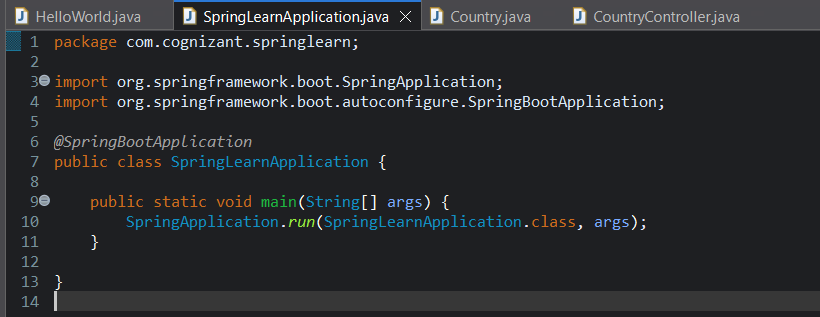
Name : Arshiya Tabassum A

Superset ID : 6424209

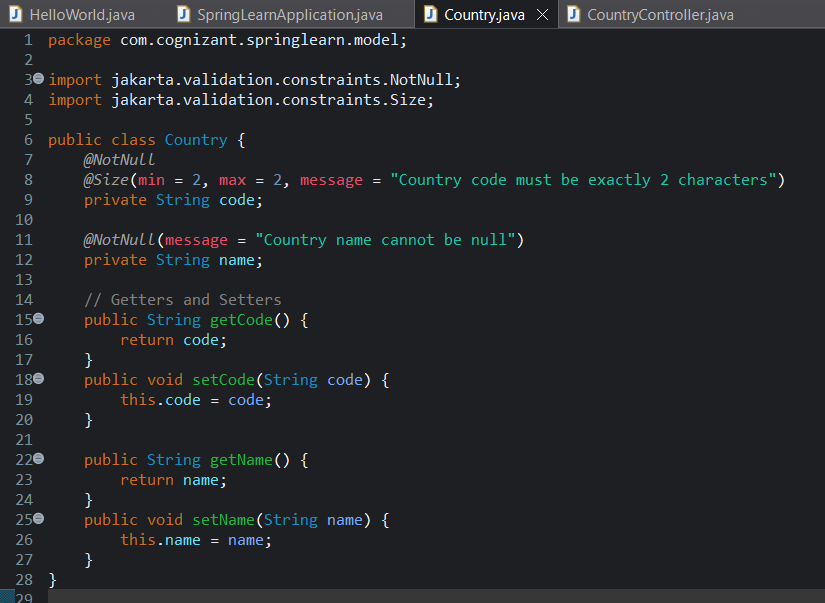
**WEEK 4**

**Significance of HTTP Method Types in RESTful Web Services   
Code :**

**Springlearnapplication.java**

****

**Country.java**

****

**Countrycontroller.java**

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.model.Country;

import jakarta.validation.Valid;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.web.bind.annotation.\*

import java.util.\*;

*@RestController*

*@RequestMapping*("/countries")

public class CountryController {

private static final Logger ***LOGGER*** = LoggerFactory.*getLogger*(CountryController.class)

private List<Country> countries = new ArrayList<>();

// GET - Retrieve all countries

*@GetMapping*

public List<Country> getAllCountries() {

***LOGGER***.info("Fetching all countries");

return countries;

}

// GET - Retrieve a specific country

*@GetMapping*("/{code}")

public Country getCountry(*@PathVariable* String code) {

return countries.stream()

.filter(c -> c.getCode().equalsIgnoreCase(code))

.findFirst()

.orElseThrow(() -> new RuntimeException("Country not found"));

}

// POST - Create a new country

*@PostMapping*

public Country addCountry(*@RequestBody* *@Valid* Country country) {

***LOGGER***.info("Adding new country");

countries.add(country);

return country;

}

// PUT - Update an existing country

*@PutMapping*

public Country updateCountry(*@RequestBody* *@Valid* Country updatedCountry) {

Country existing = getCountry(updatedCountry.getCode());

existing.setName(updatedCountry.getName());

return existing;

}

// DELETE - Delete a country by code

*@DeleteMapping*("/{code}")

public void deleteCountry(*@PathVariable* String code) {

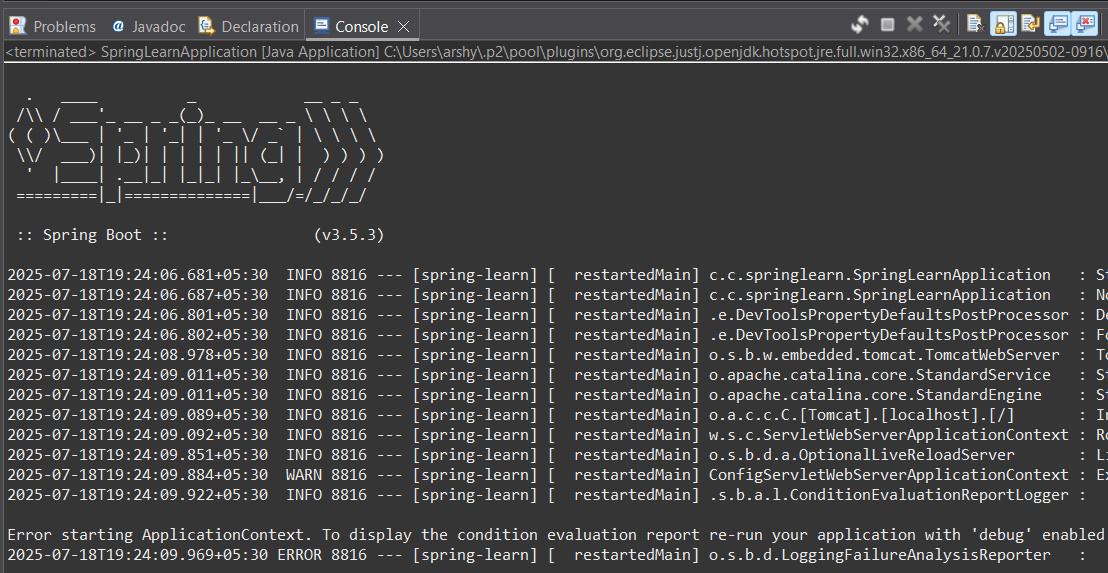
Country existing = getCountry(code);

countries.remove(existing);

}

}

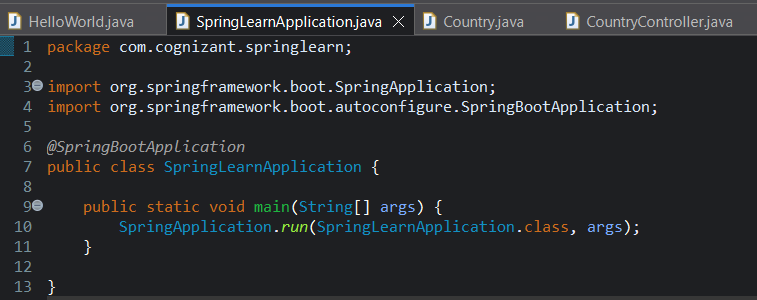
**Output :**

****

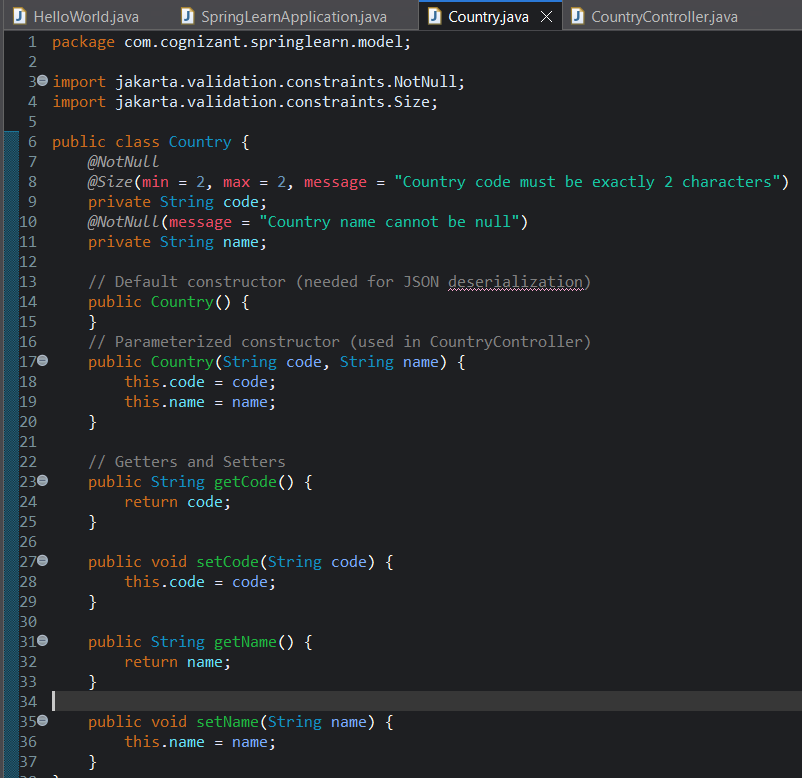
**RESTful Web Service resource naming guidelines**

