



Java Spring

Spring MVC & Freemarker template

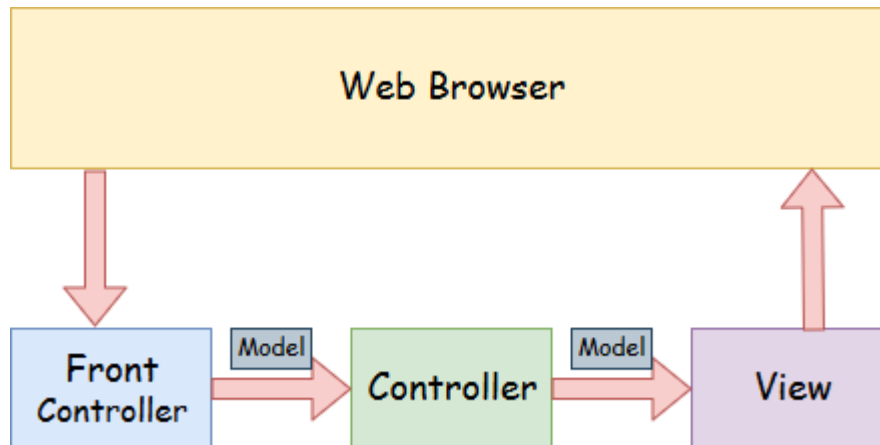
ỨNG DỤNG WEB QUẢN LÝ NHÂN SỰ CƠ BẢN

Môn học: Lập trình ứng dụng mạng

A. Tổng quan

1. Mục tiêu:

Làm quen với Spring MVC – một framework java nổi tiếng, được sử dụng để xây dựng các ứng dụng web. Tuân theo mẫu thiết kế Model – View – Controller.



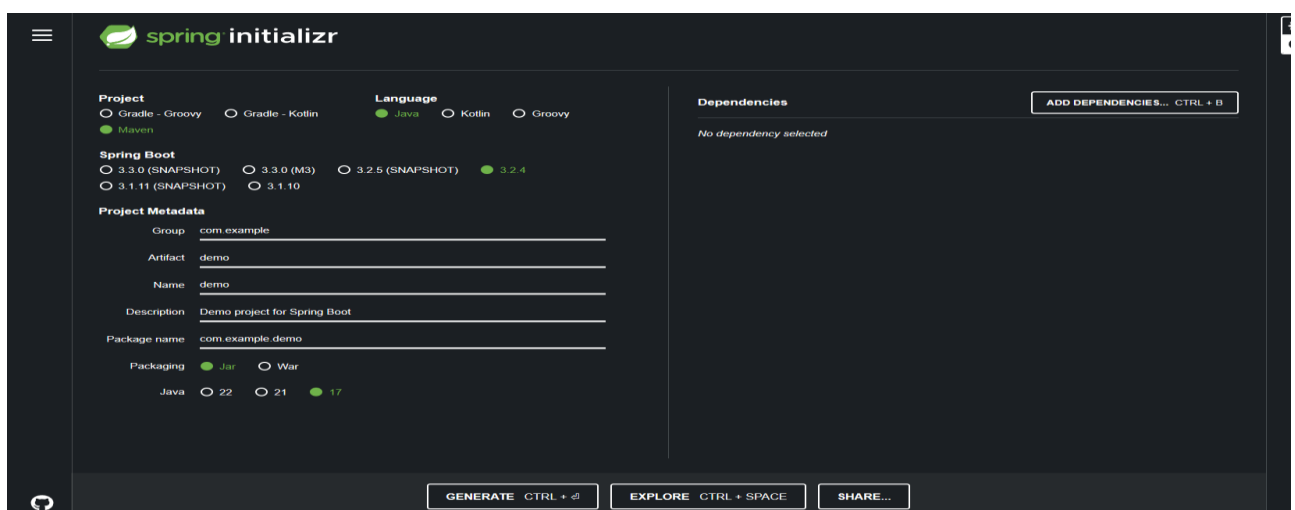
2. Chuẩn bị môi trường:

IDE : Bất kì IDE nào có thể hỗ trợ ngôn ngữ java. (IntelliJ, NetBean, Eclipse, Vscode,...)
Database: MongoDB, MySQL ,

B. Thực hành

1. Khởi tạo project:

Bước 1: Truy cập vào trang web [Spring Initializr](https://spring.io/guides-topics/docs/spring-initializr/) để khởi tạo Spring project.



