Spring boot, JPA, JSP project – MVC application

Model:

Student.java

package com.soar.spring.jpa.crud.entity;  
  
  
import lombok.AllArgsConstructor;  
import lombok.Builder;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
  
import javax.persistence.\*;  
import java.util.Objects;  
  
@Entity  
@Data  
@AllArgsConstructor  
@NoArgsConstructor  
@Builder  
@Table(name="student",  
 uniqueConstraints = @UniqueConstraint(  
 name = "emailid\_unique",  
 columnNames = "email\_address"  
 )  
)  
public class Student {  
  
 @Id  
 @SequenceGenerator(  
 name="student\_sequence",  
 sequenceName = "student\_sequence",  
 allocationSize=1  
 )  
 @GeneratedValue(  
 strategy = GenerationType.*SEQUENCE*,  
 generator="student\_sequence"  
 )  
 private Long studentId;  
 private String firstName;  
 private String lastName;  
  
 @Column(  
 name="email\_address",  
 nullable = false  
 )  
 private String emailId;  
  
 @Embedded  
 private Guardian guardian;  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (!(o instanceof Student)) return false;  
 Student student = (Student) o;  
 return Objects.*equals*(getStudentId(), student.getStudentId()) &&  
 Objects.*equals*(getFirstName(), student.getFirstName()) &&  
 Objects.*equals*(getLastName(), student.getLastName()) &&  
 Objects.*equals*(getEmailId(), student.getEmailId()) &&  
 Objects.*equals*(getGuardian(), student.getGuardian());  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.*hash*(getStudentId(), getFirstName(), getLastName(), getEmailId(), getGuardian());  
 }  
}

Guardian.java

package com.soar.spring.jpa.crud.entity;  
  
import lombok.AllArgsConstructor;  
import lombok.Builder;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
  
import javax.persistence.AttributeOverride;  
import javax.persistence.AttributeOverrides;  
import javax.persistence.Column;  
import javax.persistence.Embeddable;  
  
@Embeddable  
@Data  
@Builder  
@AllArgsConstructor  
@NoArgsConstructor  
@AttributeOverrides(  
 {  
 @AttributeOverride(  
 name = "name",  
 column = @Column(name = "guardian\_name")  
 ),  
 @AttributeOverride(  
 name = "email",  
 column = @Column(name = "guardian\_email")  
 ),  
 @AttributeOverride(  
 name = "mobile",  
 column = @Column(name = "guardian\_mobile")  
 )  
 }  
)  
public class Guardian {  
  
 private String name;  
 private String email;  
 private String mobile;  
  
}

Repository:

package com.soar.spring.jpa.crud.repository;  
  
import com.soar.spring.jpa.crud.entity.Student;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.data.jpa.repository.Query;  
import org.springframework.stereotype.Repository;  
  
@Repository  
public interface StudentRepository extends JpaRepository<Student, Long> {  
  
 @Query("select s from Student s where s.emailId = ?1")  
 Student findStudentByEmail(String email);  
  
}

Service:

package com.soar.spring.jpa.crud.service;  
  
import com.soar.spring.jpa.crud.entity.Student;  
import org.springframework.data.jpa.repository.Modifying;  
import org.springframework.stereotype.Component;  
  
import java.io.ByteArrayInputStream;  
import java.io.FileInputStream;  
import java.io.IOException;  
import java.util.List;  
  
@Component  
public interface StudentService {  
  
 Student saveStudent(Student student);  
  
 Student deleteStudent(Student student);  
  
 @Modifying  
 Student updateStudent(Long id, Student student);  
  
 Student getStudent(Long id);  
  
 Student deleteStudentById(Long studentId);  
  
 List<Student> getAllStudents();  
  
 ByteArrayInputStream getStudentsToExcel(List<Student> students) throws IOException;  
  
 List<Student> showFiles(FileInputStream f, String name);  
  
 Student findStudentByEmail(String email);  
}

package com.soar.spring.jpa.crud.service;  
  
import com.soar.spring.jpa.crud.entity.Guardian;  
import com.soar.spring.jpa.crud.entity.Student;  
import com.soar.spring.jpa.crud.repository.StudentRepository;  
  
import lombok.Data;  
import lombok.extern.java.Log;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
import org.springframework.transaction.annotation.Transactional;  
  
import javax.persistence.EntityManager;  
import java.io.\*;  
import java.util.\*;  
import java.util.stream.Collectors;  
  
import org.apache.poi.ss.usermodel.\*;  
import org.apache.poi.xssf.usermodel.XSSFWorkbook;  
import org.apache.poi.EncryptedDocumentException;  
import org.apache.poi.hssf.usermodel.HSSFWorkbook;  
  
@Log  
@Data  
@Service("StudentServiceImpl")  
public class StudentServiceImpl implements StudentService {  
  
 private List<Student> studentListIn = new ArrayList<>();  
  
 @Autowired  
 private EntityManager entityManager;  
  
 @Autowired  
 StudentRepository studentRepository;  
  
 @Override  
 @Transactional  
 public Student saveStudent(Student student) {  
 entityManager.persist(student);  
 *log*.info("Student is saved.");  
 return student;  
 }  
  
