# Spring REST Hands-on

## Hands-on 1: Create Spring Boot Project with Maven **SpringLearnApplication.java:** package com.cognizant.springlearn; import org.slf4j.Logger; import org.slf4j.LoggerFactory; import org.springframework.boot.SpringApplication; import org.springframework.boot.autoconfigure.SpringBootApplication; @SpringBootApplication public class SpringLearnApplication { private static final Logger LOGGER = LoggerFactory.getLogger(SpringLearnApplication.class); public static void main(String[] args) { LOGGER.info("START"); SpringApplication.run(SpringLearnApplication.class, args); LOGGER.info("END"); } }

## **Hands-on 2: Load SimpleDateFormat Bean**

date-format.xml:  
<beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://www.springframework.org/schema/beans  
 https://www.springframework.org/schema/beans/spring-beans.xsd">  
 <bean id="dateFormat" class="java.text.SimpleDateFormat">  
 <constructor-arg value="dd/MM/yyyy" />  
 </bean>  
</beans>  
SpringLearnApplication.java - displayDate():  
public static void displayDate() {  
 LOGGER.info("START");  
 ApplicationContext context = new ClassPathXmlApplicationContext("date-format.xml");  
 SimpleDateFormat format = context.getBean("dateFormat", SimpleDateFormat.class);  
 try {  
 Date date = format.parse("31/12/2018");  
 LOGGER.debug("Parsed Date: {}", date);  
 } catch (ParseException e) {  
 LOGGER.error("ParseException", e);  
 }  
 LOGGER.info("END");  
}

## **Hands-on 3: Incorporate Logging**

application.properties:  
logging.level.org.springframework=info  
logging.level.com.cognizant.springlearn=debug  
logging.pattern.console=%d{yyMMdd}|%d{HH:mm:ss.SSS}|%-20.20thread|%5p|%-25.25logger{25}|%25M|%m%n  
**Logger setup in SpringLearnApplication.java:**import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
private static final Logger LOGGER = LoggerFactory.getLogger(SpringLearnApplication.class);  
Use in methods:  
LOGGER.info("START");  
LOGGER.debug("Some debug log");  
LOGGER.info("END");

## **Hands-on 4: Load Country Bean**

**country.xml:**  
<bean id="country" class="com.cognizant.springlearn.Country">  
 <property name="code" value="IN"/>  
 <property name="name" value="India"/>  
</bean>  
**Country.java:**  
public class Country {  
 private String code;  
 private String name;  
 public Country() {  
 LOGGER.debug("Inside Country Constructor");  
 }  
 public String getCode() {  
 LOGGER.debug("Getting code");  
 return code;  
 }  
 public void setCode(String code) {  
 LOGGER.debug("Setting code");  
 this.code = code;

}  
 public String getName() {  
 LOGGER.debug("Getting name");  
 return name;  
 }  
 public void setName(String name) {  
 LOGGER.debug("Setting name");  
 this.name = name;  
 }  
 @Override  
 public String toString() {  
 return "Country [code=" + code + ", name=" + name + "]";  
 }  
}  
  
**SpringLearnApplication.java - displayCountry():**  
public static void displayCountry() {  
 LOGGER.info("START");  
 ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");  
 Country country = context.getBean("country", Country.class);  
 LOGGER.debug("Country : {}", country.toString());  
 LOGGER.info("END");  
}

## **Hands-on 5: Singleton vs Prototype Scope**

**To test Singleton:**  
ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");  
Country country1 = context.getBean("country", Country.class);  
Country country2 = context.getBean("country", Country.class);  
  
To test Prototype:  
<bean id="country" class="com.cognizant.springlearn.Country" scope="prototype" />

