## URL

<http://localhost:8080/StudentManagement/>

# 

# Adding Security

## Students-lister.jsp

* Put this code after the add button (which is actually <a>)

<c:url var="logoutUrl" value = "/logout" />

<a href="${logoutUrl}"

class="btn btn-primary btn-sm mb-3 mx-auto"> Logout </a>

## StudentsController

@RequestMapping(value = "/403")

public ModelAndView accesssDenied(Principal user) {

ModelAndView model = new ModelAndView();

if (user != null) {

model.addObject("msg", "Hi " + user.getName() + ", you do not have permission to access this page!");

} else {

model.addObject("msg", "You do not have permission to access this page!");

}

model.setViewName("403");

return model;

}

## 403.jsp

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<!DOCTYPE html>

<html>

<body>

<h1>HTTP Status 403 - Access is denied</h1>

<h2>${msg}</h2>

<br>

<c:url var="listUrl" value= "/students/list"/>

<hr>

<a href="${listUrl}">Back to Students List</a>

</body>

</html>

## 

## StudentWebSecurityConfigurerAdapter

@Override

protected void configure(HttpSecurity http) throws Exception {

http.authorizeRequests()

.antMatchers("/", "/student/save", "/student/add-begin", "/student/403")

.hasAnyAuthority("NORMAL\_USER", "ADMIN\_USER")

.antMatchers("/student/update-begin", "/student/delete")

.hasAuthority("ADMIN\_USER")

.anyRequest().authenticated().and()

.formLogin()

.loginProcessingUrl("/login").successForwardUrl("/students/list")

.permitAll().and().

logout()

.logoutSuccessUrl("/login").permitAll().and().

exceptionHandling()

.accessDeniedPage("/students/403").and()

.cors().and().csrf().disable();

}

## db-insert-scripts.txt (resources folder)

Cleanup Code

delete from roles;

delete from users\_roles;

delete from users;

# ROLE TABLE

insert into roles (name) values ('NORMAL\_USER');

insert into roles (name) values ('ADMIN\_USER');

# USER TABLE

Password Mapping

"mypassword" ->

$2a$12$ncVKTp7LOrPWiHqyEDginOO/4o4abR9rvbYbv5nX1wejkjMWHdAfm

"javafsd" ->

$2a$12$vRl2/tMcwdmSvN5whfiWdelqFgwxBCMh5LB6zJ9jW/WDHuqbbXxt.

insert into users (username, password) values (

'varun', '$2a$12$vRl2/tMcwdmSvN5whfiWdelqFgwxBCMh5LB6zJ9jW/WDHuqbbXxt.');

insert into users (user\_id, username, password) values (

'sanjay', '$2a$12$vRl2/tMcwdmSvN5whfiWdelqFgwxBCMh5LB6zJ9jW/WDHuqbbXxt.');

# USERS\_ROLES TABLE

insert into users\_roles (user\_id, role\_id) values (

(select user\_id from users where username = 'varun'),

(select role\_id from roles where name = 'ADMIN\_USER')

);

insert into users\_roles (user\_id, role\_id) values (

(select user\_id from users where username = 'sanjay'),

(select role\_id from roles where name = 'NORMAL\_USER')

);

## MyCustomUserDetails (part of security sub-package)

import java.util.ArrayList;

import java.util.Collection;

import java.util.List;

import org.springframework.security.core.GrantedAuthority;

import org.springframework.security.core.authority.SimpleGrantedAuthority;

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

import com.greatlearning.studentmanagement.entity.Role;

import com.greatlearning.studentmanagement.entity.User;

public class MyCustomUserDetails

implements UserDetails{

private User user;

public MyCustomUserDetails(User user) {

this.user = user;

}

@Override

public Collection<? extends GrantedAuthority> getAuthorities() {

List<Role> roles = user.getRoles();

List<SimpleGrantedAuthority> authorities = new ArrayList<>();

for (Role role : roles) {

authorities.add(new SimpleGrantedAuthority(role.getName()));

}

return authorities;

}

@Override

public String getPassword() {

return user.getPassword();

}

@Override

public String getUsername() {

return user.getUsername();

}

@Override

public boolean isAccountNonExpired() {

return true;

}

@Override

public boolean isAccountNonLocked() {

return true;

}

@Override

public boolean isCredentialsNonExpired() {

return true;

}

@Override

public boolean isEnabled() {

return true;

}

}

## Role (to be part of entity package)

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Table;

@Entity

@Table(name = "roles")

public class Role {

@Id

@Column(name = "role\_id")

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Integer id;

@Column(name = "name")