**Code :**

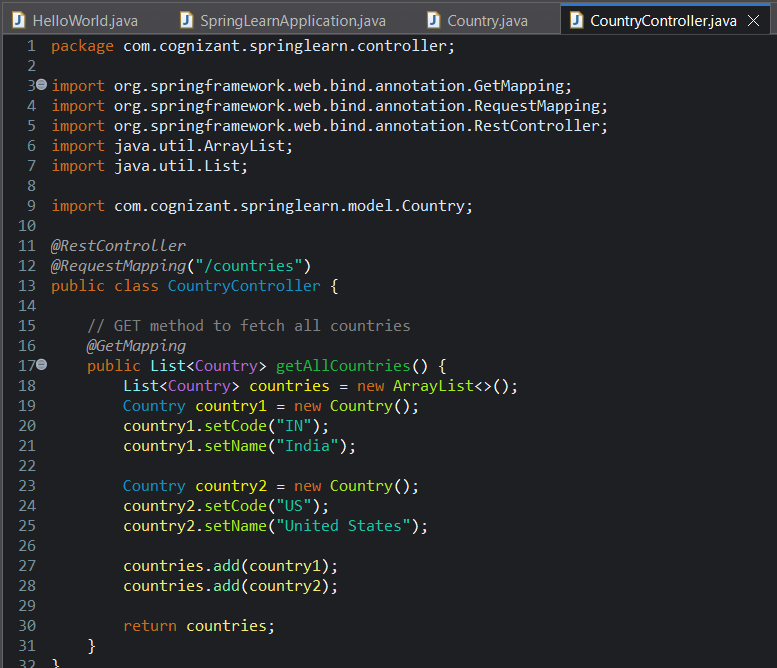
**Springlearnapplication.java**

****

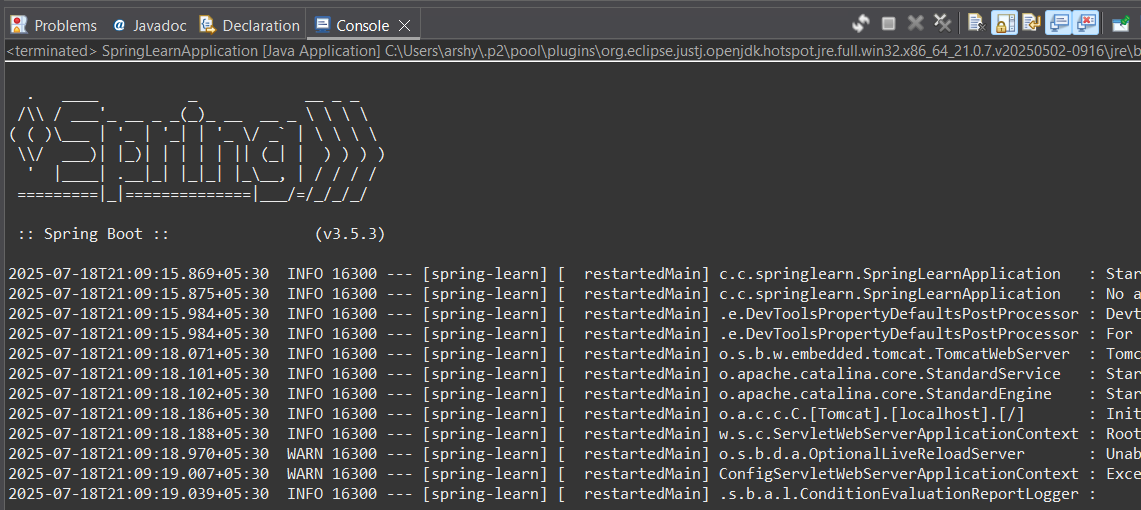
**Country.java**

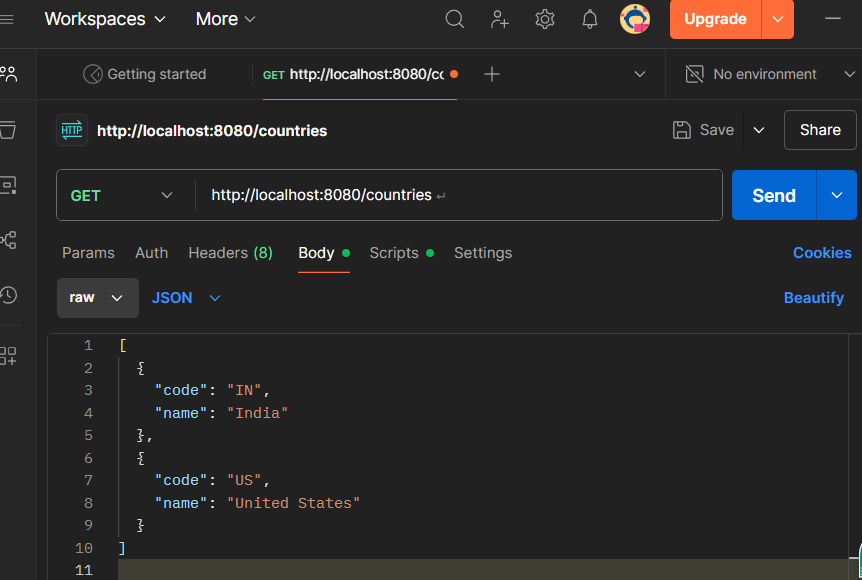
****

**Country.java**

****

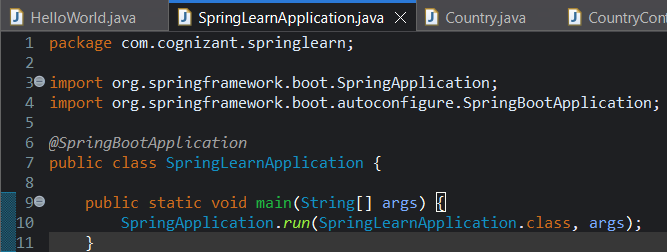
**Output :**

****

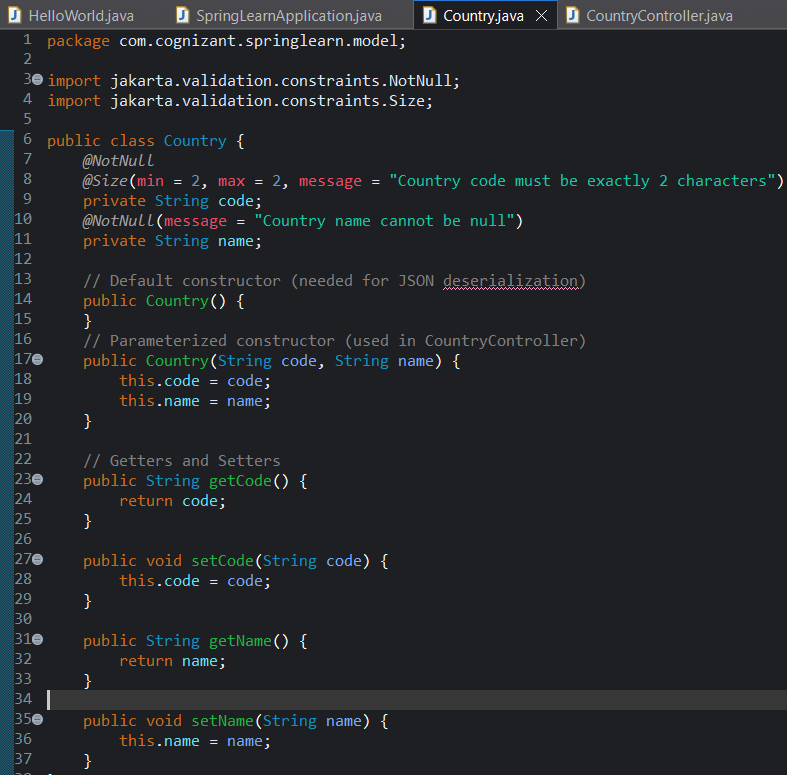
****

**Create RESTful Web Service to handle POST request of Country**

**Springlearnapplication.java**

****

**Country.java**

****

**CountryController.java**

package com.cognizant.springlearn.controller;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import java.util.ArrayList;

import java.util.List;

import com.cognizant.springlearn.model.Country;

