**MANDATORY HANDS-ON**

**(includes all other hands-on of week 4)**

**WEEK-4**

**SPRING REST USING SPRING BOOT**

**SPRING REST HANDS-ON (PDF 1)**

**Hands on 1**

**Create a Spring Web Project using Maven**mvn clean package -Dhttp.proxyHost=proxy.cognizant.com -Dhttp.proxyPort=6050 -Dhttps.proxyHost=proxy.cognizant.com -Dhttps.proxyPort=6050 -Dhttp.proxyUser=123456

@SpringBootApplication

public class SpringLearnApplication {

public static void main(String[] args) {

SpringApplication.run(SpringLearnApplication.class, args);

}

}

**Hands on 2**

**Spring Core – Load SimpleDateFormat from Spring Configuration XML**

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>

**displayDate() in SpringLearnApplication.java:**

public void displayDate() {

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

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

try {

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

System.out.println(date);

} catch (ParseException e) {

e.printStackTrace();

}

}

**Hands on 3**

**Spring Core - 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:

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

public void displayDate() {

LOGGER.info("START");

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

LOGGER.info("END");

}

**Hands on 4**

**Spring Core – Load Country from Spring Configuration XML**

**Country.java**

public class Country {

private String code;

private String name;

public Country() {

LOGGER.debug("Inside Country Constructor.");

}

public void setCode(String code) {

LOGGER.debug("Setting code: {}", code);

this.code = code;

}

public void setName(String name) {

LOGGER.debug("Setting name: {}", name);

this.name = name;

}

public String getCode() {

LOGGER.debug("Getting code: {}", code);

return code;

}

public String getName() {

LOGGER.debug("Getting name: {}", name);

return name;

}

@Override

public String toString() {

return "Country [code=" + code + ", name=" + name + "]";

}

}

country.xml

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

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

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

</bean>

displayCountry() method

public void displayCountry() {

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

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

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

}

**Hands on 5**

**Spring Core – Demonstration of Singleton Scope and Prototype Scope**

Default (Singleton)

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

Country country1 = context.getBean("country", Country.class);

Country country2 = context.getBean("country", Country.class);

Prototype Scope in country.xml

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

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

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

</bean>

**Hands on 6**

**Spring Core – Load list of countries from Spring Configuration XML**   
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>

displayCountries() method

public void displayCountries() {

LOGGER.info("START");

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

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

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

LOGGER.info("END");

}

**SPRING REST HANDS-ON(PDF 3)**

**Problem Statement - Display Employee List and Edit Employee form using RESTful Web Service**

Activity 1: Create Static Data in employee.xml

**Step 1: Define Skills, Departments, and Employees**

<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

http://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="Angular"/>

</bean>

<bean id="department1" class="com.cognizant.model.Department">

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

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

</bean>

<bean id="department2" class="com.cognizant.model.Department">

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

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

</bean>

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

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

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

<property name="salary" value="40000"/>

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

<property name="dateOfBirth" value="1990-01-01"/>

<property name="department" ref="department1"/>

<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>

</beans>

Activity 2: Create Spring REST Service

Step 1: Configure Spring to Load XML

@Configuration

@ImportResource("classpath:employee.xml")

public class SpringConfig {}

Step 2: Create EmployeeDao.java

@Component

public class EmployeeDao {

private static List<Employee> EMPLOYEE\_LIST;

@Autowired

public EmployeeDao(List<Employee> employeeList) {

EmployeeDao.EMPLOYEE\_LIST = employeeList;

}

public List<Employee> getAllEmployees() {

return EMPLOYEE\_LIST;

}

public Employee getEmployeeById(int id) {

return EMPLOYEE\_LIST.stream()

.filter(e -> e.getId() == id)

.findFirst()

.orElse(null);

}

}

Step 3: Create EmployeeService.java

@Service

public class EmployeeService {

@Autowired

private EmployeeDao employeeDao;

@Transactional

public List<Employee> getAllEmployees() {

return employeeDao.getAllEmployees();

}

@Transactional

public Employee getEmployeeById(int id) {

return employeeDao.getEmployeeById(id);

}

}

@Service

public class EmployeeService {

@Autowired

private EmployeeDao employeeDao;

@Transactional

public List<Employee> getAllEmployees() {

return employeeDao.getAllEmployees();

}

@Transactional

public Employee getEmployeeById(int id) {

return employeeDao.getEmployeeById(id);

}

}

Step 4: Create EmployeeController.java

@RestController

@RequestMapping("/employees")

@CrossOrigin(origins = "http://localhost:4200")

public class EmployeeController {

@Autowired

private EmployeeService employeeService;

@GetMapping

public List<Employee> getAllEmployees() {

return employeeService.getAllEmployees();

}

@GetMapping("/{id}")

public Employee getEmployeeById(@PathVariable int id) {

return employeeService.getEmployeeById(id);