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;

}

}

## User

import java.util.ArrayList;

import java.util.List;

import javax.persistence.CascadeType;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.JoinTable;

import javax.persistence.ManyToMany;

import javax.persistence.Table;

@Entity

@Table(name = "users")

public class User {

@Id

@Column(name = "user\_id")

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(name = "username")

private String username;

@Column(name = "password")

private String password;

@ManyToMany(cascade = CascadeType.ALL, fetch = FetchType.EAGER)

@JoinTable(

name = "users\_roles", joinColumns = @JoinColumn(name = "user\_id"),

inverseJoinColumns = @JoinColumn(name = "role\_id")

)

private List<Role> roles = new ArrayList<>();

public List<Role> getRoles() {

return roles;

}

public void setRoles(List<Role> roles) {

this.roles = roles;

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

}

## UserRepository

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.jpa.repository.Query;

import com.greatlearning.studentmanagement.entity.User;

public interface UserRepository

extends JpaRepository<User, Long>{

@Query("SELECT u FROM User u WHERE u.username = ?1")

User getUserByUsername(String username);

}

## StudentUserDetailsService

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

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

import org.springframework.security.core.userdetails.UserDetailsService;

import org.springframework.security.core.userdetails.UsernameNotFoundException;

import org.springframework.stereotype.Repository;

import com.greatlearning.studentmanagement.entity.User;

import com.greatlearning.studentmanagement.repository.UserRepository;

import com.greatlearning.studentmanagement.security.MyCustomUserDetails;

@Repository

public class StudentUserDetailsService

implements UserDetailsService{

@Autowired

private UserRepository userRepository;

@Override

public UserDetails loadUserByUsername(String username)

throws UsernameNotFoundException {

User userEntity = userRepository.getUserByUsername(username);

if (userEntity == null) {

// Throw exception

throw new UsernameNotFoundException("Could not find user");

}

return new MyCustomUserDetails(userEntity);

}

## StudentWebSecurityConfigurerAdapter

import org.springframework.context.annotation.Bean;

import org.springframework.security.authentication.AuthenticationProvider;

import org.springframework.security.authentication.dao.DaoAuthenticationProvider;

import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;

import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

import org.springframework.security.core.userdetails.UserDetailsService;

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

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

import com.greatlearning.studentmanagement.service.StudentUserDetailsService;

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

**@Configuration**

public class StudentWebSecurityConfigurerAdapter

extends WebSecurityConfigurerAdapter{

// Customize Authentication

@Override

protected void configure(AuthenticationManagerBuilder auth) throws Exception {

AuthenticationProvider authenticationProvider = customAuthenticationProvider();

auth.authenticationProvider(authenticationProvider);

}

@Bean

public UserDetailsService customUserDetailsService() {

return new StudentUserDetailsService();

}

@Bean

public PasswordEncoder bcryptPasswordEncoder() {

return new BCryptPasswordEncoder();

}

@Bean

public DaoAuthenticationProvider customAuthenticationProvider() {

DaoAuthenticationProvider daoProvider = new DaoAuthenticationProvider();

daoProvider.setUserDetailsService(customUserDetailsService());

daoProvider.setPasswordEncoder(bcryptPasswordEncoder());

return daoProvider;

}

}

## POM.xml

* + - <dependency>
      * <groupId>org.springframework.boot</groupId>
    - <artifactId>spring-boot-starter-security</artifactId>
    - </dependency>

# Delete Student

### Students-lister.jsp

* Add the below code fragment after the 5th TD and after the <a> tag

<c:url var="deleteUrl" value="/student/delete?studentId=${tempStudent.id}" />

<a href="${deleteUrl}"

class="btn btn-danger btn-sm"

onclick="if (!(confirm('Are you sure you want to delete this Student?'))) return false">

Delete

</a>

### StudentController

@RequestMapping("/delete")

public String delete(

@RequestParam("studentId") int studentId) {

studentService.deleteById(studentId);

return "redirect:/students/list";

}

### StudentServiceImpl

@Override

public void deleteById(int theId) {

studentRepository.deleteById(theId);

}

### StudentService

void deleteById(int theId);

# Update Student

## Students-lister.jsp

### Section-1

<thead class="thead-dark">

<tr>

<th>FirstName</th>

<th>LastName</th>

<th>Course</th>

<th>Country</th>

<th>Action</th>

</tr>

</thead>

### Section-2

After the end of the 4th TD

<td>

<!-- Add "update" button/link -->

<c:url var="updateUrl" value="/student/update-begin?studentId=${tempStudent.id}" />

