CONGRESSO CATARINENSE DE CIÊNCIA DA COMPUTAÇÃO
2021. Ano 8, Vol. 8. ISSN: 2319-0655

Framework para criação de tabelas em banco de dados SQL

Adriano Zimmermann
Estudante do curso Bacharelado em Ciências da Computação do Instituto Federal Catarinense
Instituto Federal Catarinense
adrianorslsc@hotmail.com,

Abstract. *This article will demonstrate how the creation was made and how the created framework works, the proposal of this framework and facilitate the creation of scripts for sql database, the language was made in Java to be used in the creation of scripts for MySQL.*

Resumo. *Este artigo demonstrará como foi feita a criação e como funciona o framework criado, a proposta deste framework e facilitar a criação de scripts para banco de dados sql, a linguagem foi feita em Java para ser usada na criação de scripts para o MySQL.*

Introdução

Um script SQL geralmente contém instruções SQL, comandos condicionais, variáveis, comentários, palavras-chave e outros elementos textuais importantes para lidar com dados e representar a lógica de algum algoritmo. Com isso há a necessidade de criar tabelas dentro dos padrões do banco de dados de modo que não causem erros na inserção de dados. Para facilitar essa criação de tabelas a ideia deste framework e gerar um padrão de formatação das tabelas agilizando assim o processo, onde mesmo uma pessoa com pouco conhecimento em banco de dados poderá criar a tabela no formato aceitado pelo MySQL.

Objetivo geral do framework

Este framework foi desenvolvido para criar tabelas dentro dos padrões do MySQL com a possibilidade de salvar as tabelas criadas em formato Json e Csv, o framework também facilitará a conexão com o banco de dados.

Design patterns

O padrão de projeto utilizado foi a fachada (Facade), este padrão de projeto e usado para encapsular funcionalidades criando assim uma interface simplificada e assim facilitar a utilização do framework criado.

Diagrama de classes

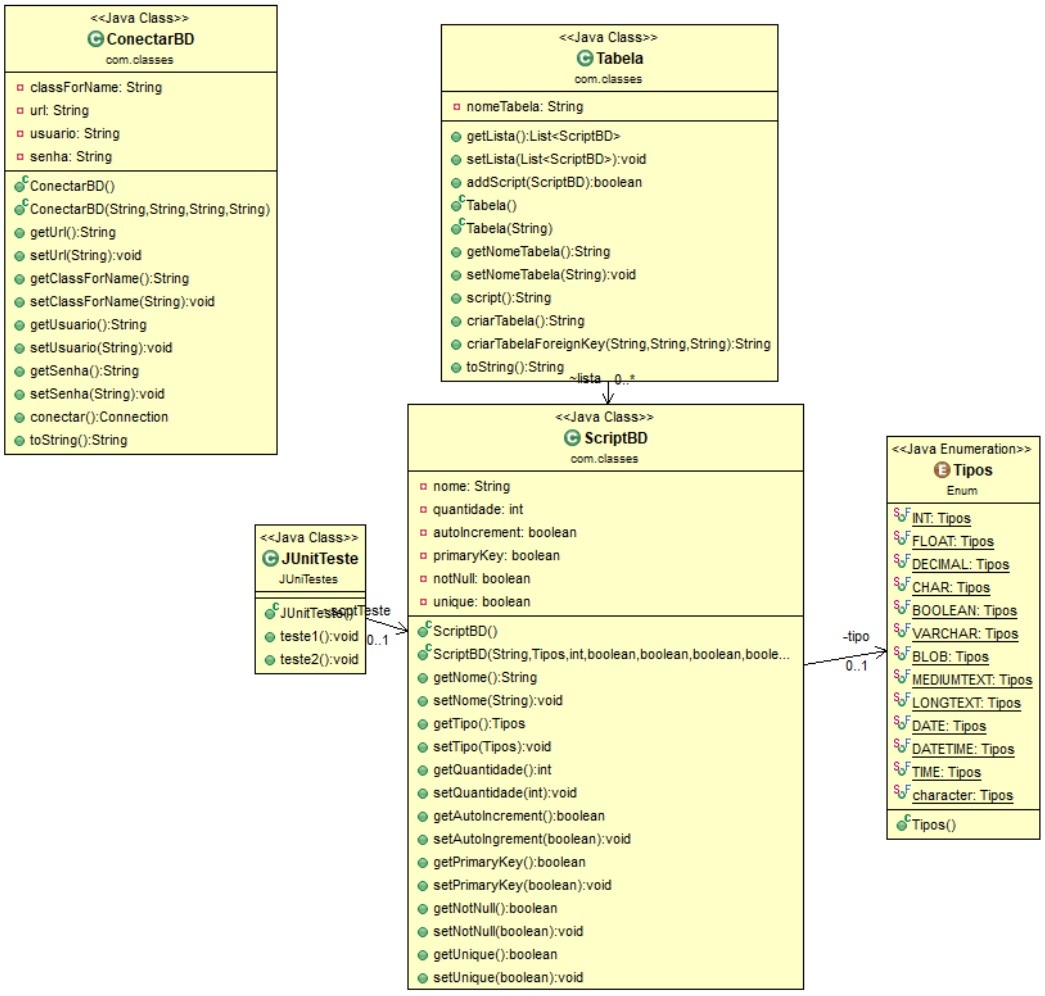

Imagem 1 (Fonte autor)