*@RestController*

*@RequestMapping*("/countries")

public class CountryController {

// Logger for debugging

private static final Logger ***LOGGER*** = LoggerFactory.*getLogger*(CountryController.class);

// GET method to fetch all countries

*@GetMapping*

public List<Country> getAllCountries() {

List<Country> countries = new ArrayList<>();

Country country1 = new Country();

country1.setCode("IN");

country1.setName("India");

Country country2 = new Country();

country2.setCode("US");

country2.setName("United States");

countries.add(country1);

countries.add(country2);

return countries;

}

// POST method (Step 3 will use this)

*@PostMapping*

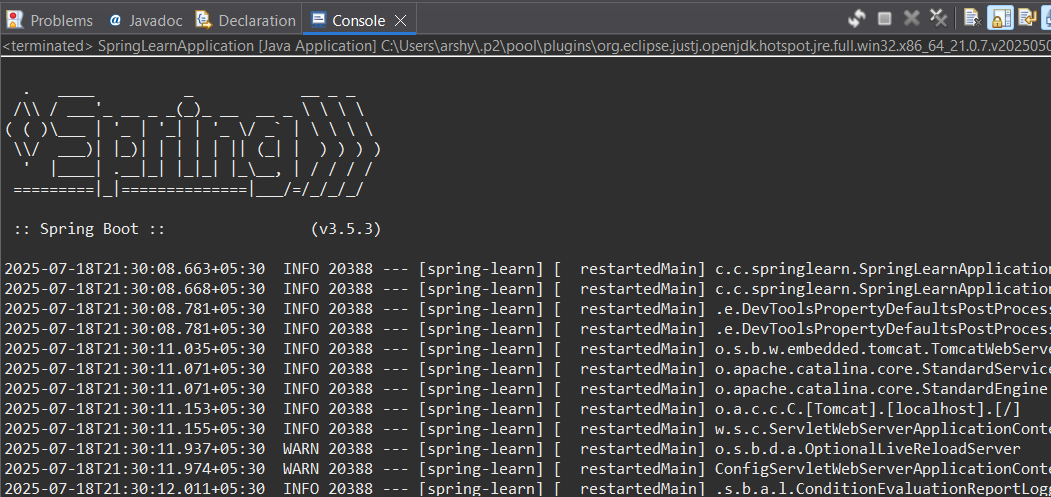
public void addCountry() {

***LOGGER*.info("Start"); // This will print "Start" in console when you call the POST API**

}

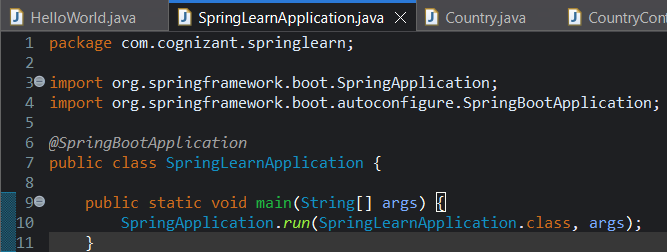
}

**Output :**

****

**Read country data as a bean in RESTful Web Service**

**Springlearnapplication.java**

****

**Country.java**

package com.cognizant.springlearn.model;

import jakarta.validation.constraints.NotNull;

import jakarta.validation.constraints.Size;

public class Country {

*@NotNull*

*@Size*(min = 2, max = 2, message = "Country code must be exactly 2 characters")

private String code;

*@NotNull*(message = "Country name cannot be null")

private String name;

// Default constructor (needed for JSON deserialization)

public Country() {

}

// Parameterized constructor (used in CountryController)

public Country(String code, String name) {

this.code = code;

this.name = name;

}

// Getters and Setters

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**

package com.cognizant.springlearn.controller;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.web.bind.annotation.\*;

import java.util.ArrayList;

import java.util.List;

import com.cognizant.springlearn.model.Country;

*@RestController*

*@RequestMapping*("/countries")

public class CountryController {

private static final Logger ***LOGGER*** = LoggerFactory.*getLogger*(CountryController.class);

// Sample list (GET request)

*@GetMapping*

public List<Country> getAllCountries() {

List<Country> countries = new ArrayList<>();

Country country1 = new Country();

country1.setCode("IN");

country1.setName("India");

Country country2 = new Country();

country2.setCode("US");

country2.setName("United States");

countries.add(country1);

countries.add(country2);

return countries;

}

// POST request to add a country

*@PostMapping*

public Country addCountry(*@RequestBody* Country country) {

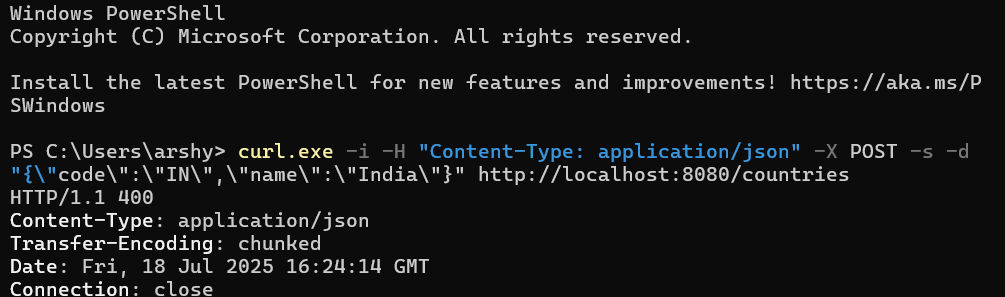
***LOGGER***.info("Start - Received Country: Code={}, Name={}", country.getCode(), country.getName());

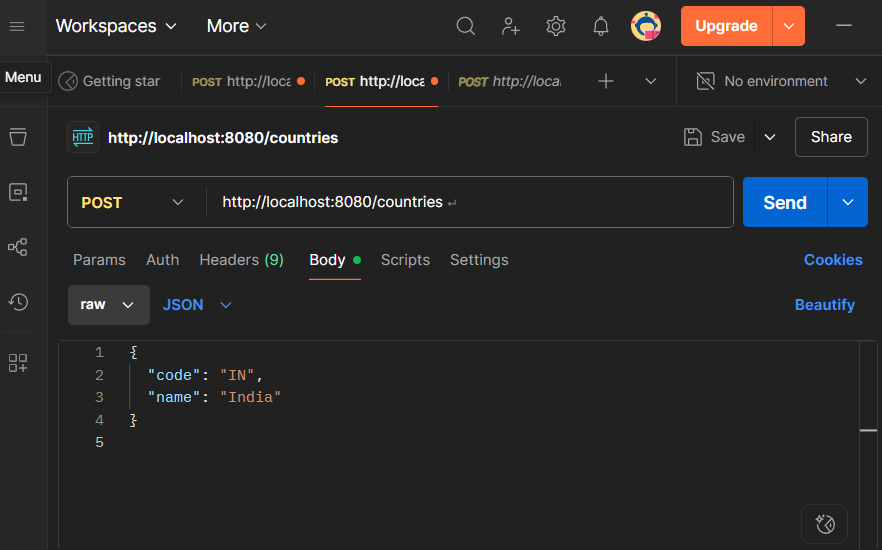
return country; // Just echoing back for now