Bước 2: Ở phía bên phải màn hình, chọn build tool, Language, Spring Boot version và điền các thông tin cần thiết của project.

Project
☐ Gradle - Groovy ☐ Gradle - Kotlin
☒ **Maven**

Language
☒ **Java** ☐ Kotlin ☐ Groovy

Spring Boot
☐ 3.3.0 (SNAPSHOT) ☐ 3.3.0 (M3) ☐ 3.2.5 (SNAPSHOT) ☒ **3.2.4**
☐ 3.1.11 (SNAPSHOT) ☐ 3.1.10

Project Metadata

Group

com.example

Artifact

demo

Name

demo

Description

Demo project for Spring Boot

Package name

com.example.demo

Packaging

☒ **Jar** ☐ War

Java

☐ 22 ☐ 21 ☒ **17**

Bước 3: Thêm các dependencies cần thiết. Ở trong project này sẽ sử dụng MongoDB và Freemarker templates.

Dependencies

ADD DEPENDENCIES... CTRL + B

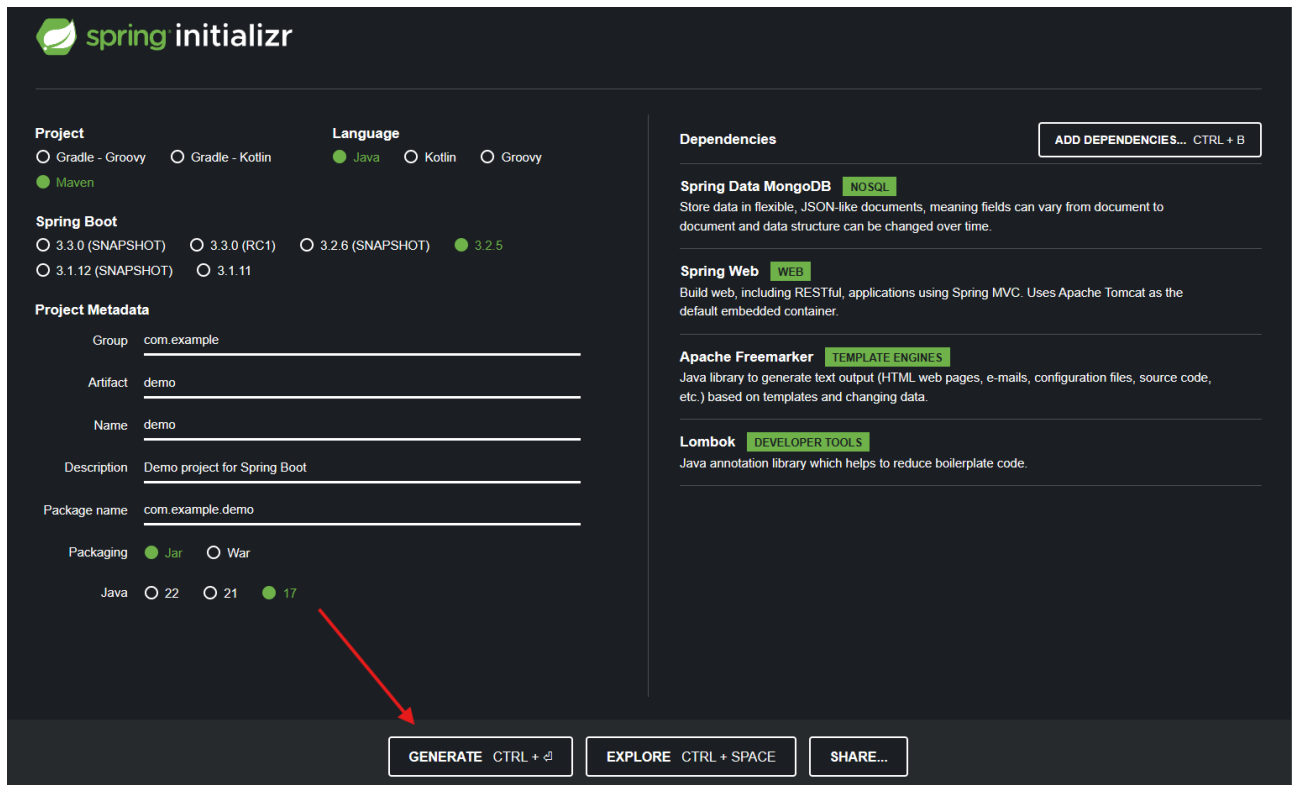
Spring Data MongoDB **NOSQL**
Store data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time.