## **Hands-on 6: Load List of Countries**

**country.xml:**  
<bean id="in" class="com.cognizant.springlearn.Country">  
 <property name="code" value="IN"/>  
 <property name="name" value="India"/>  
</bean>  
<bean id="us" class="com.cognizant.springlearn.Country">  
 <property name="code" value="US"/>  
 <property name="name" value="United States"/>  
</bean>  
<bean id="de" class="com.cognizant.springlearn.Country">  
 <property name="code" value="DE"/>  
 <property name="name" value="Germany"/>  
</bean>  
<bean id="jp" class="com.cognizant.springlearn.Country">  
 <property name="code" value="JP"/>  
 <property name="name" value="Japan"/>  
</bean>  
<bean id="countryList" class="java.util.ArrayList">  
 <constructor-arg>  
 <list>  
 <ref bean="in"/>  
 <ref bean="us"/>  
 <ref bean="de"/>  
 <ref bean="jp"/>  
 </list>  
 </constructor-arg>  
</bean>  
**SpringLearnApplication.java - displayCountries():**  
public static void displayCountries() {  
 LOGGER.info("START");  
 ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");  
 List<Country> countries = (List<Country>) context.getBean("countryList");  
 for (Country c : countries) {  
 LOGGER.debug("Country: {}", c);  
 }  
 LOGGER.info("END");  
}

**SPRING REST HANDSON-2**

## ****REST - Hello World Web Service****

## HelloController.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.RestController;

@RestControllerpublic class HelloController {

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

@GetMapping("/hello")

public String sayHello() {

LOGGER.info("START - sayHello");

LOGGER.info("END - sayHello");

return "Hello World!!";

}

}

## ****2. REST - Country Web Service****

### country.xml

<bean id="country" class="com.cognizant.springlearn.model.Country">

<property name="code" value="IN" />

<property name="name" value="India" /></bean>

### CountryController.java

package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.model.Country;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

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

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

@RestControllerpublic class CountryController {

@GetMapping("/country")

public Country getCountryIndia() {

ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");

Country country = (Country) context.getBean("country", Country.class);

return country;

}

}

## ****3. REST - Get All Countries****

country.xml

<bean id="in" class="com.cognizant.springlearn.model.Country">

<property name="code" value="IN"/>

<property name="name" value="India"/></bean><bean id="us" class="com.cognizant.springlearn.model.Country">

<property name="code" value="US"/>

<property name="name" value="United States"/></bean><bean id="jp" class="com.cognizant.springlearn.model.Country">

<property name="code" value="JP"/>

<property name="name" value="Japan"/></bean><bean id="de" class="com.cognizant.springlearn.model.Country">

<property name="code" value="DE"/>

<property name="name" value="Germany"/></bean>

<bean id="countryList" class="java.util.ArrayList">

<constructor-arg>

<list>

<ref bean="in"/>

<ref bean="us"/>

<ref bean="jp"/>

<ref bean="de"/>

</list>

</constructor-arg></bean>

### CountryController.java

@GetMapping("/countries")public List<Country> getAllCountries() {

ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");

List<Country> countryList = context.getBean("countryList", List.class);

return countryList;

}

## ****4. REST - Get Country by Code (Case Insensitive)****

### CountryService.java

package com.cognizant.springlearn.service;

import com.cognizant.springlearn.model.Country;

import com.cognizant.springlearn.service.exception.CountryNotFoundException;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import org.springframework.stereotype.Service;

import java.util.List;

@Servicepublic class CountryService {

public Country getCountry(String code) throws CountryNotFoundException {

ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");

List<Country> countries = context.getBean("countryList", List.class);

return countries.stream()

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

.findFirst()

.orElseThrow(() -> new CountryNotFoundException());

}

}

### CountryController.java

@Autowiredprivate CountryService countryService;

@GetMapping("/countries/{code}")public Country getCountry(@PathVariable String code) throws CountryNotFoundException {

return countryService.getCountry(code);

}

## ****5. REST - CountryNotFoundException****

### CountryNotFoundException.java

package com.cognizant.springlearn.service.exception;

import org.springframework.http.HttpStatus;

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

@ResponseStatus(value = HttpStatus.NOT\_FOUND, reason = "Country not found")public class CountryNotFoundException extends Exception {

private static final long serialVersionUID = 1L;

