Spring BOOT

TP : Validation des champs et gestion des exception avec Spring Boot, RestController, Bean Validation, Spring Data JPA et H2.

SOMMAIRE

I- Objectifs :

- ✓ Valider les champs d'un message Rest avec l'API Bean Validation ;
- ✓ Développer vos propres règles de validation ;
- ✓ Gestion des exceptions avec l'annotation ***@ControllerAdvice***.

II- Outils utilisés :

Dans ce TP, nous allons utiliser les outils suivants :

- ✓ Eclipse avec le plugin Maven ;
- ✓ JDK 1.8 ;
- ✓ Connection à Internet pour permettre à Maven de télécharger les dépendances nécessaires (Spring Boot 2.2.0, …) ;
- ✓ POSTMAN ou un autre outil pour tester les méthodes POST, PUT et DELETE ;

III- Développement de l'application

1. pom.xml

- Votre fichier pom.xml devrait être le suivant :

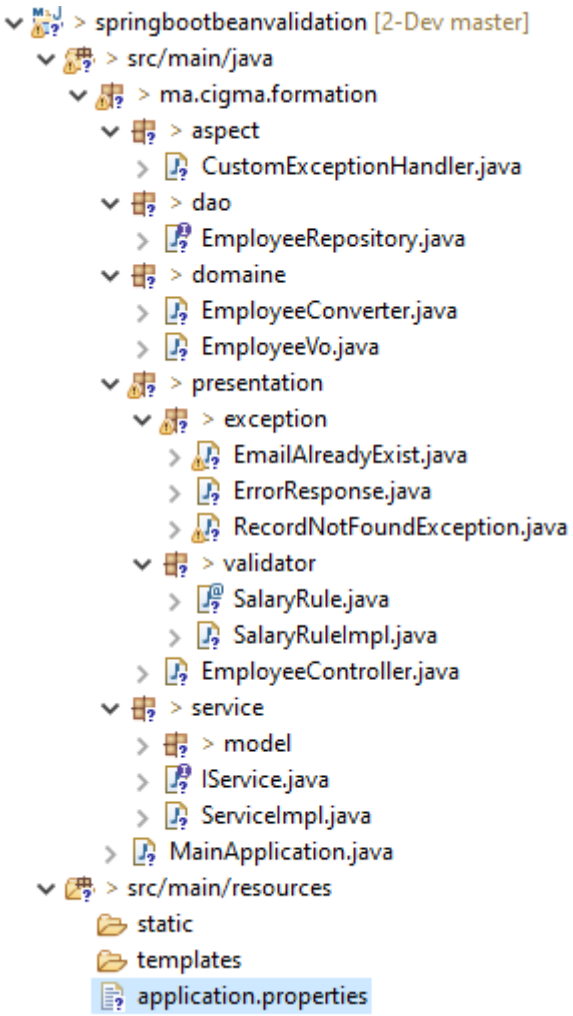
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<parent>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-parent</artifactId>
		<version>2.6.1</version>
		<relativePath />
	</parent>
	<groupId>ma.cigma.formation</groupId>
	<artifactId>springbootbeanvalidation</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<name>springbootexample</name>
	<description>L'utilisation de l'API Bean Validation</description>

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

	<dependencies>
		<dependency>
			<groupId>org.springframework.boot</groupId>
```

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

            <dependency>
                <groupId>org.projectlombok</groupId>
                <artifactId>lombok</artifactId>
                <optional>true</optional>
            </dependency>

            <!--
https://mvnrepository.com/artifact/com.fasterxml.jackson.dataformat/jackson-dataformat-xml -
->
            <dependency>
                <groupId>com.fasterxml.jackson.dataformat</groupId>
                <artifactId>jackson-dataformat-xml</artifactId>
            </dependency>

            <!-- https://mvnrepository.com/artifact/com.h2database/h2 -->
            <dependency>
                <groupId>com.h2database</groupId>
                <artifactId>h2</artifactId>
                <scope>runtime</scope>
            </dependency>

            <!-- https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-
starter-data-jpa -->
            <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-data-jpa</artifactId>
            </dependency>

            <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-validation</artifactId>
            </dependency>


            <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-test</artifactId>
                <scope>test</scope>
                <exclusions>
                    <exclusion>
                        <groupId>org.junit.vintage</groupId>
                        <artifactId>junit-vintage-engine</artifactId>
                    </exclusion>
                </exclusions>
```

```
            </dependency>
        </dependencies>

        <build>
            <plugins>
                <plugin>
                    <groupId>org.springframework.boot</groupId>
                    <artifactId>spring-boot-maven-plugin</artifactId>
                </plugin>
            </plugins>
        </build>

</project>
```

