**Universidade do Grande Rio - UNIGRANRIO**

Matheus Acácio Rodrigues

5306645

**AP3 - Frameworks**

Spring MVC e Hibernate

Duque de Caxias, Xerém

2021

**SPRING MVC E HIBERNATE**

O Spring MVC é um framework Java criado pela Spring, que implementa o padrão MVC em aplicações web, criando as camadas de Model, View e Controller. Ele já tem todas as funcionalidades que precisamos para atender as requisições HTTP, delegar responsabilidades de processamento de dados para outros componentes e preparar a resposta que precisa ser dada. Um dos pontos positivos do Framework Spring é sua fácil integração com outros frameworks e web services. Um desses frameworks é o Hibernate, uma ORM que modela em forma de objeto-relacional o banco de dados.

Para que possa haver persistência dos dados, é necessário que os dados sejam salvos em banco, uma forma que torna esse trabalho mais fácil sem precisar se preocupar muito com comandos SQL é a utilização de ORMs, que podem converter comandos SQL em scripts da linguagem utilizada na aplicação, como por exemplo o Java. O Hibernate é uma ORM para linguagem Java que possibilita fazer essas funções.

**DESENVOLVIMENTO**

Para exemplificação foi feito um projeto web que cadastre Alunos, o curso do aluno e sua cidade. O projeto foi inicializado utilizando o Maven para administrar as dependências e o Spring Boot para as configurações iniciais do projeto, utilizado o PostgresSQL para armazenar os dados e o Hibernate para modelar o banco. Nos códigos abaixo é possivel visualizar o arquivo pom.xml e as dependências que ela gerencia, como por exemplo, o Spring Web, o JPA e o PostresSQL.

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <parent>

        <groupId>org.springframework.boot</groupId>

        <artifactId>spring-boot-starter-parent</artifactId>

        <version>2.5.6</version>

        <relativePath/> <!-- lookup parent from repository -->

    </parent>

    <groupId>com.ap3frameworks</groupId>

    <artifactId>ap3</artifactId>

    <version>0.0.1-SNAPSHOT</version>

    <name>ap3</name>

    <description>Demo project for Spring Boot</description>

    <properties>

        <java.version>11</java.version>

    </properties>

<dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-data-jpa</artifactId>

        </dependency>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-thymeleaf</artifactId>

        </dependency>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-web</artifactId>

        </dependency>

No código abaixo é apresentado o controller do Aluno, contendo as métodos para cadastrar aluno, voltar para Home, e ir para pagina de listar alunos.

@Controller

public class AlunoController {

    @Autowired

    private AlunoDao alunoRepository;

    @GetMapping("/index")

    public ModelAndView index() {

        ModelAndView mv = new ModelAndView();

        mv.setViewName("home/index");

        mv.addObject("aluno", new Aluno());

        return mv;

    }

    @GetMapping("/inserirAlunos")

    public ModelAndView InsertAlunos(Aluno aluno) {

        ModelAndView mv = new ModelAndView();

        mv.setViewName("Aluno/formAluno");

        mv.addObject("aluno", new Aluno());

        return mv;

    }

    @PostMapping("InsertAlunos")

    public ModelAndView inserirAluno(Aluno aluno) {

        ModelAndView mv = new ModelAndView();

        mv.setViewName("redirect:/alunos-adicionados");

        alunoRepository.save(aluno);

        return mv;

    }

    @GetMapping("alunos-adicionados")

    public ModelAndView listagemAlunos() {

        ModelAndView mv = new ModelAndView();

        mv.setViewName("Aluno/listAlunos");

        mv.addObject("alunosList", alunoRepository.findAll());

        return mv;

    }

}

Abaixo os códigos do Dominio do Aluno já contendo a modelagem objeto-relacional dos atributos do aluno e os códigos de configuração do hibernate e do PostgresSQL

@Entity

public class Aluno {

    @Id

    @GeneratedValue(strategy = GenerationType.AUTO)

    private Integer id;

    @Column(name = "name")

    public String name;

    @Column(name = "city")

    private String city;

    @Column(name = "course")

    private String course;

    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 String getCity() {

        return city;

    }

    public void setCity(String city) {

        this.city = city;

    }

    public String getCourse() {

        return course;

    }

    public void setCourse(String course) {

        this.course = course;

    }

}

spring.datasource.url=jdbc:postgresql://ec2-35-169-43-5.compute-1.amazonaws.com:5432/d5uegrea682oo9

spring.datasource.username=qhhwykebbjecmd

spring.datasource.password=f0d4d2ac2cd0e24f9471684915e61ab87b2c6d40287a4d79c4dc43969a9ed9b5

spring.jpa.hibernate.ddl-auto=update

OBS: Foi utilizado um banco de dados hospedado no site da Heroku na nuvem, ao rodar a aplicação é possivel ver os dados que já estão armazenados nesse banco de forma remota. Por esse motivo as credencias inseridas na configuração do banco são um pouco diferentes de um banco que roda localmente.