Spring Web **WEB**
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Apache Freemarker **TEMPLATE ENGINES**
Java library to generate text output (HTML web pages, e-mails, configuration files, source code, etc.) based on templates and changing data.

Lombok **DEVELOPER TOOLS**
Java annotation library which helps to reduce boilerplate code.

Bước 4: Nhấn Generate để tải thư mục project về máy.



The Spring Initializr web interface is shown with a dark theme. It includes sections for Project, Language, Spring Boot, Project Metadata, Dependencies, and a bottom bar with buttons for GENERATE, EXPLORE, and SHARE. A red arrow points to the '17' version number under the Java language section.

Project

☐ Gradle - Groovy ☐ Gradle - Kotlin ☒ Java ☐ Kotlin ☐ Groovy

☒ Maven

Spring Boot

☐ 3.3.0 (SNAPSHOT) ☐ 3.3.0 (RC1) ☐ 3.2.6 (SNAPSHOT) ☒ 3.2.5

☐ 3.1.12 (SNAPSHOT) ☐ 3.1.11

Project Metadata

Group

Artifact

Name

Description

Package name

Packaging ☒ Jar ☐ War

Java ☐ 22 ☐ 21 ☒ 17

Dependencies ADD DEPENDENCIES... CTRL + B

Spring Data MongoDB NOSQL
Store data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time.

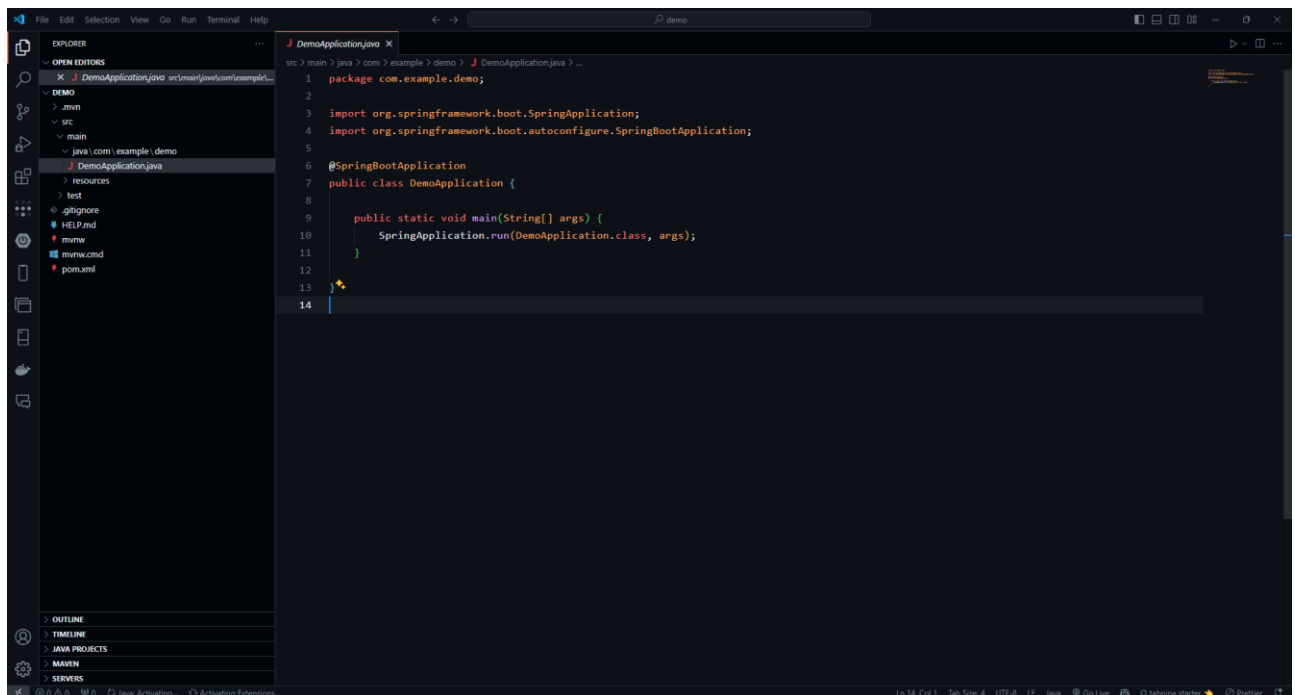
Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Apache Freemarker TEMPLATE ENGINES
Java library to generate text output (HTML web pages, e-mails, configuration files, source code, etc.) based on templates and changing data.

