SPRING REST HANDS ON 4

Superset ID : 6384831

Name : Mohana Priya N

E-mail : mohanapriya.2205056@srec.ac.in

1) Hands on 1: Create RESTful Web Service to handle POST request of Country

Solution:

//Country.java

```
package com.example.demo.controller;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
@RestController
@RequestMapping("/countries")
public class CountryController {
    private static final Logger LOGGER = LoggerFactory.getLogger(CountryController.class);
    @PostMapping
    public void addCountry() {
        LOGGER.info("Start");
    }
}
```

2) Hands on 2: Read country data as a bean in RESTful Web Service

Solution:

//Country.java

```
package com.example.demo.model;
public class Country {
    private String code;
    private String name;
    public Country() {}
    public String getCode() { return code; }
    public void setCode(String code) { this.code = code; }
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
}
```

//CountryController.java

```
@PostMapping
public Country addCountry(@RequestBody Country country) {
    LOGGER.info("Start");
    LOGGER.info("Country received: code={}, name={}", country.getCode(), country.getName());
    return country;
}
```

3) Hands on 3: Validating country code

Solution:

//Country.java

```
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
public class Country {
    @NotNull
    @Size(min = 2, max = 2, message = "Country code should be 2 characters")
    private String code;
    private String name;
}
```

// CountryController.java

```
import javax.validation.*;
import java.util.*;
import org.springframework.http.HttpStatus;
import org.springframework.web.server.ResponseStatusException;
```

```
@PostMapping
public Country addCountry(@RequestBody Country country) {
    LOGGER.info("Start");
    ValidatorFactory factory = Validation.buildDefaultValidatorFactory();
    Validator validator = factory.getValidator();
    Set<ConstraintViolation<Country>> violations = validator.validate(country);
    List<String> errors = new ArrayList<>();
    for (ConstraintViolation<Country> violation : violations) {
        errors.add(violation.getMessage());
    }
    if (!errors.isEmpty()) {
        throw new ResponseStatusException(HttpStatus.BAD_REQUEST, errors.toString());
    }
    LOGGER.info("Country received: code={}, name={}", country.getCode(), country.getName());
    return country;
}
```

4) Hands on 4: Include global exception handler for validation errors

Solution:

// GlobalExceptionHandler.java

```
package com.example.demo.exception;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.http.*;
import org.springframework.web.bind.MethodArgumentNotValidException;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.context.request.WebRequest;
import org.springframework.web.servlet.mvc.method.annotation.ResponseEntityExceptionHandler;
import java.util.*;
import java.util.stream.Collectors;
```

```
@ControllerAdvice
public class GlobalExceptionHandler extends ResponseEntityExceptionHandler {
   private static final Logger LOGGER = LoggerFactory.getLogger(GlobalExceptionHandler.class);
   @Override
   protected ResponseEntity<Object>
handleMethodArgumentNotValid(MethodArgumentNotValidException ex,
                                                       HttpHeaders headers,
                                                       HttpStatus status,
                                                       WebRequest request) {
       LOGGER.info("Start handling validation error");
       Map<String, Object> body = new LinkedHashMap<>();
       body.put("timestamp", new Date());
       body.put("status", status.value());
       List<String> errors = ex.getBindingResult()
                           .getFieldErrors()
                           .stream()
                           .map(x -> x.getDefaultMessage())
                           .collect(Collectors.toList());
       body.put("errors", errors);
       LOGGER.info("End handling validation error");
       return new ResponseEntity<>(body, headers, status);
   }
}
```

// **CountryController.java**

```
import javax.validation.*;
import java.util.*;
import org.springframework.http.HttpStatus;
import org.springframework.web.server.ResponseStatusException;
@PostMapping
public Country addCountry(@RequestBody @Valid Country country) {
   LOGGER.info("Start");
```

```
    LOGGER.info("Country received: code={}, name={}", country.getCode(), country.getName());
    return country;
}
```

5) Hands on 5: Implement REST service for updating an employee

Solution:

(i) USING PUT:

//Employee.java

```
package com.example.demo.model;
import com.fasterxml.jackson.annotation.JsonFormat;
import javax.validation.constraints.*;
import java.util.Date;
import java.util.List;
public class Employee {
    @NotNull(message = "Id cannot be null")
    private Integer id;
    @NotNull
    @NotBlank
    @Size(min = 1, max = 30)
    private String name;
    @NotNull
    @Min(value = 0, message = "Salary should be zero or above")
    private Double salary;
    @NotNull
    private Boolean permanent;
    @JsonFormat(shape = JsonFormat.Shape.STRING, pattern = "dd/MM/yyyy")
    private Date dateOfBirth;
    @NotNull
    private Department department;
    @NotNull
    private List<Skill> skills;
}
```

//Department.java

```
package com.example.demo.model;
import javax.validation.constraints.*;
public class Department {
    @NotNull(message = "Department id cannot be null")
    private Integer id;
    @NotNull
    @NotBlank
    @Size(min = 1, max = 30)
    private String name;
}
```

//Skill.java

```
package com.example.demo.model;
import javax.validation.constraints.*;
public class Skill {
    @NotNull(message = "Skill id cannot be null")
    private Integer id;
    @NotNull
    @NotBlank
    @Size(min = 1, max = 30)
    private String name;
}
```

//EmployeeNotFoundException.java

```
package com.example.demo.exception;
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ResponseStatus;
@ResponseStatus(HttpStatus.NOT_FOUND)
public class EmployeeNotFoundException extends RuntimeException {
    public EmployeeNotFoundException(String message) {
        super(message);
    }
}
```

//EmployeeDao.java

```
package com.example.demo.dao;
import com.example.demo.exception.EmployeeNotFoundException;
import com.example.demo.model.Employee;
import org.springframework.stereotype.Repository;
import java.util.*;
@Repository
public class EmployeeDao {
    private static List<Employee> employeeList = new ArrayList<>();
    public void updateEmployee(Employee employee) {
        boolean found = false;
        for (int i = 0; i < employeeList.size(); i++) {
            if (employeeList.get(i).getId().equals(employee.getId())) {
                employeeList.set(i, employee);
                found = true;
                break;
            }
        }
        if (!found) {
            throw new EmployeeNotFoundException("Employee not found with id: " + employee.getId());
        }
    }
    public List<Employee> getAllEmployees() {
        return employeeList;
    }
}
```

//EmployeeService.java

```
package com.example.demo.service;
import com.example.demo.dao.EmployeeDao;
import com.example.demo.exception.EmployeeNotFoundException;
import com.example.demo.model.Employee;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
@Service
public class EmployeeService {
   @Autowired
   private EmployeeDao employeeDao;
   public void updateEmployee(Employee employee) throws EmployeeNotFoundException {
      employeeDao.updateEmployee(employee);
   }
}
```

//EmployeeController.java

```
package com.example.demo.controller;
import com.example.demo.exception.EmployeeNotFoundException;
import com.example.demo.model.Employee;
import com.example.demo.service.EmployeeService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
import javax.validation.Valid;
@RestController
@RequestMapping("/employees")
public class EmployeeController {
   @Autowired
   private EmployeeService employeeService;
   @PutMapping
   public void updateEmployee(@RequestBody @Valid Employee employee) throws
EmployeeNotFoundException {
        employeeService.updateEmployee(employee);
    }
}
```