2. L'arborescence de votre projet

Votre projet devrait avoir l'arborescence suivante :



3. application.properties

Configuer votre fichier application.properties comme suit :

```
spring.datasource.url=jdbc:h2:mem:testdb
#spring.datasource.url = jdbc:h2:file:C:/sauvegarde/data
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
spring.jpa.hibernate.ddl-auto=update
# Enabling H2 Console
spring.h2.console.enabled=true
 # Custom H2 Console URL
spring.h2.console.path=/h2
```

4. Le modèle

Créer la classe ***Employee*** suivante :

```
package ma.cigma.formation.service.model;

import java.io.Serializable;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@AllArgsConstructor
@NoArgsConstructor
@Entity
public class Employee implements Serializable {
	private static final long serialVersionUID = 5448552240001397099L;
	@Id
	@GeneratedValue
	private Long id;
	private String firstName;
	private String lastName;
	private Double salary;
	private String email;
}
```

5. La couche dao

Créer l'interface ***EmployeeRepository*** suivante :

```
package ma.cigma.formation.dao;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;

import ma.cigma.formation.service.model.Employee;

public interface EmployeeRepository extends JpaRepository<Employee, Long> {
	List<Employee> findByEmail(String mail);
}
```

6. La couche domaine (le value object)

Créer la classe ***EmployeeVo*** suivante :

```
package ma.cigma.formation.domaine;

import javax.validation.constraints.Email;
import javax.validation.constraints.NotEmpty;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class EmployeeVo {
	@NotNull(message = "ID must not be empty")
	private Long id;
	@NotEmpty(message = "First name must not be empty")
	@Size(min = 10, max = 30, message = "message size must be between 10 et 30")
	private String firstName;
	@NotEmpty(message = "Last name must not be empty")
	private String lastName;
	@NotNull(message = "Salary must not be empty")
	private Double salary;
```

```
    @NotNull(message = "email must not be empty")
    @Email(message = "email should be a valid email")
    private String email;
}
```

- Observer les annotations ***@NotNull***, ***@NotEmpty***, ***@Size***, ***@Email***. Ces dernières concernent l'API Bean Validation (Spring Boot utilise ***Hibernate Validator*** comme implémentation de cette API).
- Vous pouvez créer vos propres règles de gestion. Dans l'exemple ci-dessous, nous allons implémenter la règle suivante : « Le salaire de l'employé doit être compris entre 1000,00 et 10 000,00 ». Pour se faire voici les étapes à suivre :

1- Créer l'annotation ***@SalaryRule*** suivante :

```
package ma.cigma.formation.presentation.validator;

import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

import javax.validation.Constraint;
import javax.validation.Payload;

@Target(ElementType.FIELD)
@Retention(RetentionPolicy.RUNTIME)
@Constraint(validatedBy = {SalaryRuleImpl.class })
public @interface SalaryRule {
        String message() default "The salary must be between 1000 MAD and 10000 MAD";
        Class<?>[] groups() default {};
        Class<? extends Payload>[] payload() default {};
}
```

