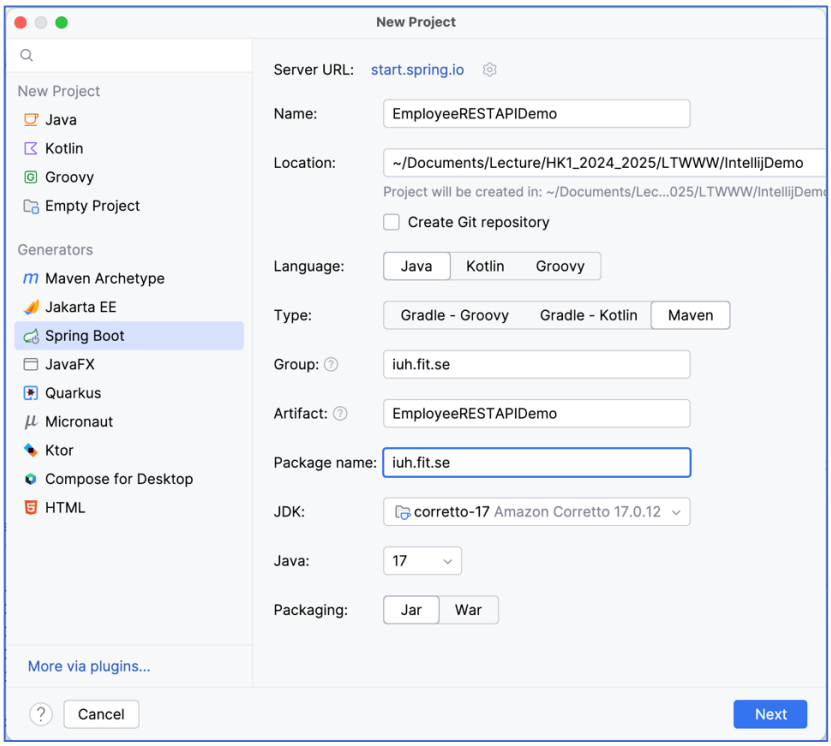
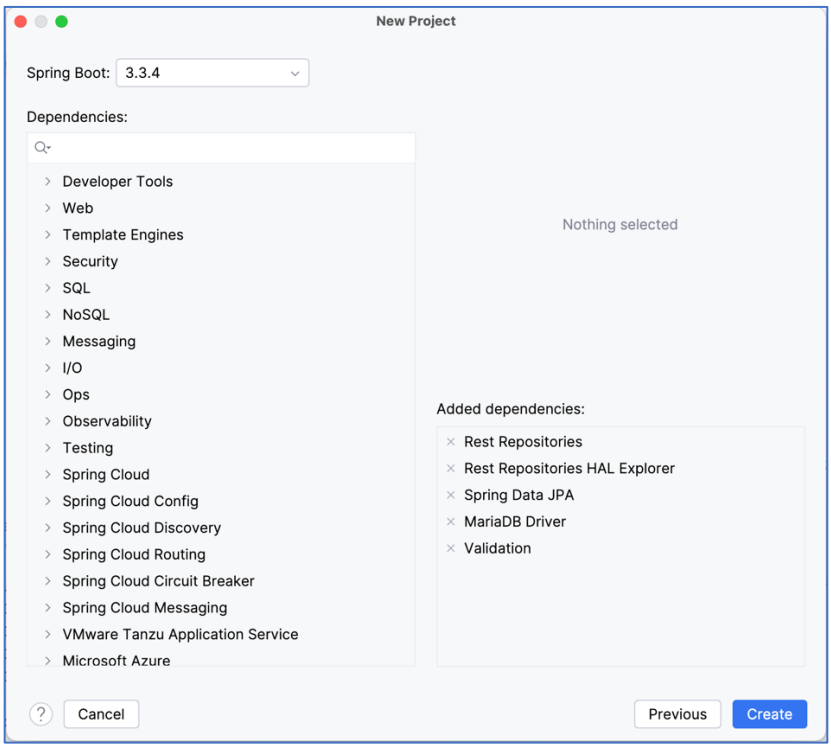
# HƯỚNG DẪN CÁCH TẠO 1 PROJECT SPRING REST API

## 1. *Tạo project và chọn dependencies:*



EmployeeRestapiDemoApplication.java

```java
package iuh.fit.se;

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

@SpringBootApplication
public class EmployeeRestapiDemoApplication {

    public static void main(String[] args) {

        SpringApplication.run(EmployeeRestapiDemoApplication.class, args);
    }

}
```