Lombok DEVELOPER TOOLS
Java annotation library which helps to reduce boilerplate code.

Buttons: GENERATE CTRL + G, EXPLORE CTRL + SPACE, SHARE...

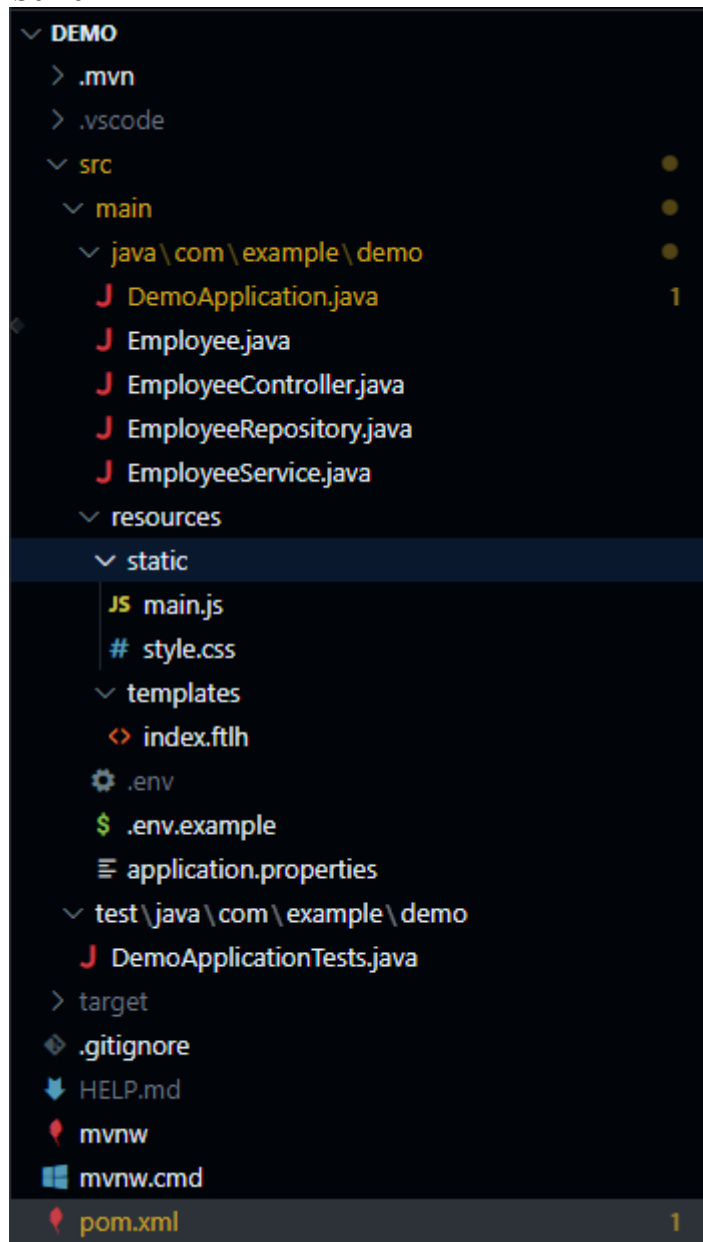
Bước 5: Nhấn chọn mở thư mục bằng IDE hỗ trợ Java.