}

}

Activity 3: Angular App Integration

Step 1: Create Service to Call REST

@Injectable({ providedIn: 'root' })

export class EmployeeService {

constructor(private http: HttpClient) {}

getEmployees(): Observable<Employee[]> {

return this.http.get<Employee[]>('http://localhost:8080/employees');

}

getEmployeeById(id: number): Observable<Employee> {

return this.http.get<Employee>(`http://localhost:8080/employees/${id}`);

}

}

Step 2: Display Employee List

ngOnInit() {

this.employeeService.getEmployees().subscribe(data => {

this.employees = data;

});

}

<tr \*ngFor="let emp of employees">

<td>{{ emp.name }}</td>

<td>{{ emp.department.name }}</td>

<td><button (click)="edit(emp.id)">Edit</button></td>

</tr>

Step 3: Load Data into Edit Form

ngOnInit() {

const id = this.route.snapshot.params['id'];

this.employeeService.getEmployeeById(id).subscribe(data => {

this.employeeForm.setValue(data);

});

}

**Create static employee list data using spring xml configuration**

employee.xml Configuration

<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

http://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="Angular"/>

</bean>

<bean id="department1" class="com.cognizant.model.Department">

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

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

</bean>

<bean id="department2" class="com.cognizant.model.Department">

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

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

</bean>

<bean id="department3" class="com.cognizant.model.Department">

<property name="id" value="3"/>

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

</bean>

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

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

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

<property name="salary" value="40000"/>

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

<property name="dateOfBirth" value="1990-01-01"/>

<property name="department" ref="department1"/>

<property name="skillList">

<list>

<ref bean="skill1"/>

<ref bean="skill2"/>

</list>

</property>

</bean>

<bean id="employee2" class="com.cognizant.model.Employee">

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

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

<property name="salary" value="35000"/>

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

<property name="dateOfBirth" value="1992-05-12"/>

<property name="department" ref="department2"/>

<property name="skillList">

<list>

<ref bean="skill1"/>

</list>

</property>

</bean>

<bean id="employee3" class="com.cognizant.model.Employee">

<property name="id" value="3"/>

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

<property name="salary" value="45000"/>

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

<property name="dateOfBirth" value="1988-10-20"/>

<property name="department" ref="department1"/>

<property name="skillList">

<list>

<ref bean="skill2"/>

</list>

</property>

</bean>

<bean id="employee4" class="com.cognizant.model.Employee">

<property name="id" value="4"/>

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

<property name="salary" value="42000"/>

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

<property name="dateOfBirth" value="1991-03-15"/>

<property name="department" ref="department3"/>

<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"/>

<ref bean="employee2"/>

<ref bean="employee3"/>

<ref bean="employee4"/>

</list>

</constructor-arg>

</bean>

</beans>

Update EmployeeDao.java

package com.cognizant.dao;

import com.cognizant.model.Employee;

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

import org.springframework.stereotype.Component;

import java.util.List;

@Component

public class EmployeeDao {

private static List<Employee> EMPLOYEE\_LIST;

@Autowired

public EmployeeDao(List<Employee> employeeList) {

EmployeeDao.EMPLOYEE\_LIST = employeeList;

}

public List<Employee> getAllEmployees() {

return EMPLOYEE\_LIST;

}

}

Load XML Configuration in Spring Boot

@Configuration

@ImportResource("classpath:employee.xml")

public class AppConfig {}

@SpringBootApplication

@ImportResource("classpath:employee.xml")

public class SpringLearnApplication {

public static void main(String[] args) {

SpringApplication.run(SpringLearnApplication.class, args);

}

}

**Create REST service to gets all employees**

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;

@Service

public class EmployeeService {

@Autowired

private EmployeeDao employeeDao;

@Transactional

public List<Employee> getAllEmployees() {

return employeeDao.getAllEmployees();

}

}

Step 2: 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.\*;

import java.util.List;

@RestController

@CrossOrigin(origins = "http://localhost:4200")

public class EmployeeController {

@Autowired

private EmployeeService employeeService;

@GetMapping("/employees")

public List<Employee> getAllEmployees() {

System.out.println("Fetching all employees...");

return employeeService.getAllEmployees();

}

}

**Create REST service for department**

Update employee.xml to Include Department List

<bean id="department1" class="com.cognizant.model.Department">

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

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

</bean>

<bean id="department2" class="com.cognizant.model.Department">

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

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

</bean>

<bean id="department3" class="com.cognizant.model.Department">

<property name="id" value="3"/>

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

</bean>

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

<constructor-arg>

<list>

<ref bean="department1"/>

<ref bean="department2"/>

<ref bean="department3"/>

</list>

</constructor-arg>

</bean>

Create DepartmentDao.java

package com.cognizant.dao;

import com.cognizant.model.Department;

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

import org.springframework.stereotype.Component;

import java.util.List;

@Component

public class DepartmentDao {

private static List<Department> DEPARTMENT\_LIST;