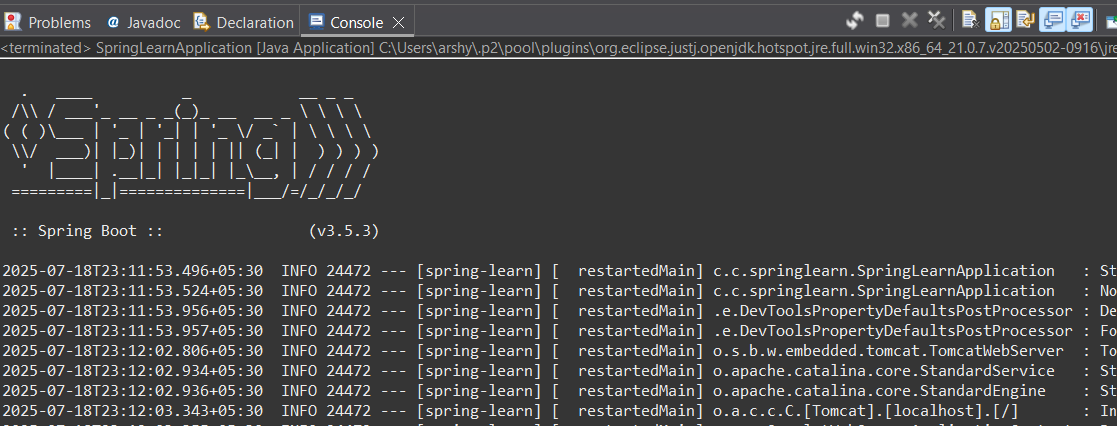
}

}

**Output :**

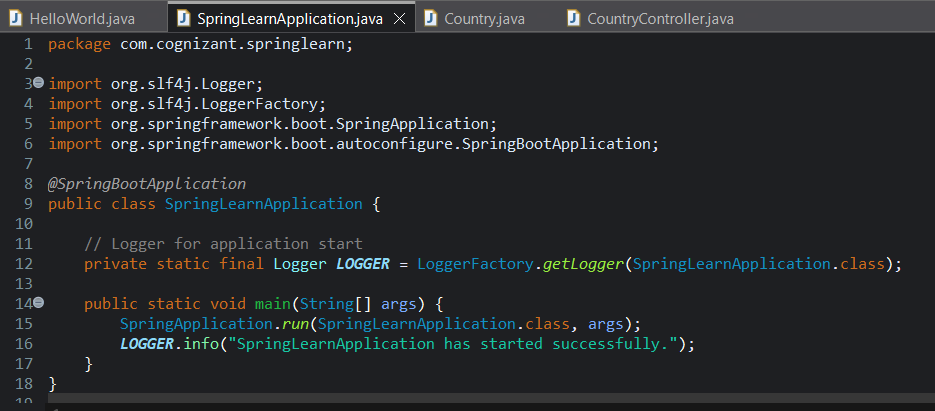
****

****

****

**Validating country code**

**Springlearnapplication.java**

****

**Country.java**

package com.cognizant.springlearn.model;

import jakarta.validation.constraints.NotNull;

import jakarta.validation.constraints.Size;

public class Country {

*@NotNull*(message = "Country code cannot be null")

*@Size*(min = 2, max = 2, message = "Country code should be 2 characters")

private String code;

*@NotNull*(message = "Country name cannot be null")

private String name;

// Default constructor (important for JSON deserialization)

public Country() {

}

// Parameterized constructor (useful for testing and initializing)

public Country(String code, String name) {

this.code = code;

this.name = name;

}

// Getters and Setters

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;

}

// Optional: toString() for debugging/logging

*@Override*

public String toString() {

return "Country{code='" + code + "', name='" + name + "'}";

}

}

**CountryController.java**

package com.cognizant.springlearn.controller;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.http.HttpStatus;

import org.springframework.web.bind.annotation.\*;

import org.springframework.web.server.ResponseStatusException;

import com.cognizant.springlearn.model.Country;

import jakarta.validation.Validation;

import jakarta.validation.Validator;

import jakarta.validation.ValidatorFactory;

import jakarta.validation.ConstraintViolation;

import java.util.ArrayList;

import java.util.List;

import java.util.Set;

*@RestController*

*@RequestMapping*("/countries")

public class CountryController {

private static final Logger ***LOGGER*** = LoggerFactory.*getLogger*(CountryController.class);

*@PostMapping*

public Country addCountry(*@RequestBody* Country country) {

***LOGGER***.info("Start");

// Create validator

ValidatorFactory factory = Validation.*buildDefaultValidatorFactory*();

Validator validator = factory.getValidator();

// Validate the country bean

Set<ConstraintViolation<Country>> violations = validator.validate(country);

List<String> errors = new ArrayList<>();

for (ConstraintViolation<Country> violation : violations) {

errors.add(violation.getMessage());

}

// If there are validation errors, throw an exception

if (!violations.isEmpty()) {

throw new ResponseStatusException(*HttpStatus*.***BAD\_REQUEST***, errors.toString());

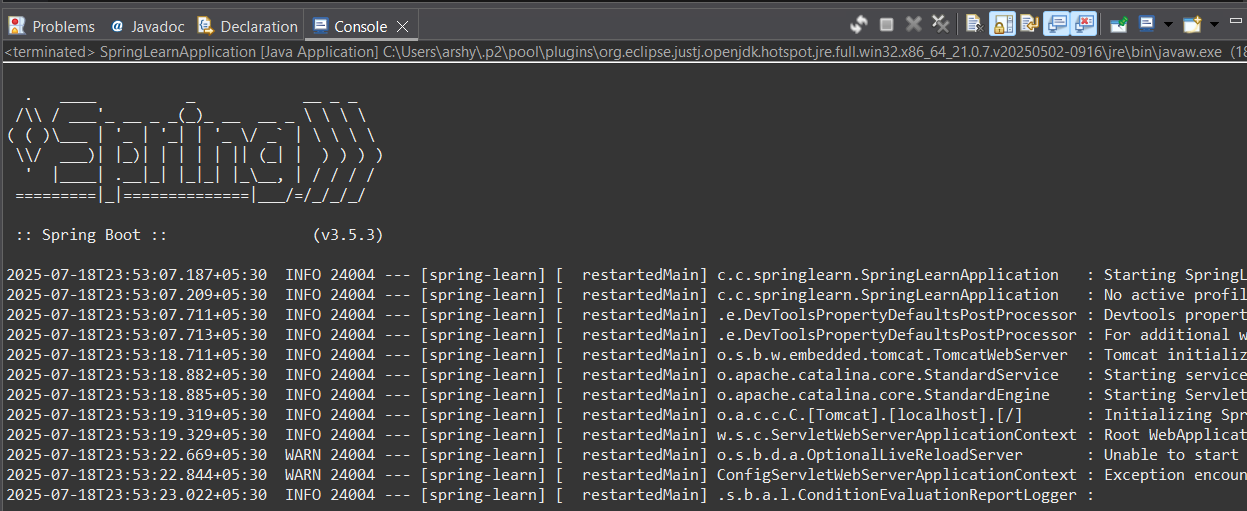
}

return country; // Return the valid country

}

}

**Output :**

****

**Question for all Learners - What needs to be done if there is another controller EmployeeController and similar validation needs to be done for Employee payload data?  
  
SME to explain the disadvantage of the above solution.  
  
This disadvantage will be overcome in the next hands on.**

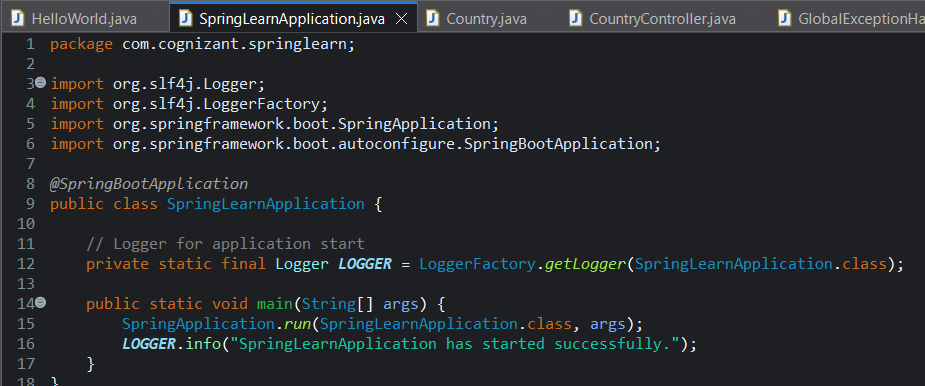
The **answer** is:

The **disadvantage** of this manual validation approach is:

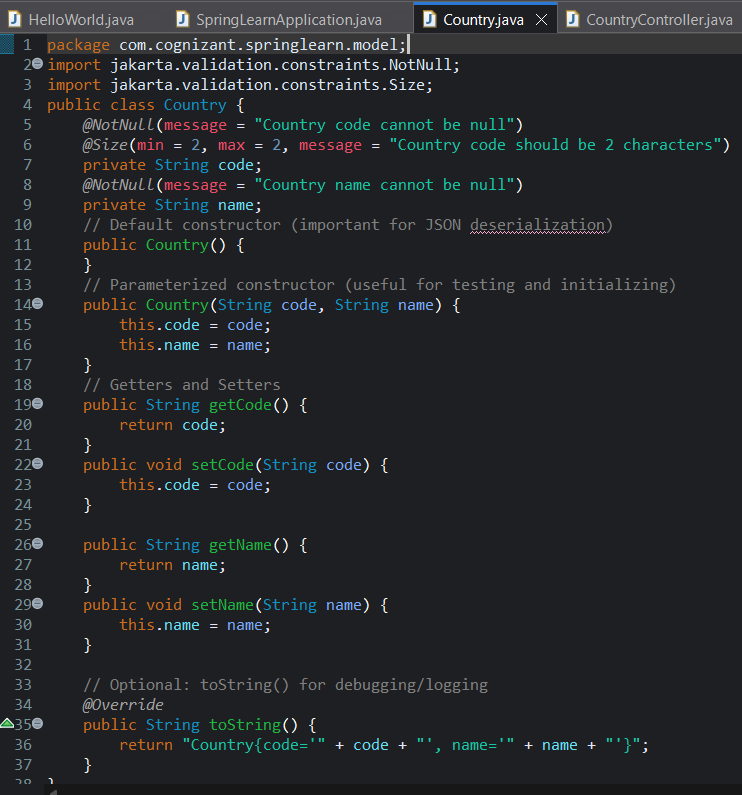
* **Code duplication** – Every controller (like CountryController, EmployeeController, etc.) must repeat the same validator setup code (ValidatorFactory, Validator, and error handling).
* **Hard to maintain** – If the validation logic changes (for example, new rules for all entities), we must manually update every controller.
* **Not scalable** – For larger applications with many controllers, the repeated code becomes difficult to manage and increases the chance of bugs.