 @Override  
 @Transactional  
 public Student deleteStudent(Student student) {  
 if (entityManager.contains(student)) {  
 entityManager.remove(student);  
 } else {  
 entityManager.remove(entityManager.merge(student));  
 }  
 return student;  
 }  
  
 @Override  
 @Transactional  
 public Student deleteStudentById(Long studentId) {  
 Student student = entityManager.find(Student.class, studentId);  
 entityManager.remove(student);  
  
 return student;  
 }  
  
 @Override  
 @Transactional  
 public Student updateStudent(Long id, Student student) {  
 Student exist = entityManager.find(Student.class, id);  
  
 exist = this.copyStudents(exist, student);  
 //entityManager.merge(student);  
 entityManager.flush();  
  
 return exist;  
 }  
  
 @Override  
 @Transactional  
 public Student getStudent(Long id) {  
 return entityManager.find(Student.class, id);  
 }  
  
 @Override  
 @Transactional  
 public List<Student> getAllStudents() {  
  
 List<Student> list = studentRepository.findAll();  
 return list;  
 }  
  
 @Override  
 @Transactional  
 public Student findStudentByEmail(String email) {  
 Student student = studentRepository.findStudentByEmail(email);  
 return student;  
 }  
  
 // download excel file  
 @Override  
 @Transactional  
 public ByteArrayInputStream getStudentsToExcel(List<Student> students) throws IOException {  
  
 String[] COLUMNs = {"Student ID:", "First Name:", "Last Name:", "E-mail:", "Guardian Name:", "Guardian E-mail:", "Guardian Mobile:"};  
 try (Workbook workbook = new XSSFWorkbook(); ByteArrayOutputStream out = new ByteArrayOutputStream()) {  
  
 Sheet sheet = workbook.createSheet("Students Report");  
  
 Font headerFont = workbook.createFont();  
 headerFont.setBold(true);  
 headerFont.setColor(IndexedColors.*BLUE*.getIndex());  
  
 CellStyle headerCellStyle = workbook.createCellStyle();  
 headerCellStyle.setFont(headerFont);  
  
 // Row for Header  
 Row headerRow = sheet.createRow(0);  
  
 // Header  
 for (int col = 0; col < COLUMNs.length; col++) {  
 Cell cell = headerRow.createCell(col);  
 cell.setCellValue(COLUMNs[col]);  
 cell.setCellStyle(headerCellStyle);  
 }  
  
 int rowIdx = 1;  
 for (Student student : students) {  
 Row row = sheet.createRow(rowIdx++);  
  
 row.createCell(0).setCellValue(student.getStudentId());  
 row.createCell(1).setCellValue(student.getFirstName());  
 row.createCell(2).setCellValue(student.getLastName());  
 row.createCell(3).setCellValue(student.getEmailId());  
 row.createCell(4).setCellValue(student.getGuardian().getName());  
 row.createCell(5).setCellValue(student.getGuardian().getEmail());  
 row.createCell(6).setCellValue(student.getGuardian().getMobile());  
  
 }  
  
 workbook.write(out);  
 return new ByteArrayInputStream(out.toByteArray());  
 }  
  
 }  
  
 @Override  
 @Transactional  
 public List<Student> showFiles(FileInputStream io, String fName) {  
 //String result = "Reading is done.";  
 if(!studentListIn.isEmpty()){  
 studentListIn.clear();  
 }  
 try {  
 List<String> listEmails = this.readFromExcel(io, fName);  
 //System.out.println("ListEmails=" + listEmails.toString());  
 for (String email : listEmails) {  
 email = email.trim();  
 Student student = studentRepository.findStudentByEmail(email);  
 //log.info("Student="+student.toString());  
 if (student != null) {  
 System.*out*.println("STUDENT:"+student.toString());  
 studentListIn.add(student);  
 }  
 }  
  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 Collections.*sort*(studentListIn, new Comparator<Student>() {  
 @Override  
 public int compare(Student o1, Student o2) {  
 return o1.getStudentId().intValue() - o2.getStudentId().intValue();  
 }  
 });  
 return studentListIn;  
 }  
  
 private Workbook getWorkbook(FileInputStream inputStream, String exelFilePath) {  
 Workbook workbook = null;  
 try {  
 if (exelFilePath.endsWith("xlsx")) {  
 workbook = new XSSFWorkbook(inputStream);  
 } else if (exelFilePath.endsWith("xls")) {  
 workbook = new HSSFWorkbook(inputStream);  
 }  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 return workbook;  
 }  
  
  
 public List<String> readFromExcel(FileInputStream io, String fName) {  
 Set<String> result = new TreeSet<>();  
 List<String> list = new ArrayList<>();  
  
 if (fName != null && !("".equals(fName))) {  
 fName = fName.trim();  
 try {  
 Workbook workbook = getWorkbook(io, fName);  
 Sheet datatypeSheet = workbook.getSheetAt(0);  
 Iterator<Row> iterator = datatypeSheet.iterator();  
  
 while (iterator.hasNext()) {  
  
 Row currentRow = iterator.next();  
 Iterator<Cell> cellIterator = currentRow.iterator();  
  
 while (cellIterator.hasNext()) {  
  
 Cell currentCell = cellIterator.next();  
 String value = currentCell.getStringCellValue();  
 if (value != null && !"".equals(value)) {  
 result.add(value);  
 }  
  