2. Cấu trúc thư mục:

Dù cho project được tạo với Maven hay Gradle thì cấu trúc chung vẫn tương tự nhau, do tuân theo một template có sẵn (tên là Archetype):

- Thư mục gốc chứa các file như pom.xml (của Maven), build.gradle và các file khác như .gitignore,... dùng để cấu hình dự án.
- Thư mục .mvn hoặc .gradle là thư mục riêng của Maven và Gradle, đừng nên đụng tới hay exclude nó ra khỏi source code.
- Code được chứa trong thư mục src.
- Thư mục build ra chứa các file class, file JAR. Với Maven là target còn Gradle là build.



Như hình, thư mục chính chúng ta cần quan tâm là src/main/java/<tên package>. Mọi code java đều nằm trong này:

- Tên package chính được đặt dạng ngược với tên miền. Ví dụ như example.com thì đặt thành com.example. Cộng thêm tên project nữa.
- Có các package con, mỗi package đại diện cho các class thuộc layer cụ thể (ví dụ như service, controller,...)
- Thư mục resources chứa các tài nguyên của ứng dụng như hình ảnh, static file, properties file,...

Ngoài ra còn có src/test dùng để chứa các test class, dùng cho unit test.

3. Cấu hình dự án trong 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/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.2.4</version>
    <relativePath /> <!-- lookup parent from repository -->
  </parent>
  <groupId>com.example</groupId>
  <artifactId>demo</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>war</packaging>
  <name>demo</name>
  <description>Demo project for Spring Boot</description>
  <properties>
    <java.version>17</java.version>
  </properties>
```

<groupId>: ID của nhóm dự án.

<artifactId>: ID của dự án (hoặc “artifact”). Trong trường hợp này, nó là demo.

<version>: Phiên bản của dự án.

<packaging>: Loại gói mà Maven sẽ tạo.

<name>: Tên của dự án.

<description>: Mô tả về dự án.

<properties>: Chứa các thuộc tính có thể được sử dụng trong tệp POM.

```
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-mongodb</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-websocket</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <scope>runtime</scope>
    <optional>true</optional>
  </dependency>
  <dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <optional>true</optional>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-tomcat</artifactId>
    <scope>provided</scope>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
```



```
</dependency>
<dependency>
    <groupId>me.paulschwarz</groupId>
    <artifactId>spring-dotenv</artifactId>
    <version>2.5.4</version>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-freemarker</artifactId>
</dependency>
</dependencies>
```

Phần **<dependencies>** trong tệp POM của Maven chứa danh sách các thư viện mà dự án cần để biên dịch, xây dựng, kiểm tra và/hoặc chạy.

- **spring-boot-starter-data-mongodb**: Thư viện này cung cấp các tiện ích để kết nối và làm việc với cơ sở dữ liệu MongoDB trong ứng dụng Spring Boot.
- **spring-boot-starter-web**: Thư viện này cung cấp các tiện ích để xây dựng ứng dụng web, bao gồm RESTful, sử dụng Spring MVC. Nó cũng giúp hỗ trợ các dịch vụ như Tomcat làm máy chủ web mặc định.
- **spring-boot-devtools**: Thư viện này cung cấp các tiện ích phát triển như tự động tải lại ứng dụng khi có thay đổi trong mã nguồn.
- **lombok**: Lombok là một thư viện Java giúp loại bỏ mã boilerplate trong mã nguồn Java của bạn bằng cách sử dụng các annotation để tạo ra các phương thức như getters, setters, constructors, và hơn thế nữa.
- **spring-boot-starter-tomcat**: Thư viện này cung cấp hỗ trợ cho Tomcat làm máy chủ web nhúng.
- **spring-boot-starter-test**: Thư viện này cung cấp các tiện ích để kiểm tra ứng dụng Spring Boot, bao gồm các thư viện như JUnit, Mockito, và Hamcrest.
- **spring-dotenv**: Thư viện này giúp ứng dụng Spring Boot của bạn đọc các biến môi trường từ tệp .env.
- **spring-boot-starter-freemarker**: Thư viện này cung cấp hỗ trợ cho FreeMarker, một thư viện tạo template.