2. ***Tạo database: employees***
3. ***Config datasource ở file: src/main/resource/application.properties***

application.properties

```properties
spring.application.name=SpringRestAPIDemo

# Setting mariaDB
spring.datasource.driver-class-name=org.mariadb.jdbc.Driver
spring.datasource.url=jdbc:mariadb://localhost:3306/employees
spring.datasource.username=root
spring.datasource.password=123456
spring.jpa.hibernate.ddl-auto=create
spring.jpa.show-sql=true
```

4. ***Tạo entities – package: iuh.fit.se.entities***

   a. *Address.java*

```java
package iuh.fit.se.entities;

@Entity
@Table(name = "address")
public class Address {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    private String address;

    @OneToOne(mappedBy = "address", fetch = FetchType.EAGER)
    @JsonIgnore
    private Employee employee;

    public Address() {
        super();
        // TODO Auto-generated constructor stub
    }

    public Address(int id, String address, Employee employee) {
        super();
        this.id = id;
        this.address = address;
```

```java
        this.employee = employee;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    public Employee getEmployee() {
        return employee;
    }

    public void setEmployee(Employee employee) {
        this.employee = employee;
    }

    @Override
    public String toString() {
        return "Address [id=" + id + ", address=" + address + ", employee=" + employee + "]";
    }

}
```

*b. Employee.java*

```java
package iuh.fit.se.entities;

@Entity
@Table(name = "employee")
public class Employee {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    @Column(name = "first_name")
    @NotEmpty(message = "First Name must not be Empty")
    private String firstName;

    @Column(name = "last_name")
    @NotEmpty(message = "Last Name must not be Empty")
    private String lastName;

    private String gender;

    @Column(name = "email")
    @NotEmpty(message = "Email must not be Empty")
    @Email(message = "Email should be valid")
    private String emailAddress;

    @Column(name = "phone_number")
    @Pattern(regexp = "\\(\\d{3}\\)\\d{3}-\\d{4}", message = "Please input phone number with
format: (NNN)NNN-NNNN")
    private String phoneNumber;

    @Past(message = "Date of birth must be less than today")
    @DateTimeFormat(pattern = "yyyy-MM-dd")
```

```java
    private Date dob;

    @CreationTimestamp
    @Temporal(TemporalType.TIMESTAMP)
    @Column(name = "created_date")
    private Date createdDate;

    @UpdateTimestamp
    @Temporal(TemporalType.TIMESTAMP)
    @Column(name = "modified_date")
    private Date modifiedDate;

    @OneToOne(fetch = FetchType.EAGER, cascade = CascadeType.ALL, orphanRemoval = true)
    @JoinColumn(name = "address_id", referencedColumnName = "id")
    @JsonIgnore
    private Address address;

    public Employee() {
        super();
        // TODO Auto-generated constructor stub
    }

    public Employee(int id, String firstName, String lastName, String gender, String
emailAddress, String phoneNumber,
                    Date dob, Date createdDate, Date modifiedDate) {
        super();
        this.id = id;
        this.firstName = firstName;
        this.lastName = lastName;
        this.gender = gender;
        this.emailAddress = emailAddress;
        this.phoneNumber = phoneNumber;
        this.dob = dob;
        this.createdDate = createdDate;
        this.modifiedDate = modifiedDate;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public String getGender() {
        return gender;
    }

    public void setGender(String gender) {
        this.gender = gender;
    }
```

```java
    public String getEmailAddress() {
        return emailAddress;
    }

    public void setEmailAddress(String emailAddress) {
        this.emailAddress = emailAddress;
    }

    public String getPhoneNumber() {
        return phoneNumber;
    }

    public void setPhoneNumber(String phoneNumber) {
        this.phoneNumber = phoneNumber;
    }

    public Date getDob() {
        return dob;
    }

    public void setDob(Date dob) {
        this.dob = dob;
    }

    public Date getCreatedDate() {
        return createdDate;
    }

    public void setCreatedDate(Date createdDate) {
        this.createdDate = createdDate;
    }

    public Date getModifiedDate() {
        return modifiedDate;
    }

    public void setModifiedDate(Date modifiedDate) {
        this.modifiedDate = modifiedDate;
    }

    public Address getAddress() {
        return address;
    }

    public void setAddress(Address address) {
        this.address = address;
    }

    @Override
    public String toString() {
        return "Employee [id=" + id + ", firstName=" + firstName + ", lastName=" + lastName +
", gender=" + gender
                + ", emailAddress=" + emailAddress + ", phoneNumber=" + phoneNumber + ",
dob=" + dob + ", createdDate="
                + createdDate + ", modifiedDate=" + modifiedDate + "]";
    }

}
```