 }  
 }  
 *log*.info("result from excel="+result.size());  
 workbook.close();  
 list = result.parallelStream().collect(Collectors.*toList*());  
 //log.info("RESULT from EXCEL:"+ list);  
 } catch (FileNotFoundException e) {  
 e.printStackTrace();  
 //log.info("File not found....");  
 } catch (IOException e) {  
 e.printStackTrace();  
 //log.info("Problem with IO...");  
 } catch (EncryptedDocumentException e) {  
 e.printStackTrace();  
 //log.info("Problem with encrypting document...");  
 }  
  
 finally {  
 if (io != null) {  
 try {  
 io.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 }  
 return list;  
 }  
  
 private Student copyStudents(Student orig, Student s){  
 orig.setFirstName(s.getFirstName());  
 orig.setLastName(s.getLastName());  
 //orig.setEmailId(s.getEmailId());  
 Guardian g = new Guardian();  
 g.setName(s.getGuardian().getName());  
 g.setMobile(s.getGuardian().getMobile());  
 g.setEmail(s.getGuardian().getEmail());  
 orig.setGuardian(g);  
 return orig;  
 }  
}

Controllers:

StudentJspController.java - @Controller

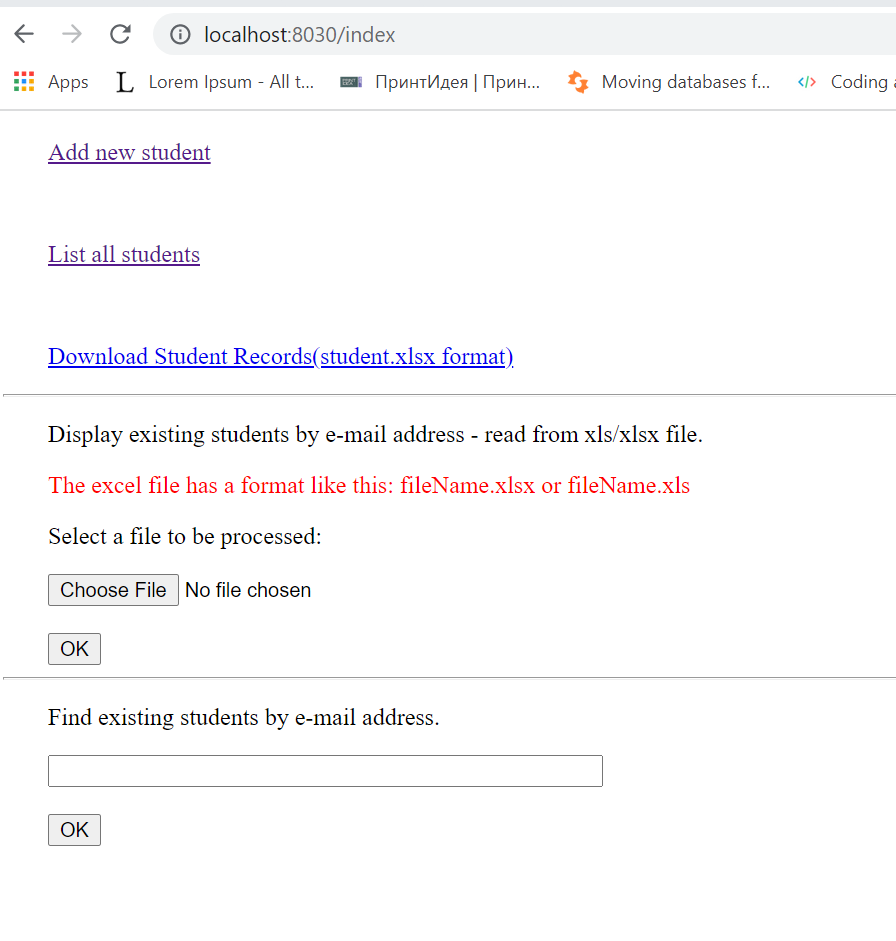
package com.soar.spring.jpa.crud.controller;  
  
import com.soar.spring.jpa.crud.entity.Student;  
import com.soar.spring.jpa.crud.service.StudentService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Controller;  
import org.springframework.web.bind.annotation.\*;  
import org.springframework.web.multipart.MultipartFile;  
import org.springframework.web.servlet.ModelAndView;  
  
import java.io.FileInputStream;  
import java.util.List;  
  
@Controller  
public class StudentJspController {  
  
 @Autowired  
 private StudentService studentService;  
  
 @RequestMapping("/index")  
 public ModelAndView homePage() {  
 ModelAndView modelAndView = new ModelAndView("home");  
 return modelAndView;  
 }  
  
 @RequestMapping(value = "/add", method = RequestMethod.*GET*)  
 public ModelAndView addStudent() {  
 ModelAndView modelAndView = new ModelAndView("addStudent");  
 modelAndView.addObject("student", new Student());  
 return modelAndView;  
 }  
  
 @RequestMapping(value = "/addnew", method = RequestMethod.*POST*)  
 public ModelAndView saveStudent(@ModelAttribute Student student) {  
 ModelAndView modelAndView = new ModelAndView("home");  
 Student studen = studentService.saveStudent(student);  
 String message = "Student was successfully added.";  
 modelAndView.addObject("message", message);  
 return modelAndView;  
 }  
  
