

# Spring Framework: Introdução

Prof. Pedro Pongelupe



PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS  
Departamento de Ciência da Computação

# Sumário

- 1 Spring
  - Prelúdio
  - Spring Boot
- 2 spring initializr
- 3 Exemplo Aluno
  - Speedrun de Conceitos
  - Nosso primeiro CRUD



# Frameworks

## O quê é um framework?

Um *framework* é uma **abstração** que une códigos entre vários projetos de software provendo uma funcionalidade genérica.

## Framework x Biblioteca

Ao contrário das bibliotecas, é o *framework* quem dita o fluxo de controle da aplicação, chamado de **Inversão de Controle**. Ou seja, o framework possui uma forma específica de se utilizar.



# História

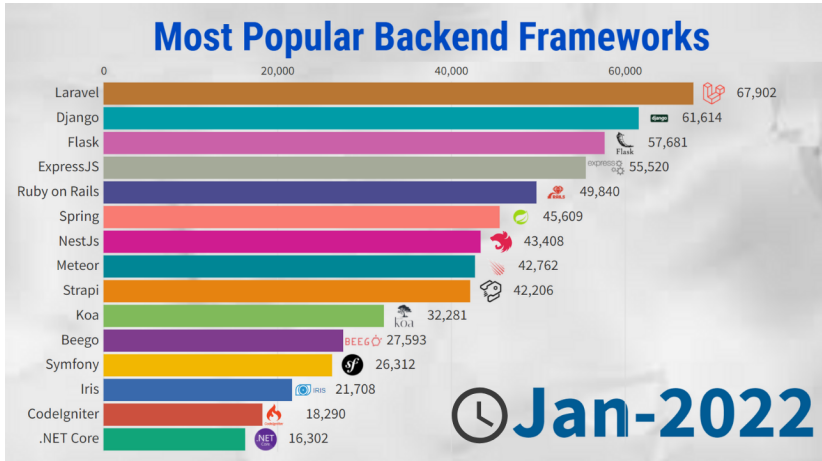
## O que é? Onde vive?

A primeira versão do Spring foi lançada em 2003. Spring é um framework de **inversão de controle** que podem ser utilizados por qualquer aplicação Java. Maaaas, as extensões web são as mais famosas.





# Frameworks





# primavera - Spring Boot

## primavera - Don L

*Eu que sou de onde a miséria seca as estações  
Vi a primavera  
Florescer entre os canhões  
E não recuar  
Eu que sou de guerra  
Dei o sangue na missão  
De regar a terra  
Se eu tombar vão ser milhões pra multiplicar*





# Spring Boot

## O que é?

Spring Boot facilita a criação de aplicações *stand-alone*, *production-grade* Spring based Applications que você pode simplesmente 'só rodar'.

## Speedrun de *hello world*

- 1 Utilize o [start.spring.io](https://start.spring.io) para criar seu projeto
- 2 Adicione seu código
- 3 Profit!



# spring initializr



## Project

☐ Gradle - Groovy ☐ Gradle - Kotlin  
☒ Maven

## Language

☒ Java ☐ Kotlin ☐ Groovy

## Spring Boot

☐ 3.2.0 (SNAPSHOT) ☐ 3.2.0 (RC1) ☐ 3.1.6 (SNAPSHOT) ☒ 3.1.5  
☐ 3.0.13 (SNAPSHOT) ☐ 3.0.12 ☐ 2.7.18 (SNAPSHOT) ☐ 2.7.17

## Project Metadata

Group

Artifact

Name

Description

Package name

Packaging ☒ Jar ☐ War

Java ☐ 21 ☒ 17 ☐ 11 ☐ 8

## Dependencies

[ADD DEPENDENCIES...](#) CTRL + B

No dependency selected

GENERATE CTRL + G

EXPLORE CTRL + SPACE

SHARE...





# spring initializr - Principais dependências

- Developer Tools
- Web\*
- Template Engines
- Security
- SQL\*
- NOSQL
- Messaging
- I/O
- OPS
- Observability
- Testing
- Spring Cloud
- Spring Cloud Config
- Spring Cloud Discovery
- Spring Cloud Routing
- Spring Cloud Circuit Breaker
- Spring Cloud Messaging
- VMware Tanzu Application Service
- Microsoft Azure
- Google Cloud
- AI



# Protocolo HTTP

## Principais características

Em linhas gerais, HTTP é um protocolo *web* para a transferência de hipertexto entre um cliente e servidor. Ou seja, é o protocolo que provê a base de funcionamento da internet. Sendo que as principais componentes desse protocolo são:

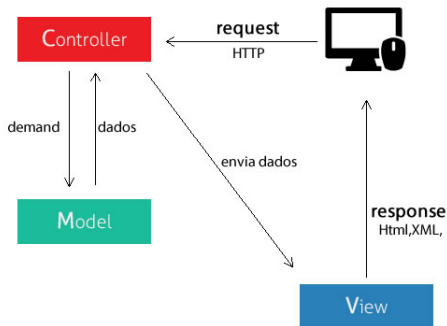
- Endereço recurso web - URL
- Verbos HTTP - GET, POST, PUT, DELETE, etc
- Body da requisição
- Cabeçalhos - *Accept*, *Connection*, etc
- Response Status Code - 2xx, 4xx, 5xx



# Padrão MVC

## Principais características

Em linhas gerais, o Padrão MVC é uma arquitetura de software formulada na década de 1980, focado no **reuso de código** e **separação de conceitos** em três camadas interconectadas.





