Spring MVC CRUD Application - Student Management

Project: SpringStudentCURD

1. Overview

This document contains notes, explanations, and code snippets for a Spring MVC CRUD application managing a student table.

2. Database Schema

student table columns: srollno, sname, semail, sbranch, syear, ssec

SQL to create table:

```
CREATE DATABASE studentdb;
USE studentdb;
CREATE TABLE student (
   srollno INT PRIMARY KEY,
   sname VARCHAR(50),
   semail VARCHAR(50),
   sbranch VARCHAR(50),
   syear VARCHAR(10),
   ssec VARCHAR(5)
);
```

3. Project Structure

See included project ZIP. Key packages: com.example.model, com.example.dao, com.example.controller

4. Important Files

dispatcher-servlet.xml (contains datasource with username root and empty password)

5. Sample Code (Student.java)

```
package com.example.model;

public class Student {
   private int srollno;
   private String sname;
   private String semail;
   private String sbranch;
   private String syear;
   private String ssec;

   public int getSrollno() { return srollno; }
```

```
    public void setSrollno(int srollno) { this.srollno = srollno; }
    public String getSname() { return sname; }
    public void setSname(String sname) { this.sname = sname; }
    public String getSemail() { return semail; }
    public void setSemail(String semail) { this.semail = semail; }
    public String getSbranch() { return sbranch; }
    public void setSbranch(String sbranch) { this.sbranch = sbranch; }
    public String getSyear() { return syear; }
    public void setSyear(String syear) { this.syear = syear; }
    public String getSsec() { return ssec; }
    public void setSsec(String ssec) { this.ssec = ssec; }
}
```

6. Sample Code (StudentDAO.java)

```
package com.example.dao;

import java.util.List;
import com.example.model.Student;

public interface StudentDAO {
    void save(Student student);
    void update(Student student);
    void delete(int srollno);
    Student get(int srollno);
    List<Student> list();
}
```

7. Sample Code (StudentDAOImpl.java)

```
package com.example.dao;

import java.util.List;
import org.springframework.jdbc.core.BeanPropertyRowMapper;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.stereotype.Repository;
import com.example.model.Student;
import org.springframework.beans.factory.annotation.Autowired;

@Repository
public class StudentDAOImpl implements StudentDAO {
```

```
    private JdbcTemplate jdbcTemplate;

    @Autowired
    public StudentDAOImpl(JdbcTemplate jdbcTemplate) {
        this.jdbcTemplate = jdbcTemplate;
    }

    @Override
    public void save(Student s) {
        String sql = "INSERT INTO student (srollno, sname, semail, sbranch, syear, ssec) VALUES
(?, ?, ?, ?, ?, ?)";
        jdbcTemplate.update(sql, s.getSrollno(), s.getSname(), s.getSemail(), s.getSbranch(),
s.getSyear(), s.getSsec());
    }

    @Override
    public void update(Student s) {
        String sql = "UPDATE student SET sname=?, semail=?, sbranch=?, syear=?, ssec=?
WHERE srollno=?";
        jdbcTemplate.update(sql, s.getSname(), s.getSemail(), s.getSbranch(), s.getSyear(),
s.getSsec(), s.getSrollno());
    }

    @Override
    public void delete(int srollno) {
        String sql = "DELETE FROM student WHERE srollno=?";
        jdbcTemplate.update(sql, srollno);
    }

    @Override
    public Student get(int srollno) {
        String sql = "SELECT * FROM student WHERE srollno=?";
        return jdbcTemplate.queryForObject(sql, new
BeanPropertyRowMapper<>(Student.class), srollno);
    }

    @Override
    public List<Student> list() {
        String sql = "SELECT * FROM student";
        return jdbcTemplate.query(sql, new BeanPropertyRowMapper<>(Student.class));
    }
}
```

8. Controller (StudentController.java)

```
package com.example.controller;

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.*;
import com.example.dao.StudentDAO;
import com.example.model.Student;

@Controller
public class StudentController {

  @Autowired
  private StudentDAO dao;

  @GetMapping("/")
  public String viewHomePage(Model model) {
    List<Student> list = dao.list();
    model.addAttribute("listStudents", list);
    return "listStudents";
  }

  @GetMapping("/new")
  public String showNewForm(Model model) {
    model.addAttribute("student", new Student());
    return "addStudent";
  }

  @PostMapping("/save")
  public String saveStudent(@ModelAttribute("student") Student student) {
    dao.save(student);
    return "redirect:/";
  }

  @GetMapping("/edit/{srollno}")
  public String showEditForm(@PathVariable("srollno") int srollno, Model model) {
    model.addAttribute("student", dao.get(srollno));
    return "editStudent";
  }
```

```
    @PostMapping("/update")
    public String updateStudent(@ModelAttribute("student") Student student) {
        dao.update(student);
        return "redirect:/";
    }

    @GetMapping("/delete/{srollno}")
    public String deleteStudent(@PathVariable("srollno") int srollno) {
        dao.delete(srollno);
        return "redirect:/";
    }
}
```

9. JSP Views

listStudents.jsp, addStudent.jsp, editStudent.jsp are included in the project.

10. How to run

SpringStudentCURD - Ready-to-run Spring MVC CRUD project (Student table)

DB Setup:
1. Create database and table in MySQL:

```
CREATE DATABASE studentdb;
USE studentdb;
CREATE TABLE student (
    srollno INT PRIMARY KEY,
    sname VARCHAR(50),
    semail VARCHAR(50),
    sbranch VARCHAR(50),
    syear VARCHAR(10),
    ssec VARCHAR(5)
);
```

2. The dispatcher-servlet.xml is pre-configured to use:
   username: root
   password: (empty)
   url:
jdbc:mysql://localhost:3306/studentdb?useSSL=false&allowPublicKeyRetrieval=true&serverTimezone=UTC

How to run:
- Import the project into Eclipse/STS as an existing Maven project.
- Ensure Tomcat (or any servlet container) is configured as runtime and added to project.

- Build and run on server. Access: http://localhost:8080/SpringStudentCURD/

Notes:
- The project uses JdbcTemplate and simple DAO pattern.
- If you want the password to be non-empty, update dispatcher-servlet.xml accordingly.