Hôm nay, chúng ta sẽ vận dụng toàn bộ kiến thức đã học để tạo ra website quản lý công việc bằng Spring Boot + Thymeleaf + MySQL.

**Cài đặt**

Chúng ta sẽ các dependencies sau:

1. spring-boot-starter-web
2. lombok
3. spring-boot-starter-thymeleaf
4. spring-boot-starter-data-jpa
5. mysql-connector-java

*pom.xml*

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<packaging>pom</packaging>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.0.5.RELEASE</version>

<relativePath /> <!-- lookup parent from repository -->

</parent>

<groupId>me.loda.spring</groupId>

<artifactId>spring-boot-learning</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>spring-boot-learning</name>

<description>Everything about Spring Boot</description>

<properties>

<java.version>1.8</java.version>

</properties>

<dependencies>

<!--spring mvc, rest-->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-thymeleaf</artifactId>

</dependency>

<!--spring jpa-->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<!-- mysql connector -->

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

</dependency>

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

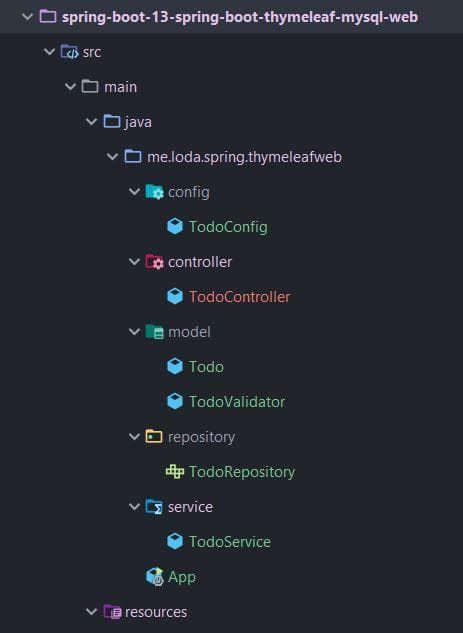
</plugin>

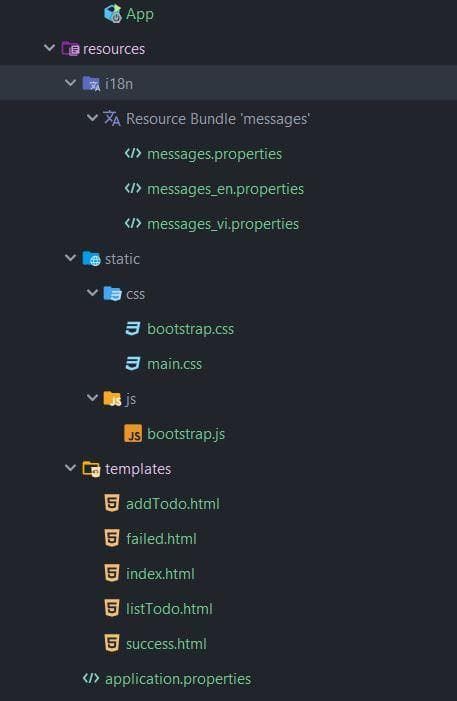
</plugins>

</build>

</project>

Cấu trúc thư mục:





**Tạo Database**

*script.sql*

CREATE SCHEMA IF NOT EXISTS `todo\_db` DEFAULT CHARACTER SET utf8mb4 ;

CREATE TABLE IF NOT EXISTS `todo\_db`.`todo` (

`id` INT(11) NOT NULL AUTO\_INCREMENT,

`title` VARCHAR(255) NULL DEFAULT NULL,

`detail` VARCHAR(255) NULL DEFAULT NULL,

PRIMARY KEY (`id`))

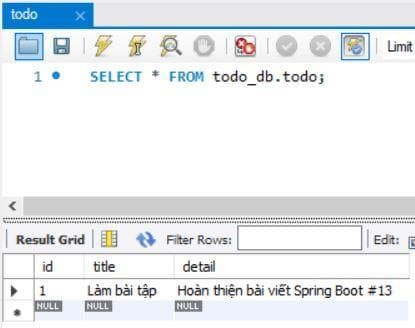
ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8mb4;

Thêm 1 record vào DB

INSERT INTO `todo\_db`.`todo` (`title`, `detail`) VALUES ('Làm bài tập', 'Hoàn thiện bài viết Spring Boot #13');

Xem thử kết quả:



**Cấu hình ứng dụng**

Cấu hình là phần cực kì quan trọng rồi, chúng ta phải cung cấp cho **Spring Boot** các thông tin về Database và Thymeleaf.

