Fundamentos de Desenvolvimento de Software
Prof. Marcelo H. Yamaguti

EXERCÍCIO DE ARMAZENAMENTO DE DADOS EM REPOSITÓRIO

Este exercício aborda a criação e execução de um *backend* em Spring Boot, com armazenamento de dados em um repositório de dados.

Siga e execute os passos do roteiro.

1. Realize os passos do roteiro anterior: "Roteiro Spring-Boot" dos passos 1. até 4., para criar e configurar um projeto Spring-Boot. Nas dependências garanta que haja:
    - Lombok Developer Tools
    - Rest Repository Web
    - H2 Database SQL
2. Acesse os códigos-fonte disponíveis no Moodle na pasta "Padrões no Spring-Boot: códigos-exemplo" da aula em que se discutiu padrões de Injeção de Dependência e de Inversão de Controle no Spring-Boot.
3. Adapte ou crie classes em seu projeto a partir destes códigos fonte (obs.: adapte o nome do 'package', se necessário).
4. Execute a aplicação e teste os *endpoints*.

• **Atividade – Implementando Repository em memória:**

1. Crie uma interface IAcervoRepository com o código a seguir (obs.: adapte o nome do 'package', se necessário.

```
//-----------------------------------
package br.pucrs.nomeusuario.exemplo;

import java.util.List;
import org.springframework.http.ResponseEntity;

public interface IAcervoRepository {
    List<Livro> getLivros();
    List<String> getTitulos();
    List<String> getAutores();
    List<Livro> getLivrosDoAutor(String autor);
    List<Livro> getLivrosDoAutor(String autor, int ano);
    Livro getLivroTitulo(String titulo);
    boolean cadastraLivroNovo(Livro livro);
    boolean removeLivro(int id);
}
//-----------------------------------
```

2. Crie uma classe AcervoRepoMemoriaImpl que implemente interface IAcervoRepository com o código a seguir. Implemente os demais métodos. Obs.: adapte o nome do 'package', se necessário.

```
//-----------------------------------
package br.pucrs.nomeusuario.exemplo;

import java.util.*;
import org.springframework.http.*;
import org.springframework.stereotype.Repository;
```

```
@Repository
public class AcervoRepoMemoriaImpl implements IAcervoRepository {
    private List<Livro> livros;

    public AcervoRepoMemoriaImpl(){
        livros = new ArrayList<>();

        livros.add(new Livro(110, "Aprendendo Java", "Maria da Silva", 2015));
        livros.add(new Livro(120, "Spring-Boot", "Jose de Souza", 2020));
        livros.add(new Livro(130, "Principios SOLID", "Pedro da Silva", 2023));
        livros.add(new Livro(140, "Padroes de Projeto", "JoanaMoura", 2023));
        livros.add(new Livro(150, "Teste Unitario", "Pedro da Silva", 2024));
    }

    @Override
    public List<Livro>getLivros(){
        return livros;
    }

    @Override
    public boolean removeLivro(int id) {
        List<Livro> tmp = livros.stream()
            .filter(livro->livro.getId() == id)
            .toList();
        return tmp.removeAll(tmp);
    }

    // Implemente os demais métodos, o mínimo para compilar

}
//-----------------------------------
```

3. Altere o início da classe ExemploController.java o construtor para:

```
public class ExemploController {
    private IAcervoRepository acervo;

    @Autowired
    public ExemploController(IAcervoRepository acervo) {
        this.acervo = acervo;
    }
```

4. Execute a aplicação e teste os *endpoints.*

• **Atividade – Implementando Repository com JDBC:**

1. Altere o arquivo pom.xml para que possua a seguinte dependência em <dependencies>:

```
<dependency>
    <groupId>
        org.springframework.boot
    </groupId>
    <artifactId>
        spring-boot-starter-jdbc
    </artifactId>
</dependency>
```

2. Verifique se a dependência a seguir está no pom.xml. Se não estiver, inclua-a.

```
<dependency>
    <groupId>
       com.h2database
    </groupId>
    <artifactId>
       H2
    </artifactId>
    <scope>
       Runtime
    </scope>
</dependency>
```

3. Feche o arquivo pom.xml e aguarde as atualizações pelo Maven.

4. Localize o arquivo application.properties (deve estar na pasta 'resources') e inclua as linhas a seguir:

```
spring.sql.init.mode=always
spring.sql.init.schema-locations=classpath*:./create.sql
spring.sql.init.data-locations=classpath*:./insert.sql
spring.sql.init.encoding=UTF-8
```

5. Na pasta 'resources' crie o arquivo create.sql com o conteúdo a seguir:

```
DROP TABLE livros IF EXISTS;
CREATE TABLE livros (id int,
                titulo VARCHAR(255),
                autor VARCHAR(255),
                ano int,
                PRIMARY KEY(id)
                );
```

6. Na pasta 'resources' crie o arquivo insert.sql com o conteúdo a seguir:

```
INSERT INTO livros (id,titulo,autor,ano) VALUES (110,'Aprendendo Java', 'Maria da Silva', 2015);
INSERT INTO livros (id,titulo,autor,ano) VALUES (120, 'Spring-Boot', 'Jose de Souza', 2020);
INSERT INTO livros (id,titulo,autor,ano) VALUES (130, 'Principios SOLID', 'Pedro da Silva', 2023);
INSERT INTO livros (id,titulo,autor,ano) VALUES (140, 'Padroes de Projeto', 'JoanaMoura', 2023);
INSERT INTO livros (id,titulo,autor,ano) VALUES (150, 'Teste Unitario', 'Pedro da Silva', 2024);
```

7. Crie uma classe AcervoRepoJdbcImpl que implemente interface IAcervoRepository com o código a seguir. Implemente os demais métodos. Obs.: adapte o nome do 'package', se necessário.:

```
//---------------------------------

package br.pucrs.nomeusuario.exemplo;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Primary;
import org.springframework.http.ResponseEntity;
```

```
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.stereotype.Repository;

@Repository
@Primary
public class AcervoRepoJdbcImpl implements IAcervoRepository {
    private JdbcTemplate jdbcTemplate;

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

    @Override
    public List<Livro> getLivros() {
        List<Livro> resp = this.jdbcTemplate.query("SELECT * FROM livros",
                (rs, rowNum) -> new Livro(rs.getInt("id"), rs.getString("titulo"), rs.getString("autor"),
                        rs.getInt("ano")));
        return resp;
    }

    @Override
    public boolean removeLivro(int id) {
        String sql = "DELETE FROM livros WHERE id = " + id;
        this.jdbcTemplate.batchUpdate(sql);
        return true;
    }

    // Implemente os demais métodos, o mínimo para compilar

}

//-----------------------------------
```

8. Execute a aplicação e teste os *endpoints.*

• **Atividade opcional – implementar os métodos que faltam na classe** AcervoRepoJdbcImpl**:**

1. Implemente os demais métodos da classe.
2. No Postman execute testes para os *endpoints*.

O roteiro descrito abordou a criação e execução de um *backend* em Spring Boot, com armazenamento de dados em um repositório de dados.

Lembretes:

- Como o Codespaces executa de forma virtual, lembre-se que continuamente fazer commit e push para o GitHub.
- Para executar os testes no Postman é necessário que a aplicação Spring-Boot já esteja executando.
- Ao final deste roteiro, não se esqueça de parar o Codespace com o comando ‘Stop Current Codespace’.