2- Créer la classe ***SalaryRuleImpl*** suivante :

```
package ma.cigma.formation.presentation.validator;

import javax.validation.ConstraintValidator;
import javax.validation.ConstraintValidatorContext;

public class SalaryRuleImpl implements ConstraintValidator<SalaryRule, Double> {

        @Override
        public boolean isValid(Double salary, ConstraintValidatorContext context) {
                if (salary == null)
                        return false;
```

```
                if (salary < 1000 || salary > 10000)
                        return false;
                return true;
        }
}
```

Annoter le champs salary dans la classe EmployeeVO par @SalaryRule comme suit :

```
package ma.cigma.formation.domaine;

import javax.validation.constraints.Email;
import javax.validation.constraints.NotEmpty;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import ma.cigma.formation.presentation.validator.SalaryRule;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class EmployeeVo {
        @NotNull(message = "ID must not be empty")
        private Long id;
        @NotEmpty(message = "First name must not be empty")
        @Size(min = 10, max = 30, message = "message size must be between 10 et 30")
        private String firstName;
        @NotEmpty(message = "Last name must not be empty")
        private String lastName;
        @NotNull(message = "Salary must not be empty")
        @SalaryRule(message = "Le salaire doit être entre 1000 MAD et 10 000 MAD")
        private Double salary;

        @NotNull(message = "email must not be empty")
        @Email(message = "email should be a valid email")
        private String email;
}
```

Créer la classe ***EmployeeConverter*** suivante :

```
package ma.cigma.formation.domaine;

import java.util.ArrayList;
```

```
import java.util.List;

import ma.cigma.formation.service.model.Employee;

public class EmployeeConverter {

	public static EmployeeVo toVo(Employee bo) {
		return new EmployeeVo(bo.getId(), bo.getFirstName(), bo.getLastName(),
bo.getSalary(), bo.getEmail());
	}

	public static Employee toBo(EmployeeVo vo) {
		return new Employee(vo.getId(), vo.getFirstName(), vo.getLastName(),
vo.getSalary(), vo.getEmail());
	}

	public static List<EmployeeVo> toVoList(List<Employee> bos) {
		List<EmployeeVo> result = new ArrayList<EmployeeVo>();
		for (Employee employee : bos) {
			result.add(toVo(employee));
		}
		return result;

	}

	public static List<Employee> toBoList(List<EmployeeVo> vos) {
		List<Employee> result = new ArrayList<Employee>();
		for (EmployeeVo employeeVo : vos) {
			result.add(toBo(employeeVo));
		}
		return result;
	}
}
```

7. La couche service

Créer l'interface ***IService*** suivante :

```
package ma.cigma.formation.service;

import java.util.List;

import ma.cigma.formation.domaine.EmployeeVo;

public interface IService {
```

```
        List<EmployeeVo> getAll();
        EmployeeVo getById(Long id);
        void save(EmployeeVo empl);
        void remove(Long id);
        boolean isEmailExist(String email);
        List<EmployeeVo> getAll(int pageId,int size);
        List<EmployeeVo> sortBy(String fieldName);
}
```

Créer la classe ***ServiceImpl*** suivante :

```
package ma.cigma.formation.service;

import java.util.List;

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

import ma.cigma.formation.dao.EmployeeRepository;
import ma.cigma.formation.domaine.EmployeeConverter;
import ma.cigma.formation.domaine.EmployeeVo;
import ma.cigma.formation.service.model.Employee;

@Service
@Transactional
public class ServiceImpl implements IService {

        @Autowired
        private EmployeeRepository employeeRepository;

        @Override
        public List<EmployeeVo> getAll() {
                List<Employee> listBos = employeeRepository.findAll();
                return EmployeeConverter.toVoList(listBos);
        }

        @Override
        public EmployeeVo getById(Long id) {
                if (!employeeRepository.existsById(id))
                        return null;
                Employee bo = employeeRepository.getById(id);
                return EmployeeConverter.toVo(bo);
```

```
	}

	@Override
	public void save(EmployeeVo empl) {
		employeeRepository.save(EmployeeConverter.toBo(empl));
	}

	@Override
	public void remove(Long id) {
		employeeRepository.deleteById(id);
	}

	@Override
	public boolean isEmailExist(String email) {
		List<Employee> list = employeeRepository.findByEmail(email);
		if (list != null && !list.isEmpty())
			return true;
		return false;
	}

	@Override
	public List<EmployeeVo> getAll(int pageId, int size) {
		Page<Employee> listBos = employeeRepository.findAll(PageRequest.of(pageId, size));
		return EmployeeConverter.toVoList(listBos.getContent());
	}

	@Override
	public List<EmployeeVo> sortBy(String fieldName) {
		List<Employee> listBos = employeeRepository.findAll(Sort.by(Direction.DESC,
fieldName));
		return EmployeeConverter.toVoList(listBos);
	}
}
```