}

## ****6. MockMVC - Test Country Web Service****

### SpringLearnApplicationTests.java

package com.cognizant.springlearn;

import com.cognizant.springlearn.controller.CountryController;

import org.junit.jupiter.api.Test;

import org.springframework.beans.factory.annotation.Autowired;import org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;

import org.springframework.boot.test.context.SpringBootTest;

import org.springframework.test.web.servlet.MockMvc;

import org.springframework.test.web.servlet.ResultActions;

import static org.junit.jupiter.api.Assertions.assertNotNull;

import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.get;

import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.\*;

@SpringBootTest@AutoConfigureMockMvcpublic class SpringLearnApplicationTests {

@Autowired

private CountryController countryController;

@Autowired

private MockMvc mvc;

@Test

public void contextLoads() {

assertNotNull(countryController);

}

@Test

public void testGetCountry() throws Exception {

ResultActions actions = mvc.perform(get("/country"));

actions.andExpect(status().isOk());

actions.andExpect(jsonPath("$.code").value("IN"));

actions.andExpect(jsonPath("$.name").value("India"));

}

@Test

public void testGetCountryException() throws Exception {

ResultActions actions = mvc.perform(get("/countries/az"));

actions.andExpect(status().isNotFound());

actions.andExpect(status().reason("Country not found"));

}

}

**SPRING HANDSON-3**

## ****1. Create Static Employee List using Spring XML****

### employee.xml

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.springframework.org/schema/beans

https://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="skill1" class="com.cognizant.model.Skill">

<property name="id" value="1" />

<property name="name" value="Java" />

</bean>

<bean id="skill2" class="com.cognizant.model.Skill">

<property name="id" value="2" />

<property name="name" value="Spring Boot" />

</bean>

<bean id="dep1" class="com.cognizant.model.Department">

<property name="id" value="1" />

<property name="name" value="HR" />

</bean>

<bean id="dep2" class="com.cognizant.model.Department">

<property name="id" value="2" />

<property name="name" value="Tech" />

</bean>

<bean id="emp1" class="com.cognizant.model.Employee">

<property name="id" value="1" />

<property name="name" value="John" />

<property name="salary" value="60000" />

<property name="permanent" value="true" />

<property name="department" ref="dep1" />

<property name="skills">

<list>

<ref bean="skill1"/>

</list>

</property>

</bean>

<bean id="emp2" class="com.cognizant.model.Employee">

<property name="id" value="2" />

<property name="name" value="Emma" />

<property name="salary" value="75000" />

<property name="permanent" value="false" />

<property name="department" ref="dep2" />

<property name="skills">

<list>

<ref bean="skill2"/>

</list>

</property>

</bean>

<bean id="employeeList" class="java.util.ArrayList">

<constructor-arg>

<list>

<ref bean="emp1"/>

<ref bean="emp2"/>

</list>

</constructor-arg>

</bean>

<bean id="departmentList" class="java.util.ArrayList">

<constructor-arg>

<list>

<ref bean="dep1"/>

<ref bean="dep2"/>

</list>

</constructor-arg>

</bean></beans>

## ****2.**** EmployeeDao.java

package com.cognizant.dao;

import com.cognizant.model.Employee;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import org.springframework.stereotype.Repository;

import java.util.List;

@Repositorypublic class EmployeeDao {

private static List<Employee> EMPLOYEE\_LIST;

public EmployeeDao() {

ApplicationContext context = new ClassPathXmlApplicationContext("employee.xml");

EMPLOYEE\_LIST = (List<Employee>) context.getBean("employeeList");

}

public List<Employee> getAllEmployees() {

return EMPLOYEE\_LIST;

}

}

## ****3.**** EmployeeService.java

package com.cognizant.service;

import com.cognizant.dao.EmployeeDao;import com.cognizant.model.Employee;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import java.util.List;

@Servicepublic class EmployeeService {

@Autowired

private EmployeeDao employeeDao;

