COGNIZANT

Digital Nurture 4.0

Deep Skilling - Java FSE

WEEK-4 HANDS ON

By Kaviya P

5.JWT-HANDSON

**Securing RESTful Web Services with Spring Security**

**EmployeeNotFoundException.java**

**package** com.cognizant.spring\_learn;

**import** org.springframework.http.HttpStatus;

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

@ResponseStatus(HttpStatus.***NOT\_FOUND***)

**public** **class** EmployeeNotFoundException **extends** Exception {

**public** EmployeeNotFoundException(String message) {

**super**(message);

}

}

**GlobalExceptionHandler.java**

**package** com.cognizant.spring\_learn;

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

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**import** org.springframework.http.HttpHeaders;

**import** org.springframework.http.HttpStatusCode;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.http.converter.HttpMessageNotReadableException;

**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.\*;

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

@ControllerAdvice

**public** **class** GlobalExceptionHandler **extends** ResponseEntityExceptionHandler {

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

@Override

**protected** ResponseEntity<Object> handleMethodArgumentNotValid(MethodArgumentNotValidException ex,

HttpHeaders headers,

HttpStatusCode status,

WebRequest request) {

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

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

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

List<String> errors = ex.getBindingResult()

.getFieldErrors()

.stream()

.map(err -> err.getDefaultMessage())

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

body.put("errors", errors);

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

}

@Override

**protected** ResponseEntity<Object> handleHttpMessageNotReadable(HttpMessageNotReadableException ex,

HttpHeaders headers,

HttpStatusCode 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) {

**for** (InvalidFormatException.Reference ref : ((InvalidFormatException) ex.getCause()).getPath()) {

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

}

}

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

}

}

**SpringLearnApplication.java**

**package** com.cognizant.spring\_learn;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

**public** **class** SpringLearnApplication {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(SpringLearnApplication.**class**, args);

}

}

**EmployeeController.java**

package com.cognizant.spring\_learn.controller;

import com.cognizant.spring\_learn.EmployeeNotFoundException;

import com.cognizant.spring\_learn.model.Employee;

import com.cognizant.spring\_learn.service.EmployeeService;

import jakarta.validation.Valid;

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

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

import java.util.List;

@RestController

public class EmployeeController {

@Autowired

private EmployeeService employeeService;

@PutMapping("/employees")

public void updateEmployee(@RequestBody @Valid Employee employee) throws EmployeeNotFoundException {

employeeService.updateEmployee(employee);

}

@GetMapping("/employees")

public List<Employee> getAllEmployees() {

return employeeService.getAllEmployees();

}

@DeleteMapping("/employees/{id}")

public void deleteEmployee(@PathVariable int id) throws EmployeeNotFoundException {

employeeService.deleteEmployee(id);

}

}

**EmployeeDao.java**

**package** com.cognizant.spring\_learn.dao;

**import** com.cognizant.spring\_learn.EmployeeNotFoundException;

**import** com.cognizant.spring\_learn.model.Employee;

**import** com.cognizant.spring\_learn.model.Department;

**import** com.cognizant.spring\_learn.model.Skill;

**import** jakarta.annotation.PostConstruct;

**import** org.springframework.stereotype.Repository;

**import** java.util.\*;

@Repository

**public** **class** EmployeeDao {

**private** **static** List<Employee> *employeeList* = **new** ArrayList<>();

@PostConstruct

**public** **void** init() {

// Create sample employee

Employee emp = **new** Employee();

emp.setId(1);

emp.setName("Default Emp");

emp.setSalary((**float**)(3000.0f));

emp.setPermanent(**true**);

emp.setDateOfBirth(**null**);

Department dept = **new** Department();

dept.setId(101);

dept.setName("HR");

emp.setDepartment(dept);

Skill skill1 = **new** Skill();

skill1.setId(1);

skill1.setName("Communication");

Skill skill2 = **new** Skill();

skill2.setId(2);

skill2.setName("Teamwork");

emp.setSkillList(Arrays.*asList*(skill1, skill2));

*employeeList*.add(emp);

}

**public** **void** deleteEmployee(**int** id) **throws** EmployeeNotFoundException {

Iterator<Employee> iterator = *employeeList*.iterator();

**while** (iterator.hasNext()) {

Employee emp = iterator.next();

**if** (emp.getId() == id) {

iterator.remove();

**return**;

}

}

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

}

**public** **void** updateEmployee(Employee updatedEmployee) **throws** EmployeeNotFoundException {

**for** (**int** i = 0; i < *employeeList*.size(); i++) {

**if** (*employeeList*.get(i).getId().equals(updatedEmployee.getId())) {

*employeeList*.set(i, updatedEmployee);

**return**;

}

}

**throw** **new** EmployeeNotFoundException("Employee not found with ID: " + updatedEmployee.getId());

}

**public** List<Employee> getAllEmployees() {

**return** *employeeList*;

}

}

**Department.java**

**package** com.cognizant.spring\_learn.model;

**import** jakarta.validation.constraints.\*;

**public** **class** Department {

@NotNull

**private** Integer id;

@NotBlank

@Size(min = 1, max = 30)

**private** String name;

**public** Integer getId() {

**return** id;

}

**public** **void** setId(Integer id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

// Getters and Setters

}

**Employee.java**

**package** com.cognizant.spring\_learn.model;

**import** com.fasterxml.jackson.annotation.JsonFormat;

**import** jakarta.validation.constraints.\*;

**import** java.util.Date;

**import** java.util.List;

**public** **class** Employee {

@NotNull

**private** Integer id;

@NotBlank

@Size(min = 1, max = 30)

**private** String name;

@NotNull

@PositiveOrZero

**private** Float salary;

@NotNull

**private** Boolean permanent;

@NotNull

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

**private** Date dateOfBirth;

@NotNull

**private** Department department;

@NotNull

**private** List<Skill> skillList;

**public** Employee()

{

}

**public** Integer getId() {

**return** id;

}

**public** **void** setId(Integer id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** Float getSalary() {

**return** salary;

}

**public** **void** setSalary(Float 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> getSkillList() {

**return** skillList;

}

**public** **void** setSkillList(List<Skill> skillList) {

**this**.skillList = skillList;

}

**public** Employee(@NotNull Integer id, @NotBlank @Size(min = 1, max = 30) String name,

@NotNull @PositiveOrZero Float salary, @NotNull Boolean permanent, @NotNull Date dateOfBirth,

@NotNull Department department, @NotNull List<Skill> skillList) {

**this**.id = id;

**this**.name = name;

**this**.salary = salary;

**this**.permanent = permanent;

**this**.dateOfBirth = dateOfBirth;

**this**.department = department;

**this**.skillList = skillList;

}

// Getters and Setters

}

**Skill.java**

**package** com.cognizant.spring\_learn.model;

**import** jakarta.validation.constraints.\*;

**public** **class** Skill {

@NotNull

**private** Integer id;

@NotBlank

@Size(min = 1, max = 30)

**private** String name;

**public** Integer getId() {

**return** id;

}

**public** **void** setId(Integer id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

// Getters and Setters

}

**EmployeeService.java**

**package** com.cognizant.spring\_learn.service;

**import** com.cognizant.spring\_learn.EmployeeNotFoundException;

**import** com.cognizant.spring\_learn.dao.EmployeeDao;

**import** com.cognizant.spring\_learn.model.Employee;

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

**import** org.springframework.stereotype.Service;

**import** java.util.List;

@Service

**public** **class** EmployeeService {

@Autowired

**private** EmployeeDao employeeDao;

**public** **void** updateEmployee(Employee employee) **throws** EmployeeNotFoundException {

employeeDao.updateEmployee(employee);

}

**public** List<Employee> getAllEmployees() {

**return** employeeDao.getAllEmployees();

}

**public** **void** deleteEmployee(**int** id) **throws** EmployeeNotFoundException {

employeeDao.deleteEmployee(id);

}

}

**SecurityConfig.java**

**package** security;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.Configuration;

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

**import** org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

**import** org.springframework.security.web.SecurityFilterChain;

@Configuration

@EnableWebSecurity

**public** **class** SecurityConfig {

@Bean

**public** SecurityFilterChain filterChain(HttpSecurity http) **throws** Exception {

http

.authorizeHttpRequests((authz) -> authz

.anyRequest().authenticated()

)

.~~httpBasic~~() // Enables basic authentication

.~~and~~()

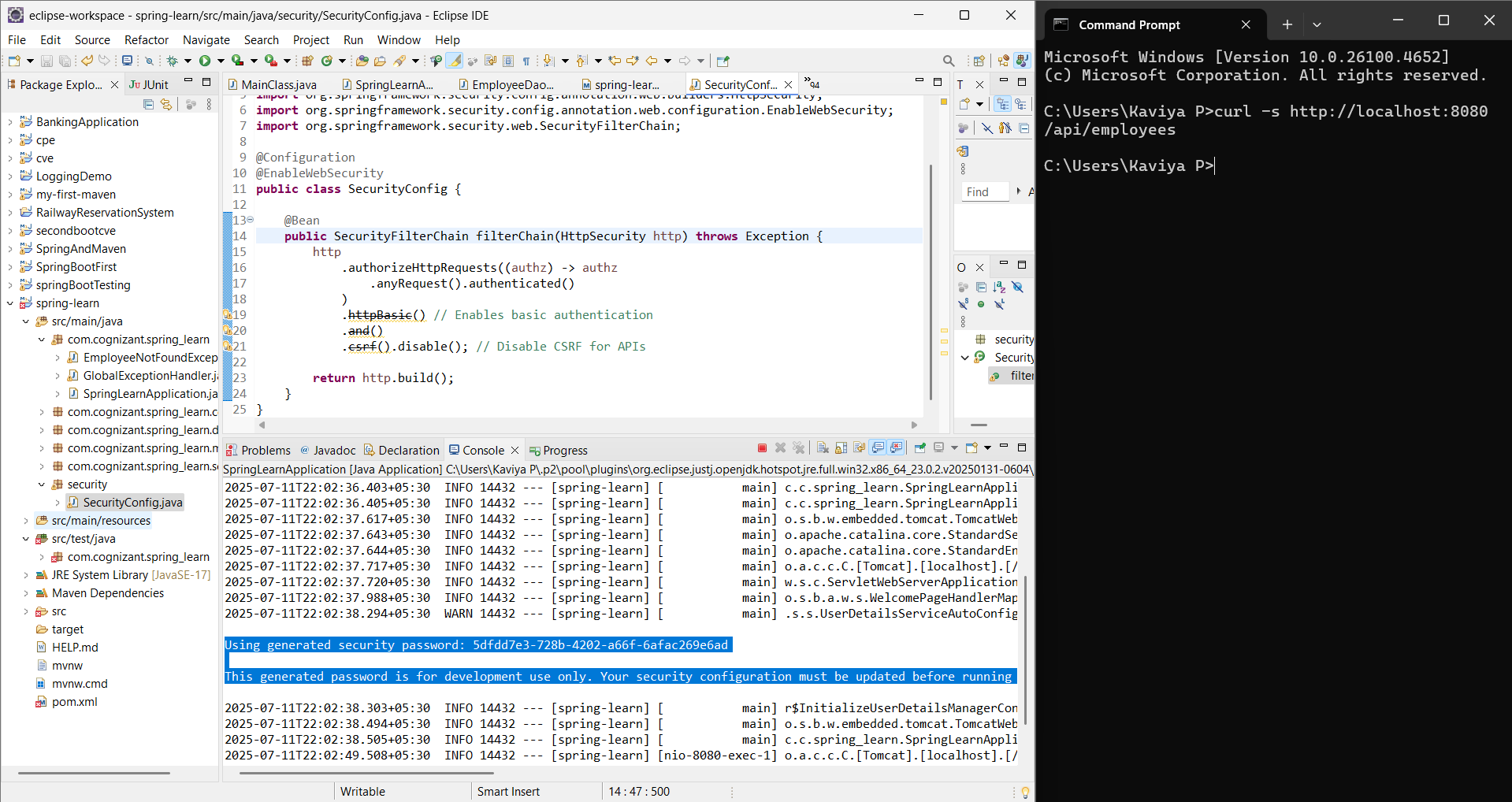
.~~csrf~~().disable(); // Disable CSRF for APIs