8. Le contrôlleur est les classes Exception

Créer la classe ***RecordNotFoundException*** suivante :

```
package ma.cigma.formation.presentation.exception;

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

@ResponseStatus(HttpStatus.NOT_FOUND)
public class RecordNotFoundException extends RuntimeException
```

```
{
        private static final long serialVersionUID = 7733586942874361333L;

        public RecordNotFoundException(String exception) {
      super(exception);
   }
}
```

Créer la classe ***ErrorResponse*** suivante :

```
package ma.cigma.formation.presentation.exception;

import java.util.List;

import javax.xml.bind.annotation.XmlRootElement;

import lombok.Getter;
import lombok.Setter;

@XmlRootElement(name = "error")
@Getter
@Setter
public class ErrorResponse
{
   public ErrorResponse(String message, List<String> details) {
      super();
      this.message = message;
      this.details = details;
   }
   private String message;
   private List<String> details;

}
```

Créer la classe ***EmailAlreadyExist*** suivante :

```
package ma.cigma.formation.presentation.exception;

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

```java
@ResponseStatus(HttpStatus.BAD_REQUEST)
public class EmailAlreadyExist extends RuntimeException{

	public EmailAlreadyExist(String exception) {
	    super(exception);
	  }
}
```

Créer la classe ***CustomExceptionHandler*** suivante :

```java
package ma.cigma.formation.aspect;

import java.util.ArrayList;
import java.util.List;

import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.validation.ObjectError;
import org.springframework.web.bind.MethodArgumentNotValidException;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.context.request.WebRequest;
import org.springframework.web.servlet.mvc.method.annotation.ResponseEntityExceptionHandler;

import ma.cigma.formation.presentation.exception.EmailAlreadyExist;
import ma.cigma.formation.presentation.exception.ErrorResponse;
import ma.cigma.formation.presentation.exception.RecordNotFoundException;

@SuppressWarnings({ "unchecked", "rawtypes" })
@ControllerAdvice
public class CustomExceptionHandler extends ResponseEntityExceptionHandler {

	@Override
	protected ResponseEntity<Object>
handleMethodArgumentNotValid(MethodArgumentNotValidException ex,
			HttpHeaders headers, HttpStatus status, WebRequest request) {
		List<String> details = new ArrayList<>();
		for (ObjectError error : ex.getBindingResult().getAllErrors()) {
			details.add(error.getDefaultMessage());
```

```
                }
                ErrorResponse error = new ErrorResponse("Validation Failed", details);
                return new ResponseEntity(error, HttpStatus.BAD_REQUEST);
        }

        @ExceptionHandler(Exception.class)
        public final ResponseEntity<Object> handleAllExceptions(Exception ex, WebRequest request)
{
                List<String> details = new ArrayList<>();
                details.add(ex.getLocalizedMessage());
                ErrorResponse error = new ErrorResponse("Server Error", details);
                return new ResponseEntity(error, HttpStatus.INTERNAL_SERVER_ERROR);
        }

        @ExceptionHandler(value = { RecordNotFoundException.class })
        public final ResponseEntity<Object>
handleUserNotFoundException(RecordNotFoundException ex, WebRequest request) {
                List<String> details = new ArrayList<>();
                details.add(ex.getLocalizedMessage());
                ErrorResponse error = new ErrorResponse("Record Not Found", details);
                return new ResponseEntity(error, HttpStatus.NOT_FOUND);
        }