 @RequestMapping(value = "/edit/{studentId}", method = RequestMethod.*GET*)  
 public ModelAndView editStudent(@PathVariable Long studentId) {  
 ModelAndView modelAndView = new ModelAndView("editStudent");  
 Student student = studentService.getStudent(studentId);  
 modelAndView.addObject("student", student);  
 return modelAndView;  
 }  
  
 @RequestMapping(value = "/update/{id}")  
 public ModelAndView editingStudent(@ModelAttribute Student student, @PathVariable("id") Long id) {  
 ModelAndView modelAndView = new ModelAndView("editStudent");  
 Student studentEdited = studentService.updateStudent(id, student);  
 String message = "Student was successfully edited.";  
 modelAndView.addObject("message", message);  
 return modelAndView;  
 }  
  
 @RequestMapping(value = "/delete/{studentId}")  
 public ModelAndView deleteStudent(@PathVariable Long studentId) {  
 Student s = studentService.deleteStudentById(studentId);  
 ModelAndView modelAndView = new ModelAndView("home");  
 String message = "Student with id = " + studentId + " was successfully deleted.";  
 modelAndView.addObject("message", message);  
 return modelAndView;  
 }  
  
 @RequestMapping(value = "/list", method = RequestMethod.*GET*)  
 public ModelAndView showListOfStudents(){  
 List<Student> list = studentService.getAllStudents();  
 ModelAndView modelAndView = new ModelAndView("listOfStudents");  
 modelAndView.addObject("list", list);  
 System.*out*.println("List of students:"+list.toString());  
 return modelAndView;  
 }  
  
 // only reads and displays on jsp  
 @RequestMapping("/displayList")  
 public ModelAndView readFromExcel(@RequestParam("file") MultipartFile file) {  
 ModelAndView mav = new ModelAndView();  
 String message = "";  
 if (file.isEmpty()) {  
 mav.setViewName("home");  
 mav.addObject("message", "Please select a file to upload");  
 return mav;  
 }  
  
 String ext = "";  
 String name = file.getOriginalFilename();  
 if (name.lastIndexOf(".") != -1 && name.lastIndexOf(".") != 0) {  
 ext = name.substring(name.lastIndexOf(".") + 1);  
 }  
 // checks for correct excel file  
 if (!(ext.equalsIgnoreCase("xls") || ext.equalsIgnoreCase("xlsx"))) {  
 mav.setViewName("home");  
 mav.addObject("message", "Please enter a correct excel file.");  
 return mav;  
 }  
 try {  
 FileInputStream f = (FileInputStream) file.getInputStream();  
 // does starting job  
 List<Student> studentsIn = studentService.showFiles(f, name);  
 System.*out*.println("STUDENTS IN:"+studentsIn.toString());  
 mav.setViewName("showFromExcel");  
 mav.addObject("message", message);  
 mav.addObject("studentsIn", studentsIn);  
  
  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 // get list of existing students by email search from xls file  
  
  
 return mav;  
 }  
  
}

StudentDownloadController.java - @RestController

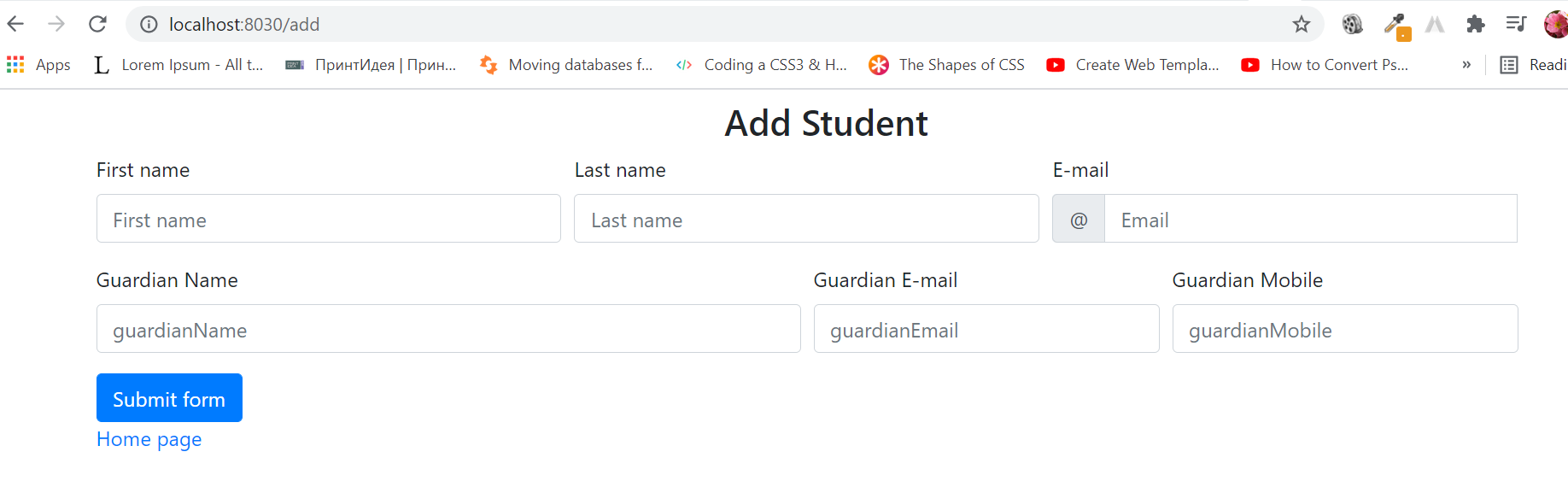
package com.soar.spring.jpa.crud.controller;  
  
import com.soar.spring.jpa.crud.entity.Student;  
import com.soar.spring.jpa.crud.service.StudentService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.core.io.InputStreamResource;  
import org.springframework.http.HttpHeaders;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.RestController;  
  
import java.io.ByteArrayInputStream;  
import java.io.IOException;  
import java.util.List;  
  
