OOAD

HANDS-ON ASSIGNMENT

MVC

NAME: SANMAT SANJAYAKUMAR PAYAGOUDAR

SRN: PES1UG20CS385

The Model-View-Controller (MVC) architecture pattern is one of the most widely used software development patterns. It is a design pattern that divides an application into three interconnected components: the model, the view, and the controller. The model represents the data and business logic, the view displays the data to the user, and the controller handles the user's actions and updates the model and view. In this write-up, we will discuss the advantages of using the MVC architecture pattern and the features of a popular MVC framework.

MVC Architecture Pattern: The MVC architecture pattern is used to separate concerns in an application. It is based on the principle of separation of concerns, where each component is responsible for a specific aspect of the application. The model represents the data and business logic, the view represents the user interface, and the controller manages the interaction between the model and view. The model and view do not communicate directly; instead, they communicate through the controller. This separation of concerns allows for easier maintenance, testing, and scalability of the application.

Advantages of MVC Pattern: The MVC architecture pattern has several advantages. Firstly, it allows for better organization of code. By separating the application into three components, each component can be developed and maintained independently, making it easier to manage and debug. Secondly, it promotes code reuse. Because the components are separated, they can be reused in different applications or different parts of the same application. Thirdly, it allows for better scalability. As the application grows, the components can be scaled individually without affecting the others. Lastly, it facilitates testing. Because the components are separated, each component can be tested independently, making it easier to identify and fix issues.

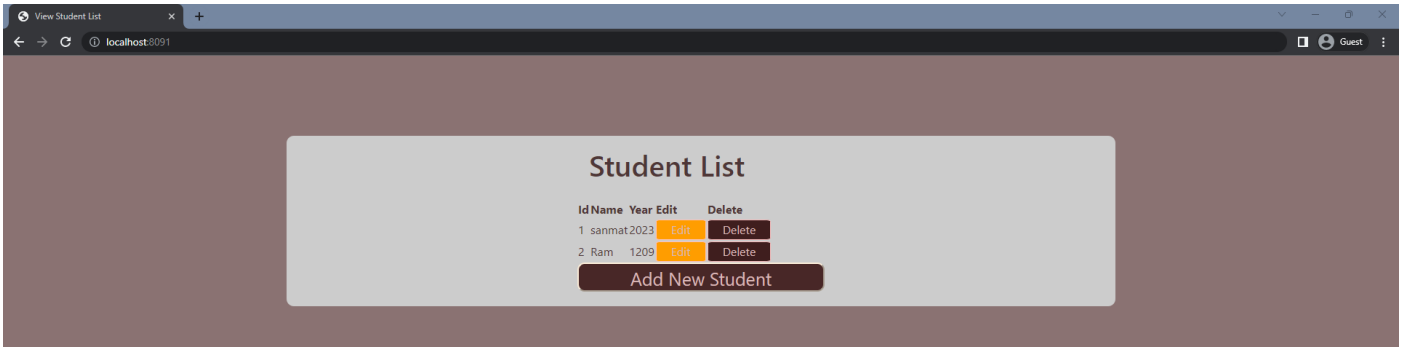
Features of MVC Framework:

A MVC framework is a software framework that implements the MVC pattern. It provides a set of tools and libraries for developers to build applications using the MVC architecture. Some of the features of a MVC framework are:

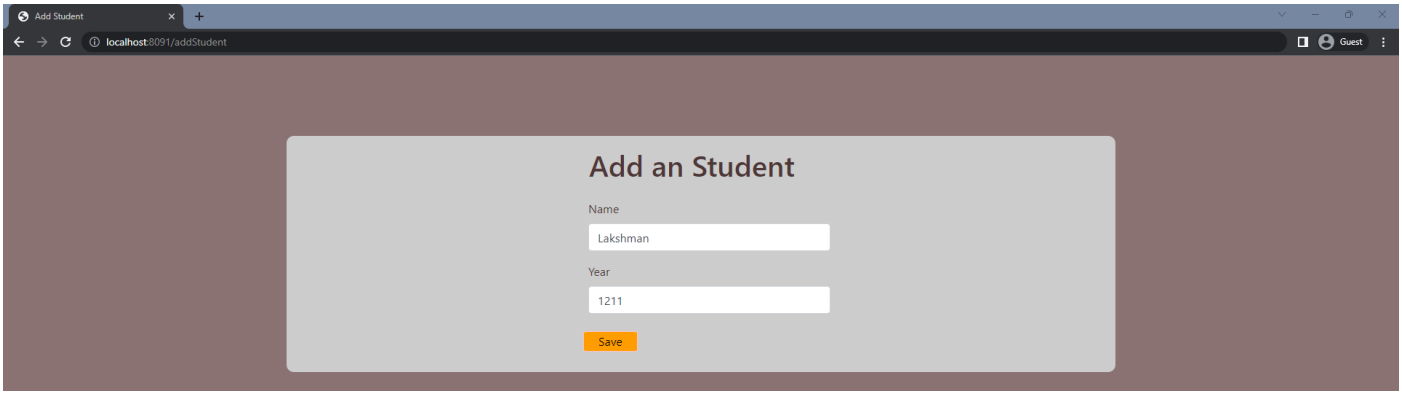
Routing, Templating, Validation, Security, Testing