**Include global exception handler for validation errors**

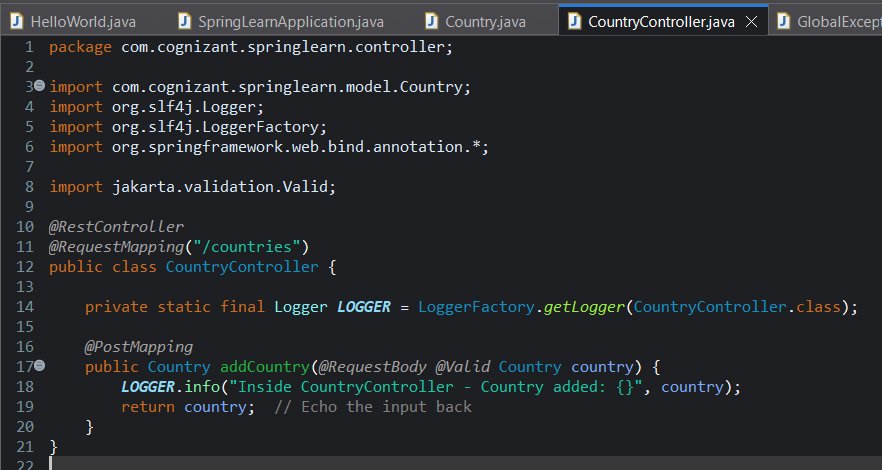
**SpringLearnApplication.java**



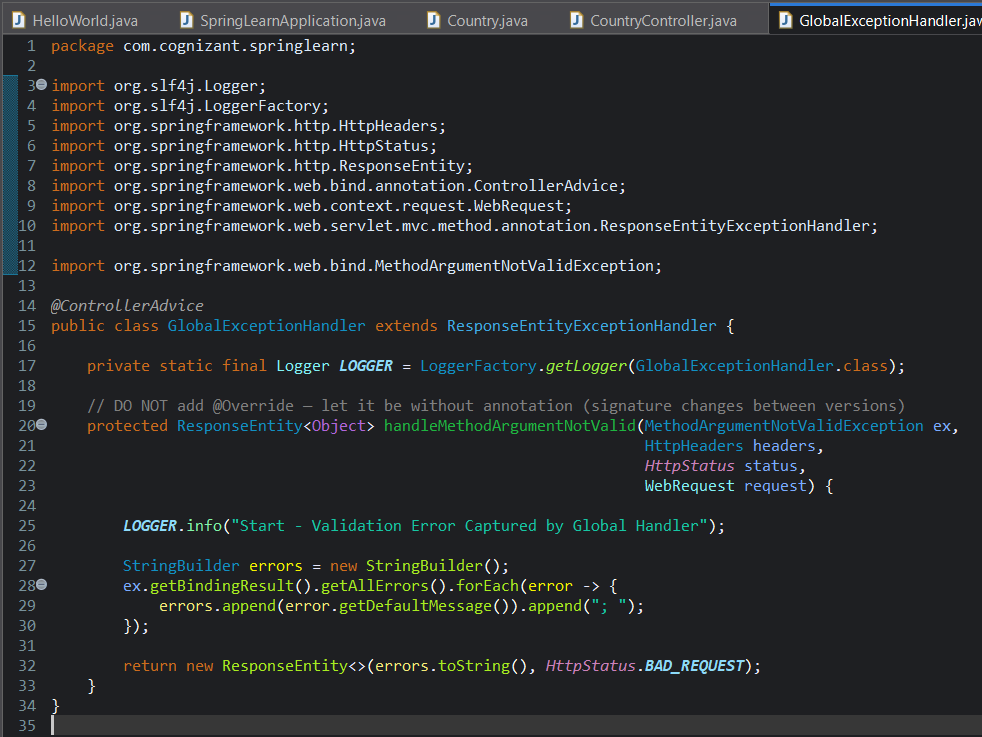
**Country.java**

****

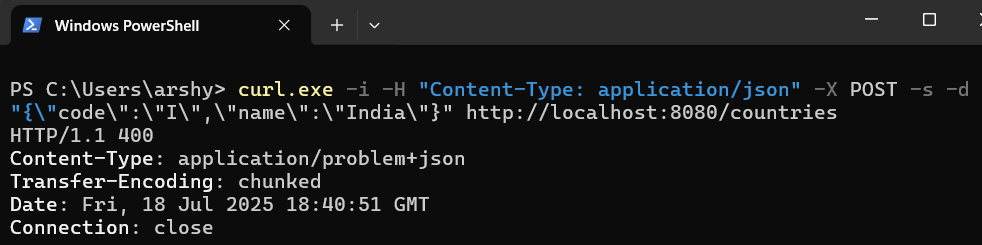
**CountryController.java**

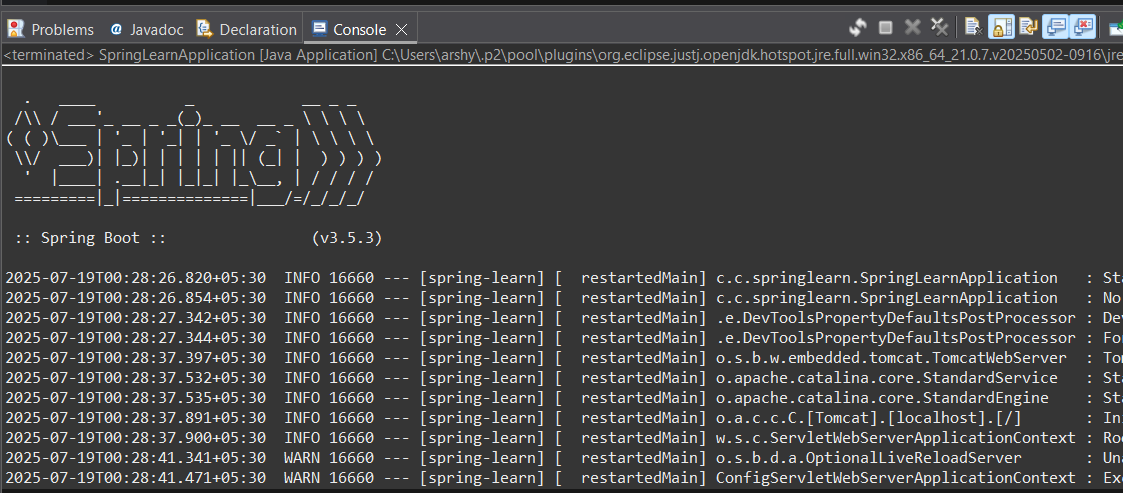
****

**GlobalExceptionHandler.java**

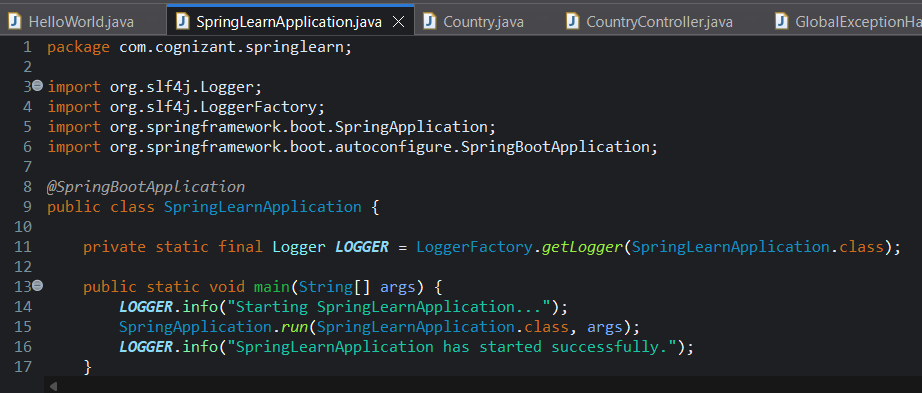
****

Output :





**Response with bad request in global exception handler   
SpringlearnApplication.java**

****

**Country.java**

package com.cognizant.springlearn.model;

import jakarta.validation.constraints.NotNull;

import jakarta.validation.constraints.Size;

public class Country {

*@NotNull*(message = "Country code cannot be null")