@RestController  
public class StudentDownloadController {  
  
 @Autowired  
 StudentService studentService;  
  
 @GetMapping(value = "/download/students")  
 public ResponseEntity<InputStreamResource> excelAllStudentsReport() throws IOException {  
  
 List<Student> students = studentService.getAllStudents();  
  
 ByteArrayInputStream in = studentService.getStudentsToExcel(students);  
 // return IOUtils.toByteArray(in);  
  
  
 HttpHeaders headers = new HttpHeaders();  
 headers.add("Content-Disposition", "attachment; filename=studets.xlsx");  
  
 return ResponseEntity  
 .*ok*()  
 .headers(headers)  
 .body(new InputStreamResource(in));  
 }  
  
}

View:

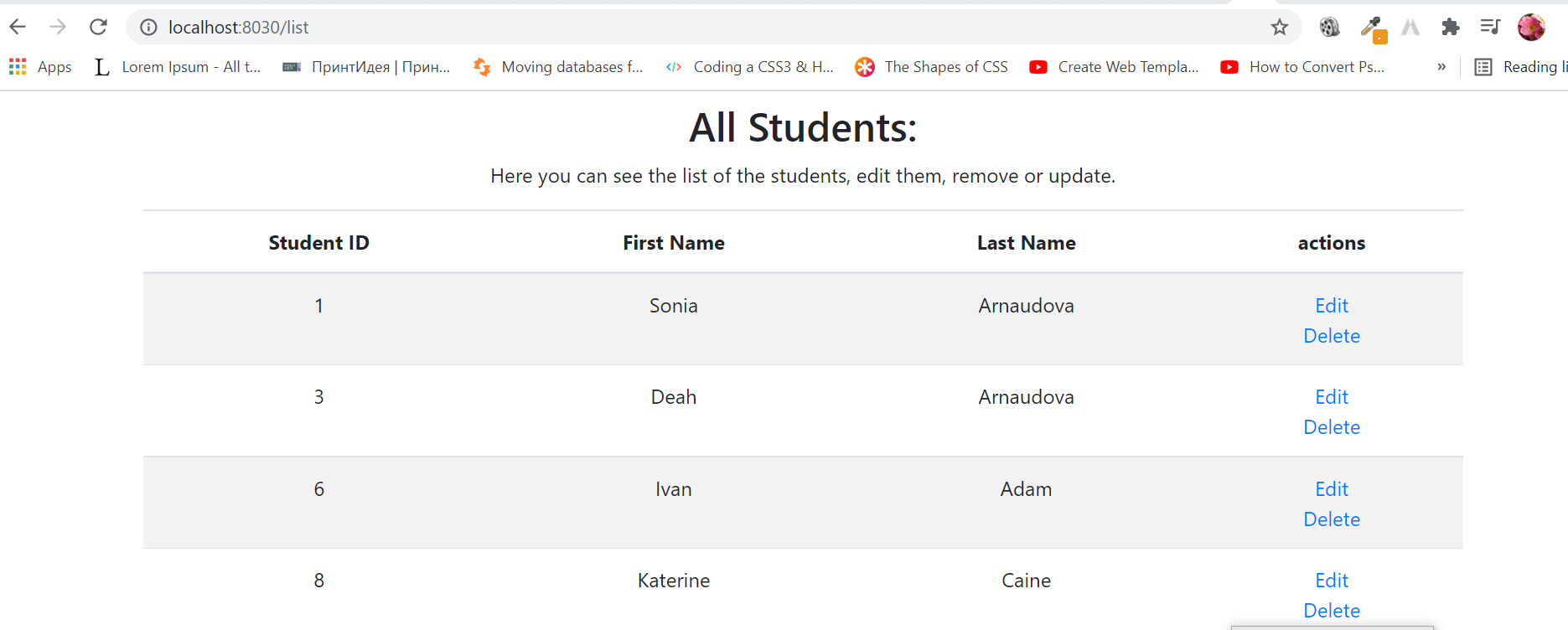
1. home.jsp:



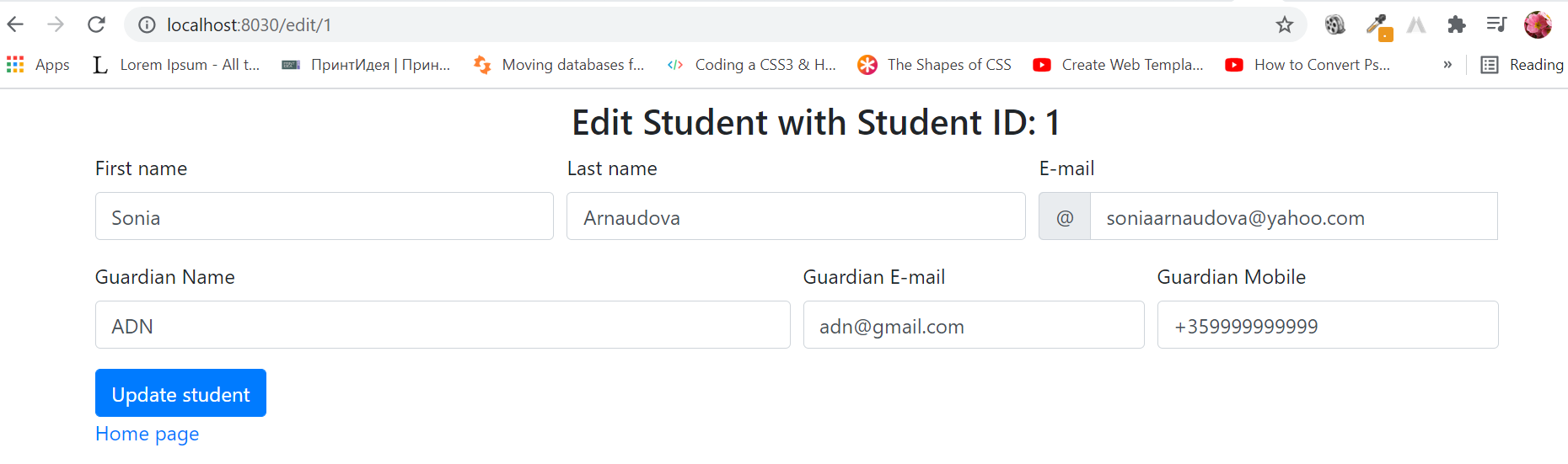
2.addStudent.jsp



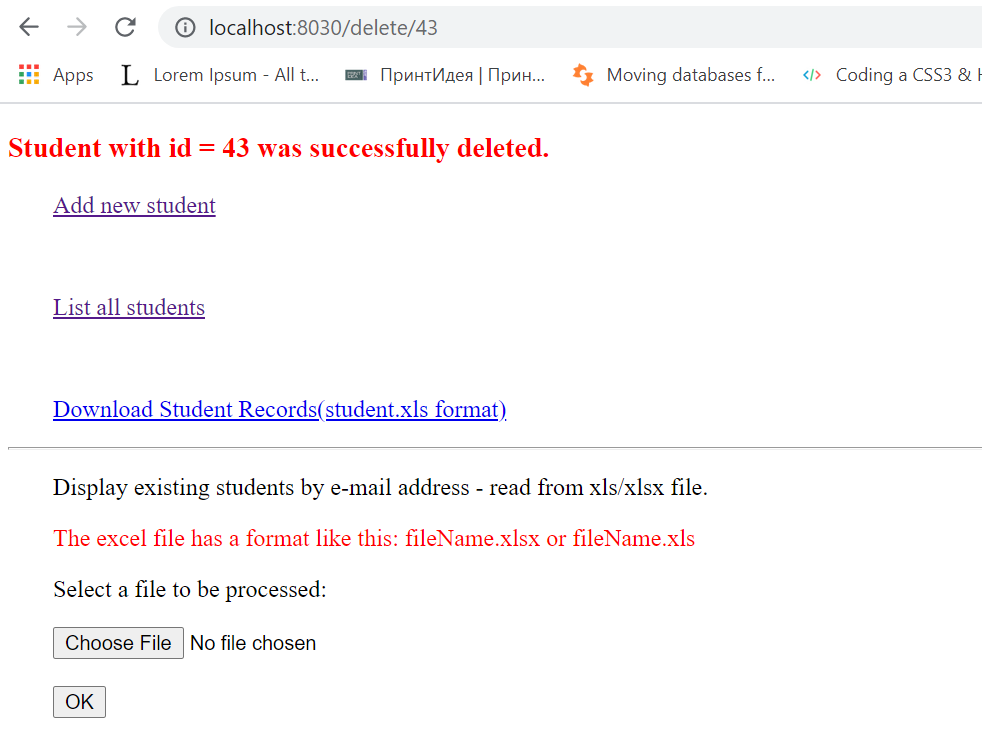
1. listOfStudent.jsp:



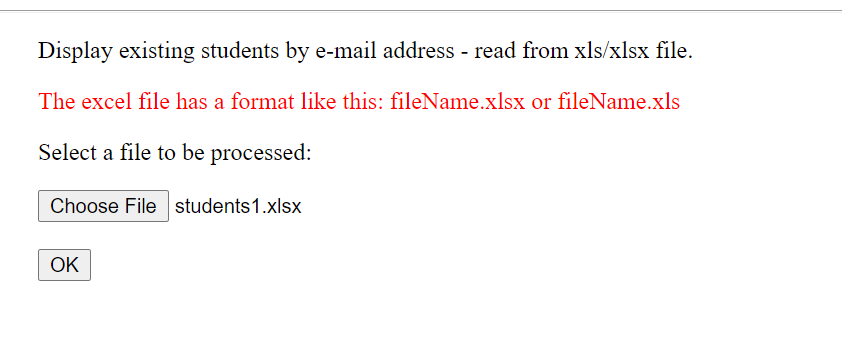
1. editStudent.jsp:



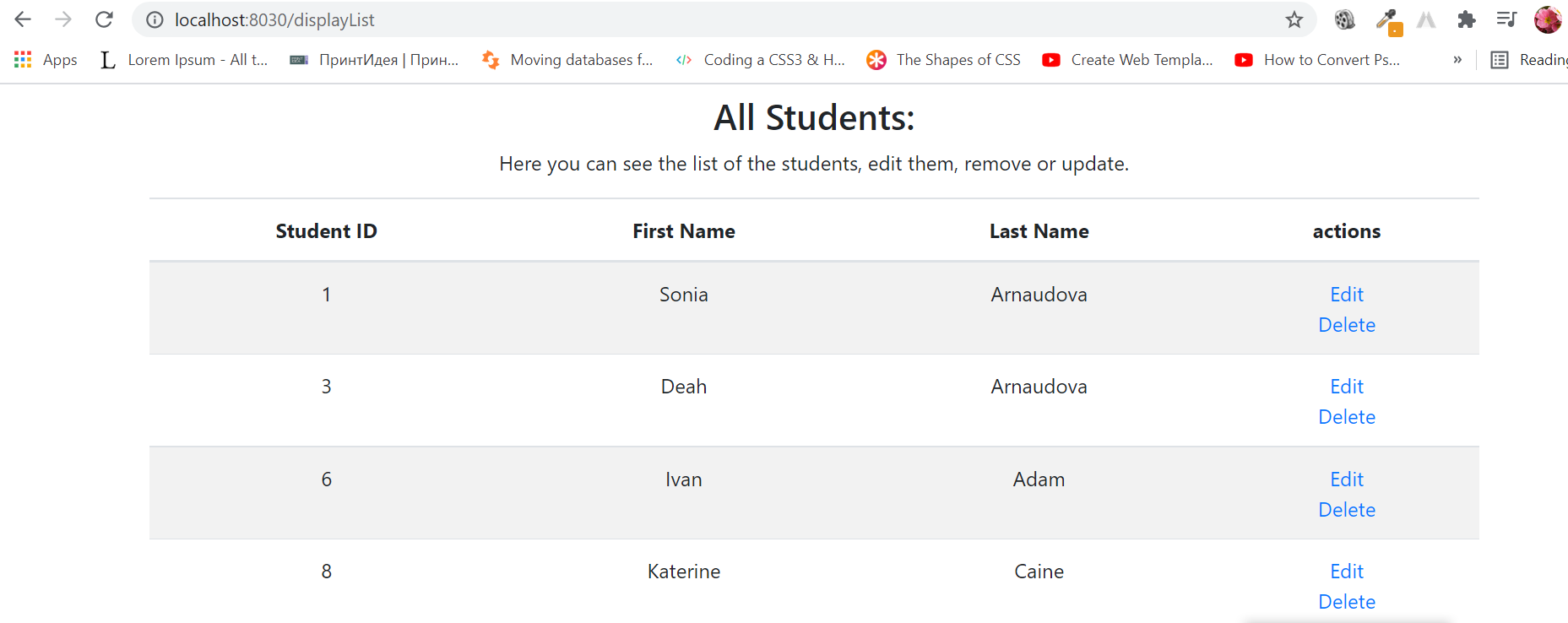
1. delete student:



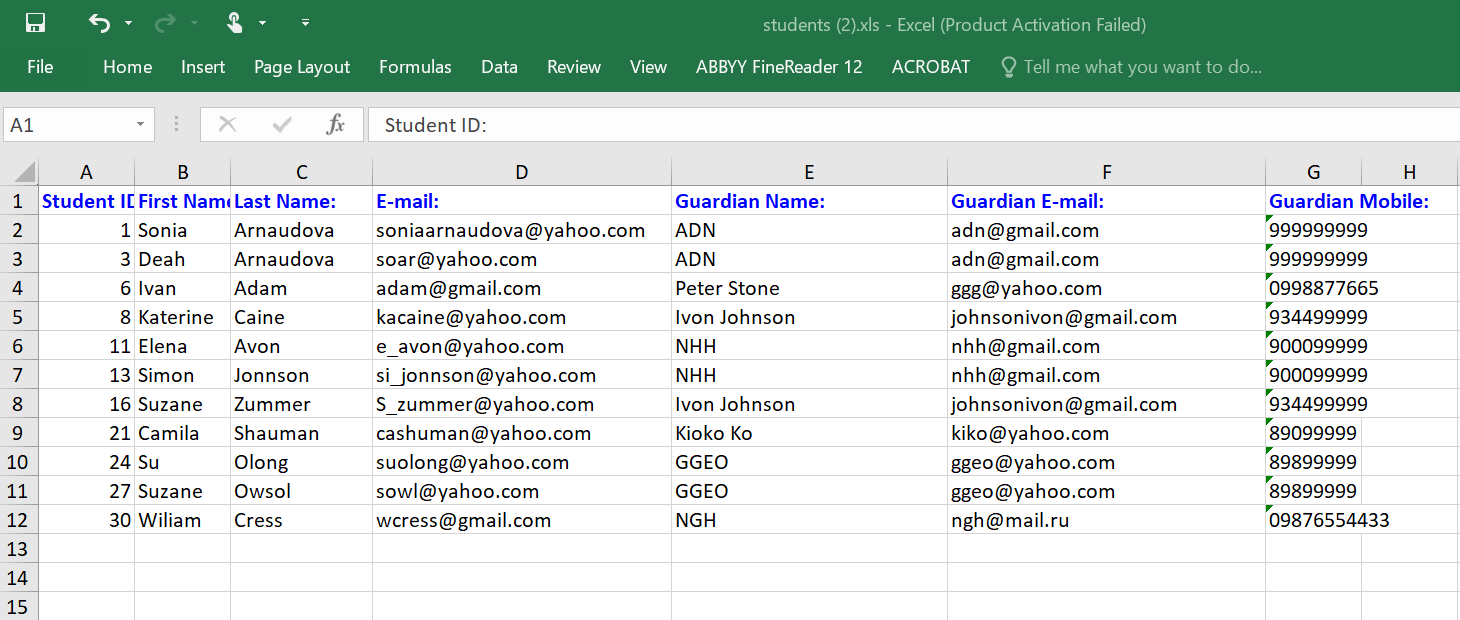
6.Read from xsl/xsls file:



The result:



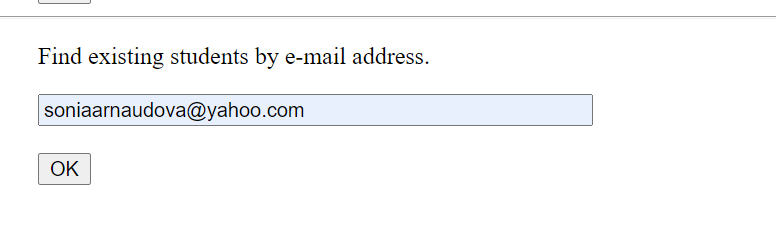
1. Download all students in xsl file:



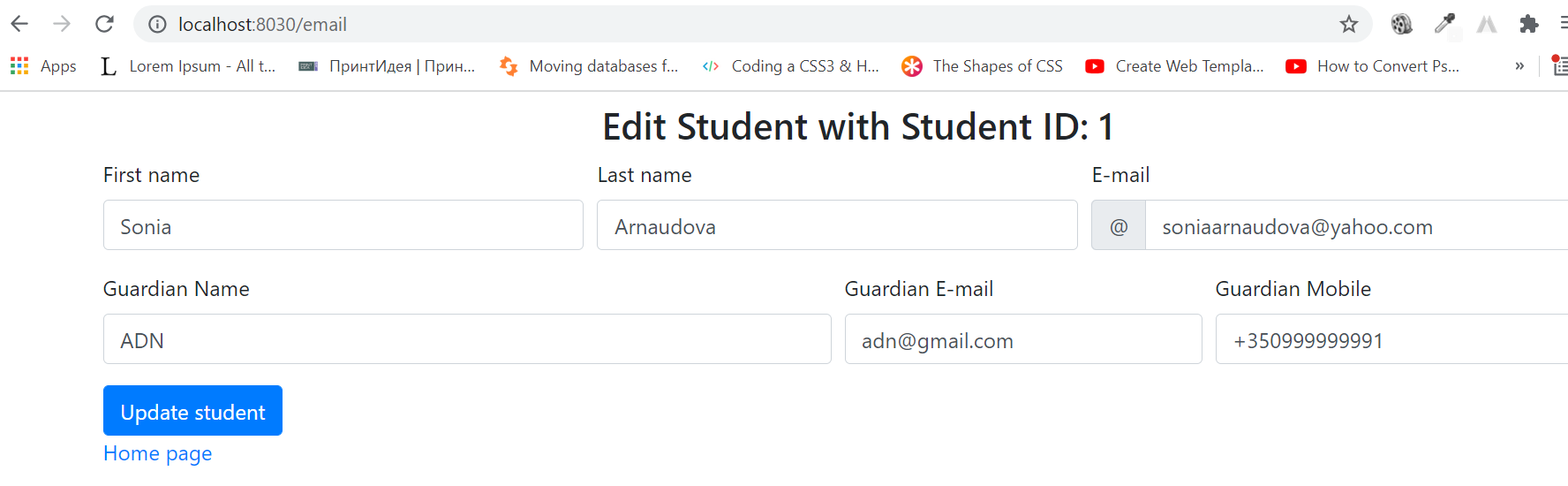
1. Find student by e-mail address:

In our controller class we add this method:

@RequestMapping("/email")  
public ModelAndView getStudentByEmail(@RequestParam("email") String email){  
 ModelAndView mv = new ModelAndView();  
 Student student = null;  
 if (email != null && !"".equals(email)) {  
 student = studentService.findStudentByEmail(email);  
 if(student != null){  
 mv.setViewName("editStudent");  
 mv.addObject("student", student);  
 mv.addObject("message", "");  
 mv.addObject("studentId",student.getStudentId());  
 }else {  
 mv.setViewName("error");  
 mv.addObject("message", "No such student.");  
 }  
 }  
  
 return mv;  
}



Goes to editStudent.jsp:



If doesn’t exist goes to error.jsp:

