1. Spring-rest-handson

Hands-on 1: Create Spring Boot Application

SpringLearnApplication.java

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 from XML

date-format.xml

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">

   <bean id="dateFormat" class="java.text.SimpleDateFormat">
      <constructor-arg value="dd/MM/yyyy"/>
   </bean>
</beans>
```

SpringLearnApplication.java

java

```
public 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("Date: {}", date.toString());
   } catch (ParseException e) {
      LOGGER.error("Parse Error", e);
   }
   LOGGER.info("END");
}
```

Hands-on 3: Logging Configuration

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
```

Hands-on 4: Load Country from XML

Country.java

java

```
package com.cognizant.springlearn;

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

public class Country {
   private static final Logger LOGGER = LoggerFactory.getLogger(Country.class);

   private String code;
   private String name;
```

```
    public Country() {
        LOGGER.debug("Inside Country Constructor");
    }

    public String getCode() {
        LOGGER.debug("Inside getCode()");
        return code;
    }

    public void setCode(String code) {
        LOGGER.debug("Inside setCode()");
        this.code = code;
    }

    public String getName() {
        LOGGER.debug("Inside getName()");
        return name;
    }

    public void setName(String name) {
        LOGGER.debug("Inside setName()");
        this.name = name;
    }

    @Override
    public String toString() {
        return "Country [code=" + code + ", name=" + name + "]";
    }
}
```

country.xml (in src/main/resources)

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="country" class="com.cognizant.springlearn.Country">
        <property name="code" value="IN"/>
        <property name="name" value="India"/>
    </bean>
</beans>
```

SpringLearnApplication.java

java

```
public 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 Scope**:

java

```
ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");
Country country = context.getBean("country", Country.class);
Country anotherCountry = context.getBean("country", Country.class);
```

To test **Prototype Scope**, modify country.xml:

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: Load List of Countries from XML

country.xml

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

java

```
public void displayCountries() {
   LOGGER.info("START");
   ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");
   List<Country> countryList = context.getBean("countryList", ArrayList.class);
   for (Country c : countryList) {
      LOGGER.debug("Country: {}", c.toString());
   }
   LOGGER.info("END");
}
```

Final Integration: Call All Methods in main()

SpringLearnApplication.java

java

```
public static void main(String[] args) {
   LOGGER.info("START");
   SpringApplication.run(SpringLearnApplication.class, args);

   SpringLearnApplication app = new SpringLearnApplication();
```

```
    app.displayDate();
    app.displayCountry();
    app.displayCountries();

    LOGGER.info("END");
}
```

---

2. Spring-rest-handson

HelloController.java

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;

@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

java

```
package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.Country;
```

```
import com.cognizant.springlearn.service.CountryService;
import com.cognizant.springlearn.service.exception.CountryNotFoundException;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.springframework.web.bind.annotation.*;

import java.util.ArrayList;
import java.util.List;

@RestController
public class CountryController {

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

    @RequestMapping("/country")
    public Country getCountryIndia() {
        LOGGER.info("START");
        ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");
        Country country = context.getBean("country", Country.class);
        LOGGER.debug("Country: {}", country.toString());
        LOGGER.info("END");
        return country;
    }

    @GetMapping("/countries")
    public List<Country> getAllCountries() {
        LOGGER.info("START");
        ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");
        List<Country> countries = (ArrayList<Country>) context.getBean("countryList");
        LOGGER.debug("Countries: {}", countries);
        LOGGER.info("END");
        return countries;
    }

    @GetMapping("/countries/{code}")
    public Country getCountry(@PathVariable String code) throws CountryNotFoundException {
        LOGGER.info("START");
        CountryService service = new CountryService();
        Country country = service.getCountry(code);
        LOGGER.debug("Country: {}", country);
        LOGGER.info("END");
        return country;
```

```
    }
}
```

CountryService.java

java

```
package com.cognizant.springlearn.service;

import com.cognizant.springlearn.Country;
import com.cognizant.springlearn.service.exception.CountryNotFoundException;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

import java.util.ArrayList;
import java.util.List;

public class CountryService {
    public Country getCountry(String code) throws CountryNotFoundException {
        ApplicationContext context = new ClassPathXmlApplicationContext("country.xml");
        List<Country> countries = (ArrayList<Country>) context.getBean("countryList");
        return countries.stream()
                .filter(c -> c.getCode().equalsIgnoreCase(code))
                .findFirst()
                .orElseThrow(() -> new CountryNotFoundException("Country not found"));
    }
}
```