*@Size*(min = 2, max = 2, message = "Country code should be 2 characters")

private String code;

*@NotNull*(message = "Country name cannot be null")

private String name;

// Default constructor (required for JSON deserialization)

public Country() {

}

// Parameterized constructor

public Country(String code, String name) {

this.code = code;

this.name = name;

}

// Getters and Setters

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;

}

// toString() for debugging/logging

*@Override*

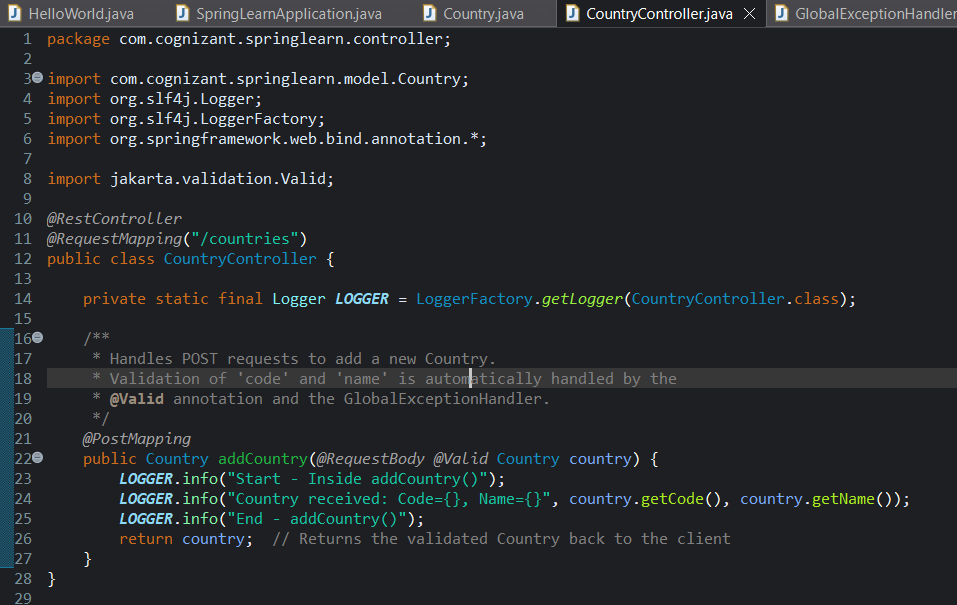
public String toString() {

return "Country{code='" + code + "', name='" + name + "'}";

}

}

**CountryController.java**

****

**GlobalExceptionalHandler.java**

package com.cognizant.springlearn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.http.HttpHeaders;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.MethodArgumentNotValidException;

import org.springframework.web.bind.annotation.ControllerAdvice;

import org.springframework.web.context.request.WebRequest;

import org.springframework.web.servlet.mvc.method.annotation.ResponseEntityExceptionHandler;

import java.util.Date;

import java.util.LinkedHashMap;

import java.util.List;

import java.util.Map;

import java.util.stream.Collectors;

*@ControllerAdvice*

public class GlobalExceptionHandler extends ResponseEntityExceptionHandler {

private static final Logger ***LOGGER*** = LoggerFactory.*getLogger*(GlobalExceptionHandler.class);

// Do NOT use @Override to avoid compilation error in Jakarta-based Spring Boot 3

protected ResponseEntity<Object> handleMethodArgumentNotValid(MethodArgumentNotValidException ex,

HttpHeaders headers,

*HttpStatus* status,

WebRequest request) {

***LOGGER***.info("Start");

Map<String, Object> body = new LinkedHashMap<>();

body.put("timestamp", new Date());

body.put("status", status.value());

List<String> errors = ex.getBindingResult()

.getFieldErrors()

.stream()

.map(error -> error.getDefaultMessage())

.collect(Collectors.*toList*());

body.put("errors", errors);

***LOGGER***.info("End");

return new ResponseEntity<>(body, headers, status);

}

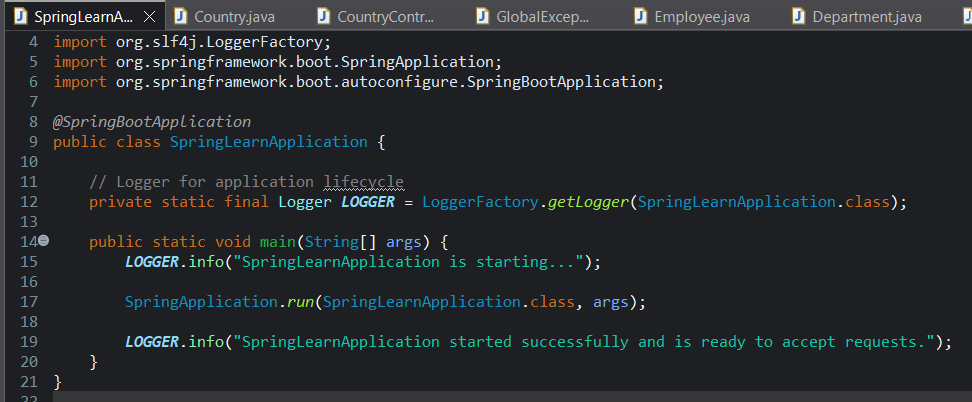
}

**Output :**

****

**Implement REST service for updating an employee**

**SpringlearnApplication.java**



**Employee.java**

package com.cognizant.springlearn.model;

import com.fasterxml.jackson.annotation.JsonFormat;

import jakarta.validation.constraints.\*;

import java.util.Date;

import java.util.List;

public class Employee {

*@NotNull*(message = "Employee ID cannot be null")

private Long id;

*@NotNull*(message = "Employee name cannot be null")

*@NotBlank*(message = "Employee name cannot be blank")

*@Size*(min = 1, max = 30, message = "Employee name must be 1-30 characters")

private String name;

*@NotNull*(message = "Salary cannot be null")

*@PositiveOrZero*(message = "Salary must be zero or positive")

private Double salary;

*@NotNull*(message = "Permanent status cannot be null")

private Boolean permanent;

*@NotNull*(message = "Date of birth cannot be null")

*@JsonFormat*(shape = *JsonFormat*.*Shape*.***STRING***, pattern = "dd/MM/yyyy")

private Date dateOfBirth;

*@NotNull*(message = "Department cannot be null")

private Department department; // Make sure Department.java exists in the same package

*@NotNull*(message = "Skills cannot be null")

private List<Skill> skills; // Make sure Skill.java exists in the same package

// Getters and Setters

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public Double getSalary() {

return salary;

}

public void setSalary(Double salary) {

this.salary = salary;

}

public Boolean getPermanent() {

return permanent;

}

public void setPermanent(Boolean permanent) {

this.permanent = permanent;

}

public Date getDateOfBirth() {

return dateOfBirth;

}

public void setDateOfBirth(Date dateOfBirth) {

this.dateOfBirth = dateOfBirth;

}

public Department getDepartment() {

return department;

}

public void setDepartment(Department department) {

this.department = department;

}

public List<Skill> getSkills() {

return skills;

}

public void setSkills(List<Skill> skills) {

this.skills = skills;

}

// toString for logging/debugging

*@Override*

public String toString() {

return "Employee{id=" + id + ", name='" + name + "', salary=" + salary +

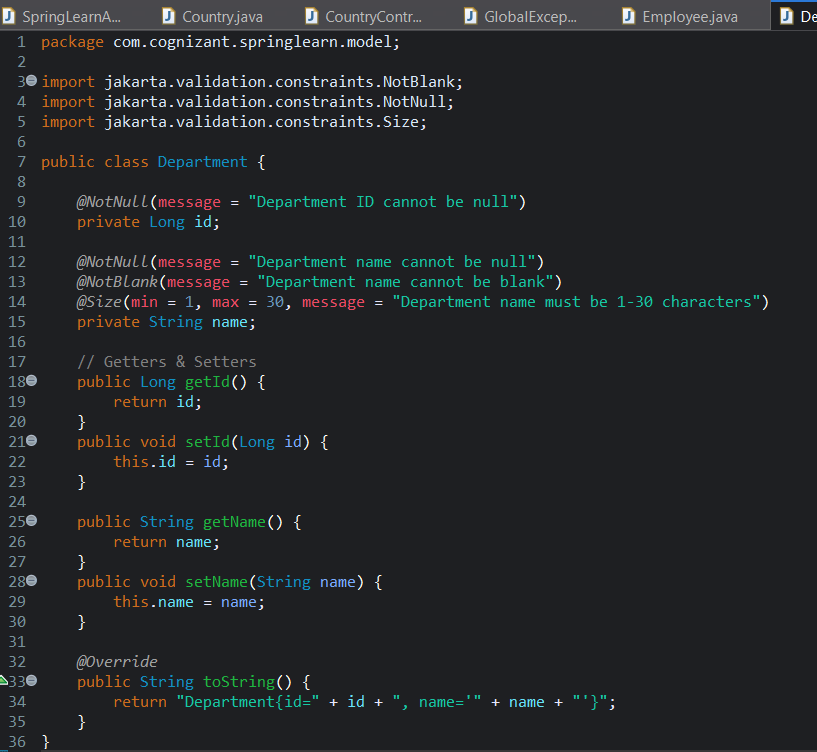
", permanent=" + permanent + ", dateOfBirth=" + dateOfBirth +