## 5. *Tạo package xử lý lỗi: iuh.fit.se.exceptions*

a. Throw lỗi trong trường hợp không tìm thấy dữ liệu

*ItemNotFoundException.java*

```
package iuh.fit.se.exceptions;

import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ResponseStatus;

@ResponseStatus(value = HttpStatus.NOT_FOUND)
public class ItemNotFoundException extends RuntimeException {
    public ItemNotFoundException(String message) {
        super(message);
    }
}
```

b. Throw lỗi trong trường hợp có lỗi khi thực hiện Validate

*ValidationException.java*

```
package iuh.fit.se.exceptions;

import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ResponseStatus;

import java.util.Map;

@ResponseStatus(value = HttpStatus.BAD_REQUEST)
public class ValidationException extends RuntimeException {
    private String message;
    private Map<String, Object> errors;

    public ValidationException(String message, Map<String, Object> errors) {
        super();
        this.message = message;
        this.errors = errors;
    }

    public String getMessage() {
        return message;
    }

    public void setMessage(String message) {
        this.message = message;
    }

    public Map<String, Object> getErrors() {
        return errors;
    }

    public void setErrors(Map<String, Object> errors) {
        this.errors = errors;
    }
}
```

c. Catch lỗi Exception

*GlobalExceptionHandler.java*

```java
package iuh.fit.se.exceptions;

import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.RestControllerAdvice;
import org.springframework.web.servlet.mvc.method.annotation.ResponseEntityExceptionHandler;

import java.util.LinkedHashMap;
import java.util.Map;

@RestControllerAdvice
public class GlobalExceptionHandler extends ResponseEntityExceptionHandler {
    @ExceptionHandler(ItemNotFoundException.class)
    public ResponseEntity<Map<String, Object>> userNotFoundException(ItemNotFoundException ex)
{
        Map<String, Object> errors = new LinkedHashMap<String, Object>();
        errors.put("status", HttpStatus.NOT_FOUND.value());
        errors.put("message", ex.getMessage());
        return new ResponseEntity<Map<String, Object>>(errors, HttpStatus.NOT_FOUND);
    }

    @ExceptionHandler(ValidationException.class)
    public ResponseEntity<Map<String, Object>> validationException(ValidationException ex) {
        Map<String, Object> errors = new LinkedHashMap<String, Object>();
        errors.put("status", HttpStatus.BAD_REQUEST.value());
        errors.put("errors", ex.getErrors());
        errors.put("message", ex.getMessage());
        return new ResponseEntity<Map<String, Object>>(errors, HttpStatus.BAD_REQUEST);
    }

    @ExceptionHandler(Exception.class)
    public ResponseEntity<Map<String, Object>> globalExceptionHandler(Exception ex) {
        Map<String, Object> errors = new LinkedHashMap<String, Object>();
        errors.put("status", HttpStatus.INTERNAL_SERVER_ERROR.value());
        errors.put("message", ex.getMessage());
        return new ResponseEntity<Map<String, Object>>(errors,
HttpStatus.INTERNAL_SERVER_ERROR);
    }
}
```

## 6. *Tạo package: iuh.fit.se.repositories*

```java
package iuh.fit.se.repositories;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.rest.core.annotation.RepositoryRestResource;
import iuh.fit.se.entities.Address;

@RepositoryRestResource
    public interface AddressRepository extends JpaRepository<Address, Integer>{
}
```

```java
package iuh.fit.se.repositories;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.rest.core.annotation.RepositoryRestResource;

import iuh.fit.se.entities.Employee;

@RepositoryRestResource
    public interface EmployeeRepository extends JpaRepository<Employee, Integer> {
}
```

## 7. *Tạo package: iuh.fit.se.services*

```java
package iuh.fit.se.services;

import iuh.fit.se.entities.Address;

public interface AddressService {
    public Address save(Address address);
}
```

```java
package iuh.fit.se.services;

import java.util.List;

import org.springframework.data.domain.Page;

import iuh.fit.se.entities.Employee;

public interface EmployeeService {

    public Employee findById(int id);
    public List<Employee> findAll();
    public Page<Employee> findAllWithPaging(int pageNo, int pageSize, String sortBy, String
sortDirection);
    public Employee save(Employee employee);
    public Employee update(int id, Employee employee);
    public int delete(int id);
}
```