```
<build>
    <finalName>${artifactId}</finalName>
    <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>
```

Phần <build> trong tệp POM của Maven chứa các cấu hình để xây dựng dự án.

- <finalName>: Tên cuối cùng của dự án khi nó được xây dựng. Trong trường hợp này, nó sẽ lấy giá trị từ biến \${artifactId}.
- <plugins>: Chứa danh sách các plugin Maven sẽ sử dụng trong quá trình xây dựng.
- <configuration>: Chứa các cấu hình cho plugin. Trong trường hợp này, nó định nghĩa một danh sách các phụ thuộc để loại trừ khỏi quá trình xây dựng.

4. Kết nối cơ sở dữ liệu:

Trong dự án này chúng ta sử dụng MongoDB để quản lý và lưu trữ dữ liệu của dự án. Tạo file .env ở trong thư mục resources với các biến tương tự:

```
MONGO_DATABASE=
MONGO_USER=
MONGO_PASSWORD=
MONGO_CLUSTER=
```

Sau đó cấu hình url và database để khởi tạo kết nối tới MongoDB ở trong application.properties.

```
spring.data.mongodb.database=${env.MONGO_DATABASE}
spring.data.mongodb.uri=mongodb+srv://${env.MONGO_USER}:${env.MONGO_PASSWORD}@${env.MONGO_CLUSTER}
```

5. Repository, Model, Service, Controller & View:

1. Model: Employee.java

```
@Document(collection = "employee")
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Employee {
    @Id
    private ObjectId objectId;

    private String id;
    private String name;
    private int age;
    private int salary;
}
```

- **@Document(collection = "employee")**: là một annotation của Spring Data MongoDB, được sử dụng để chỉ định rằng class này sẽ được ánh xạ tới collection "employee" trong MongoDB.
- **@Data**: Annotation này sẽ tự động tạo ra các getter, setter, equals, hashCode và toString cho class.
- **@AllArgsConstructor**: tạo ra một constructor với tất cả các trường làm tham số.
- **@NoArgsConstructor**: tạo ra một constructor không tham số.

Khởi tạo các thông tin cần thiết của nhân viên như id, tên, tuổi, lương. Lưu ý ObjectId không nằm trong các thông tin của nhân viên mà chính là id record để lưu trữ trong MongoDB.

2. Repository: EmployeeRepository.java

```
@Repository
public interface EmployeeRepository extends MongoRepository<Employee,
ObjectId> {
    Employee findById(String id);
}
```

- **@Repository**: được sử dụng để đánh dấu một class như một DAO (Data Access Object) hoặc một Repository. Spring sẽ tự động quản lý các bean của các class được đánh dấu bằng annotation này.
- **public interface EmployeeRepository extends MongoRepository<Employee, ObjectId>**: là khai báo của interface EmployeeRepository giúp đơn giản hóa việc truy cập dữ liệu trong MongoDB. Nó kế thừa từ MongoRepository với hai tham số kiểu là Employee và ObjectId.
- **Employee findById(String id)**: là một phương thức được định nghĩa trong EmployeeRepository. Nó được sử dụng để tìm một Employee dựa trên id được cung cấp.

3. Service: EmployeeService.java

```
@Service
public class EmployeeService {
    @Autowired
    private EmployeeRepository employeeRepository;

    @Autowired
    private MongoTemplate mongoTemplate;
```

- **@Service**: được sử dụng để đánh dấu một class như một Service.
- **@Autowired**: được sử dụng để tự động chèn (inject) các bean vào các trường hoặc phương thức. Trong trường hợp này, Spring sẽ tự động chèn một instance của EmployeeRepository và MongoTemplate vào các trường tương ứng.