", department=" + department + ", skills=" + skills + "}";

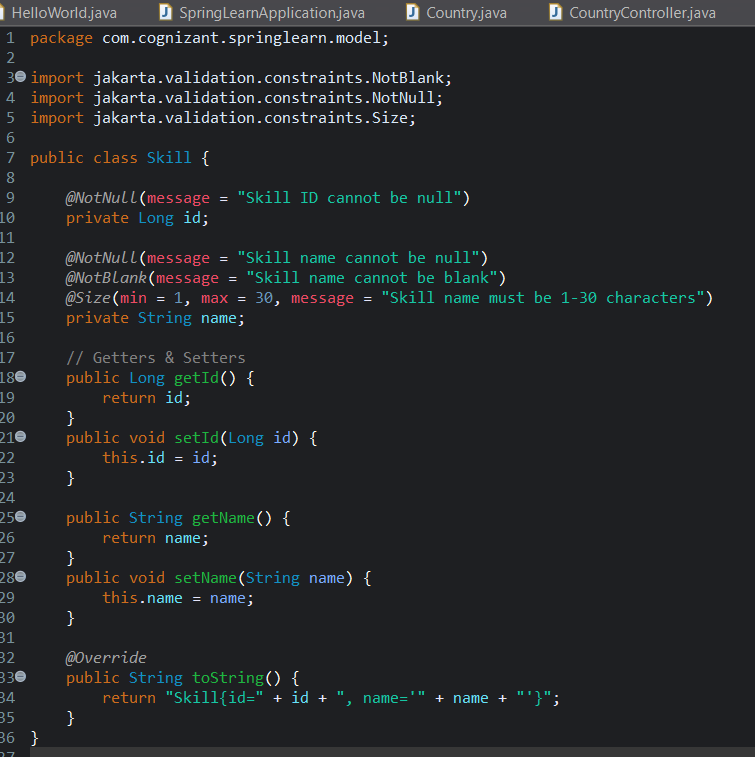
}

}

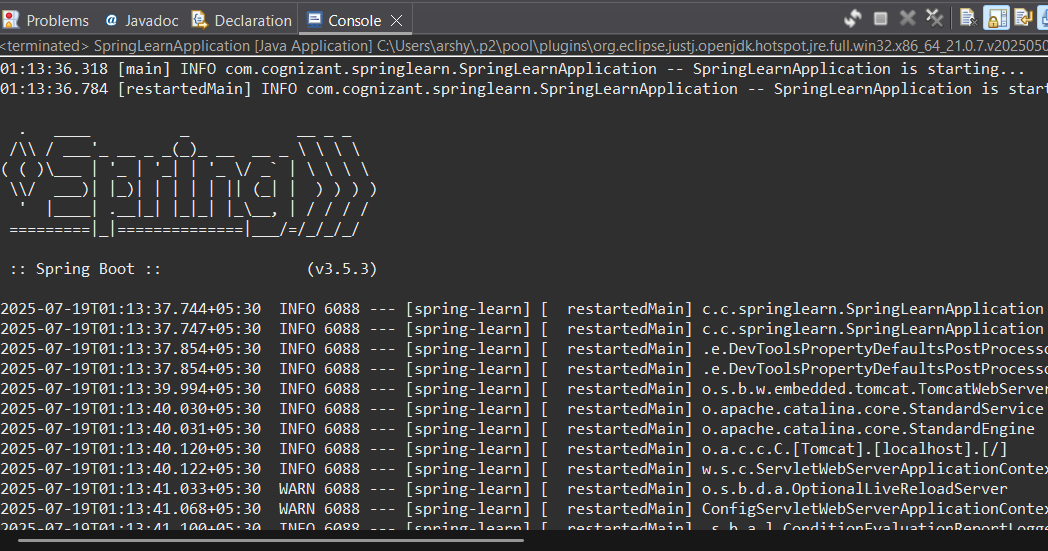
**Department.java**



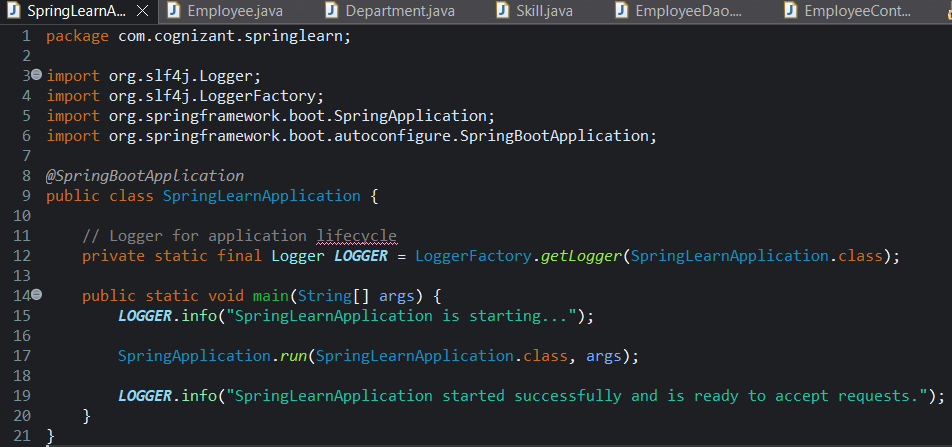
**Skill.java**

****

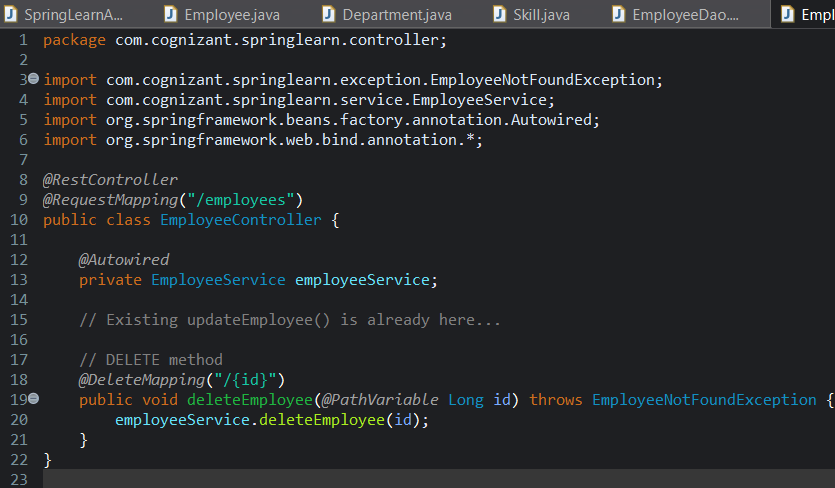
**Output :**

****

**Implement REST DELETE Service   
SpringlearnApplication.java**

****

**EmployeeController.java**

****

**EmployeeDao.java**

package com.cognizant.springlearn.dao;

import com.cognizant.springlearn.exception.EmployeeNotFoundException;

import com.cognizant.springlearn.model.Employee;

import org.springframework.stereotype.Repository;

import java.util.ArrayList;

import java.util.Iterator;

import java.util.List;

*@Repository*

public class EmployeeDao {

private static List<Employee> *EMPLOYEE\_LIST* = new ArrayList<>();

static {

// Dummy data for testing (you can add more)

Employee emp = new Employee();

emp.setId(1L);

emp.setName("John");

emp.setSalary(5000.0);

emp.setPermanent(true);

*EMPLOYEE\_LIST*.add(emp);

}

// Existing updateEmployee() would already be here...