@Transactional

public List<Employee> getAllEmployees() {

return employeeDao.getAllEmployees();

}

}

## ****4.** EmployeeController.java**

package com.cognizant.controller;

import com.cognizant.model.Employee;

import com.cognizant.service.EmployeeService;

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

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

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

import java.util.List;

@RestControllerpublic class EmployeeController {

@Autowired

private EmployeeService employeeService;

@GetMapping("/employees")

public List<Employee> getAllEmployees() {

return employeeService.getAllEmployees();

}

}

## ****5. Department REST Service****

### DepartmentDao.java

package com.cognizant.dao;

import com.cognizant.model.Department;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import org.springframework.stereotype.Repository;

import java.util.List;

@Repositorypublic class DepartmentDao {

private static List<Department> DEPARTMENT\_LIST;

public DepartmentDao() {

ApplicationContext context = new ClassPathXmlApplicationContext("employee.xml");

DEPARTMENT\_LIST = (List<Department>) context.getBean("departmentList");

}

public List<Department> getAllDepartments() {

return DEPARTMENT\_LIST;

}

}

### DepartmentService.java

package com.cognizant.service;

import com.cognizant.dao.DepartmentDao;

import com.cognizant.model.Department;

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

import org.springframework.stereotype.Service;

import java.util.List;

@Servicepublic class DepartmentService {

@Autowired

private DepartmentDao departmentDao;

public List<Department> getAllDepartments() {

return departmentDao.getAllDepartments();

}

}

### DepartmentController.java

package com.cognizant.controller;

import com.cognizant.model.Department;

import com.cognizant.service.DepartmentService;

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

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

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

import java.util.List;

@RestControllerpublic class DepartmentController {

@Autowired

private DepartmentService departmentService;

@GetMapping("/departments")

public List<Department> getAllDepartments() {

return departmentService.getAllDepartments();

}

}

**SPRING HANDS ON 4**

## 1. ****POST Country with Validation****

### CountryController.java

@RestController@RequestMapping("/countries")public class CountryController {

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

@PostMapping

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

LOGGER.info("START");

LOGGER.debug("Country: {}", country);

LOGGER.info("END");

return country;

}

}

### Country.java

package com.cognizant.model;

import jakarta.validation.constraints.NotNull;import jakarta.validation.constraints.Size;

public class Country {

@NotNull

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

private String code;

@NotNull(message = "Country name must not be null")

private String name;

}

## 2. ****Global Exception Handler for Validation****

### GlobalExceptionHandler.java

package com.cognizant.springlearn;

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

import org.springframework.http.\*;import org.springframework.web.bind.MethodArgumentNotValidException;

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

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

importorg.springframework.web.servlet.mvc.method.annotation.ResponseEntityExceptionHandler;

import com.fasterxml.jackson.databind.exc.InvalidFormatException;

import org.springframework.http.converter.HttpMessageNotReadableException;

import java.util.\*;

import java.util.stream.Collectors;

@ControllerAdvicepublic 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: handle validation");

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: handle validation");

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

}

@Override

protected ResponseEntity<Object> handleHttpMessageNotReadable(HttpMessageNotReadableException ex,

HttpHeaders headers,

HttpStatus status,

WebRequest request) {

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 ife) {

for (InvalidFormatException.Reference ref : ife.getPath()) {

body.put("message", "Incorrect format for field '" + ref.getFieldName() + "'");

}

}

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

}

}

## 3. ****PUT (Update) Employee with Validation****

### EmployeeController.java

@PutMapping("/employees")public void updateEmployee(@RequestBody @Valid Employee employee) throws EmployeeNotFoundException {

employeeService.updateEmployee(employee);

}

### EmployeeService.java

public void updateEmployee(Employee employee) throws EmployeeNotFoundException {

employeeDao.updateEmployee(employee);