Ngoài ra, tùy chỉnh một số thông tin để giúp chúng ta lập trình đơn giản hơn.

*application.properties*

#Chạy ứng dụng trên port 8085

server.port=8085

# Bỏ tính năng cache của thymeleaf để lập trình cho nhanh

spring.thymeleaf.cache=false

# Các message tĩnh sẽ được lưu tại thư mục i18n

spring.messages.basename=i18n/messages

# Bỏ properties này đi khi deploy

# Nó có tác dụng cố định ngôn ngữ hiện tại chỉ là Tiếng Việt

spring.mvc.locale-resolver=fixed

# Mặc định ngôn ngữ là tiếng việt

spring.mvc.locale=vi\_VN

# Đổi thành tiếng anh bằng cách bỏ comment ở dứoi

#spring.mvc.locale=en\_US

spring.datasource.url=jdbc:mysql://localhost:3306/todo\_db?useSSL=false

spring.datasource.username=root

spring.datasource.password=root

## Hibernate Properties

# The SQL dialect makes Hibernate generate better SQL for the chosen database

spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5InnoDBDialect

# Hibernate ddl auto (create, create-drop, validate, update)

spring.jpa.hibernate.ddl-auto = update

**Tạo Model**

Tạo model Todo liên kết tới bảng todo trong Database.

Phần này chúng ta sử dụng [Lombok](https://loda.me/general-huong-dan-su-dung-lombok-giup-code-java-nhanh-hon-69-loda1552789752787) và [Hibernate](https://loda.me/hibernate-la-gi-loda1554623701594) *Todo.java*

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import lombok.Data;

@Entity

@Data

public class Todo {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String title;

private String detail;

}

Ngoài ra, chúng ta tạo thêm một đối tượng là TodoValidator, có trách nhiệm kiểm tra xem một object Todo là hợp lệ hay không.

import org.thymeleaf.util.StringUtils;

/\*

Đối tượng này dùng để kiểm tra xem một Object Todo có hợp lệ không

\*/

public class TodoValidator {

/\*\*

\* Kiểm tra một object Todo có hợp lệ không

\* @param todo

\* @return

\*/

public boolean isValid(Todo todo) {

return Optional.ofNullable(todo)

.filter(t -> !StringUtils.isEmpty(t.getTitle())) // Kiểm tra title khác rỗng

.filter(t -> !StringUtils.isEmpty(t.getDetail())) // Kiểm tra detail khác rỗng

.isPresent(); // Trả về true nếu hợp lệ, ngược lại false

}

}

Vậy là xong phần chuẩn bị Model.

**TodoConfig**

Trong ứng dụng của mình, tôi muốn tự tạo ra Bean TodoValidator.

Đây là lúc sử dụng @Configuration và @Bean đã học tại bài [Spring Boot #6](https://loda.me/spring-boot-6-configuration-va-bean-loda1557885506910)

*config/TodoConfig.java*

@Configuration

public class TodoConfig {

/\*\*

\* Tạo ra Bean TodoValidator để sử dụng sau này

\* @return

\*/

@Bean

public TodoValidator validator() {

return new TodoValidator();

}

}

**Tầng Repository**

Tầng Repository, chịu trách nhiệm giao tiếp với Database. Chúng ta sử dụng **Spring JPA**.

*repository/TodoRepository.java*

@Repository

public interface TodoRepository extends JpaRepository<Todo, Long> {

}

**Tầng Service**

Tầng Service, chị trách nhiệm thực hiện các xử lý logic, business, hỗ trợ cho tầng Controller.

*service/TodoService.java*

@Service

public class TodoService {

@Autowired

private TodoRepository todoRepository;

@Autowired

private TodoValidator validator;

/\*\*

\* Lấy ra danh sách List<Todo>

\*

\* @param limit - Giới hạn số lượng lấy ra

\*

\* @return Trả về danh sách List<Todo> dựa theo limit, nếu limit == null thì trả findAll()

\*/

public List<Todo> findAll(Integer limit) {

return Optional.ofNullable(limit)

.map(value -> todoRepository.findAll(PageRequest.of(0, value)).getContent())

.orElseGet(() -> todoRepository.findAll());

}

/\*\*

\* Thêm một Todo mới vào danh sách công việc cần làm

\*

\* @return Trả về đối tượng Todo sau khi lưu vào DB, trả về null nếu không thành công

\*/

public Todo add(Todo todo) {

if (validator.isValid(todo)) {

return todoRepository.save(todo);

}

return null;

}

}

**Tầng Controller**

Tầng Controller, nơi đón nhận các request từ phía người dùng, và chuyển tiếp xử lý xuống tầng Service.

*controller/TodoController.java*

@Controller

public class TodoController {

@Autowired

private TodoService todoService;

/\*

@RequestParam dùng để đánh dấu một biến là request param trong request gửi lên server.

Nó sẽ gán dữ liệu của param-name tương ứng vào biến

\*/

@GetMapping("/listTodo")

public String index(Model model, @RequestParam(value = "limit", required = false) Integer limit) {

// Trả về đối tượng todoList.

model.addAttribute("todoList", todoService.findAll(limit));

// Trả về template "listTodo.html"

return "listTodo";

}

@GetMapping("/addTodo")

public String addTodo(Model model) {

model.addAttribute("todo", new Todo());

return "addTodo";

}

/\*

@ModelAttribute đánh dấu đối tượng Todo được gửi lên bởi Form Request

\*/

@PostMapping("/addTodo")

public String addTodo(@ModelAttribute Todo todo) {

return Optional.ofNullable(todoService.add(todo))

.map(t -> "success") // Trả về success nếu save thành công

.orElse("failed"); // Trả về failed nếu không thành công

}

}

**Templates**

Tầng Controller đã trả về templates, nhiệm vụ tiếp theo là sử dụng Template Engine để xử lý các templates này và trả về webpage cho người dùng.

Template Engine chúng ta sử dụng sẽ là **Thymeleaf**, đã học tại các bài Spring Boot [#8](https://loda.me/spring-boot-8-tao-web-helloworld-voi-controller-thymeleaf-loda1558189401113), [#9](https://loda.me/spring-boot-9-giai-thich-cach-thymeleaf-van-hanh-expression-demo-full-loda1558267496214), [#10](https://loda.me/spring-boot-10-request-mapping-post-mapping-model-attribute-request-param-web-to-do-voi-thymeleaf-loda1558661736676).

*index.html*

<!DOCTYPE html>

<htmlxmlns:th="http://www.thymeleaf.org"><head><metacharset="UTF-8"/><title>Loda To-do</title><metahttp-equiv="Content-Type"content="text/html; charset=UTF-8"/><!--css-->

<linkth:href="@{/css/bootstrap.css}"rel="stylesheet"/><linkth:href="@{/css/main.css}"rel="stylesheet"/><!--js-->

<scriptth:src="@{/js/bootstrap.js}"></script></head><body><h1th:text="#{loda.message.hello}"></h1><ath:href="@{/listTodo}"th:text="#{loda.value.viewListTodo}"class="btn btn-primary"></a><ath:href="@{/addTodo}"th:text="#{loda.value.addTodo}"class="btn btn-success"></a></body></html>

*listTodo.html*

<!DOCTYPE html>

<htmlxmlns:th="http://www.thymeleaf.org"><head><metacharset="UTF-8"/><title>Loda To-do</title><metahttp-equiv="Content-Type"content="text/html; charset=UTF-8"/><!--css-->

<linkth:href="@{/css/bootstrap.css}"rel="stylesheet"/><linkth:href="@{/css/main.css}"rel="stylesheet"/><!--js-->

<scriptth:src="@{/js/bootstrap.js}"></script></head><body><h1th:text="#{loda.value.listTodo} + ':'"></h1><ul><!--Duyệt qua toàn bộ phần tử trong biến "todoList"-->

<lith:each="todo : ${todoList}"><!--Với mỗi phần tử, lấy ra title và detail-->

<spanth:text="\*{todo.getTitle()}"></span> : <spanth:text="\*{todo.getDetail()}"></span></li></ul><ath:href="@{/addTodo}"th:text="#{loda.value.addTodo}"class="btn btn-success"></a></body></html>

*addTodo.html*

<!DOCTYPE html>

<htmlxmlns:th="http://www.thymeleaf.org"><head><metacharset="UTF-8"/><title>Loda To-do</title><metahttp-equiv="Content-Type"content="text/html; charset=UTF-8"/><!--css-->

<linkth:href="@{/css/bootstrap.css}"rel="stylesheet"/><linkth:href="@{/css/main.css}"rel="stylesheet"/><!--js-->

<scriptth:src="@{/js/bootstrap.js}"></script></head><body><h1>To-do</h1><formth:action="@{/addTodo}"th:object="${todo}"method="post"><p>title: <inputtype="text"th:field="\*{title}"/></p><p>detail: <inputtype="text"th:field="\*{detail}"/></p><p><buttontype="submit"class="btn btn-success">Add</button></p></form></body></html>

*success.html*

<!DOCTYPE html>

<htmlxmlns:th="http://www.thymeleaf.org"><head><metacharset="UTF-8"/><title>Loda To-do</title><metahttp-equiv="Content-Type"content="text/html; charset=UTF-8"/><linkth:href="@{/css/bootstrap.css}"rel="stylesheet"/><linkth:href="@{/css/main.css}"rel="stylesheet"/></head><body><h1>To-do</h1><h1 style="color: green"th:text="#{loda.message.success}"></h1><ath:href="@{/listTodo}"th:text="#{loda.value.viewListTodo}"class="btn btn-primary"></a></body></html>

*failed.html*

<!DOCTYPE html>

<htmlxmlns:th="http://www.thymeleaf.org"><head><metacharset="UTF-8"/><title>Loda To-do</title><metahttp-equiv="Content-Type"content="text/html; charset=UTF-8"/><linkth:href="@{/css/bootstrap.css}"rel="stylesheet"/><linkth:href="@{/css/main.css}"rel="stylesheet"/></head><body><h1>To-do</h1><h1 style="color: red"th:text="#{loda.message.failed}"></h1><ath:href="@{/listTodo}"th:text="#{loda.value.viewListTodo}"class="btn btn-primary"></a></body></html>

**i18n**

Trong các template, tôi có sử dụng các message tĩnh, những message này hỗ trợ đa ngôn ngữ.

Chúng ta định nghĩa các message này tại thư mục i18n.

*i18n/messages\_vi.properties*

loda.message.hello=Welcome to TodoApp

loda.message.success=Thêm Todo thành công!

loda.message.failed=Thêm Todo không thành công!

loda.value.addTodo=Thêm công việc

loda.value.viewListTodo=Xem danh sách công việc

loda.value.listTodo=Danh sách công việc

*i18n/messages\_en.properties*

loda.message.hello=Welcome to TodoApp

loda.message.success=Add To-do Successfully!

loda.message.failed=Add To-do Failed!

loda.value.addTodo=Add To-do

loda.value.viewListTodo=View To-do list

loda.value.listTodo=To-do list

**Chạy thử ứng dụng**

Chạy ứng dụng:

*App.java*

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class App {

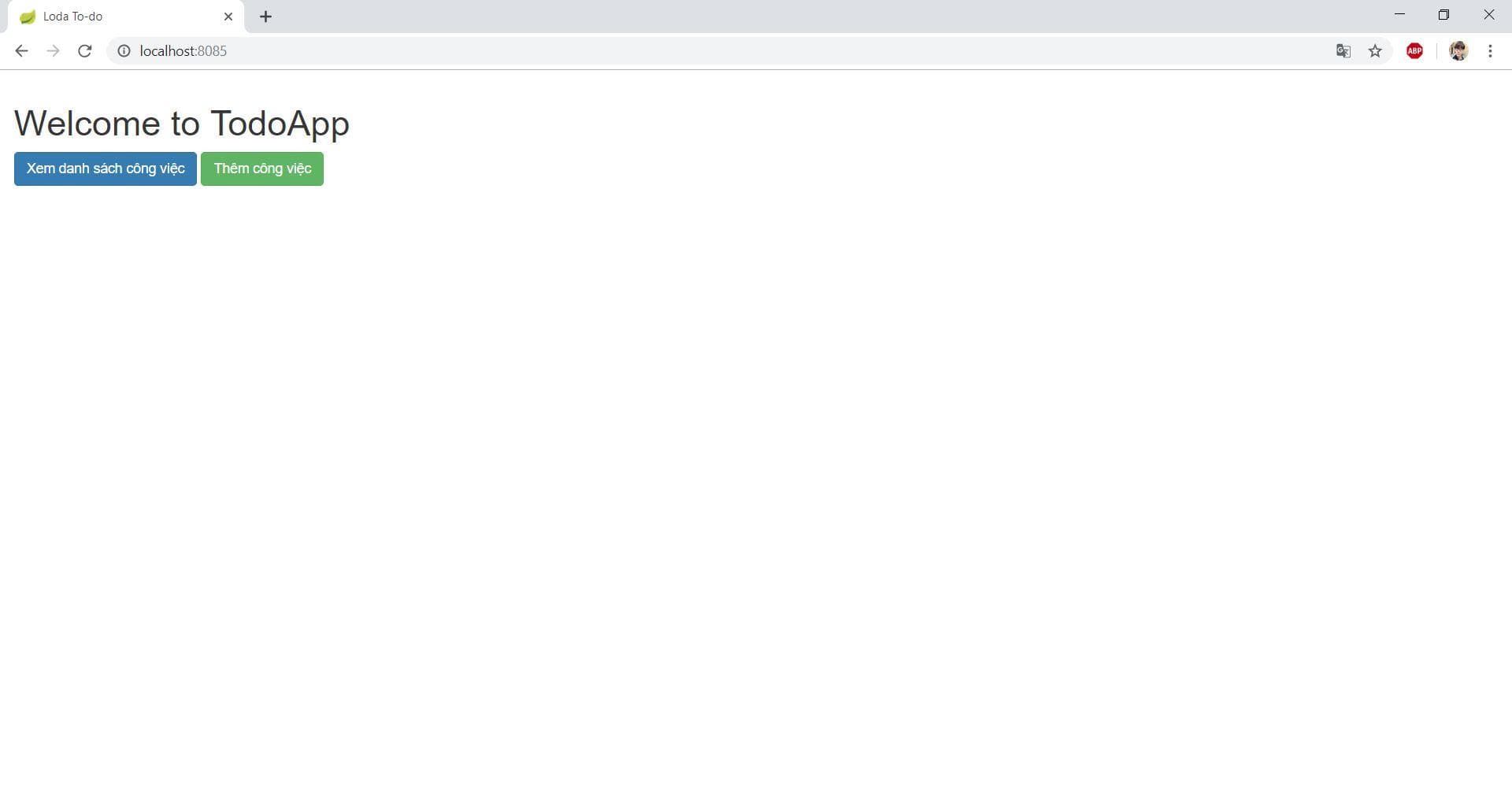
public static void main(String[] args) {

SpringApplication.run(App.class, args);