@Autowired

public DepartmentDao(List<Department> departmentList) {

DepartmentDao.DEPARTMENT\_LIST = departmentList;

}

public List<Department> getAllDepartments() {

return DEPARTMENT\_LIST;

}

}

Create 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 org.springframework.transaction.annotation.Transactional;

import java.util.List;

@Service

public class DepartmentService {

@Autowired

private DepartmentDao departmentDao;

@Transactional

public List<Department> getAllDepartments() {

return departmentDao.getAllDepartments();

}

}

Create 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.\*;

import java.util.List;

@RestController

@CrossOrigin(origins = "http://localhost:4200")

public class DepartmentController {

@Autowired

private DepartmentService departmentService;

@GetMapping("/departments")

public List<Department> getAllDepartments() {

System.out.println("Fetching all departments...");

return departmentService.getAllDepartments();

}

}

Load XML Config in Spring Boot

@SpringBootApplication

@ImportResource("classpath:employee.xml")

public class SpringLearnApplication {

public static void main(String[] args) {

SpringApplication.run(SpringLearnApplication.class, args);

}

}

SPRING REST HANDS-ON (PDF 2)

REST - Country Web Service

package com.cognizant.spring-learn.controller;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

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

@RestController

public class HelloController {

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

@GetMapping("/hello")

public String sayHello() {

LOGGER.info("START: sayHello()");

String message = "Hello World!!";

LOGGER.info("END: sayHello()");

return message;

}

}

REST - Get country based on country code

country.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

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

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

<constructor-arg>

<list>

<bean class="com.cognizant.spring-learn.model.Country">

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

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

</bean>

<bean class="com.cognizant.spring-learn.model.Country">

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

<property name="name" value="United States"/>

</bean>

<bean class="com.cognizant.spring-learn.model.Country">

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

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

</bean>

</list>

</constructor-arg>

</bean>

</beans>

CountryService

package com.cognizant.spring-learn.service;

import com.cognizant.spring-learn.model.Country;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class CountryService {

public Country getCountry(String code) {

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

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

return countryList.stream()

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

.findFirst()

.orElse(null);

}

}

CountryController Implementation

package com.cognizant.spring-learn.controller;

import com.cognizant.spring-learn.model.Country;

import com.cognizant.spring-learn.service.CountryService;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

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

@RestController

public class CountryController {

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

@Autowired

private CountryService countryService;

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

public Country getCountry(@PathVariable String code) {

LOGGER.info("START: getCountry()");

Country country = countryService.getCountry(code);

LOGGER.info("END: getCountry()");

return country;

}

}

Add Bean Scanning in SpringLearnApplication.java

@SpringBootApplication

@ComponentScan(basePackages = "com.cognizant.spring-learn")

public class SpringLearnApplication {

public static void main(String[] args) {

SpringApplication.run(SpringLearnApplication.class, args);

}

}

**JWT HANDS-ON**

**Create authentication service that returns JWT**

AuthenticationController.java

package com.cognizant.springsecurity.controller;

import com.cognizant.springsecurity.util.JwtUtil;

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

import org.springframework.http.ResponseEntity;

import org.springframework.security.core.Authentication;

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

@RestController

public class AuthenticationController {

@Autowired

private JwtUtil jwtUtil;

@GetMapping("/authenticate")

public ResponseEntity<?> authenticate(Authentication authentication) {

String username = authentication.getName();

String token = jwtUtil.generateToken(username);

return ResponseEntity.ok().body(new TokenResponse(token));

}

public static class TokenResponse {

private String token;

public TokenResponse(String token) { this.token = token; }

public String getToken() { return token; }

public void setToken(String token) { this.token = token; }

}

}

SecurityConfig.java

package com.cognizant.springsecurity.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.authentication.configuration.EnableGlobalAuthentication;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.core.userdetails.\*;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import org.springframework.security.crypto.password.PasswordEncoder;

import org.springframework.security.web.SecurityFilterChain;

@Configuration

public class SecurityConfig {

@Bean

public InMemoryUserDetailsManager userDetailsService() {

UserDetails user = User

.withUsername("user")

.password(passwordEncoder().encode("pwd"))

.roles("USER")

.build();

return new InMemoryUserDetailsManager(user);

}

@Bean

public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

http

.csrf().disable()

.authorizeHttpRequests()

.requestMatchers("/authenticate").authenticated()

.and()

.httpBasic();

return http.build();

}

@Bean

public PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

}

JwtUtil.java

package com.cognizant.springsecurity.util;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import org.springframework.stereotype.Component;

import java.util.Date;

@Component

public class JwtUtil {

private String secretKey = "secret";

private long expirationMs = 60 \* 60 \* 1000;

public String generateToken(String username) {

return Jwts.builder()

.setSubject(username)

.setIssuedAt(new Date(System.currentTimeMillis()))

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

.signWith(SignatureAlgorithm.HS256, secretKey)

.compact();

}

}