Lista 4 - Injeção de dependência

Prazo de entrega: 07/09/2024 até 23:59 - (Sábado)

Instruções de entrega: Você deve ter um repositório em seu github para a disciplina de Programação procedural chamado **desenvolvimento-backend**
Crie uma pasta dentro dele chamada Listas e dentro dela crie uma pasta chamada Lista4.
Para cada exercício da lista deve ser criado um projeto spring boot (pode utilizar o spring initializr)
Ao terminar a lista, suba seus exercícios no github e envie o link no formulário abaixo.

Link para o formulário de envio: https://forms.gle/ztptrwL7eKDZyV5w6

MONITORIAS: Serão realizados quatro dias de monitorias para auxiliar com a resolução dos exercícios. As datas e horários das monitorias são:

Dia	Horários		
Segunda-Feira	19h até 21h		
Terça-Feira	19h até 21h	-	-
Quarta-Feira	09h até 11h	19h até 21h	-
Quinta-Feira	19h até 21h	-	-
Sexta-Feira	09h até 11h	19h até 21h	22h até 23:59

OBS: Se você não conseguir participar das monitorias nos dias e horários propostos, você pode solicitar uma monitoria em uma data e horário diferente, desde que seja comunicado com pelo menos um dia de antecedência 🙂

Exercícios

1. Você foi contratado por uma plataforma de leilões de obras de arte para a criação de um sistema de gerenciamento das obras.

O sistema em questão deve ser construído com Spring Boot, deve utilizar arquitetura **MVC** e a **biblioteca Jackson** para realizar a persistência das obras em um arquivo Json. Além disso, deve-se usar o conceito de **injeção de dependência**.

Você deve criar um programa em Spring boot que contenha uma classe chamada Obra

```
public class Obra {
    private int id;
    private String nome;
    private String autor;
    private BigDecimal preco;

    @JsonProperty("ja_foi_vendida")
    private boolean jaFoiVendida;

    public Obra() {
    }

    public Obra(int id, String nome, String autor, BigDecimal
preco, boolean jaFoiVendida) {
        this.id = id;
        this.nome = nome;
```

```
        this.autor = autor;
        this.preco = preco;
        this.jaFoiVendida = jaFoiVendida;
    }

    public int getId() {
        return id;
    }

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

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public String getAutor() {
        return autor;
    }

    public void setAutor(String autor) {
        this.autor = autor;
    }

    public BigDecimal getPreco() {
        return preco;
    }

    public void setPreco(BigDecimal preco) {
        this.preco = preco;
    }

    public boolean jaFoiVendida() {
        return jaFoiVendida;
    }

    public void setJaFoiVendida(boolean jaFoiVendida) {
        this.jaFoiVendida = jaFoiVendida;
    }
}
```

Requisitos:

- Criar uma classe ObraRepository
- **Adicionar um File e um ObjectMapper ao ObraRepository**
- O File deve manipular um arquivo tb_obras.json para obter, salvar e atualizar as obras
- Criar os métodos getAll, save e update (devem manipular o arquivo tb_obras.json)
- Seu repository deve permitir o cadastro, a listagem e a atualização de obras (a atualização deve permitir alterar o nome da obra, o altor, o preço e o status de venda)
- Criar a classe ObraService com os métodos listarObras, cadastrarNovaObra e atualizarObra (injetar ObraRepository via construtor)
- Criar a classe ObraController com os métodos POST, PUT e GET (cadastrar, atualizar e listar obras, respectivamente)
- Ser capaz de cadastrar e listar as obras através da rota http://localhost:8080/obras

Exemplo de entrada:
POST *http://localhost:8080/app/obras*

body

```
{
    "id": 1,
    "nome": "Os retirantes",
    "autor": "Candido Portinari",
    "preco": 11400000.00,
    "ja_foi_vendida": false
}
```

Saída esperada:

- *HTTP status: 201 (Created)*

Exemplo de entrada:
PUT *http://localhost:8080/app/obras*

body

```
{
    "id": 1,
    "nome": "Os retirantes",
    "autor": "Candido Portinari",
    "preco": 12500000.00,
    "ja_foi_vendida": true
}
```

Saída esperada:

- *HTTP status: 200 (OK)*

Exemplo de entrada:
GET http://localhost:8080/app/*obras*

Saída esperada:

- Deve listar todos as obras. Suponha que existam 2 obras. Seriam listados nesse formato:

```
[
    {
        "id": 1,
        "nome": "Os retirantes",
        "autor": "Candido Portinari",
        "preco": 11400000.00,
        "ja_foi_vendida": true
    },
    {
        "id": 2,
        "nome": "A Tentação de Santo Antão",
        "autor": "Salvador Dalí",
        "preco": 65850000.00,
        "ja_foi_vendida": false
    }
]
```