<a

href="${updateUrl}"

class="btn btn-info btn-sm"> Update

</a>

</td>

## StudentController

### Method 1

@RequestMapping("/update-begin")

public String handleUpdateStudentBegin(

@RequestParam("studentId") int studentId,

Model model) {

Student student = studentService.findById(studentId);

model.addAttribute("student", student);

return "student-details";

}

### Method 2

@PostMapping("/save")

public String saveStudent(

@RequestParam("studentId") int studentId,

@RequestParam("firstName") String firstName,

@RequestParam("lastName") String lastName,

@RequestParam("course") String course,

@RequestParam("country") String country) {

System.out.println(studentId);

Student student = null;

if (studentId != 0) {

student = studentService.findById(studentId);

student.setFirstName(firstName);

student.setLastName(lastName);

student.setCourse(course);

student.setCountry(country);

} else{

// Add flow

student = new Student(firstName, lastName, course, country);

}

studentService.save(student);

// use a redirect to prevent duplicate submissions

return "redirect:/students/list";

}

## StudentServiceImpl

public Student findById(int id) {

return studentRepository.findById(id).get();

}

## StudentService

public interface StudentService {

List<Student> findAll();

void save(Student thestudent);

**Student findById(int theId);**

}

# Add Student

## students-details.jsp

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<!DOCTYPE html>

<html>

<head>

<!-- Required meta tags -->

<meta charset="utf-8">

<meta name="viewport"

content="width=device-width, initial-scale=1, shrink-to-fit=no">

<!-- Bootstrap CSS -->

<link rel="stylesheet"

href="https://stackpath.bootstrapcdn.com/bootstrap/4.2.1/css/bootstrap.min.css"

integrity="sha384-GJzZqFGwb1QTTN6wy59ffF1BuGJpLSa9DkKMp0DgiMDm4iYMj70gZWKYbI706tWS"

crossorigin="anonymous">

<title>Save Student</title>

</head>

<body>

<div class="container">

<h3>Student Directory</h3>

<hr>

<p class="h4 mb-4">Student</p>

<c:url var = "saveUrl" value = "/student/save" />

<form action="${saveUrl}" method="POST">

<!-- Add hidden form field to handle update -->

<input type="hidden" name="studentId" value="${student.id}" />

<div class="form-inline">

<input type="text" name="firstName" value="${student.firstName}"

class="form-control mb-4 col-4" placeholder="First Name">

</div>

<div class="form-inline">

<input type="text" name="lastName" value="${student.lastName}"

class="form-control mb-4 col-4" placeholder="Last Name">

</div>

<div class="form-inline">

<input type="text" name="course" value="${student.course}"

class="form-control mb-4 col-4" placeholder="Course">

</div>

<div class="form-inline">

<input type="text" name="country" value="${student.country}"

class="form-control mb-4 col-4" placeholder="Country">

</div>

<button type="submit" class="btn btn-info col-2">Save</button>

</form>

<c:url var="listUrl" value= "/students/list"/>

<hr>

<a href="${listUrl}">Back to Students List</a>

</div>

</body>

</html>

## students-lister.jsp

* Add the following code just before the <table> tag

<c:url var="addUrl" value = "/student/add-begin" />

<!-- Add a button -->

<a href="${addUrl}"

class="btn btn-primary btn-sm mb-3"> Add Student </a>

## StudentController

package com.greatlearning.studentmanagement.controller;

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

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

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

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

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

import com.greatlearning.studentmanagement.entity.Student;

import com.greatlearning.studentmanagement.service.StudentService;

@Controller

@RequestMapping("/student")

public class StudentController {

@Autowired

private StudentService studentService;

@RequestMapping("/add-begin")

public String handleAddStudentBegin(Model model) {

// create model attribute to bind form data

Student student = new Student();

model.addAttribute("student", student);

return "student-details";

}

@PostMapping("/save")

public String saveStudent(

@RequestParam("studentId") int studentId,

@RequestParam("firstName") String firstName,

@RequestParam("lastName") String lastName,

@RequestParam("course") String course,

@RequestParam("country") String country) {

System.out.println(studentId);

Student student = null;

if (studentId != 0) {

// TODO - Update flow

} else{

// Add flow

student = new Student(firstName, lastName, course, country);

}

studentService.save(student);

// use a redirect to prevent duplicate submissions

return "redirect:/students/list";

}

}

## Student Service Impl

public void save(Student theStudent) {

studentRepository.save(theStudent);

}

## Student Service

public interface StudentService {

List<Student> findAll();

void save(Student thestudent);

