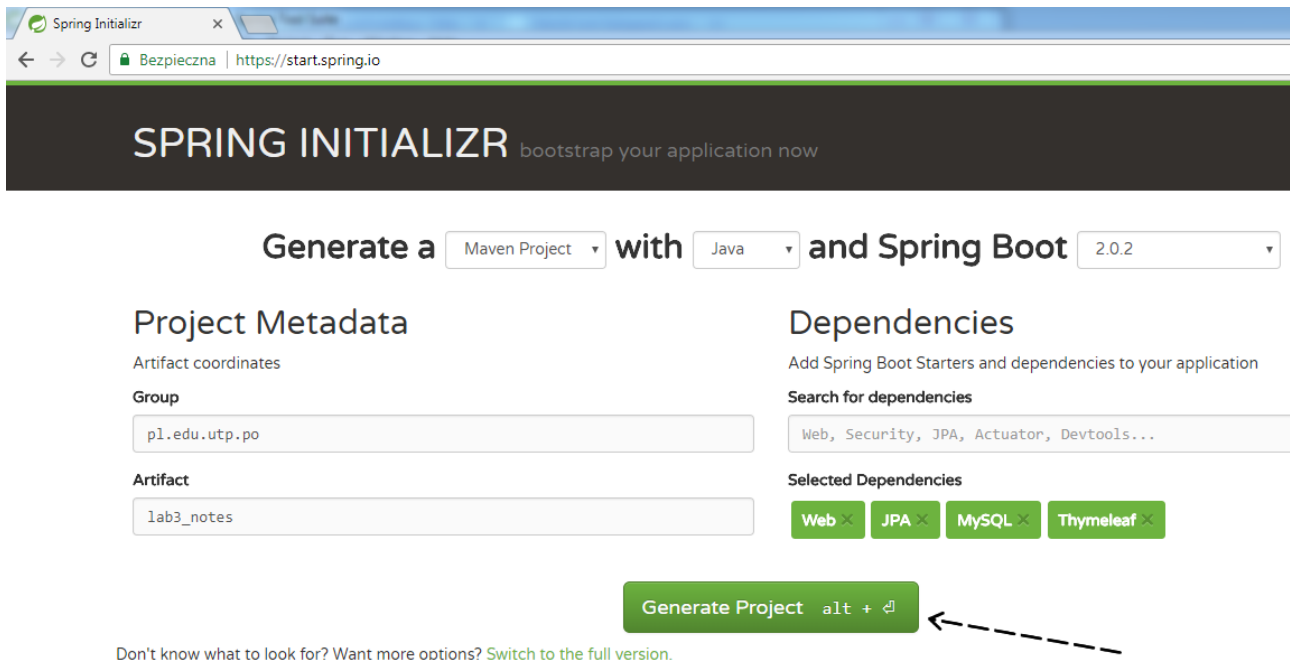
Importance:

Your Note:

### The list of your notes:

Importance	Timestamp	Note
OPTIONAL	2018-05-23 13:42:50.0	Buy tickets tomorrow.
STANDARD	2018-05-23 13:42:29.0	Call my brother.
URGENT	2018-05-23 13:42:10.0	Exam in 8 days!

2. Co będziemy potrzebować... ?



Generate a Maven Project with Java and Spring Boot 2.0.2

**Project Metadata**

Artifact coordinates

Group  
pl.edu.utp.po

Artifact  
lab3\_notes

**Dependencies**

Add Spring Boot Starters and dependencies to your application

Search for dependencies  
Web, Security, JPA, Actuator, Devtools...

Selected Dependencies  
Web × JPA × MySQL × Thymeleaf ×

Generate Project alt + ⌘

Don't know what to look for? Want more options? [Switch to the full version.](#)



Możesz również dodać moduł DevTools, który ułatwi Ci tworzenie aplikacji. Np. za każdym razem, kiedy zmienisz coś w szablonie Thymeleaf, nie będziesz musiał restartować aplikacji, aby zobaczyć zmiany (wystarczy odświeżyć stronę). Należy dodać wewnątrz znaczników `<dependencies> ...</dependencies>` w pliku pom.xml następującą zależność:

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-devtools</artifactId>
  <optional>true</optional>
</dependency>
```

3. W poprzednich ćwiczeniach używałeś pliku persistence.xml, aby skonfigurować Hibernate. W projekcie Spring Boot należy użyć pliku **application.properties**. Przykładowa zawartość tego pliku dla bazy danych o nazwie **baza\_prog\_ob** pliku poniżej.

```
## Spring DATASOURCE (DataSourceAutoConfiguration & DataSourceProperties)
spring.datasource.url = jdbc:mysql://localhost:3306/baza_prog_ob?useSSL=false
spring.datasource.username = root
spring.datasource.password = 1234

## Hibernate Properties
# The SQL dialect makes Hibernate generate better SQL for the chosen database
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL57Dialect