(ii) JSON format error Handler

//GlobalExceptionHandler.java

```
package com.example.demo.exception;
import com.fasterxml.jackson.databind.exc.InvalidFormatException;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.http.*;
import org.springframework.http.converter.HttpMessageNotReadableException;
import org.springframework.web.bind.MethodArgumentNotValidException;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.context.request.WebRequest;
import org.springframework.web.servlet.mvc.method.annotation.ResponseEntityExceptionHandler;
import java.util.*;
import java.util.stream.Collectors;
@ControllerAdvice
public class GlobalExceptionHandler extends ResponseEntityExceptionHandler {
    private static final Logger LOGGER = LoggerFactory.getLogger(GlobalExceptionHandler.class);
    @Override
    protected ResponseEntity<Object>
handleMethodArgumentNotValid(MethodArgumentNotValidException ex,
                                                          HttpHeaders headers,
                                                          HttpStatus status,
                                                          WebRequest request) {
        LOGGER.info("Start - handleMethodArgumentNotValid");
        Map<String, Object> body = new LinkedHashMap<>();
        body.put("timestamp", new Date());
        body.put("status", status.value());
        List<String> errors = ex.getBindingResult()
                .getFieldErrors()
                .stream()
                .map(x -> x.getDefaultMessage())
                .collect(Collectors.toList());
        body.put("errors", errors);
        LOGGER.info("End - handleMethodArgumentNotValid");
        return new ResponseEntity<>(body, headers, status);
    }
    @Override
    protected ResponseEntity<Object>
handleHttpMessageNotReadable(HttpMessageNotReadableException ex,
                                                                  HttpHeaders headers,
                                                                  HttpStatus status,
                                                                  WebRequest request) {
        LOGGER.info("Start - handleHttpMessageNotReadable");
        Map<String, Object> body = new LinkedHashMap<>();
        body.put("timestamp", new Date());
        body.put("status", status.value());
        body.put("error", "Bad Request");
        if (ex.getCause() instanceof InvalidFormatException) {
            InvalidFormatException ife = (InvalidFormatException) ex.getCause();
            for (InvalidFormatException.Reference ref : ife.getPath()) {
                body.put("message", "Incorrect format for field '" + ref.getFieldName() + "'");
            }
        } else {
            body.put("message", "Malformed JSON or invalid input format");
        }
        LOGGER.info("End - handleHttpMessageNotReadable");
        return new ResponseEntity<>(body, headers, status);
    }
```

```
}
```

6) Hands on 6: Implement REST DELETE Service

Solution:

//EmployeeDao.java

```
package com.example.demo.dao;
import com.example.demo.exception.EmployeeNotFoundException;
import com.example.demo.model.Employee;
import org.springframework.stereotype.Repository;
import java.util.ArrayList;
import java.util.List;
@Repository
public class EmployeeDao {
    private static List<Employee> employeeList = new ArrayList<>();
    static {
    }
    public void deleteEmployeeById(Integer id) {
        boolean removed = employeeList.removeIf(employee -> employee.getId().equals(id));
        if (!removed) {
            throw new EmployeeNotFoundException("Employee not found with id: " + id);
        }
    }
    public List<Employee> getAllEmployees() {
        return employeeList;
    }
    public void addEmployee(Employee employee) {
        employeeList.add(employee);
    }
}
```

//EmployeeService.java

```
package com.example.demo.service;
```

```
import com.example.demo.dao.EmployeeDao;
import com.example.demo.exception.EmployeeNotFoundException;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
@Service
public class EmployeeService {
   @Autowired
   private EmployeeDao employeeDao;
   public void deleteEmployee(Integer id) throws EmployeeNotFoundException {
      employeeDao.deleteEmployeeById(id);
   }
}
```

//EmployeeController.java

```
package com.example.demo.controller;
import com.example.demo.exception.EmployeeNotFoundException;
import com.example.demo.service.EmployeeService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
@RestController
@RequestMapping("/employees")
public class EmployeeController {
   @Autowired
   private EmployeeService employeeService;
   @DeleteMapping("/{id}")
   public void deleteEmployee(@PathVariable Integer id) {
      employeeService.deleteEmployee(id);
   }
}
```

//EmployeeNotFoundException.java

```
package com.example.demo.exception;
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ResponseStatus;
@ResponseStatus(HttpStatus.NOT_FOUND)
public class EmployeeNotFoundException extends RuntimeException {
    public EmployeeNotFoundException(String message) {
        super(message);
    }
}
```