**return** http.build();

}

}

**OUTPUT**

****

**Creating users and roles in Spring Security**   
**EmployeeNotFoundException.java**

**package** com.cognizant.spring\_learn;

**import** org.springframework.http.HttpStatus;

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

@ResponseStatus(HttpStatus.***NOT\_FOUND***)

**public** **class** EmployeeNotFoundException **extends** Exception {

**public** EmployeeNotFoundException(String message) {

**super**(message);

}

}

**GlobalExceptionHandler.java**

**package** com.cognizant.spring\_learn;

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

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**import** org.springframework.http.HttpHeaders;

**import** org.springframework.http.HttpStatusCode;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.http.converter.HttpMessageNotReadableException;

**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.\*;

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

@ControllerAdvice

**public** **class** GlobalExceptionHandler **extends** ResponseEntityExceptionHandler {

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

@Override

**protected** ResponseEntity<Object> handleMethodArgumentNotValid(MethodArgumentNotValidException ex,

HttpHeaders headers,

HttpStatusCode status,

WebRequest request) {

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

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

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

List<String> errors = ex.getBindingResult()

.getFieldErrors()

.stream()

.map(err -> err.getDefaultMessage())

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

body.put("errors", errors);

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

}

@Override

**protected** ResponseEntity<Object> handleHttpMessageNotReadable(HttpMessageNotReadableException ex,

HttpHeaders headers,

HttpStatusCode 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) {

**for** (InvalidFormatException.Reference ref : ((InvalidFormatException) ex.getCause()).getPath()) {

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

}

}

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

}

}

**SpringLearnApplication.java**

**package** com.cognizant.spring\_learn;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

**public** **class** SpringLearnApplication {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(SpringLearnApplication.**class**, args);

}

}

**EmployeeController.java**

package com.cognizant.spring\_learn.controller;

import com.cognizant.spring\_learn.EmployeeNotFoundException;

import com.cognizant.spring\_learn.model.Employee;

import com.cognizant.spring\_learn.service.EmployeeService;

import jakarta.validation.Valid;

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

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

import java.util.List;

@RestController

public class EmployeeController {

@Autowired

private EmployeeService employeeService;

@PutMapping("/employees")

public void updateEmployee(@RequestBody @Valid Employee employee) throws EmployeeNotFoundException {

employeeService.updateEmployee(employee);

}

@GetMapping("/employees")

public List<Employee> getAllEmployees() {

return employeeService.getAllEmployees();

}

@DeleteMapping("/employees/{id}")

public void deleteEmployee(@PathVariable int id) throws EmployeeNotFoundException {

employeeService.deleteEmployee(id);

}

}

**EmployeeDao.java**

**package** com.cognizant.spring\_learn.dao;

**import** com.cognizant.spring\_learn.EmployeeNotFoundException;

**import** com.cognizant.spring\_learn.model.Employee;

**import** com.cognizant.spring\_learn.model.Department;

**import** com.cognizant.spring\_learn.model.Skill;

**import** jakarta.annotation.PostConstruct;

**import** org.springframework.stereotype.Repository;

**import** java.util.\*;

@Repository

**public** **class** EmployeeDao {

**private** **static** List<Employee> *employeeList* = **new** ArrayList<>();

@PostConstruct

**public** **void** init() {

// Create sample employee

Employee emp = **new** Employee();

emp.setId(1);

emp.setName("Default Emp");

emp.setSalary((**float**)(3000.0f));

emp.setPermanent(**true**);

emp.setDateOfBirth(**null**);

Department dept = **new** Department();

dept.setId(101);

dept.setName("HR");

emp.setDepartment(dept);

Skill skill1 = **new** Skill();

skill1.setId(1);

skill1.setName("Communication");

Skill skill2 = **new** Skill();

skill2.setId(2);

skill2.setName("Teamwork");

emp.setSkillList(Arrays.*asList*(skill1, skill2));

*employeeList*.add(emp);

}

**public** **void** deleteEmployee(**int** id) **throws** EmployeeNotFoundException {

Iterator<Employee> iterator = *employeeList*.iterator();

**while** (iterator.hasNext()) {

Employee emp = iterator.next();

**if** (emp.getId() == id) {

iterator.remove();

**return**;

}

}

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

}

**public** **void** updateEmployee(Employee updatedEmployee) **throws** EmployeeNotFoundException {

**for** (**int** i = 0; i < *employeeList*.size(); i++) {

**if** (*employeeList*.get(i).getId().equals(updatedEmployee.getId())) {

*employeeList*.set(i, updatedEmployee);

**return**;

}

}

**throw** **new** EmployeeNotFoundException("Employee not found with ID: " + updatedEmployee.getId());

}

**public** List<Employee> getAllEmployees() {

**return** *employeeList*;

}

}

**Department.java**

**package** com.cognizant.spring\_learn.model;

**import** jakarta.validation.constraints.\*;

**public** **class** Department {

@NotNull

**private** Integer id;

@NotBlank

@Size(min = 1, max = 30)

**private** String name;

**public** Integer getId() {

**return** id;

}

**public** **void** setId(Integer id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

// Getters and Setters

}

**Employee.java**

**package** com.cognizant.spring\_learn.model;

**import** com.fasterxml.jackson.annotation.JsonFormat;

**import** jakarta.validation.constraints.\*;

**import** java.util.Date;

**import** java.util.List;

**public** **class** Employee {

@NotNull

**private** Integer id;

@NotBlank

@Size(min = 1, max = 30)

**private** String name;

@NotNull

@PositiveOrZero

**private** Float salary;

@NotNull

**private** Boolean permanent;

@NotNull

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

**private** Date dateOfBirth;

@NotNull

**private** Department department;

@NotNull

**private** List<Skill> skillList;

**public** Employee()

{

}

**public** Integer getId() {

**return** id;

}

**public** **void** setId(Integer id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** Float getSalary() {

**return** salary;

}

**public** **void** setSalary(Float 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> getSkillList() {

**return** skillList;

}

**public** **void** setSkillList(List<Skill> skillList) {

**this**.skillList = skillList;

}

**public** Employee(@NotNull Integer id, @NotBlank @Size(min = 1, max = 30) String name,

@NotNull @PositiveOrZero Float salary, @NotNull Boolean permanent, @NotNull Date dateOfBirth,

@NotNull Department department, @NotNull List<Skill> skillList) {

**this**.id = id;

**this**.name = name;

**this**.salary = salary;

**this**.permanent = permanent;

**this**.dateOfBirth = dateOfBirth;

**this**.department = department;

**this**.skillList = skillList;

}

// Getters and Setters

}

**Skill.java**

**package** com.cognizant.spring\_learn.model;

**import** jakarta.validation.constraints.\*;

**public** **class** Skill {

@NotNull

**private** Integer id;

@NotBlank

@Size(min = 1, max = 30)

**private** String name;

**public** Integer getId() {

**return** id;

}

**public** **void** setId(Integer id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

// Getters and Setters

}

**EmployeeService.java**

**package** com.cognizant.spring\_learn.service;

**import** com.cognizant.spring\_learn.EmployeeNotFoundException;

**import** com.cognizant.spring\_learn.dao.EmployeeDao;

**import** com.cognizant.spring\_learn.model.Employee;

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

**import** org.springframework.stereotype.Service;

**import** java.util.List;

@Service

**public** **class** EmployeeService {

@Autowired

**private** EmployeeDao employeeDao;

**public** **void** updateEmployee(Employee employee) **throws** EmployeeNotFoundException {

employeeDao.updateEmployee(employee);

}

**public** List<Employee> getAllEmployees() {

**return** employeeDao.getAllEmployees();

}

**public** **void** deleteEmployee(**int** id) **throws** EmployeeNotFoundException {

employeeDao.deleteEmployee(id);

}

}

**SecurityConfig.java**

**package** security;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.Configuration;

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

**import** org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

**import** org.springframework.security.web.SecurityFilterChain;

@Configuration

@EnableWebSecurity

**public** **class** SecurityConfig {

@Bean

**public** SecurityFilterChain filterChain(HttpSecurity http) **throws** Exception {

http

.authorizeHttpRequests((authz) -> authz

.anyRequest().authenticated()

)

.~~httpBasic~~() // Enables basic authentication

.~~and~~()

.~~csrf~~().disable(); // Disable CSRF for APIs

**return** http.build();

}

}

**DepartmentService.java**

**package** com.cognizant.spring\_learn.service;

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

**import** org.springframework.stereotype.Service;

**import** com.cognizant.spring\_learn.dao.DepartmentDao;

**import** com.cognizant.spring\_learn.model.Department;

**import** java.util.List;

@Service

**public** **class** DepartmentService {

@Autowired

**private** DepartmentDao departmentDao;

**public** List<Department> getAllDepartments() {

**return** departmentDao.getAllDepartments();

}

}

**CountryController.java**

**package** com.cognizant.spring\_learn.controller;

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.Map;

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

@RestController

@RequestMapping

**public** **class** CountryController {

@GetMapping("/countries")

**public** List<Map<String, String>> getCountries() {

List<Map<String, String>> countries = **new** ArrayList<>();

countries.add(Map.*of*("code", "US", "name", "United States"));

countries.add(Map.*of*("code", "DE", "name", "Germany"));

countries.add(Map.*of*("code", "IN", "name", "India"));

countries.add(Map.*of*("code", "JP", "name", "Japan"));

**return** countries;

}

}

**c**

**package** com.cognizant.spring\_learn.dao;

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

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

**import** org.springframework.beans.factory.annotation.Qualifier;

**import** org.springframework.stereotype.Repository;

**import** com.cognizant.spring\_learn.model.Department;

**import** java.util.List;

@Repository

**public** **class** DepartmentDao {

**public** **static** List<Department> *DEPARTMENT\_LIST*;

@Autowired

**public** DepartmentDao(@Qualifier("departmentList") List<Department> departmentList) {

*DEPARTMENT\_LIST* = departmentList;