}

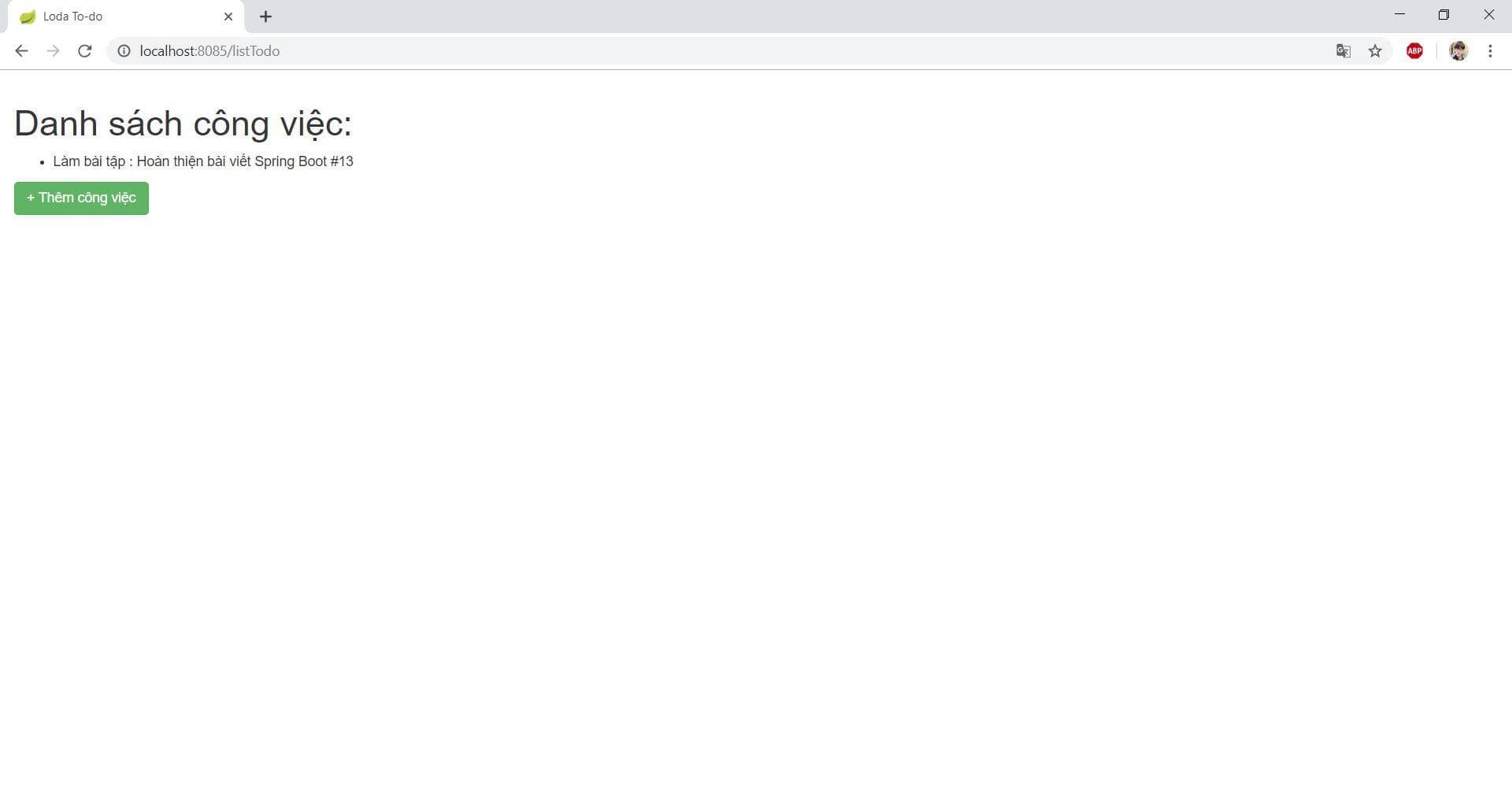
}

Truy cập địa chỉ: <http://localhost:8085/>



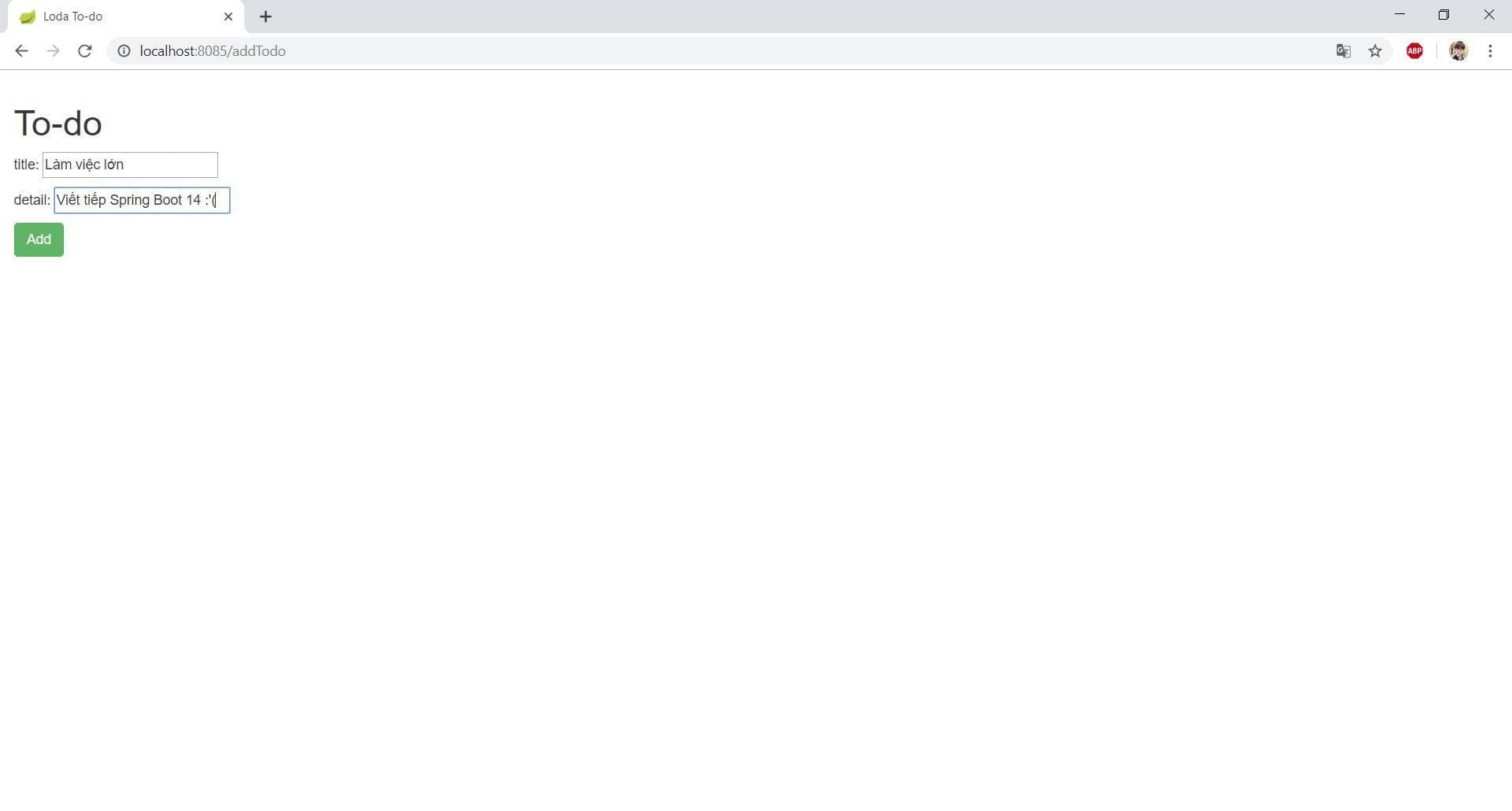
Vì chúng ta cấu hình Locale là vi, nên ngôn ngữ đều hiện Tiếng Việt, rất tuyệt :3

Bấm vào **Xem danh sách công việc** để tới /listTodo

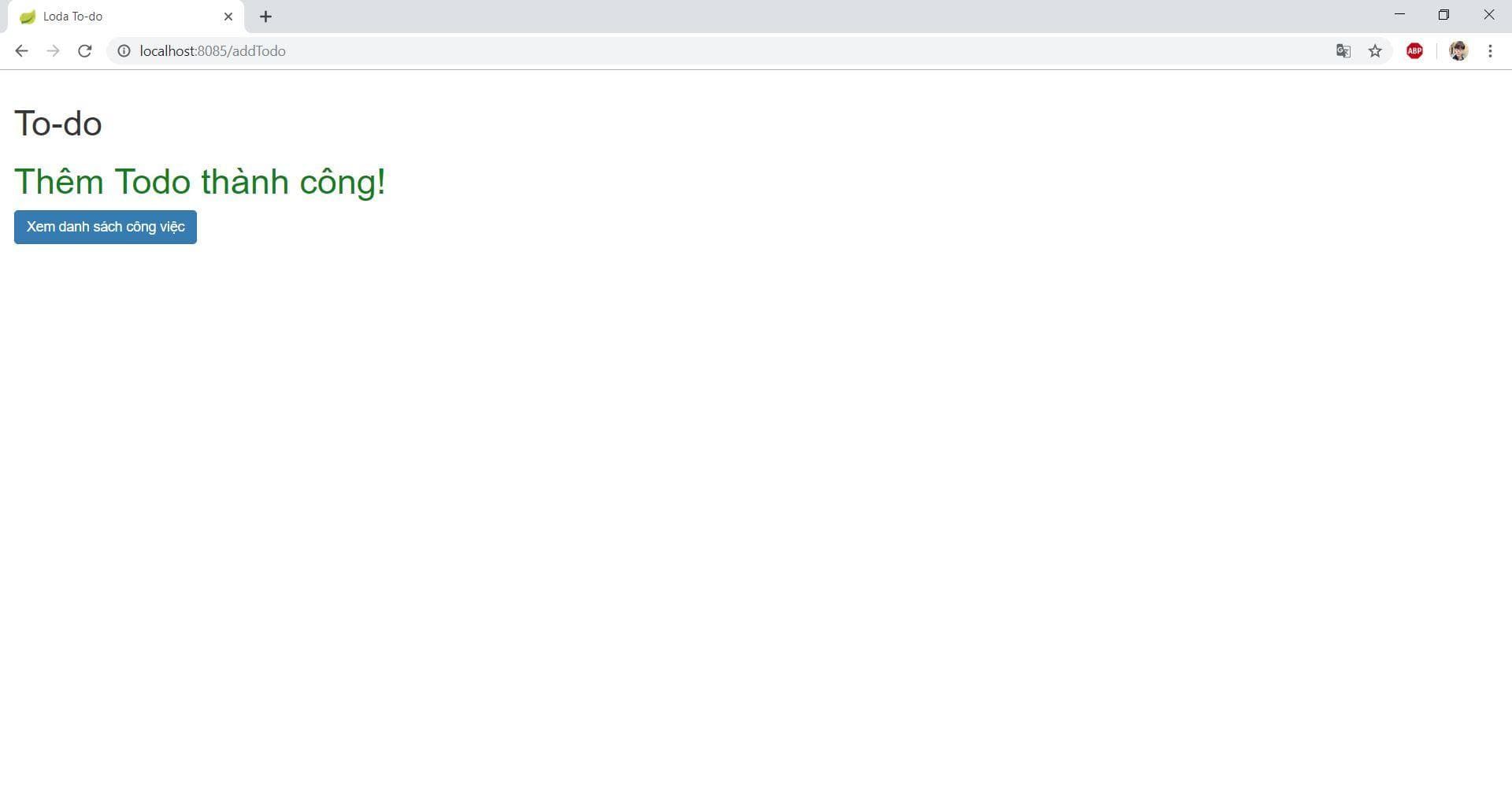


Vì chúng ta đã insert 1 bản ghi vào Database từ trước, nên ở đây nó hiện ra 1 việc cần làm.

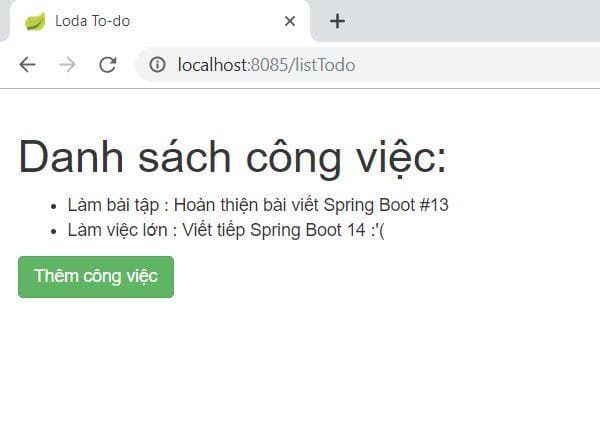
Bấm vào **Thêm công việc** để tới /addTodo



Bấm **Add** để lưu thông tin vào Database.

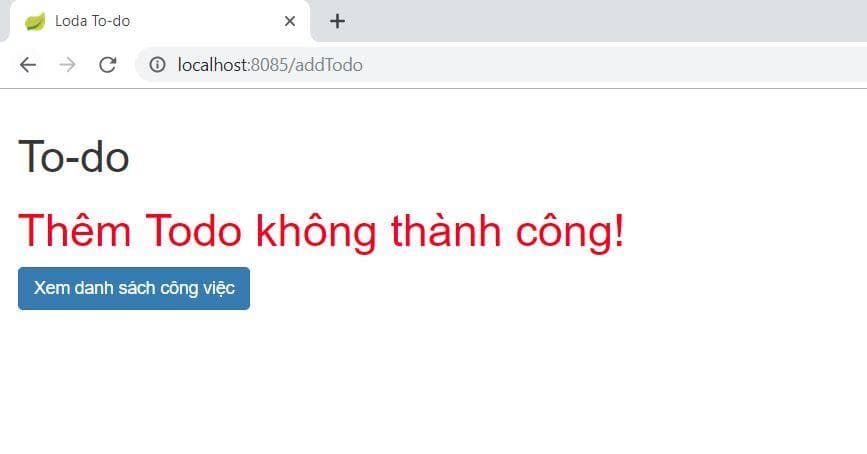


Vậy là giờ chúng ta có 2 công việc :3



Bây giờ giả sử dụng ta gửi lên request tạo ra một Todo không hợp lệ.

TodoValidator sẽ trả về null -> thêm thất bại



**Kết**