## 8. *Tạo package implement: iuh.fit.se.services.impl*

```java
package iuh.fit.se.services.impl;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import iuh.fit.se.entities.Address;
import iuh.fit.se.repositories.AddressRepository;
import iuh.fit.se.services.AddressService;

@Service
public class AddressServiceImpl implements AddressService{
    private AddressRepository addressRepository;

    @Autowired
    public AddressServiceImpl(AddressRepository addressRepository) {
        this.addressRepository = addressRepository;
    }

    @Override
    public Address save(Address address) {
        return this.addressRepository.save(address);
    }
}
```

```java
package iuh.fit.se.services.impl;

import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;
import java.util.Set;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
```

```java
import iuh.fit.se.entities.Address;
import iuh.fit.se.entities.Employee;
import iuh.fit.se.exceptions.ItemNotFoundException;
import iuh.fit.se.exceptions.ValidationException;
import iuh.fit.se.repositories.EmployeeRepository;
import iuh.fit.se.services.EmployeeService;
import jakarta.validation.ConstraintViolation;
import jakarta.validation.Validation;
import jakarta.validation.Validator;
import jakarta.validation.ValidatorFactory;

@Service
public class EmployeeServiceImpl implements EmployeeService{
    private EmployeeRepository employeeRepository;

    @Autowired
    public EmployeeServiceImpl(EmployeeRepository employeeRepository) {
        this.employeeRepository = employeeRepository;
    }

    @Override
    public Employee findById(int id) {
        return employeeRepository.findById(id)
                .orElseThrow(()-> new ItemNotFoundException("Can not find Employee with id: "
+ id));
    }

    @Override
    public List<Employee> findAll() {
        return employeeRepository.findAll();
    }

    @Override
    public Page<Employee> findAllWithPaging(int pageNo, int pageSize, String sortBy, String
sortDirection) {
        Sort sort = sortDirection.equalsIgnoreCase(Sort.Direction.ASC.name()) ?
Sort.by(sortBy).ascending()
                : Sort.by(sortBy).descending();

        Pageable pageable = PageRequest.of(pageNo, pageSize, sort);
        return employeeRepository.findAll(pageable);
    }

    @Transactional
    @Override
    public Employee save(Employee employee) {

        ValidatorFactory factory = Validation.buildDefaultValidatorFactory();
        Validator validator = factory.getValidator();
        Set<ConstraintViolation<Employee>> violations = validator.validate(employee);

        if(!violations.isEmpty()) {
            Map<String, Object> errors = new LinkedHashMap<String, Object>();
            violations.forEach(violation -> {
                errors.put(violation.getPropertyPath().toString(), violation.getMessage());
            });

            throw new ValidationException("An error occurred while adding the employee",
errors);
        }
        else {
            employeeRepository.save(employee);
        }

        return employee;
    }

    @Override
```

```
    public Employee update(int id, Employee employee) {
        this.findById(id);

        ValidatorFactory factory = Validation.buildDefaultValidatorFactory();
        Validator validator = factory.getValidator();
        Set<ConstraintViolation<Employee>> violations = validator.validate(employee);

        if(!violations.isEmpty()) {
            Map<String, Object> errors = new LinkedHashMap<String, Object>();
            violations.forEach(violation -> {
                errors.put(violation.getPropertyPath().toString(), violation.getMessage());
            });

            throw new ValidationException("An error occurred while adding the employee",
errors);
        }
        else {
            employee.setId(id);
            employeeRepository.save(employee);
        }

        return employee;
    }

    @Override
    public int delete(int id) {
        Employee employee = this.findById(id);
        employeeRepository.delete(employee);
        return id;
    }
}
```

## 9. *Tạo package: iuh.fit.se.controllers*

```
package iuh.fit.se.controllers;

import java.util.LinkedHashMap;
import java.util.Map;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.rest.webmvc.RepositoryRestController;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import iuh.fit.se.entities.Employee;
import iuh.fit.se.services.EmployeeService;

@RepositoryRestController
public class EmployeeController {

    private EmployeeService employeeService;

    @Autowired
    public EmployeeController(EmployeeService employeeService) {
        this.employeeService = employeeService;
    }

    @GetMapping("/employees/{id}")
    public ResponseEntity<Employee> getEmployeeById(@PathVariable int id) {
        return ResponseEntity.status(HttpStatus.OK).body(employeeService.findById(id));
    }

    @PostMapping("/employees")
```