# Hibernate ddl auto (create, create-drop, validate, update)
spring.jpa.hibernate.ddl-auto = update
```

4. Na poniższym rysunku przedstawiono prostą bazę danych **baza\_prog\_ob** z tabelą **notes** i przykładowymi wpisami. Taką właśnie bazę danych należy wykorzystać w ćwiczeniu.

notes	
id	BIGINT(20)
importance	VARCHAR(25)
text	VARCHAR(255)
timestamp	TIMESTAMP

id	importance	text	timestamp
4	URGENT	Exam in 8 days!	2018-05-14 11:43:25
5	STANDARD	Call my brother.	2018-05-14 11:43:31
6	OPTIONAL	Buy tickets tomorrow.	2018-05-14 11:43:47

5. Przykładowa organizacja kodu znajduje się poniżej (patrz rysunek na końcu instrukcji).

- **Lab3JpaAppcilation. Java:** Najważniejsza klasa już gotowa na listingu poniżej, teraz dopisać jeszcze kilka linijek kodu ;)
- **NoteController.java:** Tutaj znajdzie się nasz kotroler sterujący całą aplikacją. Z pewnością będziesz chciał utworzyć w tej klasie dwie metody obsługujące żądania typu GET i POST konieczne do wyświetlania zawartości tabeli oraz do obsługi formularza.
- **Note.java, ImportanceEnum.java:** W warstwie domenowej należy zdefiniować odpowiednio encję wykorzystywaną w aplikacji (zgodnie z pkt 4) oraz Enum'a, który reprezentował będzie możliwe poziomy ważności dodawanych notatek.
- **NoteRepo.java:** Zauważ, że w tym pliku wystarczy zdefiniować tylko interfejs (który rozszerza interfejs `org.springframework.data.jpa.repository.JpaRepository`). Spring boot automatycznie zaimplementuje potrzebne metody. Ponieważ chcielibyśmy wyświetlać notatki w zależności daty ich dodania (pole `TIMESTAMP`, ostatnio dodana notatka powinna pojawiać się na samej górze), musimy podać sygnaturę dodatkowej metody, która pozwoli zwrócić notatki w pożądanej kolejności:

```
List<Note> findByOrderByTimestampDesc();
```

Zauważ, że i tej metody nie musisz implementować, wystarczy w tym przypadku tylko powyższa sygnatura w interfejsie `NoteRepo`!

- **NoteService.java, NoteServiceImpl.java:** Czas na usługę, w której zdefiniujemy dwie metody. Pierwsza z nich pozwoli zwrócić odpowiednio przygotowaną listę notatek, natomiast druga obsłuży dąwanie nowej notatki z formularza do bazy danych.

```
List<Note> listOfNotes(); oraz void addNote(Note n);
```

Zauważ, że dzięki mechanizmowi autowiązania w pliku **NoteServiceImpl.java** użyjesz komponentu repozytorium do współpracy z bazą danych, a w kotrolerze z kolei niniejszą usługę. **W kotrolerze dostęp do naszej usługi możemy uzyskać w poniższy sposób.** Należy zauważyć, że pole jest typu interfejsowego `NoteService`! Komunikacja między warstwą usług i prezentacji (kontrolera) powinna odbywać się właśnie przez interfejsy.

```
@Autowired
private NoteService noteService;
```



Być może zastanawiasz się, czy nie można posłużyć się bezpośrednio repozytorium `NoteRepo`. Oczywiście, można i takie rozwiązanie w literaturze można często spotkać, szczególnie w prostych przykładach. Jednak bardziej eleganckim rozwiązaniem jest użycie usług w kotrolerze niż bezpośrednie odwoływanie się do repozyterium. Chociaż metody `listOfNotes` oraz `addNote` są w naszym przypadku bardzo proste, w praktyce często podobne metody mogą zawierać wiele linijek kodu. Umieszczenie takiego kodu bezpośrednio w kotrolerze nie jest najlepszym pomysłem i z pewnością staje się on wtedy mniej czytelny.

- **notes.html:** Pozostaje jeszcze zdefiniowanie tzw. widoku, którego nazwa będzie zwracana przez daną metodę w kontrolerze. Ponieważ wykorzystujemy popularne szablony Thymeleaf do zdefiniowania widoku zapoznaj się z atrybutami **th:\*** (w szczególności **th:text**, **th:action**, **th:object**, **th:each**, **th:value**, **th:field**). Jak powiązać formularz z naszym obiektem domenowym klasy Note? Z pewnością powinieneś się zapoznać ze składnią **\*{}** oraz **\${}** oraz sposobem jej wykorzystywania przy tworzeniu formularzy www.



Jak wypełnić listę rozwijaną **Importance** wartościami z naszego Enum'a? Możemy wykorzystać operator **T()** języka wyrażeń SpEL, który pozwala odwoływać się do metod statycznych. Zapewne zainteresuje cię metoda `static T[] values()` dostępna dla klasy Enum, która pozwoli na pobranie wszystkich zdefiniowanych stałych (poziomów ważności notatki).



Model, Widok, Kontroler, jak to właściwie działa? Po wpisaniu adresu w przeglądarce (np. [localhost:8080/list](http://localhost:8080/list)) wykonanie programu zostaje oddelegowane do właściwej metody w kontrolerze, która jest oznaczona odpowiednią adnotacją np.

`@GetMapping("/list")`. W tejże metodzie wykorzystamy naszą usługę do pobrania listy notatek i prześlemy ją do widoku za pomocą modelu np.

```
model.addAttribute("notes", noteService.listOfNotes());
```

W widoku z kolei mamy dostęp do przekazanej listy notatek za pomocą specjalnej notacji `${notes}`. Jak się z pewnością domyślasz, należałoby jedynie zrenderować do formatu html zawartość tejże listy, aby wyświetlić wszystkie notatki w przeglądarce.

```
package pl.edu.utp.po;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Lab3NotesApplication {

    public static void main(String[] args) {
        SpringApplication.run(Lab3NotesApplication.class, args);
    }
}
```