// NEW Delete Method

public void deleteEmployee(Long id) throws EmployeeNotFoundException {

Iterator<Employee> iterator = *EMPLOYEE\_LIST*.iterator();

boolean found = false;

while (iterator.hasNext()) {

Employee emp = iterator.next();

if (emp.getId().equals(id)) {

iterator.remove();

found = true;

break;

}

}

if (!found) {

throw new EmployeeNotFoundException("Employee with ID " + id + " not found");

}

}

// Helper: return all employees (for testing)

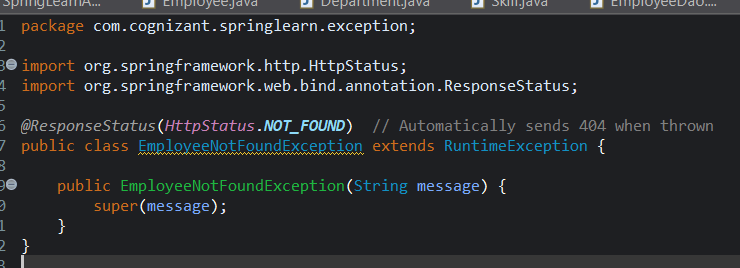
public List<Employee> getAllEmployees() {

return *EMPLOYEE\_LIST*;

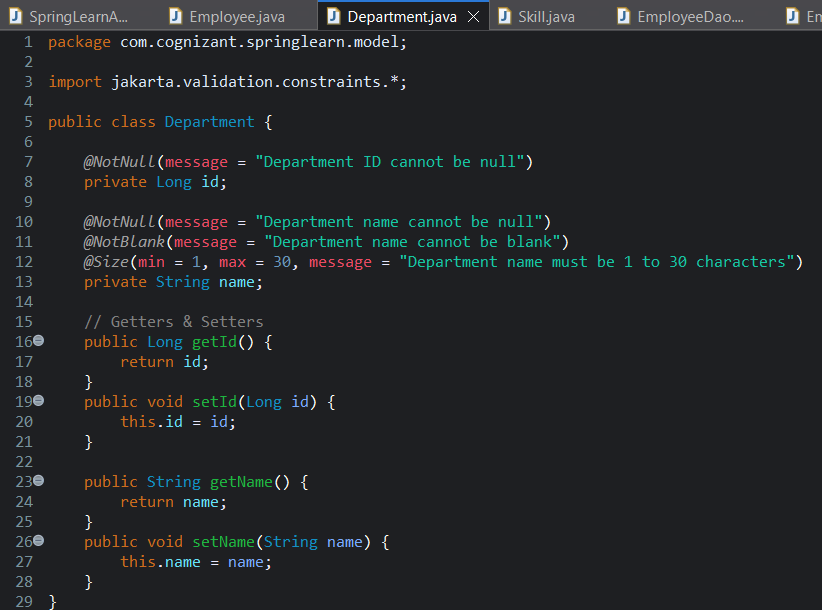
}

}

**EmployeeNotFoundException.java**

****

**Department.java**

****

**Employee.java**

package com.cognizant.springlearn.model;

import com.fasterxml.jackson.annotation.JsonFormat;

import jakarta.validation.constraints.\*;

import java.util.Date;

import java.util.List;

public class Employee {

*@NotNull*(message = "Employee ID cannot be null")

private Long id;

*@NotNull*(message = "Employee name cannot be null")

*@NotBlank*(message = "Employee name cannot be blank")

*@Size*(min = 1, max = 30, message = "Employee name must be 1 to 30 characters")

private String name;

*@NotNull*(message = "Salary cannot be null")

*@DecimalMin*(value = "0.0", inclusive = true, message = "Salary must be zero or positive")

private Double salary;

*@NotNull*(message = "Permanent flag cannot be null")

private Boolean permanent;

*@JsonFormat*(shape = *JsonFormat*.*Shape*.***STRING***, pattern = "dd/MM/yyyy")

private Date dateOfBirth;

private Department department; // You need Department.java class

private List<Skill> skills; // You need Skill.java class

// Getters & setters (Generate them in Eclipse or manually)

public Long getId() { return id; }

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

public String getName() { return name; }

public void setName(String name) { this.name = name; }

public Double getSalary() { return salary; }

public void setSalary(Double salary) { this.salary = salary; }

public Boolean getPermanent() { return permanent; }

public void setPermanent(Boolean permanent) { this.permanent = permanent; }

public Date getDateOfBirth() { return dateOfBirth; }

public void setDateOfBirth(Date dateOfBirth) { this.dateOfBirth = dateOfBirth; }

public Department getDepartment() { return department; }

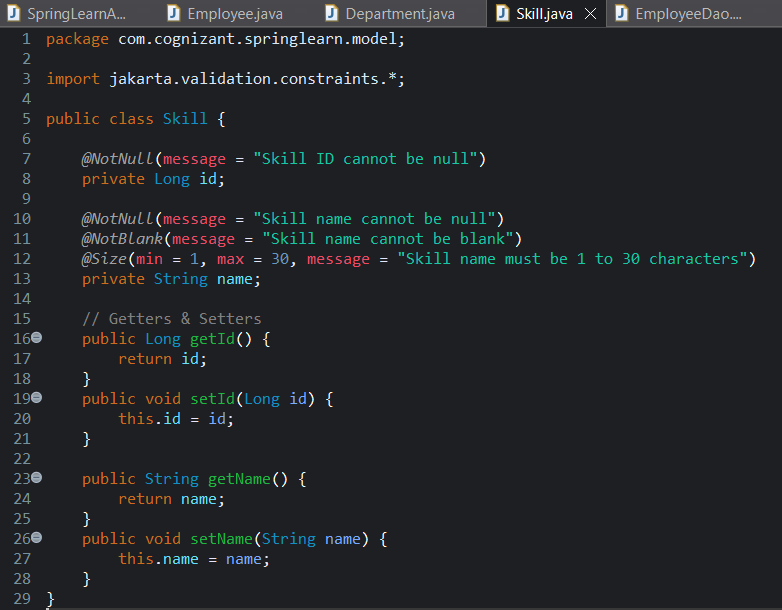
public void setDepartment(Department department) { this.department = department; }

public List<Skill> getSkills() { return skills; }

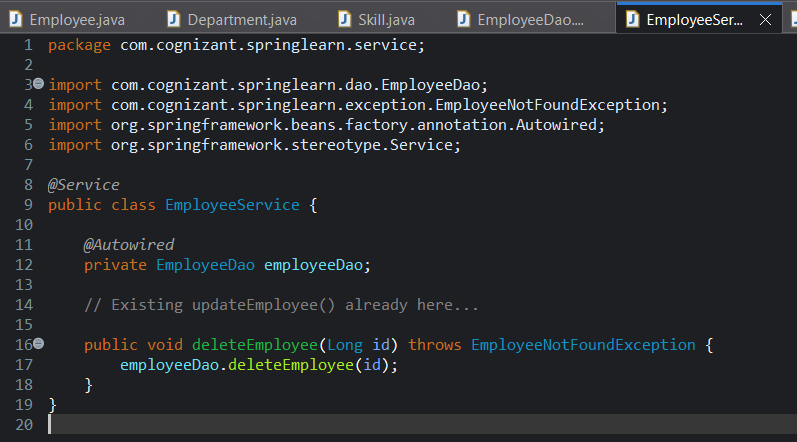
public void setSkills(List<Skill> skills) { this.skills = skills; }

}

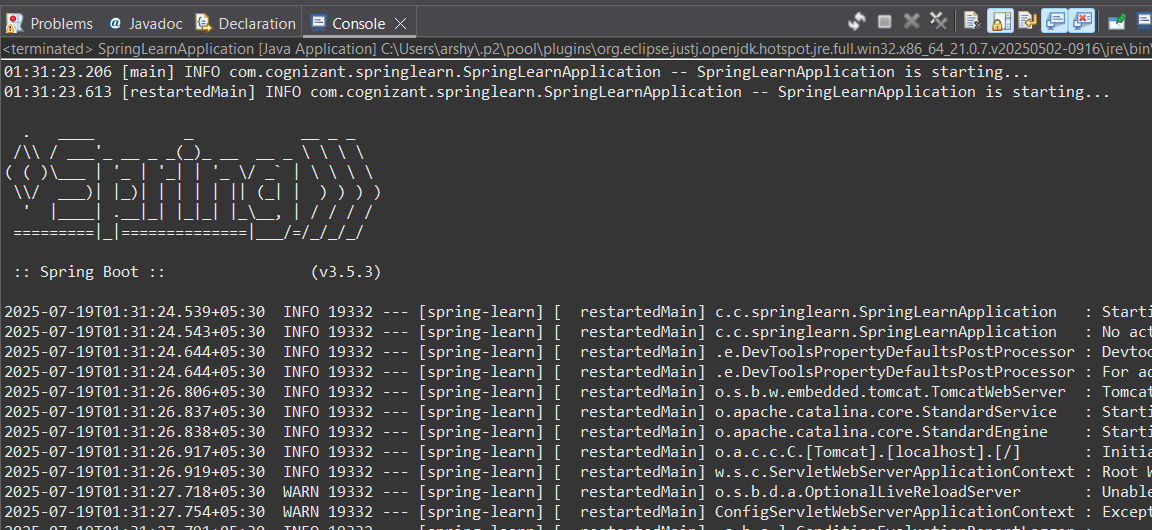
**Skill.java**



**EmployeeService.java**

****

**Output :**

****