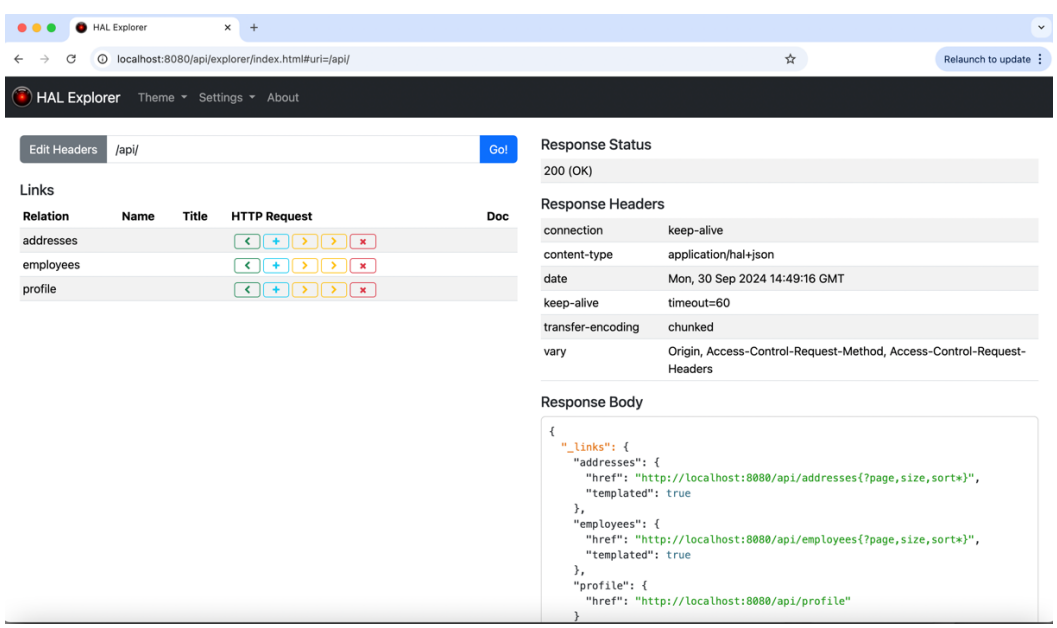
```
    public ResponseEntity<Employee> saveEmployee(@RequestBody Employee employee) {
        return ResponseEntity.status(HttpStatus.OK).body(employeeService.save(employee));
    }

    @PutMapping("/employees/{id}")
    public ResponseEntity<Employee> updateEmployee(@PathVariable int id, @RequestBody
Employee employee) {
        Map<String, Object> response = new LinkedHashMap<String, Object>();
        return ResponseEntity.status(HttpStatus.OK).body(employeeService.update(id,
employee));
    }

    @DeleteMapping("/employees/{id}")
    public ResponseEntity<Integer> deleteEmployee(@PathVariable int id) {
        return ResponseEntity.status(HttpStatus.OK).body(employeeService.delete(id));
    }

    @GetMapping("/employees")
    public ResponseEntity<Page<Employee>> getEmployees(
            @RequestParam(defaultValue = "1", required = false) int pageNo,
            @RequestParam(defaultValue = "2", required = false) int pageSize,
            @RequestParam(defaultValue = "id", required = false) String sortBy,
            @RequestParam(defaultValue = "ASC", required = false) String sortDirection) {

        Page<Employee> page = employeeService.findAllWithPaging(pageNo, pageSize, sortBy,
sortDirection);
        return ResponseEntity.status(HttpStatus.OK).body(page);
    }
}
```

## 10. *Config path api:*

***application.properties***

```
# Setting API
spring.data.rest.base-path=/api
```

## 11. *Access HAL explorer:* http://locahost:8080/api



## 12. *Springdoc – openapi*

- ***Dependency***

```
<!--https://mvnrepository.com/artifact/org.springdoc/
springdoc-openapi-starter-webmvc-ui -->
<dependency>
    <groupId>org.springdoc</groupId>
    <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
    <version>2.6.0</version>
</dependency>
```

- ***Config springdoc:***

```
package iuh.fit.se.configs;

import java.util.List;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import io.swagger.v3.oas.models.OpenAPI;
import io.swagger.v3.oas.models.info.Info;
import io.swagger.v3.oas.models.servers.Server;

@Configuration
public class OpenAPIConfiguration {
    @Bean
    OpenAPI defineOpenApi() {
       Server server = new Server();
       server.setUrl("http://localhost:8080");
       server.setDescription("Employee Management REST API Documentation");

       Info information = new Info()
             .title("Employee Management REST API Documentation")
             .version("1.0")
             .description("This API exposes endpoints to manage employees.");

       return new OpenAPI().info(information).servers(List.of(server));
    }
}
```