CountryNotFoundException.java

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 {
    public CountryNotFoundException(String message) {
        super(message);
    }
```

}

SpringLearnApplicationTests.java

java

```
package com.cognizant.springlearn;

import com.cognizant.springlearn.controller.CountryController;
import org.junit.jupiter.api.Test;
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.*;

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;

@SpringBootTest
@AutoConfigureMockMvc
public 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("$.code").exists());
        actions.andExpect(jsonPath("$.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"));
    }
}
```

---

3. Spring-rest-handson

employee.xml (in src/main/resources)

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

    <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>

    <bean id="dept1" class="com.cognizant.springlearn.model.Department">
        <property name="id" value="1"/>
        <property name="name" value="Information Technology"/>
    </bean>

    <bean id="dept2" class="com.cognizant.springlearn.model.Department">
        <property name="id" value="2"/>
        <property name="name" value="HR"/>
    </bean>
```

```
<bean id="employee1" class="com.cognizant.springlearn.model.Employee">
   <property name="id" value="1"/>
   <property name="name" value="John"/>
   <property name="salary" value="50000"/>
   <property name="permanent" value="true"/>
   <property name="department" ref="dept1"/>
   <property name="skills">
      <list>
         <ref bean="skill1"/>
         <ref bean="skill2"/>
      </list>
   </property>
</bean>

<bean id="employee2" class="com.cognizant.springlearn.model.Employee">
   <property name="id" value="2"/>
   <property name="name" value="Jane"/>
   <property name="salary" value="60000"/>
   <property name="permanent" value="false"/>
   <property name="department" ref="dept2"/>
   <property name="skills">
      <list>
         <ref bean="skill1"/>
      </list>
   </property>
</bean>

<bean id="employeeList" class="java.util.ArrayList">
   <constructor-arg>
      <list>
         <ref bean="employee1"/>
         <ref bean="employee2"/>
      </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

java

```
package com.cognizant.springlearn.dao;

import com.cognizant.springlearn.model.Employee;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.springframework.stereotype.Repository;

import java.util.ArrayList;

@Repository
public class EmployeeDao {

    private static ArrayList<Employee> EMPLOYEE_LIST;

    public EmployeeDao() {
        ApplicationContext context = new ClassPathXmlApplicationContext("employee.xml");
        EMPLOYEE_LIST = (ArrayList<Employee>) context.getBean("employeeList");
    }

    public ArrayList<Employee> getAllEmployees() {
        return EMPLOYEE_LIST;
    }
}
```

EmployeeService.java

java

```
package com.cognizant.springlearn.service;

import com.cognizant.springlearn.dao.EmployeeDao;
import com.cognizant.springlearn.model.Employee;
import org.springframework.beans.factory.annotation.Autowired;
```

```
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.util.ArrayList;

@Service
public class EmployeeService {

    @Autowired
    private EmployeeDao employeeDao;

    @Transactional
    public ArrayList<Employee> getAllEmployees() {
        return employeeDao.getAllEmployees();
    }
}
```

EmployeeController.java

java

```
package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.model.Employee;
import com.cognizant.springlearn.service.EmployeeService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.ArrayList;

@RestController
public class EmployeeController {

    @Autowired
    private EmployeeService employeeService;

    @GetMapping("/employees")
    public ArrayList<Employee> getAllEmployees() {
        return employeeService.getAllEmployees();
    }
}
```

DepartmentDao.java

java

```
package com.cognizant.springlearn.dao;

import com.cognizant.springlearn.model.Department;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.springframework.stereotype.Repository;

import java.util.ArrayList;

@Repository
public class DepartmentDao {

    private static ArrayList<Department> DEPARTMENT_LIST;

    public DepartmentDao() {
        ApplicationContext context = new ClassPathXmlApplicationContext("employee.xml");
        DEPARTMENT_LIST = (ArrayList<Department>) context.getBean("departmentList");
    }

    public ArrayList<Department> getAllDepartments() {
        return DEPARTMENT_LIST;
    }
}
```

DepartmentService.java

java