STUDENT MANAGEMENT SYSTEM

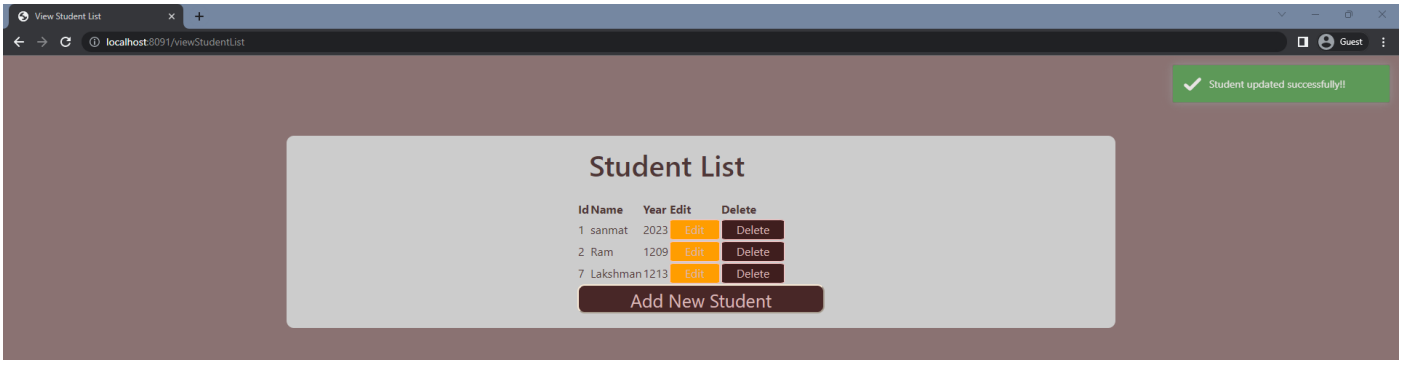
Student List



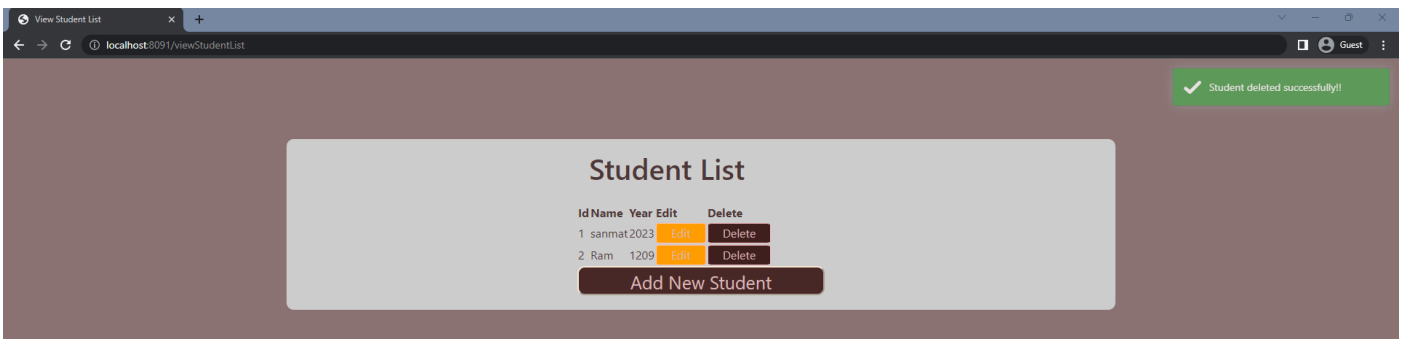
Add Student



Edit Student Details



Delete Student



Console

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.5.1)