```
public List<Employee> allEmployees() {
    return employeeRepository.findAll();
}

public Employee getEmployeeById(String id) {
    return employeeRepository.findById(id);
}

// add employee to database

public Employee addEmployee(Employee employee) {
    employeeRepository.insert(employee);
    return employee;
}
```

```
// edit employee
public void editEmployee(Employee employee, String employeeId) {
    Query select = Query.query(Criteria.where("id").is(employeeId));
    Update update = new Update();
    update.set("name", employee.getName());
    update.set("age", employee.getAge());
    update.set("salary", employee.getSalary());

    mongoTemplate.update(Employee.class)
        .matching(select)
        .apply(update)
        .first();
}

//delete employee
public void deleteEmployee(String id) {
    Employee employee = employeeRepository.findById(id);
    ObjectId employeeId = employee.getObjectId();
    employeeRepository.deleteById(employeeId);
}
```

- **allEmployees()**: Phương thức này trả về danh sách tất cả các nhân viên từ cơ sở dữ liệu. Nó gọi phương thức `findAll()` của `employeeRepository`.
- **getEmployeeById(String id)**: Phương thức này trả về một nhân viên dựa trên id được cung cấp. Nó gọi phương thức `findById(id)` của `employeeRepository`.
- **addEmployee(Employee employee)**: Phương thức này thêm một nhân viên mới vào cơ sở dữ liệu.
- **editEmployee(Employee employee, String employeeId)**: Phương thức này chỉnh sửa thông tin của một nhân viên dựa trên `employeeId` được cung cấp.
- **deleteEmployee(String id)**: Phương thức này xóa một nhân viên từ cơ sở dữ liệu dựa trên id được cung cấp.

4.Controller: EmployeeController.java

```
@Controller
public class EmployeeController {
```

```
@Autowired
private EmployeeService employeeService;
```

- **@Controller:** Đây là một annotation của Spring MVC, được sử dụng để đánh dấu một class như một Controller. Controller là một thành phần quan trọng trong mô hình MVC (Model-View-Controller), nó xử lý các yêu cầu từ người dùng và trả về một phản hồi.
- **@Autowired:** được sử dụng để tự động chèn (inject) các bean vào các trường hoặc phương thức.
- **private EmployeeService employeeService:** Nó giữ một tham chiếu đến một instance của EmployeeService, cho phép EmployeeController thực hiện các thao tác trên cơ sở dữ liệu thông qua EmployeeService

```
@GetMapping("/")
public String index(@ModelAttribute("model") ModelMap model) {
    model.addAttribute("employees", employeeService.allEmployees());
    return "index";
}

@PostMapping("/add")
public String addEmployee(Employee employee) {
    employeeService.addEmployee(employee);
    return "redirect:/";
}

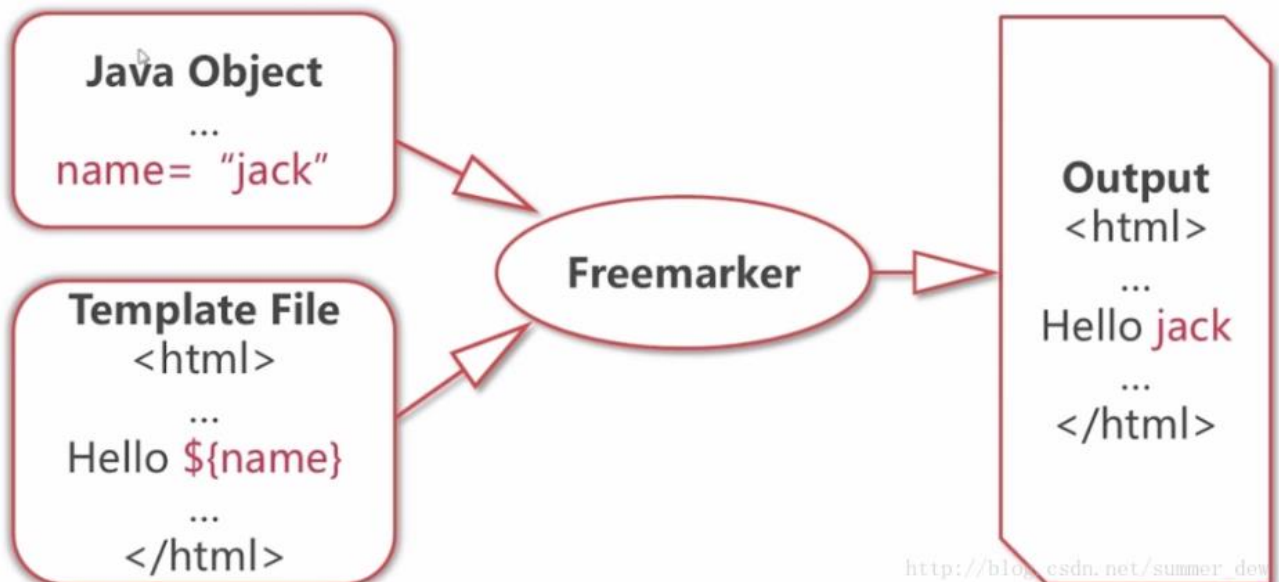
@GetMapping("/edit/{id}")
public String editEmployee(@PathVariable("id") String id, Employee
employee) {
    employee.setId(id);
    employeeService.editEmployee(employee, id);
    return "redirect:/";
}