- ***Add setting springdoc at application.properties***

```
# Paths to include
springdoc.pathsToMatch=/**
springdoc.paths-to-exclude=/api/profile/**
springdoc.swagger-ui.operationsSorter=method
```

- ***Access URL: http://localhost:8080/swagger-ui/index.html***

## 13. *Config logger để ghi log file*

### *src/main/resources/logback-spring.xml*

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

<configuration>
        <include resource="org/springframework/boot/logging/logback/defaults.xml"/>
        <include resource="org/springframework/boot/logging/logback/console-appender.xml" />
        <include resource="org/springframework/boot/logging/logback/file-appender.xml" />
        <root level="INFO">
            <appender-ref ref="CONSOLE" />
            <appender-ref ref="FILE" />
        </root>
</configuration>
```

### *application.properties*

```
# Logging
logging.level.org.springframework.web=debug
logging.level.org.hibernate=error
logging.file.name=logs/myapplication.log
logging.config=classpath:logback-spring.xml
```

### *Cách ghi log file:*

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

private final static Logger logger =
LoggerFactory.getLogger(EmployeeController.class.getName());

logger.info("");
logger.error("");
logger.debug("");
logger.trace("");
logger.warn("");
```

# REST API

1. ***Core principals (Nguyên tắc cốt lõi):***
    a. Client – Server: tách biệt giữa máy khách và máy chủ, mỗi bên sẽ đảm nhận một vai trò cụ thể.
    b. Stateless: các request độc lập, không phụ thuộc vào trạng thái của máy chủ (Server).
    c. Cachecability: lưu trữ tạm thời để tăng tốc độ phản hồi (response)
    d. Layered System: hỗ trợ các lớp trung gian để tăng cường bảo mật và mở rộng.
    e. Code on demand: máy chủ có thể cung cấp mã để cải thiện chức năng phía máy khách (Client).
    f. Uniform Interface: tương tác được chuẩn hoá giữa các thành phần.
2. ***HTTP methods:***
    a. GET: lấy dữ liệu từ resource
    b. POST: tạo mới resource
    c. PUT: cập nhật toàn bộ resource
    d. PATCH: cập nhật 1 phần resource
    e. DELETE: xoá resource
    f. HEAD: lấy thông tin tiêu đề mà không có body
    g. OPTIONS: kiển tra các tuỳ chọn khả dụng của resource
3. ***Status codes:***
    a. 2xx: thành công
        i. 200 – OK: thành công
        ii. 201 – Created: tạo resource thành công
    b. 3xx: chuyển hướng
        i. 301 – Moved Permanently: resource được chuyển hướng đến URI mới
    c. 4xx: lỗi từ Client
        i. 401 – Unauthorized: yêu cầu xác thực
        ii. 403 – Forbidden: không có quyền truy cập
        iii. 404 – Not Found: resource không tồn tại
    d. 5xx: lỗi từ Server

        i. 500 – Internal Server Error

4. ***Security:***
    a. Authentication: JWT, Oauth 2.0, …
    b. Authorization: phân quyền
    c. HTTPS: mã hoá TLS/SSL để bảo vệ dữ liệu
    d. Rate Limiting: giới hạn số request
    e. CORS: cấu hình để kiểm soát quyền truy cập
    f. Security Headers: ví dụ Content-Security-Policy
5. ***Resource Naming:***
    a. Danh từ (nouns) và là số nhiều (plurals):
        i. users
        ii. accounts
        iii. customers
        iv. products
        v. …
    b. Sử dụng dấu gạch nối (hyphens):
        i. product-categories
    c. Dùng chữ thường (lowercase)
6. ***Best practices:***
    a. Versioning (phiên bản): sử dụng version trong URI
        i. /v1/users
        ii. /v2/users
    b. Filtering: lọc data bằng tham số
        i. /users?status=1
    c. Sorting: sắp xếp bằng tham số
        i. /users?sort=name
    d. Pagintion: phân trang bằng tham số
        i. /users?page=1&limit=20
    e. Error handling: cung cấp rõ ràng status code và error message
    f. Documentation: tạo tài liệu API đầy đủ, VD sử dụng Swagger …
    g. Caching: sử dụng cache để cải thiện hiệu suất