**Exercise-1: Spring Boot Setup (REST Basics)**

**Code:**

@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);

displayDate();

displayCountry();

displayCountries();

LOGGER.info("END");

}

public static void displayDate() {

ApplicationContext context = new ClassPathXmlApplicationContext("date-format.xml");

SimpleDateFormat format = context.getBean("dateFormat", SimpleDateFormat.class);

try {

Date date = format.parse("31/12/2018");

LOGGER.debug("Date: {}", date);

} catch (ParseException e) {

LOGGER.error("Parse error", e);

}

}

public static void displayCountry() {

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

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

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

}

public static void displayCountries() {

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

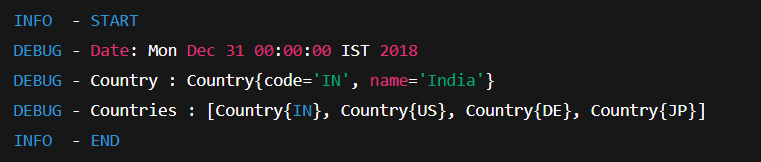
List<Country> list = context.getBean("countryList", ArrayList.class);

LOGGER.debug("Countries : {}", list);

}

}

**Output:**

****

**Exercise: REST Controller**

**HelloController.java**

@RestController

public class HelloController {

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

@GetMapping("/hello")

public String sayHello() {

LOGGER.info("START");

LOGGER.info("END");

return "Hello World!!";

}

}

**CountryController.java**

@RestController

@RequestMapping("/countries")

public class CountryController {

@Autowired

private CountryService service;

@GetMapping

public List<Country> getAllCountries() {

return service.getAllCountries();

}

@GetMapping("/{code}")

public Country getCountry(@PathVariable String code) throws CountryNotFoundException {

return service.getCountry(code);

}

}

**CountryService.java**

@Service

public class CountryService {

public List<Country> getAllCountries() {

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

return context.getBean("countryList", ArrayList.class);

}

public Country getCountry(String code) throws CountryNotFoundException {

return getAllCountries().stream()

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

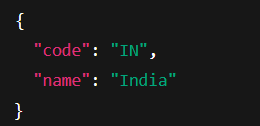
.findFirst()

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

}

}

**Output:**

****

**Exercise: Testing with MockMVC**

**SpringLearnApplicationTests.java**

@SpringBootTest

@AutoConfigureMockMvc

public class SpringLearnApplicationTests {

@Autowired

private MockMvc mvc;

@Autowired

private CountryController countryController;

@Test

public void contextLoads() {

assertNotNull(countryController);

}

@Test

public void testGetCountry() throws Exception {

mvc.perform(get("/countries/IN"))

.andExpect(status().isOk())

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

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

}

}

**Exercise: JWT and Spring Security**

**AuthenticationController.java**

@RestController

public class AuthenticationController {

@GetMapping("/authenticate")

public Map<String, String> authenticate(@RequestHeader("Authorization") String authHeader) {

String user = getUser(authHeader);

String token = generateJwt(user);

Map<String, String> map = new HashMap<>();

map.put("token", token);

return map;

}

private String getUser(String authHeader) {

String base64Credentials = authHeader.substring("Basic ".length());

String credentials = new String(Base64.getDecoder().decode(base64Credentials));

return credentials.split(":")[0];

}

private String generateJwt(String user) {

return Jwts.builder()

.setSubject(user)

.setIssuedAt(new Date())

.setExpiration(new Date(System.currentTimeMillis() + 1200000))

.signWith(SignatureAlgorithm.HS256, "secretkey")

.compact();

}

}

**SecurityConfig.java**

@Override

protected void configure(HttpSecurity http) throws Exception {

http.csrf().disable().httpBasic().and()

.authorizeRequests()

.antMatchers("/authenticate").permitAll()

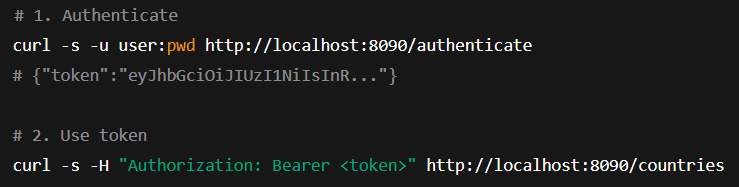
.anyRequest().authenticated()

.and()

.addFilter(new JwtAuthorizationFilter(authenticationManager()));

}

**Output:**

****

**Exercise: Employee Service Overview**

**Employee.java**

public class Employee {

private int id;

private String name;

private double salary;

private boolean permanent;

private Date dateOfBirth;

private Department department;

private List<Skill> skillList;

// Getters, setters, toString

}

**Department.java**

public class Department {

private int id;

private String name;

// Getters, setters, toString

}

**Skill.java**

public class Skill {

private int id;

private String name;

// Getters, setters, toString

}

**employee.xml**

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

<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">

<!-- Departments -->

<bean id="dept1" class="com.cognizant.springlearn.model.Department">

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

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

</bean>

<bean id="dept2" class="com.cognizant.springlearn.model.Department">

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

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

</bean>

<!-- Skills -->

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

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

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

</bean>

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

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

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

</bean>

<!-- Employees -->

<bean id="employee1" class="com.cognizant.springlearn.model.Employee">

<property name="id" value="101"/>

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

<property name="salary" value="55000"/>

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

<property name="dateOfBirth">

<bean class="java.text.SimpleDateFormat">

<constructor-arg value="dd/MM/yyyy"/>

</bean>

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

</property>

<property name="department" ref="dept1"/>

<property name="skillList">

<list>

<ref bean="skill1"/>

<ref bean="skill2"/>

</list>

</property>

</bean>

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

<constructor-arg>

<list>

<ref bean="employee1"/>

</list>

</constructor-arg>

</bean>

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

<constructor-arg>

<list>

<ref bean="dept1"/>

<ref bean="dept2"/>

</list>

</constructor-arg>

</bean>

</beans>

**EmployeeDao.java**

@Repository

public class EmployeeDao {

public static List<Employee> EMPLOYEE\_LIST;

public EmployeeDao() {

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

EMPLOYEE\_LIST = context.getBean("employeeList", ArrayList.class);

}

public List<Employee> getAllEmployees() {

return EMPLOYEE\_LIST;

}

}

**DepartmentDao.java**

@Repository

public class DepartmentDao {

public static List<Department> DEPARTMENT\_LIST;

public DepartmentDao() {

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

DEPARTMENT\_LIST = context.getBean("departmentList", ArrayList.class);

}

public List<Department> getAllDepartments() {

return DEPARTMENT\_LIST;

}

}

**EmployeeService.java**

@Service

public class EmployeeService {

@Autowired

private EmployeeDao employeeDao;

public List<Employee> getAllEmployees() {

return employeeDao.getAllEmployees();

}

}

**DepartmentService.java**

@Service

public class DepartmentService {

@Autowired

private DepartmentDao departmentDao;

public List<Department> getAllDepartments() {

return departmentDao.getAllDepartments();

}

}

**EmployeeController.java**

@RestController

public class EmployeeController {

@Autowired

private EmployeeService employeeService;

@GetMapping("/employees")

public List<Employee> getAllEmployees() {

return employeeService.getAllEmployees();

}

}

**DepartmentController.java**

@RestController

public class DepartmentController {

@Autowired

private DepartmentService departmentService;

@GetMapping("/departments")

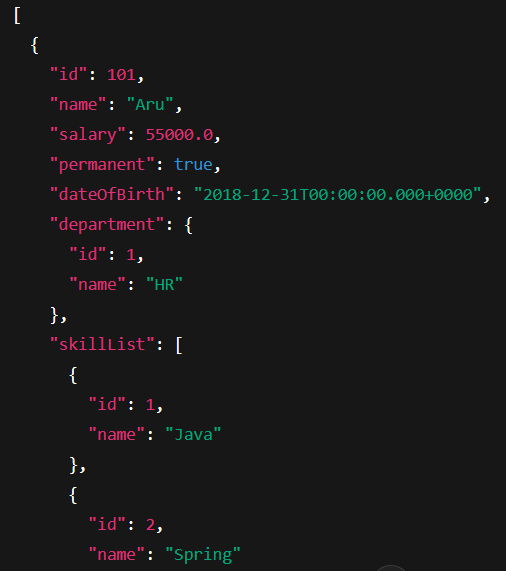
public List<Department> getAllDepartments() {

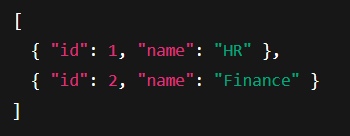
return departmentService.getAllDepartments();

}

}

**Output:**

****

****