# ORM - Mapeamento Objeto-Relacional

## Principais características

Em linhas gerais, os ORM traduzem as classes de um sistema Orientado por Objetos em tabelas de banco de dados. Ou seja, Com esta técnica, o programador não precisa se preocupar com os comandos em linguagem SQL; ele irá usar uma interface de programação simples que faz todo o trabalho de persistência.

## Exemplos

- Hibernate e EclipseLink - Java
- Django - Python
- Entity Framework - C#/.net
- Sequelize - Node js



# Enunciado e classe Aluno

## Enunciado

Você e outros alunos estão planejando fazer um sistema de gestão acadêmica para desbancar o SGA. O sistema de vocês chama-se SGA 2, vocês fizeram o planejamento das entidades do sistema, pensaram as interfaces e dividiram todas as tarefas em sprints. A solução projetada terá uma arquitetura cliente-servidor utilizando o Spring Boot para o back-end e o postgres para o banco de dados. Sua tarefa da primeira sprint é implementar um CRUD de aluno no back-end, ou seja, expor os seguintes 4 endpoints:



# Endpoints e classe Aluno

## Enunciado

- **POST /alunos** - Insere no banco de dados os dados do Aluno
- **GET /alunos/{matricula}** - Busca no banco de dados os dados do Aluno pela matrícula
- **PUT /alunos/{matricula}** - Atualiza no banco de dados os dados do Aluno pela matrícula
- **DELETE /alunos/{matricula}** - Apaga no banco de dados os dados do Aluno pela matrícula

### Aluno

- matricula
- nome
- idade



# Classe Aluno

```
@Entity
@Table
public class Aluno {

    @Id
    @Column
    private String matricula;

    @Column
    private String nome;

    @Column
    private int idade;

    public Aluno(String nome, String matricula, int idade) {
        this.nome = nome;
        this.matricula = matricula;
        this.idade = idade;
    }

    public Aluno() {
    }

    ... getters/setters ...
}
```



# AlunosController

```
@Controller
public class AlunosController {

    @PersistenceUnit
    private EntityManagerFactory entityManagerFactory;

    ...
}
```





# C - Create

```
...
@PostMapping("/alunos")
public @ResponseBody Aluno postAluno(@RequestBody Aluno a) {
    var entityManager = entityManagerFactory.createEntityManager();

    entityManager.getTransaction().begin();
    entityManager.persist(a);
    entityManager.getTransaction().commit();
    entityManager.close();

    return a;
}
...
```



# R - Read

```
...
@GetMapping("/{matricula}")
public @ResponseBody Aluno getAlunoByMatricula(@PathVariable String matricula) {
    var entityManager = entityManagerFactory.createEntityManager();

    entityManager.getTransaction().begin();
    var a = entityManager.find(Aluno.class, matricula);
    entityManager.getTransaction().commit();
    entityManager.close();

    return a;
}
...
```



# U - Update

```
...
@PutMapping("/{alunos/{matricula}}")
    public @ResponseBody Aluno updateAlunoByMatricula(@PathVariable String matricula,
        @RequestBody Aluno a) {
        var entityManager = entityManagerFactory.createEntityManager();

        entityManager.getTransaction().begin();
        var alunoDB = entityManager.find(Aluno.class, matricula);
        alunoDB.setMatricula(a.getMatricula());
        alunoDB.setIdade(a.getIdade());
        alunoDB.setNome(a.getNome());
        entityManager.persist(alunoDB);
        entityManager.getTransaction().commit();
        entityManager.close();

        return a;
    }
...
```



# D - Delete

```
...
    @DeleteMapping("/{matricula}")
    public @ResponseBody Aluno deleteAlunoByMatricula(@PathVariable String matricula) {
        var entityManager = entityManagerFactory.createEntityManager();

        entityManager.getTransaction().begin();
        var a = entityManager.find(Aluno.class, matricula);
        entityManager.remove(a);
        entityManager.getTransaction().commit();
        entityManager.close();

        return a;
    }
...
```



# Obrigado!!

Muito obrigado pela atenção! Alguma dúvida? Bora praticar!!!

*"Hay que endurecer sem nunca sem nunca perder a ternura"*