}

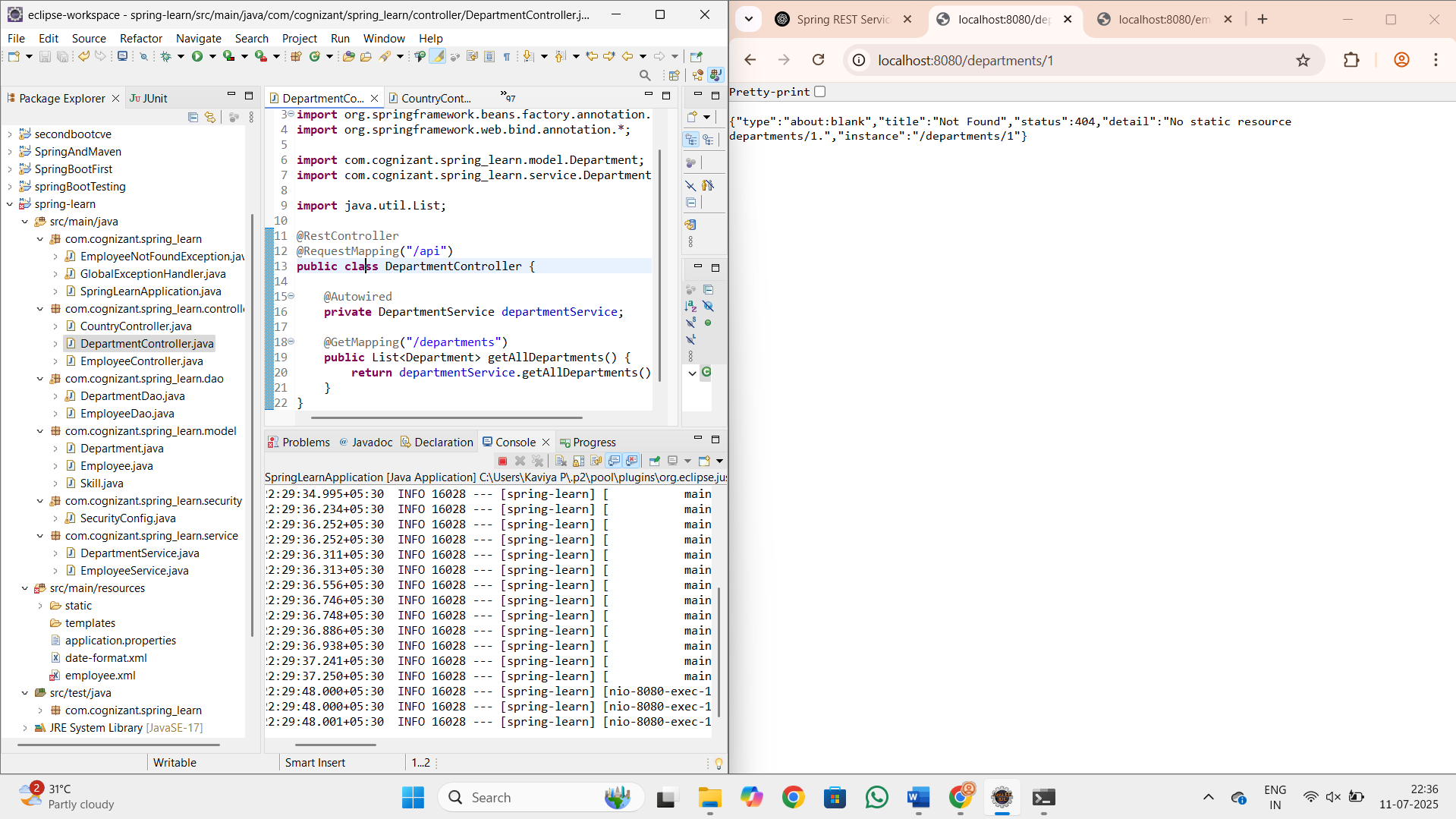
**public** List<Department> getAllDepartments() {

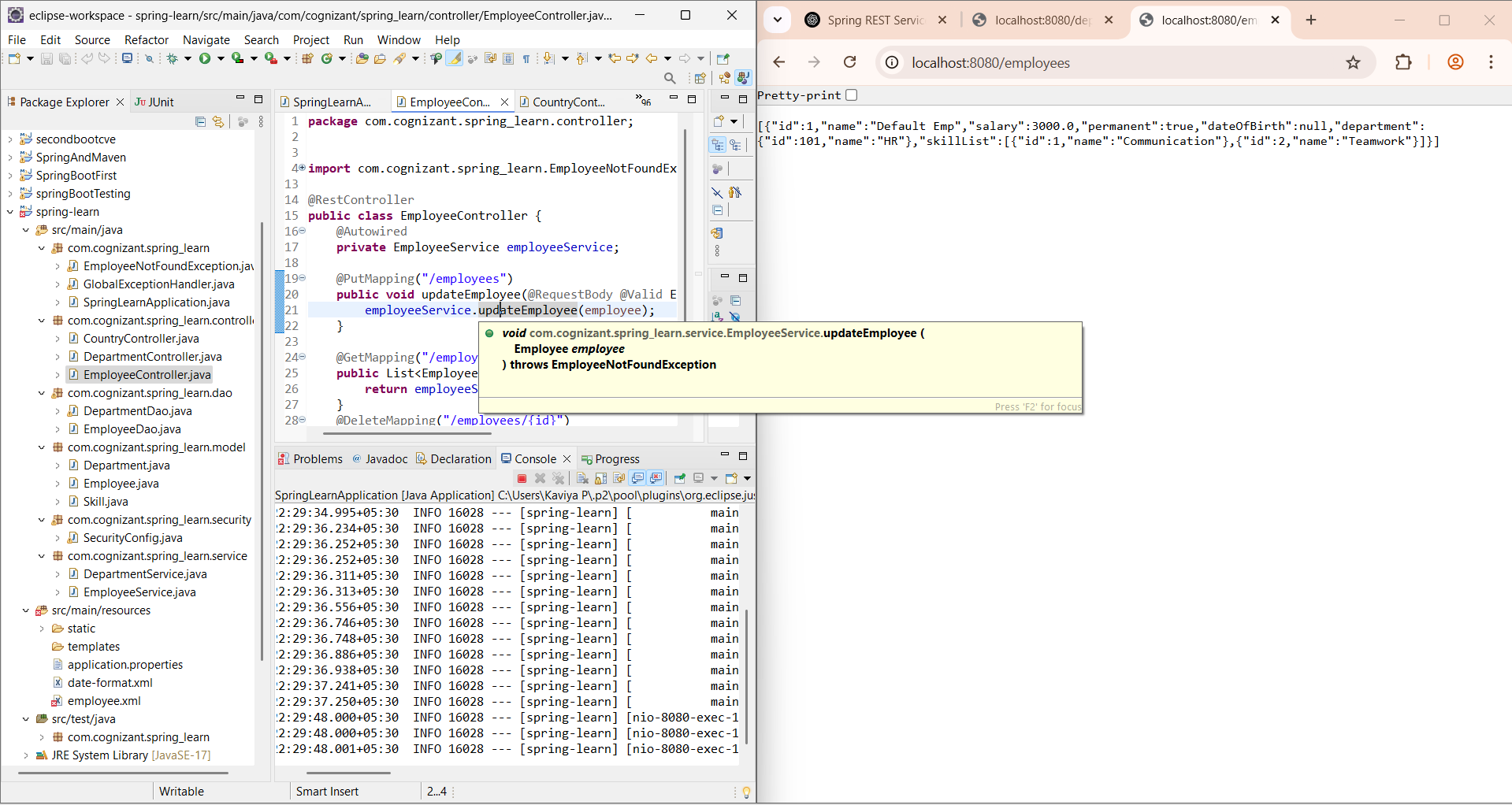
**return** *DEPARTMENT\_LIST*;

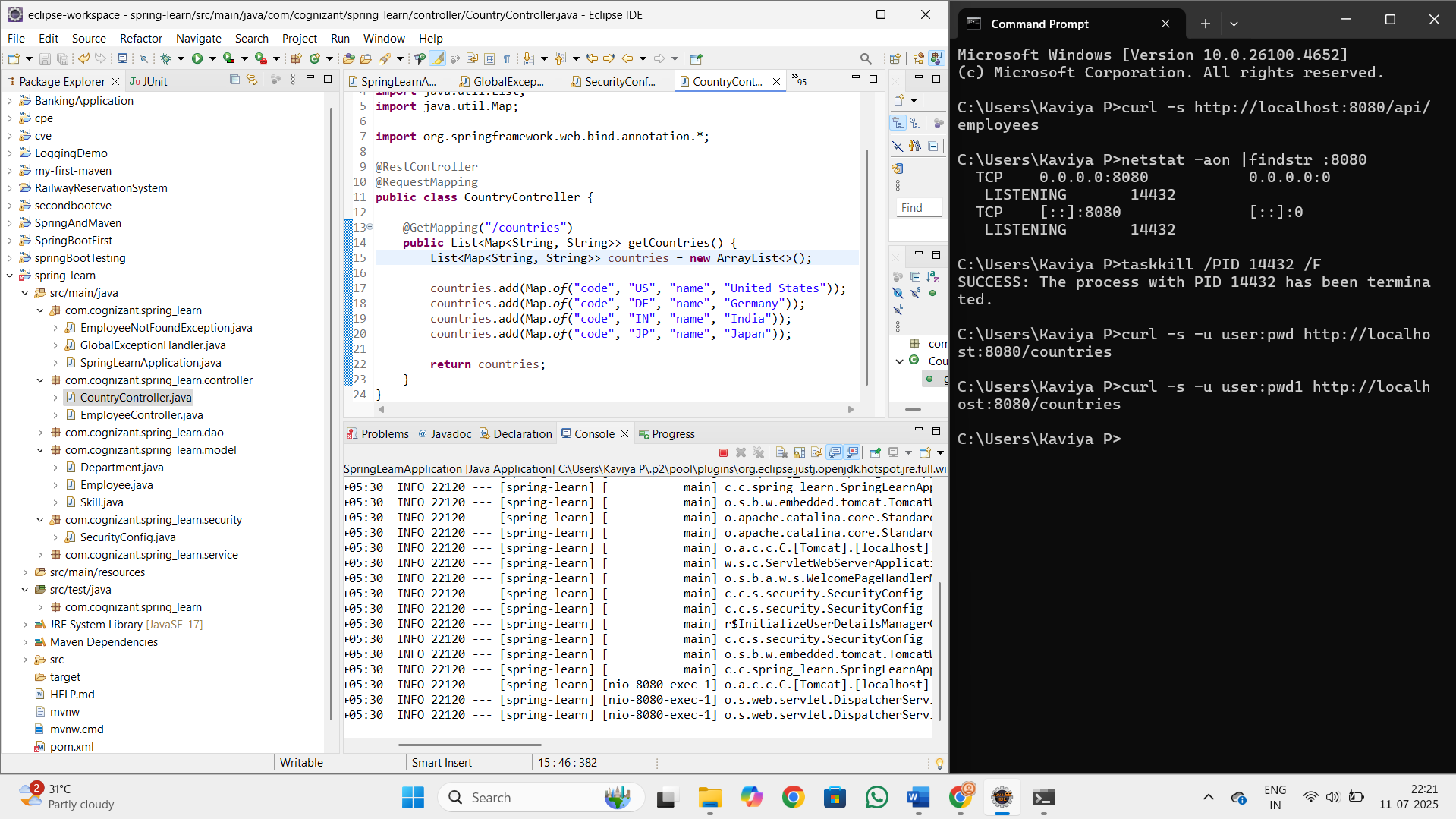
}

}

**OUTPUT**

****

****

****

**Create authentication service that returns JWT**

**JwtUtil.java**

**package** com.cognizant.spring\_learn.util;

**import** io.jsonwebtoken.Jwts;

**import** io.jsonwebtoken.SignatureAlgorithm;

**import** org.springframework.stereotype.Component;

**import** java.util.Date;

@Component

**public** **class** JwtUtil {

**private** **final** String SECRET\_KEY = "my\_secret\_key";

**public** String generateToken(String username) {

**return** Jwts.*builder*()

.setSubject(username)