```
package com.cognizant.springlearn.service;

import com.cognizant.springlearn.dao.DepartmentDao;
import com.cognizant.springlearn.model.Department;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.ArrayList;
```

```
@Service
public class DepartmentService {

    @Autowired
    private DepartmentDao departmentDao;

    public ArrayList<Department> getAllDepartments() {
        return departmentDao.getAllDepartments();
    }
}
```

DepartmentController.java

java

```
package com.cognizant.springlearn.controller;

import com.cognizant.springlearn.model.Department;
import com.cognizant.springlearn.service.DepartmentService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.ArrayList;

@RestController
public class DepartmentController {

    @Autowired
    private DepartmentService departmentService;

    @GetMapping("/departments")
    public ArrayList<Department> getAllDepartments() {
        return departmentService.getAllDepartments();
    }
}
```

---

4. Spring-rest-handson

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

```
@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.info(country.toString());
      return country;
   }
}
```

GlobalExceptionHandler.java

```
@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");
        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");
        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) {
            InvalidFormatException ife = (InvalidFormatException) ex.getCause();
            for (InvalidFormatException.Reference ref : ife.getPath()) {
                body.put("message", "Incorrect format for field '" + ref.getFieldName() + "'");
            }
        }
        return new ResponseEntity<>(body, headers, status);
    }
}
```

Employee.java

```
@JsonIgnoreProperties(ignoreUnknown = true)
public class Employee {
    @NotNull
    private Integer id;

    @NotBlank
```

```
    @Size(min = 1, max = 30)
    private String name;

    @NotNull
    @Min(0)
    private Double salary;

    @NotNull
    private Boolean permanent;

    @JsonFormat(shape = JsonFormat.Shape.STRING, pattern = "dd/MM/yyyy")
    private Date dateOfBirth;

    @NotNull
    private Department department;

    private List<Skill> skills;
    // Getters and setters
}
```

```
Department.java
public class Department {
    @NotNull
    private Integer id;

    @NotBlank
    @Size(min = 1, max = 30)
    private String name;
    // Getters and setters
}

// --- Skill.java ---

public class Skill {
    @NotNull
    private Integer id;

    @NotBlank
    @Size(min = 1, max = 30)
    private String name;
    // Getters and setters
}
```

EmployeeController.java

```
@RestController
@RequestMapping("/employees")
public class EmployeeController {

    @Autowired
    private EmployeeService employeeService;

    @PutMapping
            public void updateEmployee(@RequestBody @Valid Employee employee) throws
     EmployeeNotFoundException {
        employeeService.updateEmployee(employee);
    }

    @DeleteMapping("/{id}")
    public void deleteEmployee(@PathVariable int id) throws EmployeeNotFoundException {
        employeeService.deleteEmployee(id);
    }
}
```

EmployeeService.java

```
@Service
public class EmployeeService {

    @Autowired
    private EmployeeDao employeeDao;

    public void updateEmployee(Employee employee) throws EmployeeNotFoundException {
        employeeDao.updateEmployee(employee);
    }

    public void deleteEmployee(int id) throws EmployeeNotFoundException {
        employeeDao.deleteEmployee(id);
    }
}
```

EmployeeDao.java

```
@Repository
public class EmployeeDao {

    private static List<Employee> EMPLOYEE_LIST = new ArrayList<>();
```

```
    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().equals(updatedEmp.getId())) {
                EMPLOYEE_LIST.set(i, updatedEmp);
                found = true;
                break;
            }
        }
        if (!found) throw new EmployeeNotFoundException("Employee not found");
    }

    public void deleteEmployee(int id) throws EmployeeNotFoundException {
        boolean removed = EMPLOYEE_LIST.removeIf(e -> e.getId() == id);
        if (!removed) throw new EmployeeNotFoundException("Employee not found");
    }
}
```

EmployeeNotFoundException.java

```
@ResponseStatus(HttpStatus.NOT_FOUND)
public class EmployeeNotFoundException extends Exception {
    public EmployeeNotFoundException(String message) {
        super(message);
    }
}
```

---

5. JWT-handson

pom.xml - Add Dependencies