        @ExceptionHandler(value = EmailAlreadyExist.class)
        public final ResponseEntity<Object> handleEmailExistAlreadyException(EmailAlreadyExist ex,
WebRequest request) {
                List<String> details = new ArrayList<>();
                details.add(ex.getLocalizedMessage());
                ErrorResponse error = new ErrorResponse("Record Not Found", details);
                return new ResponseEntity(error, HttpStatus.NOT_FOUND);
        }

}
```

Créer la classe ***EmployeeController*** suivante :

```
package ma.cigma.formation.presentation;

import java.util.List;

import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.DeleteMapping;
```

```
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import ma.cigma.formation.domaine.EmployeeVo;
import ma.cigma.formation.presentation.exception.EmailAlreadyExist;
import ma.cigma.formation.presentation.exception.RecordNotFoundException;
import ma.cigma.formation.service.IService;

@RestController
public class EmployeeController {
	@Autowired
	private IService service;

	@GetMapping(value = "/employees", produces = { MediaType.APPLICATION_JSON_VALUE,
MediaType.APPLICATION_XML_VALUE })
	public List<EmployeeVo> getAll() {
		return service.getAll();
	}

	@GetMapping(value = "/employees/{id}", produces = {
MediaType.APPLICATION_JSON_VALUE,
			MediaType.APPLICATION_XML_VALUE })
	public EmployeeVo getById(@PathVariable(value = "id") Long id) {
		EmployeeVo e = service.getById(id);
		if (e == null)
			throw new RecordNotFoundException("No record with id=" + id);
		return service.getById(id);
	}

	@PostMapping(value = "/employees", produces = { MediaType.APPLICATION_JSON_VALUE,
MediaType.APPLICATION_XML_VALUE })
	public ResponseEntity<String> create(@Valid @RequestBody EmployeeVo emp) {
		if (service.isEmailExist(emp.getEmail())) {
			throw new EmailAlreadyExist("Email already exist : " + emp.getEmail());
		}

		service.save(emp);
		return new ResponseEntity<String>("Employee created with success",
HttpStatus.CREATED);
	}

	@PutMapping(value = "/employees/{id}", produces = {
MediaType.APPLICATION_JSON_VALUE,
```

```
                        MediaType.APPLICATION_XML_VALUE })
        public ResponseEntity<String> update(@Valid @PathVariable(value = "id") Long id,
@RequestBody EmployeeVo emp) {
                EmployeeVo e = service.getById(id);
                if (e == null)
                        throw new RecordNotFoundException("No record with id=" + id);
                emp.setId(id);
                service.save(emp);
                return new ResponseEntity<String>("Employee updated with success",
HttpStatus.OK);
        }

        @DeleteMapping(value = "/employees/{id}", produces = {
MediaType.APPLICATION_JSON_VALUE,
                        MediaType.APPLICATION_XML_VALUE })
        public ResponseEntity<String> delete(@PathVariable(value = "id") Long id) {
                EmployeeVo e = service.getById(id);
                if (e == null)
                        throw new RecordNotFoundException("No record with id=" + id);
                service.remove(id);
                return new ResponseEntity<String>("Employee fremoved with success",
HttpStatus.OK);
        }

        @GetMapping(value = "/employees/sort/{fieldName}", produces = {
MediaType.APPLICATION_JSON_VALUE,
                        MediaType.APPLICATION_XML_VALUE })
        public List<EmployeeVo> getAll(@PathVariable String fieldName) {
                return service.sortBy(fieldName);
        }