@GetMapping("/delete/{id}")
public String deleteEmployee(@PathVariable String id) {
    employeeService.deleteEmployee(id);
    return "redirect:/";
}
```

- **@GetMapping("/")**: Đây là một annotation của Spring MVC, được sử dụng để xử lý các yêu cầu **GET** tới đường dẫn **"/"**. Phương thức **index()** sẽ được gọi khi có yêu cầu tới đường dẫn này. Nó thêm danh sách tất cả các nhân viên vào model và trả về tên của view là **"index"**.
- **@PostMapping("/add")**: Đây là một annotation của Spring MVC, được sử dụng để xử lý các yêu cầu **POST** tới đường dẫn **"/add"**. Phương thức **addEmployee()** sẽ được gọi khi có yêu cầu tới đường dẫn này. Nó thêm một nhân viên mới vào cơ sở dữ liệu và sau đó chuyển hướng người dùng về trang chủ.
- **@GetMapping("/edit/{id}")**: Đây là một annotation của Spring MVC, được sử dụng để xử lý các yêu cầu **GET** tới đường dẫn **"/edit/{id}"**. Phương thức **editEmployee()** sẽ được gọi khi có yêu cầu tới đường dẫn này. Nó chỉnh sửa thông tin của một nhân viên dựa trên id được cung cấp và sau đó chuyển hướng người dùng về trang chủ.
- **@GetMapping("/delete/{id}")**: Đây là một annotation của Spring MVC, được sử dụng để xử lý các yêu cầu **GET** tới đường dẫn **"/delete/{id}"**. Phương thức **deleteEmployee()** sẽ được gọi khi có yêu cầu tới đường dẫn này. Nó xóa một nhân viên từ cơ sở dữ liệu dựa trên id được cung cấp và sau đó chuyển hướng người dùng về trang chủ.

5.View: Freemarker template

FreeMarker là một template engine cho nền tảng Java. Nó đọc các file mẫu và kết hợp với các đối tượng Java để tạo đầu ra văn bản như: trang web HTML, email, mã nguồn. Các mẫu (template) trong FreeMarker có đuôi mở rộng là **.ftl** (FreeMarker Template Language).



Cấu hình trong **application.properties**:

```
spring.freemarker.template-loader-path: classpath:/templates
spring.freemarker.suffix: .ftlh
```

spring.freemarker.template-loader-path: Thuộc tính này chỉ định đường dẫn nơi Spring Boot sẽ tìm kiếm các tệp mẫu FreeMarker.

spring.freemarker.suffix: Thuộc tính này chỉ định phần mở rộng mà Spring Boot sẽ sử dụng khi tìm kiếm các tệp mẫu FreeMarker.

Đuôi file .ftlh sẽ cho phép chúng ta viết các câu lệnh giống hệt trong html nhưng vẫn có thể phù hợp với định dạng của Freemarker.

Dưới đây là front end của trang web quản lý nhân sự sử dụng Freemarker template.

ID	Name	Age	Salary	Add
----	------	-----	--------	-----

#	Employee name	Age	Salary		
1	Anh Tuan	21	12,345	Delete	Edit
2	Tuan Anh	23	30,000	Delete	Edit

HẾT