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

.setExpiration(**new** Date(System.*currentTimeMillis*() + 1000 \* 60 \* 60)) // 1 hour

.signWith(SignatureAlgorithm.***HS256***, SECRET\_KEY)

.compact();

}

}

**AuthController.java**

**package** com.cognizant.spring\_learn.controller;

**import** com.cognizant.spring\_learn.util.JwtUtil;

**import** jakarta.servlet.http.HttpServletRequest;

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

**import** org.springframework.security.authentication.AuthenticationManager;

**import** org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

**import** org.springframework.security.core.Authentication;

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

**import** java.util.Base64;

**import** java.util.HashMap;

**import** java.util.Map;

@RestController

**public** **class** AuthController {

@Autowired

**private** JwtUtil jwtUtil;

@Autowired

**private** AuthenticationManager authenticationManager;

@GetMapping("/authenticate")

**public** Map<String, String> authenticate(HttpServletRequest request) {

String authHeader = request.getHeader("Authorization");

**if** (authHeader == **null** || !authHeader.startsWith("Basic ")) {

**throw** **new** RuntimeException("Missing or invalid Authorization header");

}

// Decode Base64 credentials

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

**byte**[] credDecoded = Base64.*getDecoder*().decode(base64Credentials);

String credentials = **new** String(credDecoded);

String[] values = credentials.split(":", 2);

String username = values[0];

String password = values[1];

Authentication authentication = authenticationManager.authenticate(

**new** UsernamePasswordAuthenticationToken(username, password)

);

String token = jwtUtil.generateToken(authentication.getName());

Map<String, String> response = **new** HashMap<>();

response.put("token", token);

**return** response;

}

}

**EmployeeNotFoundException.java**

**package** com.cognizant.spring\_learn;

**import** org.springframework.http.HttpStatus;

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

@ResponseStatus(HttpStatus.***NOT\_FOUND***)

**public** **class** EmployeeNotFoundException **extends** Exception {

**public** EmployeeNotFoundException(String message) {

**super**(message);

}

}

**GlobalExceptionHandler.java**

**package** com.cognizant.spring\_learn;

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

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**import** org.springframework.http.HttpHeaders;

**import** org.springframework.http.HttpStatusCode;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.http.converter.HttpMessageNotReadableException;

**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.\*;

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

@ControllerAdvice

**public** **class** GlobalExceptionHandler **extends** ResponseEntityExceptionHandler {

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

@Override

**protected** ResponseEntity<Object> handleMethodArgumentNotValid(MethodArgumentNotValidException ex,

HttpHeaders headers,

HttpStatusCode status,

WebRequest request) {

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

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

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

List<String> errors = ex.getBindingResult()

.getFieldErrors()

.stream()

.map(err -> err.getDefaultMessage())

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

body.put("errors", errors);

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

}

@Override

**protected** ResponseEntity<Object> handleHttpMessageNotReadable(HttpMessageNotReadableException ex,

HttpHeaders headers,

HttpStatusCode 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) {

**for** (InvalidFormatException.Reference ref : ((InvalidFormatException) ex.getCause()).getPath()) {

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

}

}

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

}

}

**SpringLearnApplication.java**

**package** com.cognizant.spring\_learn;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

**public** **class** SpringLearnApplication {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(SpringLearnApplication.**class**, args);

}

}

**EmployeeController.java**

package com.cognizant.spring\_learn.controller;

import com.cognizant.spring\_learn.EmployeeNotFoundException;

import com.cognizant.spring\_learn.model.Employee;

import com.cognizant.spring\_learn.service.EmployeeService;

import jakarta.validation.Valid;

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

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

import java.util.List;

@RestController

public class EmployeeController {

@Autowired

private EmployeeService employeeService;

@PutMapping("/employees")

public void updateEmployee(@RequestBody @Valid Employee employee) throws EmployeeNotFoundException {

employeeService.updateEmployee(employee);

}

@GetMapping("/employees")

public List<Employee> getAllEmployees() {

return employeeService.getAllEmployees();

}

@DeleteMapping("/employees/{id}")

public void deleteEmployee(@PathVariable int id) throws EmployeeNotFoundException {

employeeService.deleteEmployee(id);

}

}

**EmployeeDao.java**

**package** com.cognizant.spring\_learn.dao;

**import** com.cognizant.spring\_learn.EmployeeNotFoundException;

**import** com.cognizant.spring\_learn.model.Employee;

**import** com.cognizant.spring\_learn.model.Department;

**import** com.cognizant.spring\_learn.model.Skill;

**import** jakarta.annotation.PostConstruct;

**import** org.springframework.stereotype.Repository;

**import** java.util.\*;

@Repository

**public** **class** EmployeeDao {

**private** **static** List<Employee> *employeeList* = **new** ArrayList<>();

@PostConstruct

**public** **void** init() {

// Create sample employee

Employee emp = **new** Employee();

emp.setId(1);

emp.setName("Default Emp");

emp.setSalary((**float**)(3000.0f));

emp.setPermanent(**true**);

emp.setDateOfBirth(**null**);

Department dept = **new** Department();

dept.setId(101);

dept.setName("HR");

emp.setDepartment(dept);

Skill skill1 = **new** Skill();

skill1.setId(1);

skill1.setName("Communication");

Skill skill2 = **new** Skill();

skill2.setId(2);

skill2.setName("Teamwork");

emp.setSkillList(Arrays.*asList*(skill1, skill2));

*employeeList*.add(emp);

}

**public** **void** deleteEmployee(**int** id) **throws** EmployeeNotFoundException {

Iterator<Employee> iterator = *employeeList*.iterator();

**while** (iterator.hasNext()) {

Employee emp = iterator.next();

**if** (emp.getId() == id) {

iterator.remove();

**return**;

}

}

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

}

**public** **void** updateEmployee(Employee updatedEmployee) **throws** EmployeeNotFoundException {

**for** (**int** i = 0; i < *employeeList*.size(); i++) {

**if** (*employeeList*.get(i).getId().equals(updatedEmployee.getId())) {

*employeeList*.set(i, updatedEmployee);

**return**;

}

}

**throw** **new** EmployeeNotFoundException("Employee not found with ID: " + updatedEmployee.getId());

}

**public** List<Employee> getAllEmployees() {

**return** *employeeList*;

}

}

**Department.java**

**package** com.cognizant.spring\_learn.model;

**import** jakarta.validation.constraints.\*;

**public** **class** Department {

@NotNull

**private** Integer id;

@NotBlank

@Size(min = 1, max = 30)

**private** String name;

**public** Integer getId() {

**return** id;

}

**public** **void** setId(Integer id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

// Getters and Setters

}

**Employee.java**

**package** com.cognizant.spring\_learn.model;

**import** com.fasterxml.jackson.annotation.JsonFormat;

**import** jakarta.validation.constraints.\*;

**import** java.util.Date;

**import** java.util.List;

**public** **class** Employee {

@NotNull

**private** Integer id;

@NotBlank

@Size(min = 1, max = 30)

**private** String name;

@NotNull

@PositiveOrZero

**private** Float salary;

@NotNull

**private** Boolean permanent;

@NotNull

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

**private** Date dateOfBirth;

@NotNull

**private** Department department;

@NotNull

**private** List<Skill> skillList;

**public** Employee()

{

}

**public** Integer getId() {

**return** id;

}

**public** **void** setId(Integer id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** Float getSalary() {

**return** salary;

}

**public** **void** setSalary(Float 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> getSkillList() {

**return** skillList;

}

**public** **void** setSkillList(List<Skill> skillList) {

**this**.skillList = skillList;

}

**public** Employee(@NotNull Integer id, @NotBlank @Size(min = 1, max = 30) String name,

@NotNull @PositiveOrZero Float salary, @NotNull Boolean permanent, @NotNull Date dateOfBirth,

@NotNull Department department, @NotNull List<Skill> skillList) {

**this**.id = id;

**this**.name = name;

**this**.salary = salary;

**this**.permanent = permanent;

**this**.dateOfBirth = dateOfBirth;

**this**.department = department;

**this**.skillList = skillList;

}

// Getters and Setters

}

**Skill.java**

**package** com.cognizant.spring\_learn.model;

**import** jakarta.validation.constraints.\*;

**public** **class** Skill {

@NotNull

**private** Integer id;

@NotBlank

@Size(min = 1, max = 30)

**private** String name;

**public** Integer getId() {

**return** id;

}

**public** **void** setId(Integer id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

// Getters and Setters

}

**EmployeeService.java**

**package** com.cognizant.spring\_learn.service;

**import** com.cognizant.spring\_learn.EmployeeNotFoundException;

**import** com.cognizant.spring\_learn.dao.EmployeeDao;

**import** com.cognizant.spring\_learn.model.Employee;

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

**import** org.springframework.stereotype.Service;

**import** java.util.List;

@Service

**public** **class** EmployeeService {

@Autowired

**private** EmployeeDao employeeDao;

**public** **void** updateEmployee(Employee employee) **throws** EmployeeNotFoundException {

employeeDao.updateEmployee(employee);

}

**public** List<Employee> getAllEmployees() {

**return** employeeDao.getAllEmployees();

}

**public** **void** deleteEmployee(**int** id) **throws** EmployeeNotFoundException {

employeeDao.deleteEmployee(id);

}

}

**SecurityConfig.java**

**package** security;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.Configuration;

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

**import** org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

**import** org.springframework.security.web.SecurityFilterChain;

@Configuration

@EnableWebSecurity

**public** **class** SecurityConfig {

@Bean

**public** SecurityFilterChain filterChain(HttpSecurity http) **throws** Exception {

http

.authorizeHttpRequests((authz) -> authz

.anyRequest().authenticated()

)

.~~httpBasic~~() // Enables basic authentication

.~~and~~()

.~~csrf~~().disable(); // Disable CSRF for APIs

**return** http.build();

}

}

**DepartmentService.java**

**package** com.cognizant.spring\_learn.service;

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

**import** org.springframework.stereotype.Service;

**import** com.cognizant.spring\_learn.dao.DepartmentDao;

**import** com.cognizant.spring\_learn.model.Department;

**import** java.util.List;

@Service

**public** **class** DepartmentService {

@Autowired

**private** DepartmentDao departmentDao;

**public** List<Department> getAllDepartments() {

**return** departmentDao.getAllDepartments();

}

}

**CountryController.java**

**package** com.cognizant.spring\_learn.controller;

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.Map;

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

@RestController

@RequestMapping

**public** **class** CountryController {

@GetMapping("/countries")

**public** List<Map<String, String>> getCountries() {

List<Map<String, String>> countries = **new** ArrayList<>();

countries.add(Map.*of*("code", "US", "name", "United States"));

countries.add(Map.*of*("code", "DE", "name", "Germany"));

countries.add(Map.*of*("code", "IN", "name", "India"));

countries.add(Map.*of*("code", "JP", "name", "Japan"));

**return** countries;

}

}

**c**

**package** com.cognizant.spring\_learn.dao;

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

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

**import** org.springframework.beans.factory.annotation.Qualifier;

**import** org.springframework.stereotype.Repository;

**import** com.cognizant.spring\_learn.model.Department;

**import** java.util.List;

@Repository

**public** **class** DepartmentDao {

**public** **static** List<Department> *DEPARTMENT\_LIST*;

@Autowired

**public** DepartmentDao(@Qualifier("departmentList") List<Department> departmentList) {

*DEPARTMENT\_LIST* = departmentList;

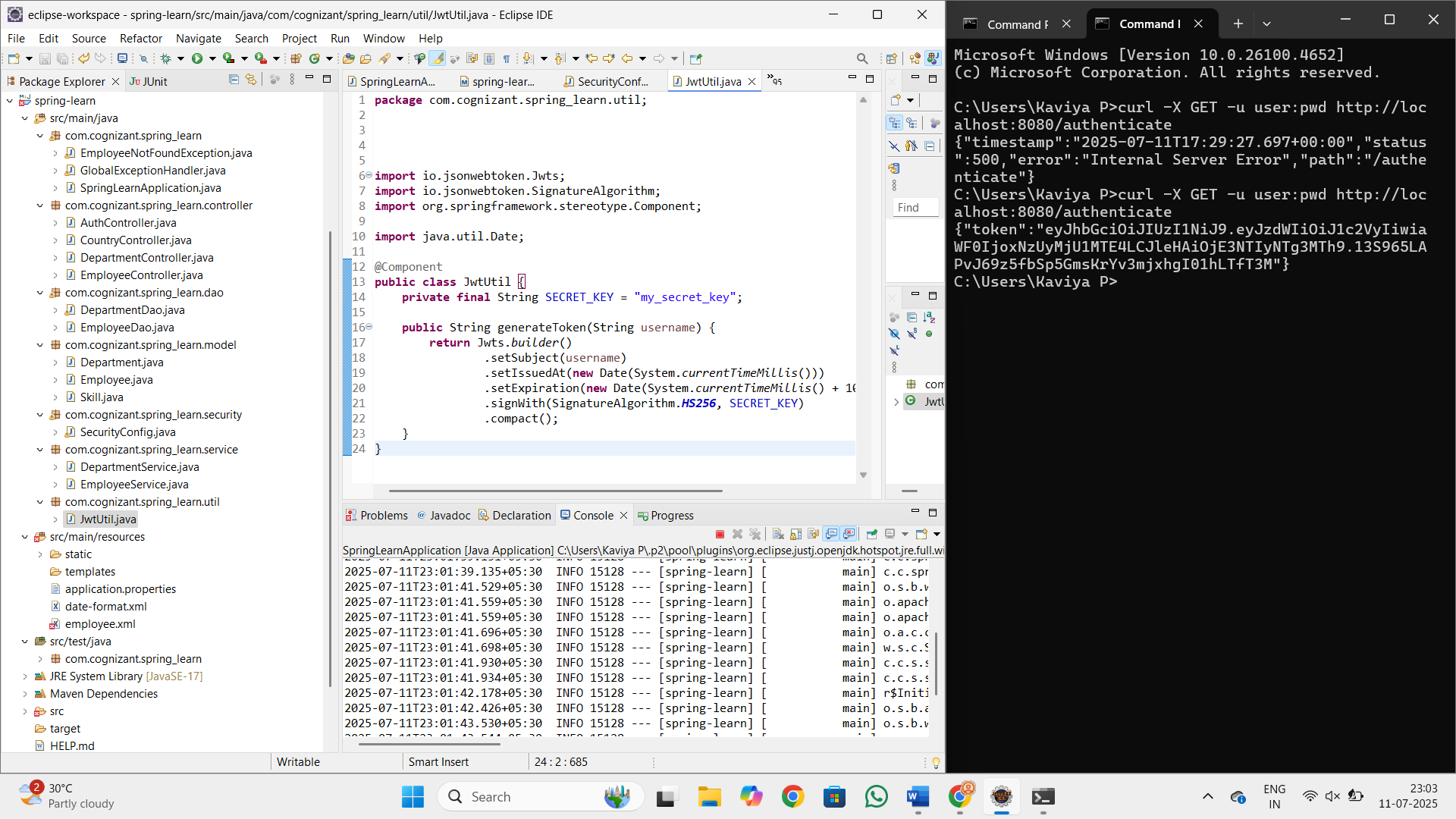
}

**public** List<Department> getAllDepartments() {

**return** *DEPARTMENT\_LIST*;

}

}

****

**Create authentication controller and configure it in SecurityConfig**

**SpringLearnApplication.java**

**package** com.cognizant.spring\_learn;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** org.springframework.context.annotation.ImportResource;

@SpringBootApplication

@ImportResource("classpath:employee.xml")

**public** **class** SpringLearnApplication {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(SpringLearnApplication.**class**, args);

}

}

**AuthController.java**

**package** com.cognizant.spring\_learn.controller;

**import** com.cognizant.spring\_learn.util.JwtUtil;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

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

**import** org.springframework.security.authentication.AuthenticationManager;

**import** org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

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

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

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

**import** java.nio.charset.StandardCharsets;

**import** java.util.Base64;

**import** java.util.HashMap;

**import** java.util.Map;

@RestController

**public** **class** AuthController {

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

@Autowired

**private** AuthenticationManager authenticationManager;

@Autowired

**private** JwtUtil jwtUtil;

@GetMapping("/authenticate")

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

***LOGGER***.info("START - /authenticate");

***LOGGER***.debug("Authorization Header: {}", authHeader);

// Decode Base64 credentials

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

String credentials = **new** String(Base64.*getDecoder*().decode(base64Credentials), StandardCharsets.***UTF\_8***);

String[] values = credentials.split(":", 2);

String username = values[0];

String password = values[1];

// Authenticate user

authenticationManager.authenticate(**new** UsernamePasswordAuthenticationToken(username, password));

// Generate JWT

String token = jwtUtil.generateToken(username);

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

map.put("token", token);

***LOGGER***.info("END - /authenticate");

**return** map;

}

}

**SecurityConfig.java**

**package** com.cognizant.spring\_learn.security;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.Configuration;

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

**import** org.springframework.security.web.SecurityFilterChain;

**import** org.springframework.security.authentication.AuthenticationManager;

**import** org.springframework.security.config.annotation.authentication.configuration.AuthenticationConfiguration;

**import** org.springframework.security.config.annotation.method.configuration.EnableMethodSecurity;

**import** org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

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

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

**import** org.springframework.security.provisioning.InMemoryUserDetailsManager;

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

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

@Configuration

@EnableWebSecurity

@EnableMethodSecurity

**public** **class** SecurityConfig {

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

@Bean

**public** AuthenticationManager authenticationManager(AuthenticationConfiguration authenticationConfiguration) **throws** Exception {

**return** authenticationConfiguration.getAuthenticationManager();

}

@Bean

**public** PasswordEncoder passwordEncoder() {

***LOGGER***.info("Initializing password encoder");

**return** **new** BCryptPasswordEncoder();

}

@Bean

**public** InMemoryUserDetailsManager userDetailsService(PasswordEncoder encoder) {

***LOGGER***.info("Creating in-memory users");

UserDetails admin = User.*withUsername*("admin")

.password(encoder.encode("pwd"))

.roles("ADMIN")

.build();

UserDetails user = User.*withUsername*("user")

.password(encoder.encode("pwd"))

.roles("USER")

.build();

**return** **new** InMemoryUserDetailsManager(admin, user);

}

@Bean

**public** SecurityFilterChain filterChain(HttpSecurity http) **throws** Exception {

http.~~csrf~~().disable()

.~~authorizeHttpRequests~~()

.requestMatchers("/authenticate").permitAll()

.requestMatchers("/countries").hasRole("USER")

.anyRequest().authenticated()

.~~and~~()

.~~httpBasic~~(); // Still needed for initial Basic Auth

**return** http.build();

}

}

**JwtUtil.java**

**package** com.cognizant.spring\_learn.util;

**import** io.jsonwebtoken.Jwts;

**import** io.jsonwebtoken.SignatureAlgorithm;

**import** jakarta.annotation.PostConstruct;

**import** org.springframework.stereotype.Component;

**import** javax.crypto.spec.SecretKeySpec;

**import** java.security.Key;

**import** java.util.Base64;

**import** java.util.Date;

@Component

**public** **class** JwtUtil {

**private** **final** String SECRET\_KEY = "my\_secret\_key";

**private** Key hmacKey;

@PostConstruct

**public** **void** init() {

**byte**[] keyBytes = Base64.*getEncoder*().encode(SECRET\_KEY.getBytes());

hmacKey = **new** SecretKeySpec(keyBytes, SignatureAlgorithm.***HS256***.getJcaName());

}

**public** String generateToken(String username) {

**return** Jwts.*builder*()

.setSubject(username)

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

.setExpiration(**new** Date(System.*currentTimeMillis*() + 1000 \* 60 \* 60)) // 1 hour

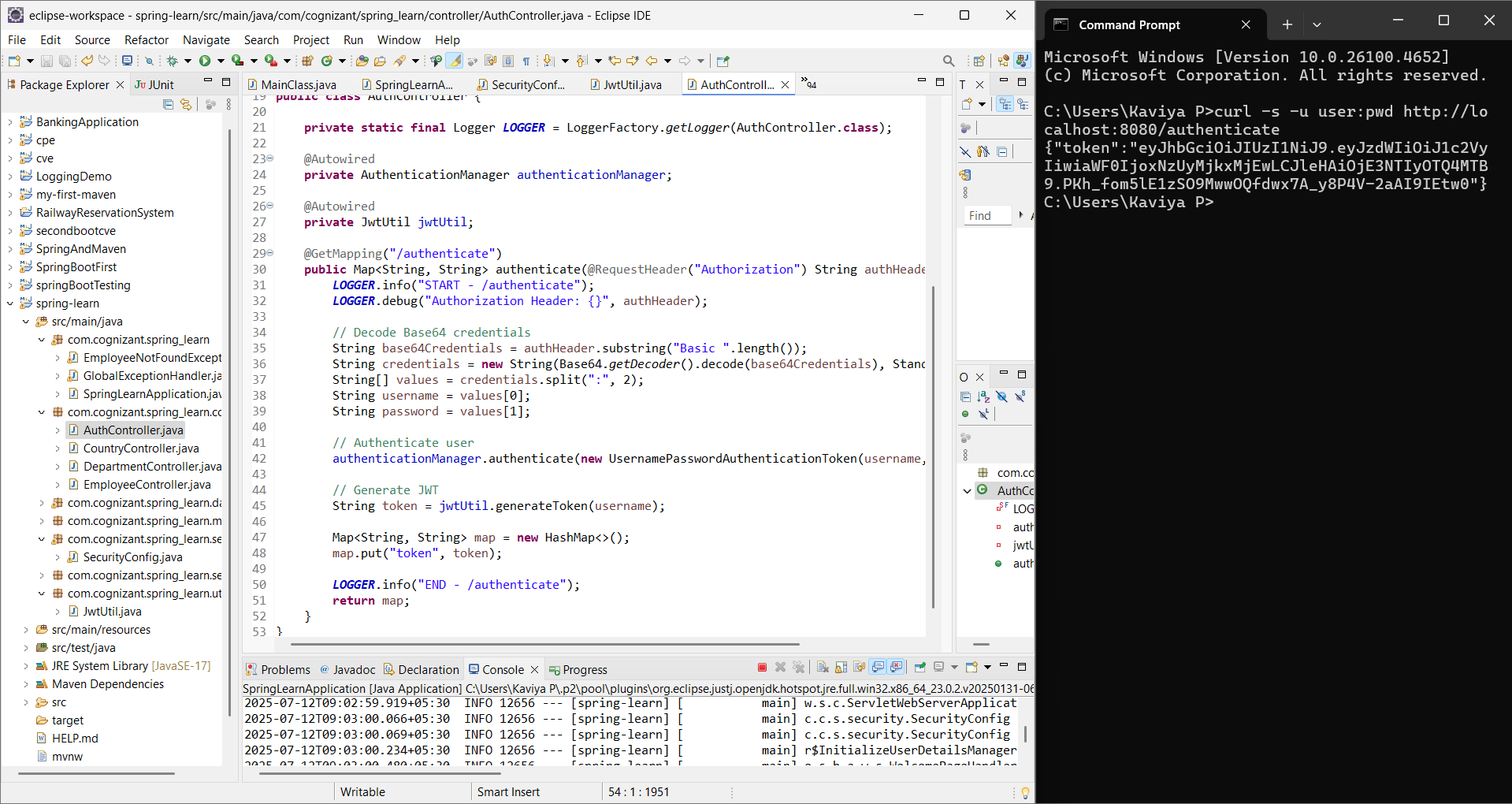
.signWith(SignatureAlgorithm.***HS256***, hmacKey)

.compact();

}

}

**OUTPUT:**

****

**Read Authorization header and decode the username and password**

**SpringLearnApplication.java**

**package** com.cognizant.spring\_learn;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** org.springframework.context.annotation.ImportResource;

@SpringBootApplication

@ImportResource("classpath:employee.xml")

**public** **class** SpringLearnApplication {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(SpringLearnApplication.**class**, args);

}

}

**AuthController.java**

**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.RequestHeader;

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

**import** java.util.Base64;

**import** java.util.HashMap;

**import** java.util.Map;

@RestController

**public** **class** AuthController {

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

@GetMapping("/authenticate")

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

***LOGGER***.info("START - /authenticate");

***LOGGER***.debug("Authorization Header: {}", authHeader);

String username = getUser(authHeader);

***LOGGER***.debug("Decoded username: {}", username);

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

map.put("token", ""); // JWT token will be added later

***LOGGER***.info("END - /authenticate");

**return** map;

}

**private** String getUser(String authHeader) {

***LOGGER***.debug("Decoding Authorization header");

**if** (authHeader != **null** && authHeader.startsWith("Basic ")) {

**try** {

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

**byte**[] decodedBytes = Base64.*getDecoder*().decode(base64Credentials);

String decoded = **new** String(decodedBytes);

***LOGGER***.debug("Decoded credentials: {}", decoded);

// credentials = "username:password"

String[] values = decoded.split(":", 2);

**return** values[0]; // Return username

} **catch** (Exception e) {

***LOGGER***.error("Error decoding Authorization header", e);

}

}

**return** **null**;

}

}

**SecurityConfig.java**

**package** com.cognizant.spring\_learn.security;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.Configuration;

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

**import** org.springframework.security.web.SecurityFilterChain;

**import** org.springframework.security.authentication.AuthenticationManager;

**import** org.springframework.security.config.annotation.authentication.configuration.AuthenticationConfiguration;

**import** org.springframework.security.config.annotation.method.configuration.EnableMethodSecurity;

**import** org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

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

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

**import** org.springframework.security.provisioning.InMemoryUserDetailsManager;

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

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

@Configuration

@EnableWebSecurity

@EnableMethodSecurity

**public** **class** SecurityConfig {

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

@Bean

**public** AuthenticationManager authenticationManager(AuthenticationConfiguration authenticationConfiguration) **throws** Exception {

**return** authenticationConfiguration.getAuthenticationManager();

}

@Bean

**public** PasswordEncoder passwordEncoder() {

***LOGGER***.info("Initializing password encoder");

**return** **new** BCryptPasswordEncoder();

}

@Bean

**public** InMemoryUserDetailsManager userDetailsService(PasswordEncoder encoder) {

***LOGGER***.info("Creating in-memory users");

UserDetails admin = User.*withUsername*("admin")

.password(encoder.encode("pwd"))

.roles("ADMIN")

.build();

UserDetails user = User.*withUsername*("user")

.password(encoder.encode("pwd"))

.roles("USER")

.build();

**return** **new** InMemoryUserDetailsManager(admin, user);

}

@Bean

**public** SecurityFilterChain filterChain(HttpSecurity http) **throws** Exception {

http.~~csrf~~().disable()

.~~authorizeHttpRequests~~()

.requestMatchers("/authenticate").permitAll()

.requestMatchers("/countries").hasRole("USER")

.anyRequest().authenticated()

.~~and~~()

.~~httpBasic~~(); // Still needed for initial Basic Auth

**return** http.build();

}

}

**JwtUtil.java**

**package** com.cognizant.spring\_learn.util;

**import** io.jsonwebtoken.Jwts;

**import** io.jsonwebtoken.SignatureAlgorithm;

**import** jakarta.annotation.PostConstruct;

**import** org.springframework.stereotype.Component;

**import** javax.crypto.spec.SecretKeySpec;

**import** java.security.Key;

**import** java.util.Base64;

**import** java.util.Date;

@Component

**public** **class** JwtUtil {

**private** **final** String SECRET\_KEY = "my\_secret\_key";

**private** Key hmacKey;

@PostConstruct

**public** **void** init() {

**byte**[] keyBytes = Base64.*getEncoder*().encode(SECRET\_KEY.getBytes());

hmacKey = **new** SecretKeySpec(keyBytes, SignatureAlgorithm.***HS256***.getJcaName());

}

**public** String generateToken(String username) {

**return** Jwts.*builder*()

.setSubject(username)

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

.setExpiration(**new** Date(System.*currentTimeMillis*() + 1000 \* 60 \* 60)) // 1 hour

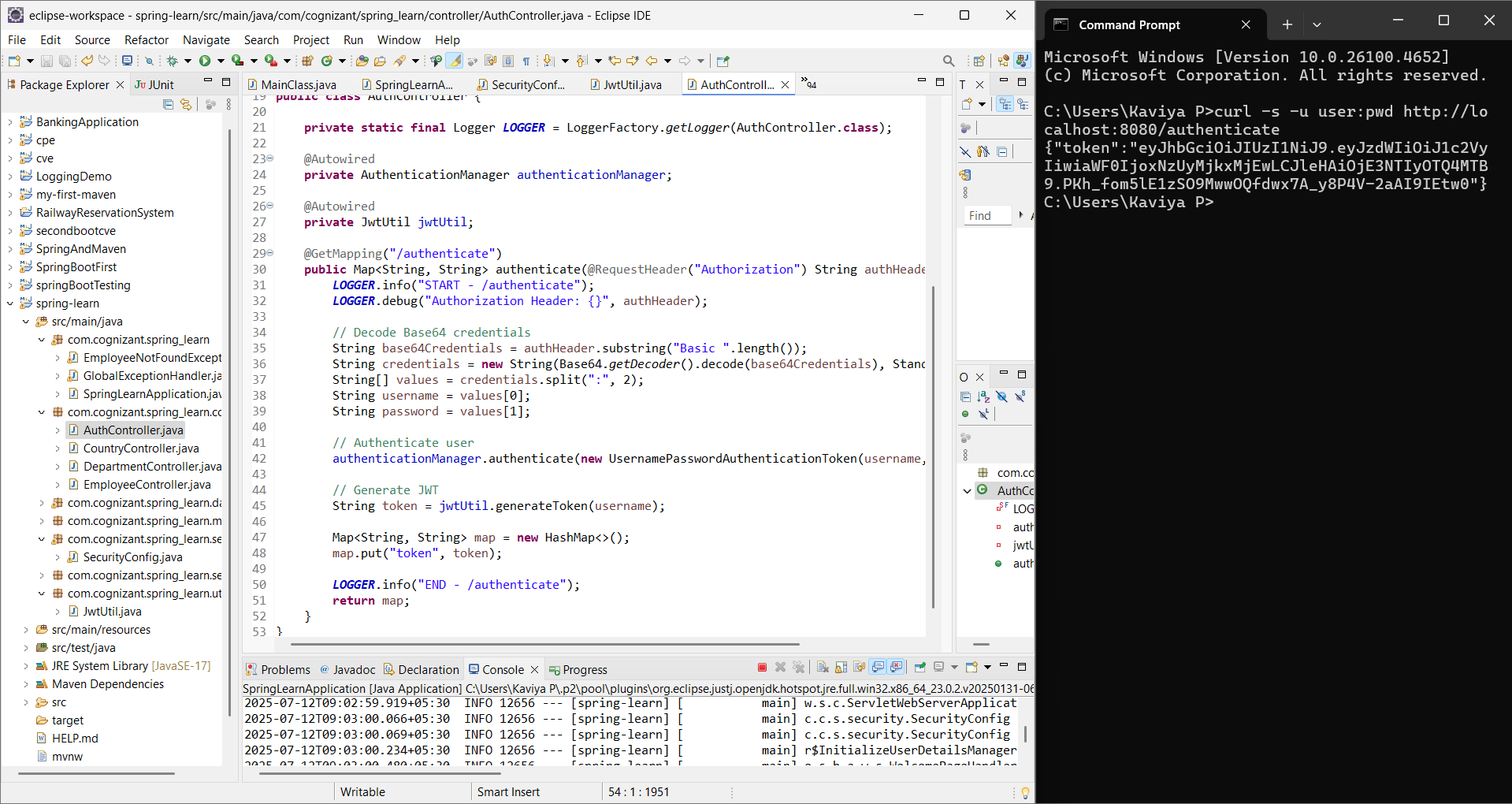
.signWith(SignatureAlgorithm.***HS256***, hmacKey)

.compact();

}

}

**OUTPUT**

****

**Generate token based on the user**

**SpringLearnApplication.java**

**package** com.cognizant.spring\_learn;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** org.springframework.context.annotation.ImportResource;

@SpringBootApplication

@ImportResource("classpath:employee.xml")

**public** **class** SpringLearnApplication {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(SpringLearnApplication.**class**, args);

}

}

**AuthController.java**

**package** com.cognizant.spring\_learn.controller;

**import** io.jsonwebtoken.JwtBuilder;

**import** io.jsonwebtoken.Jwts;

**import** io.jsonwebtoken.SignatureAlgorithm;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

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

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

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

**import** java.util.Base64;

**import** java.util.Date;

**import** java.util.HashMap;

**import** java.util.Map;

@RestController

**public** **class** AuthController {

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

@GetMapping("/authenticate")

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

***LOGGER***.info("START - /authenticate");

***LOGGER***.debug("Authorization Header: {}", authHeader);

String username = getUser(authHeader);

***LOGGER***.debug("Decoded username: {}", username);

String token = generateJwt(username);

***LOGGER***.debug("Generated JWT: {}", token);

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

map.put("token", token);

***LOGGER***.info("END - /authenticate");

**return** map;

}

**private** String getUser(String authHeader) {

***LOGGER***.debug("Decoding Authorization header");

**if** (authHeader != **null** && authHeader.startsWith("Basic ")) {

**try** {

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

**byte**[] decodedBytes = Base64.*getDecoder*().decode(base64Credentials);

String decoded = **new** String(decodedBytes);

***LOGGER***.debug("Decoded credentials: {}", decoded);

String[] values = decoded.split(":", 2);

**return** values[0];

} **catch** (Exception e) {

***LOGGER***.error("Error decoding Authorization header", e);

}

}

**return** **null**;

}

**private** String generateJwt(String user) {

***LOGGER***.debug("Generating JWT for user: {}", user);

JwtBuilder builder = Jwts.*builder*();

builder.setSubject(user);

builder.setIssuedAt(**new** Date());

builder.setExpiration(**new** Date((**new** Date()).getTime() + 1200000)); // 20 min

builder.signWith(SignatureAlgorithm.***HS256***, "secretkey");

String token = builder.compact();

**return** token;

}

}

**SecurityConfig.java**

**package** com.cognizant.spring\_learn.security;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.Configuration;

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

**import** org.springframework.security.web.SecurityFilterChain;

**import** org.springframework.security.authentication.AuthenticationManager;

**import** org.springframework.security.config.annotation.authentication.configuration.AuthenticationConfiguration;

**import** org.springframework.security.config.annotation.method.configuration.EnableMethodSecurity;

**import** org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

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

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

**import** org.springframework.security.provisioning.InMemoryUserDetailsManager;

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

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

@Configuration

@EnableWebSecurity

@EnableMethodSecurity

**public** **class** SecurityConfig {

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

@Bean

**public** AuthenticationManager authenticationManager(AuthenticationConfiguration authenticationConfiguration) **throws** Exception {

**return** authenticationConfiguration.getAuthenticationManager();

}

@Bean

**public** PasswordEncoder passwordEncoder() {

***LOGGER***.info("Initializing password encoder");

**return** **new** BCryptPasswordEncoder();

}

@Bean

**public** InMemoryUserDetailsManager userDetailsService(PasswordEncoder encoder) {

***LOGGER***.info("Creating in-memory users");

UserDetails admin = User.*withUsername*("admin")

.password(encoder.encode("pwd"))

.roles("ADMIN")

.build();

UserDetails user = User.*withUsername*("user")

.password(encoder.encode("pwd"))

.roles("USER")

.build();

**return** **new** InMemoryUserDetailsManager(admin, user);

}

@Bean

**public** SecurityFilterChain filterChain(HttpSecurity http) **throws** Exception {

http.~~csrf~~().disable()

.~~authorizeHttpRequests~~()

.requestMatchers("/authenticate").permitAll()

.requestMatchers("/countries").hasRole("USER")

.anyRequest().authenticated()

.~~and~~()

.~~httpBasic~~(); // Still needed for initial Basic Auth

**return** http.build();

}

}

**JwtUtil.java**

**package** com.cognizant.spring\_learn.util;

**import** io.jsonwebtoken.Jwts;

**import** io.jsonwebtoken.SignatureAlgorithm;

**import** jakarta.annotation.PostConstruct;

**import** org.springframework.stereotype.Component;

**import** javax.crypto.spec.SecretKeySpec;

**import** java.security.Key;

**import** java.util.Base64;

**import** java.util.Date;

@Component

**public** **class** JwtUtil {

**private** **final** String SECRET\_KEY = "my\_secret\_key";

**private** Key hmacKey;

@PostConstruct

**public** **void** init() {

**byte**[] keyBytes = Base64.*getEncoder*().encode(SECRET\_KEY.getBytes());

hmacKey = **new** SecretKeySpec(keyBytes, SignatureAlgorithm.***HS256***.getJcaName());

}

**public** String generateToken(String username) {

**return** Jwts.*builder*()

.setSubject(username)

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

.setExpiration(**new** Date(System.*currentTimeMillis*() + 1000 \* 60 \* 60)) // 1 hour

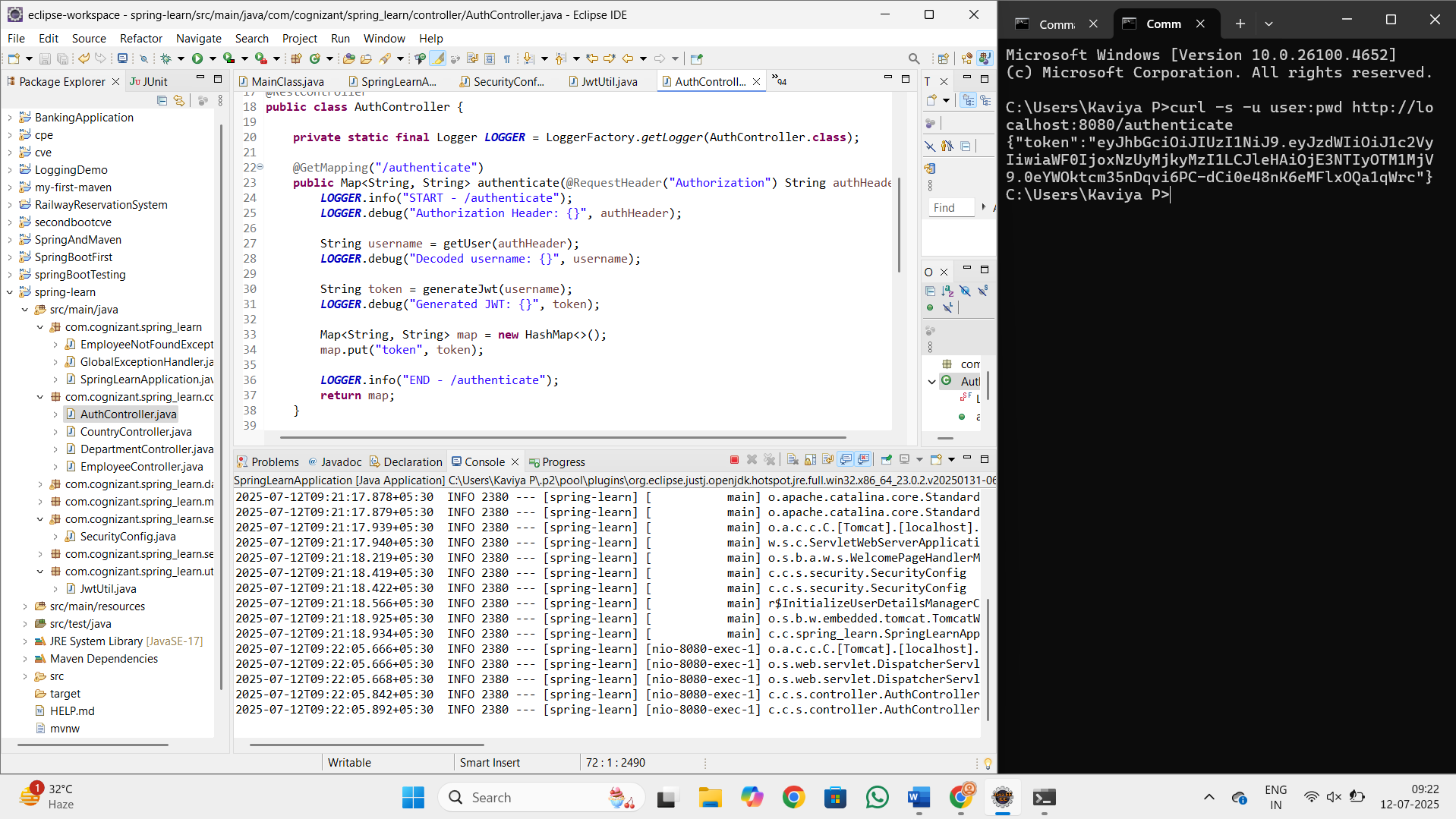
.signWith(SignatureAlgorithm.***HS256***, hmacKey)

.compact();

}

}

**OUTPUT**

****

**Authorize based on JWT**

**SpringLearnApplication.java**

**package** com.cognizant.spring\_learn;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** org.springframework.context.annotation.ImportResource;

@SpringBootApplication

@ImportResource("classpath:employee.xml")

**public** **class** SpringLearnApplication {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(SpringLearnApplication.**class**, args);

}

}

**AuthController.java**

**package** com.cognizant.spring\_learn.controller;

**import** io.jsonwebtoken.JwtBuilder;

**import** io.jsonwebtoken.Jwts;

**import** io.jsonwebtoken.SignatureAlgorithm;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

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

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

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

**import** java.util.Base64;

**import** java.util.Date;

**import** java.util.HashMap;

**import** java.util.Map;

@RestController

**public** **class** AuthController {

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

@GetMapping("/authenticate")

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

***LOGGER***.info("START - /authenticate");

***LOGGER***.debug("Authorization Header: {}", authHeader);

String username = getUser(authHeader);

***LOGGER***.debug("Decoded username: {}", username);

String token = generateJwt(username);

***LOGGER***.debug("Generated JWT: {}", token);

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

map.put("token", token);

***LOGGER***.info("END - /authenticate");

**return** map;

}

**private** String getUser(String authHeader) {

***LOGGER***.debug("Decoding Authorization header");

**if** (authHeader != **null** && authHeader.startsWith("Basic ")) {

**try** {

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

**byte**[] decodedBytes = Base64.*getDecoder*().decode(base64Credentials);

String decoded = **new** String(decodedBytes);

***LOGGER***.debug("Decoded credentials: {}", decoded);

String[] values = decoded.split(":", 2);

**return** values[0];

} **catch** (Exception e) {

***LOGGER***.error("Error decoding Authorization header", e);

}

}

**return** **null**;

}

**private** String generateJwt(String user) {

***LOGGER***.debug("Generating JWT for user: {}", user);

JwtBuilder builder = Jwts.*builder*();

builder.setSubject(user);

builder.setIssuedAt(**new** Date());

builder.setExpiration(**new** Date((**new** Date()).getTime() + 1200000)); // 20 min

builder.signWith(SignatureAlgorithm.***HS256***, "secretkey");

String token = builder.compact();

**return** token;

}

}

**JwtAuthorizationFilter.java**

**package** com.cognizant.spring\_learn.security;

**import** java.io.IOException;

**import** java.util.ArrayList;

**import** jakarta.servlet.FilterChain;

**import** jakarta.servlet.ServletException;

**import** jakarta.servlet.http.HttpServletRequest;

**import** jakarta.servlet.http.HttpServletResponse;

**import** io.jsonwebtoken.Claims;

**import** io.jsonwebtoken.Jws;

**import** io.jsonwebtoken.JwtException;

**import** io.jsonwebtoken.Jwts;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**import** org.springframework.security.authentication.AuthenticationManager;

**import** org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

**import** org.springframework.security.core.context.SecurityContextHolder;

**import** org.springframework.security.web.authentication.www.BasicAuthenticationFilter;

**public** **class** JwtAuthorizationFilter **extends** BasicAuthenticationFilter {

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

**public** JwtAuthorizationFilter(AuthenticationManager authenticationManager) {

**super**(authenticationManager);

***LOGGER***.info("JwtAuthorizationFilter Constructor");

}

@Override

**protected** **void** doFilterInternal(HttpServletRequest req, HttpServletResponse res, FilterChain chain)

**throws** IOException, ServletException {

***LOGGER***.info("JWT Filter - Start");

String header = req.getHeader("Authorization");

***LOGGER***.debug("Authorization Header: {}", header);

**if** (header == **null** || !header.startsWith("Bearer ")) {

chain.doFilter(req, res);

**return**;

}

UsernamePasswordAuthenticationToken authentication = getAuthentication(req);

**if** (authentication != **null**) {

SecurityContextHolder.*getContext*().setAuthentication(authentication);

}

chain.doFilter(req, res);

***LOGGER***.info("JWT Filter - End");

}

**private** UsernamePasswordAuthenticationToken getAuthentication(HttpServletRequest request) {

String token = request.getHeader("Authorization");

**if** (token != **null**) {

**try** {

Jws<Claims> jws = Jwts.*parser*()

.setSigningKey("secretkey") // should match what's used in generateJwt()

.parseClaimsJws(token.replace("Bearer ", ""));

String user = jws.getBody().getSubject();

***LOGGER***.debug("JWT subject (user): {}", user);

**if** (user != **null**) {

**return** **new** UsernamePasswordAuthenticationToken(user, **null**, **new** ArrayList<>());

}

} **catch** (JwtException e) {

***LOGGER***.error("JWT parsing failed", e);

**return** **null**;

}

}

**return** **null**;

}

}

**SecurityConfig.java**

**package** com.cognizant.spring\_learn.security;

**import** org.slf4j.Logger;

**import** org.slf4j.LoggerFactory;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.Configuration;

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

**import** org.springframework.security.web.SecurityFilterChain;

**import** org.springframework.security.authentication.AuthenticationManager;

**import** org.springframework.security.config.annotation.authentication.configuration.AuthenticationConfiguration;

**import** org.springframework.security.config.annotation.method.configuration.EnableMethodSecurity;

**import** org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

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

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

**import** org.springframework.security.provisioning.InMemoryUserDetailsManager;

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

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

@Configuration

@EnableWebSecurity

@EnableMethodSecurity

**public** **class** SecurityConfig {

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

@Bean

**public** AuthenticationManager authenticationManager(AuthenticationConfiguration authenticationConfiguration) **throws** Exception {

**return** authenticationConfiguration.getAuthenticationManager();

}

@Bean

**public** PasswordEncoder passwordEncoder() {

***LOGGER***.info("Initializing password encoder");

**return** **new** BCryptPasswordEncoder();

}

@Bean

**public** InMemoryUserDetailsManager userDetailsService(PasswordEncoder encoder) {

***LOGGER***.info("Creating in-memory users");

UserDetails admin = User.*withUsername*("admin")

.password(encoder.encode("pwd"))

.roles("ADMIN")

.build();

UserDetails user = User.*withUsername*("user")

.password(encoder.encode("pwd"))

.roles("USER")

.build();

**return** **new** InMemoryUserDetailsManager(admin, user);

}

@Bean

**public** SecurityFilterChain filterChain(HttpSecurity http) **throws** Exception {

http.~~csrf~~().disable()

.~~authorizeHttpRequests~~()

.requestMatchers("/authenticate").permitAll()

.requestMatchers("/countries").hasRole("USER")

.anyRequest().authenticated()

.~~and~~()

.addFilter(**new** JwtAuthorizationFilter(authenticationManager(http.getSharedObject(AuthenticationConfiguration.**class**))))

.~~httpBasic~~(); // Optional: keep for basic auth testing

**return** http.build();

}

}

**JwtUtil.java**

**package** com.cognizant.spring\_learn.util;

**import** io.jsonwebtoken.Jwts;

**import** io.jsonwebtoken.SignatureAlgorithm;

**import** jakarta.annotation.PostConstruct;

**import** org.springframework.stereotype.Component;

**import** javax.crypto.spec.SecretKeySpec;

**import** java.security.Key;

**import** java.util.Base64;

**import** java.util.Date;

@Component

**public** **class** JwtUtil {

**private** **final** String SECRET\_KEY = "my\_secret\_key";

**private** Key hmacKey;

@PostConstruct

**public** **void** init() {

**byte**[] keyBytes = Base64.*getEncoder*().encode(SECRET\_KEY.getBytes());

hmacKey = **new** SecretKeySpec(keyBytes, SignatureAlgorithm.***HS256***.getJcaName());

}

**public** String generateToken(String username) {

**return** Jwts.*builder*()

.setSubject(username)

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

.setExpiration(**new** Date(System.*currentTimeMillis*() + 1000 \* 60 \* 60)) // 1 hour

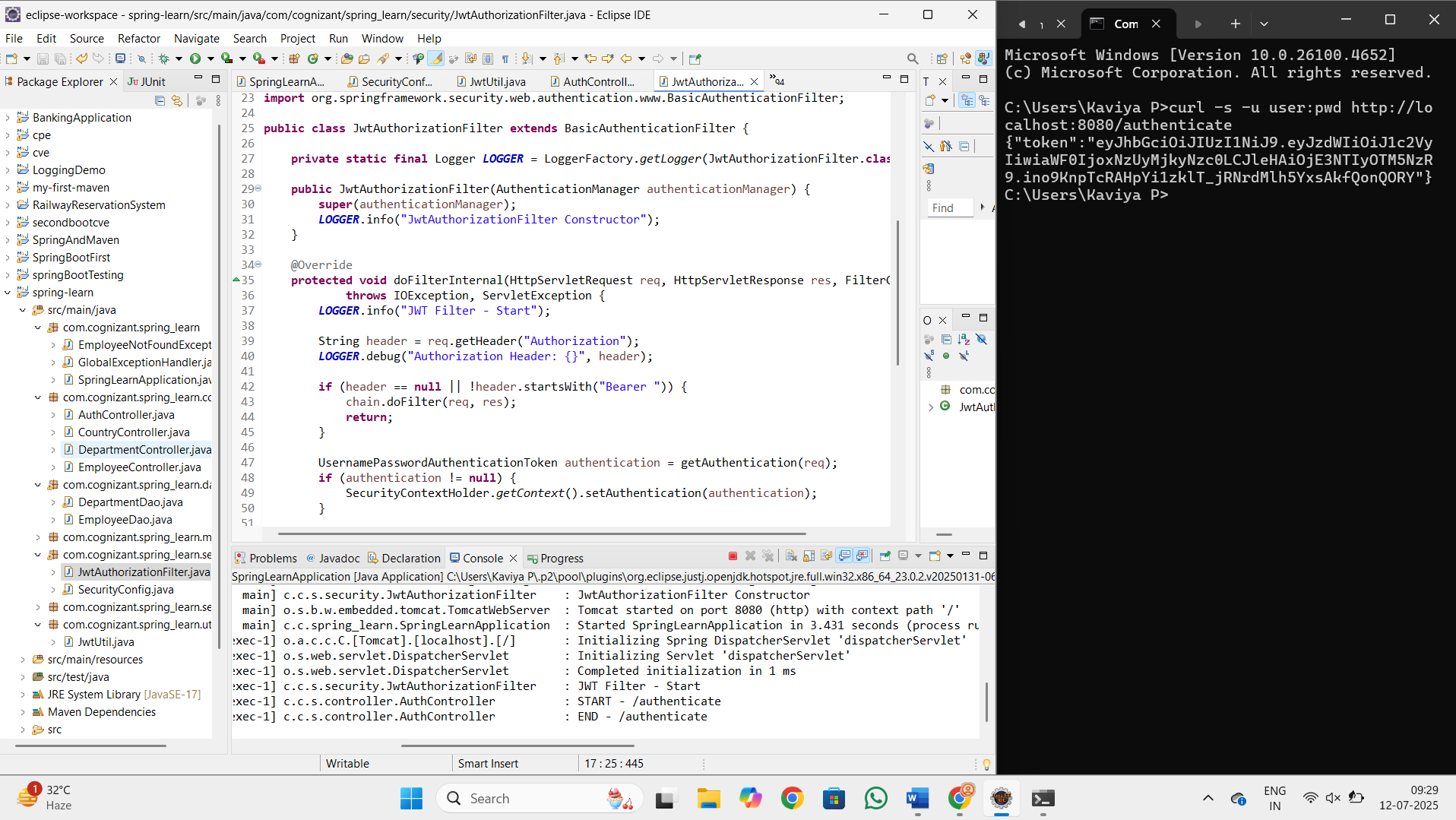
.signWith(SignatureAlgorithm.***HS256***, hmacKey)

.compact();

}

}

**OUTPUT**

****