        @GetMapping(value = "/employees/{pageId}/{size}", produces = {
MediaType.APPLICATION_JSON_VALUE,
                        MediaType.APPLICATION_XML_VALUE })
        public List<EmployeeVo> getAll(@PathVariable Integer pageId, @PathVariable Integer size) {
                return service.getAll(pageId, size);
        }
}
```

9. La classe de démarrage MainApplication

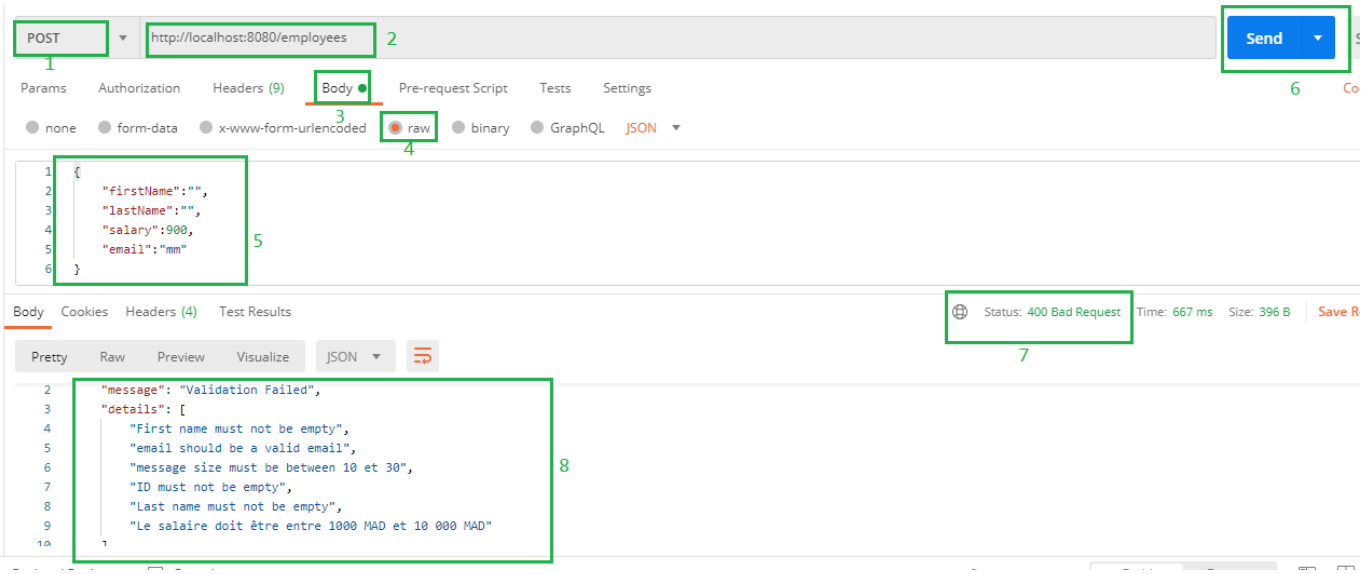
```
package ma.cigma.formation;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
```

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

import ma.cigma.formation.domaine.EmployeeVo;
import ma.cigma.formation.service.IService;

@SpringBootApplication
public class MainApplication implements CommandLineRunner {

	@Autowired
	private IService service;

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

	@Override
	public void run(String... args) throws Exception {
	service.save(new EmployeeVo(null, "Alami", "Mohamed", 12000d,
"alamimhamed@france.fr"));
	service.save(new EmployeeVo(null, "Jamali", "Ali", 5000d, "jamaliali@ss.com"));
	service.save(new EmployeeVo(null, "Amrani", "Bilal", 6000d, "amranibilal@yahoo.fr"));
	service.save(new EmployeeVo(null, "Kadiri", "Samir", 13000d, "kadirisamir@hotmail.com"));
	service.save(new EmployeeVo(null, "Choukri", "Abla", 12000d, "choukriabla@gmail.com"));
	}
}
```

IV- Les tests

1. **Test n°1 : Ajouter un employé dans les valeurs des champs ne respectent pas les règles de gestion définies**

❖ Lancer l'application, ensuite au niveau de POSTMAN suivre les étapes suivantes :



1- Choisir Post
2- Entrer le lien de votre service web : http://localhost:8080/employees
3- Cliquer sur Body
4- Cliquer sur raw
5- Entrer le message JSON suivant :