}

# List Students

## Welcome.jsp

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<!DOCTYPE html>

<html>

<head>

<title>Spring Boot - Student Management</title>

</head>

<body>

Spring Boot - Welcome to Student Management

</body>

<c:url var="listUrl" value= "/students/list"/>

<hr>

<a href="${listUrl}">Back to Students List</a>

</html>

## Students-lister.jsp

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<!-- Required meta tags -->

<meta charset="utf-8">

<meta name="viewport"

content="width=device-width, initial-scale=1, shrink-to-fit=no">

<!-- Bootstrap CSS -->

<link rel="stylesheet"

href="https://stackpath.bootstrapcdn.com/bootstrap/4.2.1/css/bootstrap.min.css"

integrity="sha384-GJzZqFGwb1QTTN6wy59ffF1BuGJpLSa9DkKMp0DgiMDm4iYMj70gZWKYbI706tWS"

crossorigin="anonymous">

<title>Students Directory</title>

</head>

<body>

<div class="container">

<h3>Students Directory</h3>

<hr>

<table class="table table-bordered table-striped">

<thead class="thead-dark">

<tr>

<th>FirstName</th>

<th>LastName</th>

<th>Course</th>

<th>Country</th>

</tr>

</thead>

<tbody>

<c:forEach items="${students}" var="tempStudent">

<tr>

<td><c:out value="${tempStudent.firstName}" /></td>

<td><c:out value="${tempStudent.lastName}" /></td>

<td><c:out value="${tempStudent.course}" /></td>

<td><c:out value="${tempStudent.country}" /></td>

</tr>

</c:forEach>

</tbody>

</table>

</div>

</body>

</html>

## StudentsController

import java.util.List;

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

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

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

import com.greatlearning.studentmanagement.entity.Student;

import com.greatlearning.studentmanagement.service.StudentService;

@Controller

@RequestMapping("/students")

public class StudentsController {

@Autowired

private StudentService studentService;

@RequestMapping("/list")

public String handleListAll(Model model) {

List<Student> theStudents = studentService.findAll();

// add to the spring model

model.addAttribute("students", theStudents);

return "students-lister";

}

}

## Student Service Impl

import java.util.List;

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

import org.springframework.stereotype.Repository;

import com.greatlearning.studentmanagement.entity.Student;

import com.greatlearning.studentmanagement.repository.StudentRepository;

@Repository

public class StudentServiceImpl

implements StudentService{

@Autowired

private StudentRepository studentRepository;

@Override

public List<Student> findAll() {

return studentRepository.findAll();

}

}

## Student Service Interface

import java.util.List;

import com.greatlearning.studentmanagement.entity.Student;

public interface StudentService {

List<Student> findAll();

}

## Student Repository

public interface StudentRepository

extends JpaRepository<Student, Integer>{

}

## Student Entity

@Entity

@Table(name = "student")

public class Student {

// define fields

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "id")

private int id;

@Column(name = "firstName")

private String firstName;

@Column(name = "lastName")

private String lastName;

@Column(name = "course")

private String course;

@Column(name = "country")

private String country;

public Student() {

}

public Student(String firstName, String lastName, String course, String country) {

this.firstName = firstName;

this.lastName = lastName;

this.course = course;

this.country = country;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getFirstName() {

return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

public String getLastName() {

return lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

public String getCourse() {

return course;

}

public void setCourse(String course) {

this.course = course;

}

public String getCountry() {

return country;

}

public void setCountry(String country) {

this.country = country;

}

@Override

public String toString() {

return "Student [id=" + id + ", firstName=" + firstName + ", lastName=" + lastName + ", course=" + course

+ ", country=" + country + "]";

}

}

## Application.properties

* + spring.jpa.hibernate.ddl-auto=update
  + spring.datasource.url=jdbc:mysql://localhost:3306/student-management
  + spring.datasource.username=test\_user
  + spring.datasource.password=test@12345
  + spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect

## Pom.xml

* + <dependency>
  + <groupId>org.springframework.boot</groupId>
  + <artifactId>spring-boot-starter-data-jpa</artifactId>
  + </dependency>
  + <dependency>
  + <groupId>mysql</groupId>
  + <artifactId>mysql-connector-java</artifactId>
  + <scope>runtime</scope>
  + </dependency>
  + <dependency>
  + <groupId>javax.servlet</groupId>
  + <artifactId>jstl</artifactId>
  + </dependency>

<dependency>

* + <groupId>org.springframework.security</groupId>
  + <artifactId>spring-security-taglibs</artifactId>
  + <version>5.2.2.RELEASE</version>

</dependency>