2023-03-31 13:25:15.779  INFO 25200 --- [  restartedMain] com.example.jspdemo.MainApplication      : No active profile set, falling back to default profiles
: default
2023-03-31 13:25:15.868  INFO 25200 --- [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'spring.devtools
.add-properties' to 'false' to disable
2023-03-31 13:25:15.868  INFO 25200 --- [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider setting the
 'logging.level.web' property to 'DEBUG'
2023-03-31 13:25:16.482  INFO 25200 --- [  restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT m
ode.
2023-03-31 13:25:16.539  INFO 25200 --- [  restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 47 ms. Foun
d 1 JPA repository interfaces.
2023-03-31 13:25:17.123  INFO 25200 --- [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8091 (http)
2023-03-31 13:25:17.133  INFO 25200 --- [  restartedMain] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2023-03-31 13:25:17.133  INFO 25200 --- [  restartedMain] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.46]
2023-03-31 13:25:17.418  INFO 25200 --- [  restartedMain] org.apache.jasper.servlet.TldScanner     : At least one JAR was scanned for TLDs yet contained no
TLDs. Enable debug logging for this logger for a complete list of JARs that were scanned but no TLDs were found in them. Skipping unneeded JARs during scann
ing can improve startup time and JSP compilation time.
2023-03-31 13:25:17.429  INFO 25200 --- [  restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2023-03-31 13:25:17.429  INFO 25200 --- [  restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in
 1557 ms
2023-03-31 13:25:17.511  INFO 25200 --- [  restartedMain] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2023-03-31 13:25:17.695  INFO 25200 --- [  restartedMain] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
it 'default'
2023-03-31 13:25:19.139  WARN 25200 --- [  restartedMain] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefor
e, database queries may be performed during view rendering. Explicitly configure spring.jpa.open-in-view to disable this warning
2023-03-31 13:25:19.451  INFO 25200 --- [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is running on port 35729
2023-03-31 13:25:19.507  INFO 25200 --- [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8091 (http) with context pat
h ''
2023-03-31 13:25:19.524  INFO 25200 --- [  restartedMain] com.example.jspdemo.MainApplication      : Started MainApplication in 4.192 seconds (JVM running f
or 4.772)
2023-03-31 13:25:25.890  INFO 25200 --- [nio-8091-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherServle
t'
2023-03-31 13:25:25.890  INFO 25200 --- [nio-8091-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2023-03-31 13:25:25.894  INFO 25200 --- [nio-8091-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 4 ms
```

Database

Data Output | Messages | Notifications

	id [PK] bigint	name character varying (255)	year integer
1	1	sanmat	2023
2	2	Ram	1209

Application Properties

```
server.port = 8091

spring.mvc.view.prefix=/WEB-INF/jsp/
spring.mvc.view.suffix=.jsp

spring.datasource.url=jdbc:postgresql://localhost:5432/sanmat
spring.datasource.username=postgres
spring.datasource.password=123

spring.jpa.hibernate.ddl-auto=update
```

Repository

```java
package com.example.jspdemo.repo;

import com.example.jspdemo.model.Student;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface IStudentRepository extends JpaRepository<Student, Long> {
}

```

Student.java

```java
package com.example.jspdemo.model;

import javax.persistence.*;

@Entity
@Table(name="Student")
public class Student {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    @Column
    private String name;

    @Column
    private int year;

    public Student() {

    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getYear() {
        return year;
    }

    public void setYear(int year) {
        this.year = year;
    }
}

```

Controller

```java
package com.example.jspdemo.controller;

import com.example.jspdemo.model.Student;
import com.example.jspdemo.service.StudentService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;

@Controller
public class StudentController {

    @Autowired
    StudentService StudentService;

    @GetMapping({"/", "/viewStudentList"})
    public String viewStudentList(@ModelAttribute("message") String message, Model model) {
        model.addAttribute(attributeName:"StudentList", StudentService.getAllStudent());
        model.addAttribute(attributeName:"message", message);

        return "ViewStudentList";
    }

    @GetMapping("/addStudent")
    public String addStudent(@ModelAttribute("message") String message, Model model) {
        model.addAttribute(attributeName:"Student", new Student());
        model.addAttribute(attributeName:"message", message);

        return "AddStudent";
    }

    @PostMapping("/saveStudent")
    public String saveStudent(Student Student, RedirectAttributes redirectAttributes) {
        if (StudentService.saveOrUpdateStudent(Student)) {
            redirectAttributes.addFlashAttribute(attributeName:"message", attributeValue:"Save Success");
            return "redirect:/viewStudentList";
        }

        redirectAttributes.addFlashAttribute(attributeName:"message", attributeValue:"Save Failure");
        return "redirect:/addStudent";
    }

    @GetMapping("/editStudent/{id}")
    public String editStudent(@PathVariable Long id, Model model) {
        model.addAttribute(attributeName:"Student", StudentService.getStudentById(id));

        return "EditStudent";
    }

    @PostMapping("/editSaveStudent")
    public String editSaveStudent(Student Student, RedirectAttributes redirectAttributes) {
        if (StudentService.saveOrUpdateStudent(Student)) {
            redirectAttributes.addFlashAttribute(attributeName:"message", attributeValue:"Edit Success");
            return "redirect:/viewStudentList";
        }

        redirectAttributes.addFlashAttribute(attributeName:"message", attributeValue:"Edit Failure");
        return "redirect:/editStudent/" + Student.getId();
    }

    @GetMapping("/deleteStudent/{id}")
    public String deleteStudent(@PathVariable Long id, RedirectAttributes redirectAttributes) {
        if (StudentService.deleteStudent(id)) {
            redirectAttributes.addFlashAttribute(attributeName:"message", attributeValue:"Delete Success");
        } else {
            redirectAttributes.addFlashAttribute(attributeName:"message", attributeValue:"Delete Failure");
        }

        return "redirect:/viewStudentList";
    }

}
```