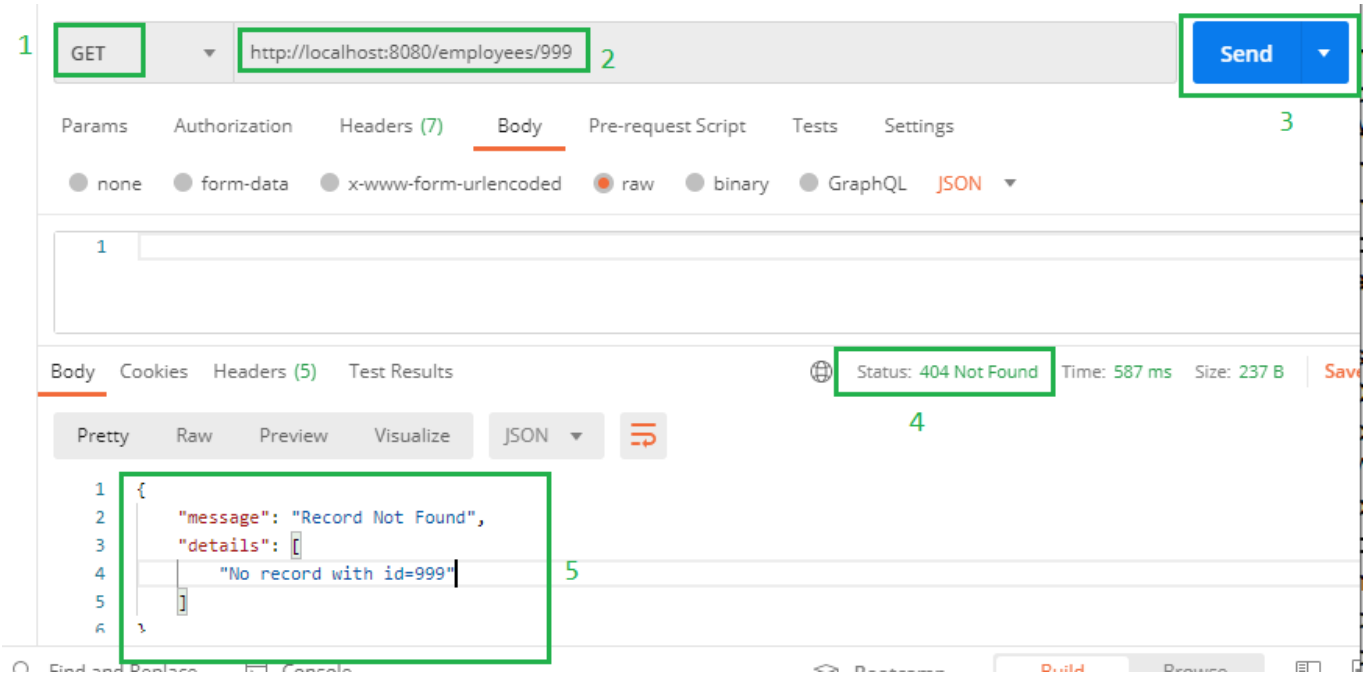
```
{
    "firstName":"",
    "lastName":"",
    "salary":900,
    "email":"mm"
}
```

6- Cliquer sur Send
7- Vérifier le code erreur 400 (Bad Request)
8- Vérifier le message qui est envoyé par le serveur :

```
{
    "message": "Validation Failed",
    "details": [
        "First name must not be empty",
        "email should be a valid email",
        "message size must be between 10 et 30",
        "ID must not be empty",
        "Last name must not be empty",
        "Le salaire doit être entre 1000 MAD et 10 000 MAD"
    ]
}
```

1. **Test n°2 : Chercher un employé dont l'id n'existe pas**

Suivre les étapes suivantes :



1- Choisir GET.
2- Entrer le lien : http://localhost:8080/employees/999.
3- Cliquer sur Send.
4- Vérifier le code erreur 404 Not Found qui est envoyé par le serveur.
5- Observer le message d'erreur envoyé par le serveur.