}

### EmployeeDao.java

public void updateEmployee(Employee updatedEmp) throws EmployeeNotFoundException {

boolean found = false;

for (int i = 0; i < EMPLOYEE\_LIST.size(); i++) {

if (EMPLOYEE\_LIST.get(i).getId() == updatedEmp.getId()) {

EMPLOYEE\_LIST.set(i, updatedEmp);

found = true;

break;

}

}

if (!found) {

throw new EmployeeNotFoundException("Employee not found");

}

}

## 4. ****DELETE Employee****

### EmployeeController.java

@DeleteMapping("/employees/{id}")public void deleteEmployee(@PathVariable int id) throws EmployeeNotFoundException {

employeeService.deleteEmployee(id);

}

### EmployeeService.java

public void deleteEmployee(int id) throws EmployeeNotFoundException {

employeeDao.deleteEmployee(id);

}

### EmployeeDao.java

public void deleteEmployee(int id) throws EmployeeNotFoundException {

boolean removed = EMPLOYEE\_LIST.removeIf(emp -> emp.getId() == id);

if (!removed) {

throw new EmployeeNotFoundException("Employee not found");

}

}

**5.JWT HANDS ON**

## 1. ****POST Country with Validation****

### CountryController.java

@RestController@RequestMapping("/countries")public class CountryController {

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

@PostMapping

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

LOGGER.info("START");

LOGGER.debug("Country: {}", country);

LOGGER.info("END");

return country;

}

}

## 2. ****Country.java (Bean with Validation)****

### Country.java

package com.cognizant.model;

import jakarta.validation.constraints.NotNull;import jakarta.validation.constraints.Size;

public class Country {

@NotNull

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

private String code;

@NotNull

private String name;

}

## 3. ****Global Validation Exception Handler****

### GlobalExceptionHandler.java

package com.cognizant.springlearn.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.ControllerAdvice;

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

importorg.springframework.web.servlet.mvc.method.annotation.ResponseEntityExceptionHandler;

import org.springframework.http.converter.HttpMessageNotReadableException;

import com.fasterxml.jackson.databind.exc.InvalidFormatException;

import java.util.\*;import java.util.stream.Collectors;

@ControllerAdvicepublic 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");

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);

}

@Override

protected ResponseEntity<Object> handleHttpMessageNotReadable(HttpMessageNotReadableException ex,

HttpHeaders headers,

HttpStatus status,

WebRequest request) {

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 cause) {

for (InvalidFormatException.Reference ref : cause.getPath()) {

body.put("message", "Incorrect format for field '" + ref.getFieldName() + "'");

}

}

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

}

}

## 4. ****PUT – Update Employee****

### EmployeeController.java

@PutMapping("/employees")public void updateEmployee(@RequestBody @Valid Employee employee) throws EmployeeNotFoundException {

employeeService.updateEmployee(employee);

}

### EmployeeService.java

public void updateEmployee(Employee employee) throws EmployeeNotFoundException {

employeeDao.updateEmployee(employee);

}

### EmployeeDao.java

public void updateEmployee(Employee updatedEmp) throws EmployeeNotFoundException {

boolean found = false;

for (int i = 0; i < EMPLOYEE\_LIST.size(); i++) {

if (EMPLOYEE\_LIST.get(i).getId() == updatedEmp.getId()) {

EMPLOYEE\_LIST.set(i, updatedEmp);

found = true;

break;

}

}

if (!found) {

throw new EmployeeNotFoundException("Employee not found");

}

}

## 5. ****DELETE – Remove Employee****

### EmployeeController.java

@DeleteMapping("/employees/{id}")public void deleteEmployee(@PathVariable int id) throws EmployeeNotFoundException {

employeeService.deleteEmployee(id);

}

### EmployeeService.java

public void deleteEmployee(int id) throws EmployeeNotFoundException {

employeeDao.deleteEmployee(id);

}

### EmployeeDao.java

public void deleteEmployee(int id) throws EmployeeNotFoundException {

boolean removed = EMPLOYEE\_LIST.removeIf(emp -> emp.getId() == id);

if (!removed) {

throw new EmployeeNotFoundException("Employee not found");

}

}