```
<dependencies>
   <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-security</artifactId>
   </dependency>
   <dependency>
      <groupId>io.jsonwebtoken</groupId>
      <artifactId>jjwt</artifactId>
      <version>0.9.0</version>
   </dependency>
</dependencies>
```

AuthenticationController.java

```
@RestController
public class AuthenticationController {
   private static final Logger LOGGER = LoggerFactory.getLogger(AuthenticationController.class);

   @GetMapping("/authenticate")
   public Map<String, String> authenticate(@RequestHeader("Authorization") String authHeader) {
      LOGGER.info("Start");
      LOGGER.debug("{}", authHeader);

      String user = getUser(authHeader);
      String token = generateJwt(user);

      Map<String, String> map = new HashMap<>();
      map.put("token", token);

      LOGGER.info("End");
      return map;
   }

   private String getUser(String authHeader) {
      String encoded = authHeader.replace("Basic ", "");
      byte[] decoded = Base64.getDecoder().decode(encoded);
      String decodedStr = new String(decoded);
      return decodedStr.split(":")[0];
   }

   private String generateJwt(String user) {
```

```
        JwtBuilder builder = Jwts.builder();
        builder.setSubject(user);
        builder.setIssuedAt(new Date());
        builder.setExpiration(new Date(System.currentTimeMillis() + 1200000));
        builder.signWith(SignatureAlgorithm.HS256, "secretkey");
        return builder.compact();
    }
}
```

JwtAuthorizationFilter.java

```
public class JwtAuthorizationFilter extends BasicAuthenticationFilter {
    private static final Logger LOGGER = LoggerFactory.getLogger(JwtAuthorizationFilter.class);

    public JwtAuthorizationFilter(AuthenticationManager authenticationManager) {
        super(authenticationManager);
        LOGGER.info("Start");
    }

    @Override
        protected void doFilterInternal(HttpServletRequest req, HttpServletResponse res, FilterChain
    chain)
            throws IOException, ServletException {
        LOGGER.info("Start");
        String header = req.getHeader("Authorization");
        LOGGER.debug("{}", header);

        if (header == null || !header.startsWith("Bearer ")) {
            chain.doFilter(req, res);
            return;
        }

        UsernamePasswordAuthenticationToken authentication = getAuthentication(req);
        SecurityContextHolder.getContext().setAuthentication(authentication);
        chain.doFilter(req, res);
        LOGGER.info("End");
    }

    private UsernamePasswordAuthenticationToken getAuthentication(HttpServletRequest request) {
        String token = request.getHeader("Authorization");
        if (token != null) {
            try {
                Jws<Claims> jws = Jwts.parser()
                    .setSigningKey("secretkey")
                    .parseClaimsJws(token.replace("Bearer ", ""));
```

```
            String user = jws.getBody().getSubject();
            LOGGER.debug("{}", user);
            if (user != null) {
               return new UsernamePasswordAuthenticationToken(user, null, new ArrayList<>());
            }
         } catch (JwtException ex) {
            return null;
         }
      }
      return null;
   }
}
```

SecurityConfig.java

```
@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {
   private static final Logger LOGGER = LoggerFactory.getLogger(SecurityConfig.class);

   @Override
   protected void configure(AuthenticationManagerBuilder auth) throws Exception {
      auth.inMemoryAuthentication()
         .withUser("admin").password(passwordEncoder().encode("pwd")).roles("ADMIN")
         .and()
         .withUser("user").password(passwordEncoder().encode("pwd")).roles("USER");
   }

   @Bean
   public PasswordEncoder passwordEncoder() {
      LOGGER.info("Start");
      return new BCryptPasswordEncoder();
   }

   @Override
   protected void configure(HttpSecurity httpSecurity) throws Exception {
      httpSecurity.csrf().disable()
         .httpBasic()
         .and()
         .authorizeRequests()
         .antMatchers("/authenticate").hasAnyRole("USER", "ADMIN")
         .anyRequest().authenticated()
         .and()
         .addFilter(new JwtAuthorizationFilter(authenticationManager()));
   }
```

}