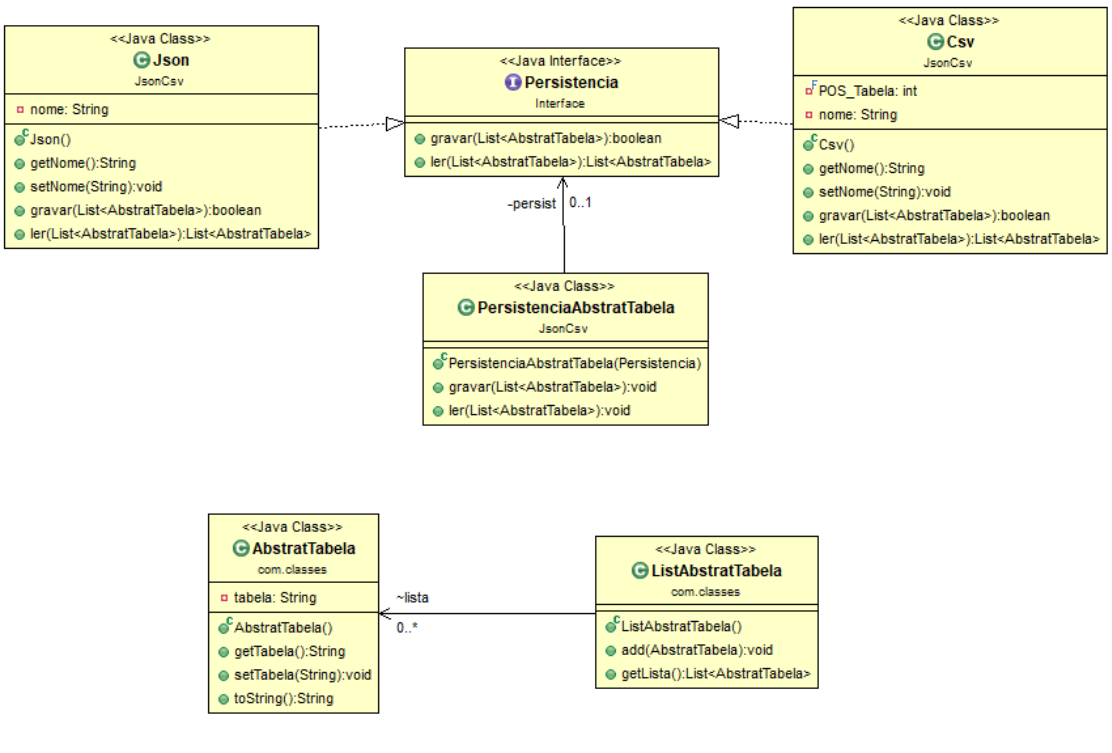

Imagem 2 (Fonte autor)

Desenvolvimento da estrutura das classes

A classe ScriptBD e responsável para formatar através do seu construtor a quais dados devem ser inseridos em cada linha do script do banco de dados, com um nome onde deve ser em String, tipo de dados que irá ser acrescentado através de um enum onde já estão inseridos quais tipos de variáveis o MySQL aceita, a quantidade de caracteres que esse tipo irá receber e através de booleanos se ele e auto incrementado, uma chave primaria, não nulo e único.

```
public ScriptBD(String nome, Tipos tipo, int quantidade, boolean autoIncrement,boolean primaryKey, boolean notNull, boolean unique) {
    this.nome = nome;
    this.tipo = tipo;
    this.quantidade = quantidade;
    this.autoIncrement = autoIncrement;
    this.primaryKey = primaryKey;
    this.notNull = notNull;
    this.unique = unique;
}
```

Imagem 3 (Fonte autor)

O método CriarScript() irá montar a linha da tabela fazendo as verificações do construtor, retornando uma String.

```
public String CriarScript() {
    String script = getNome() + " " + getTipo() +"(" + getQuantidade() + ")";
    if (getNotNull() == true) {
        script = script + " NOT NULL";
    }else {

    }
    if (getAutoIncrement() == true) {
        script = script + " AUTO_INCREMENT";
        if (getPrimaryKey() == true) {
            script = script + " PRIMARY KEY";
        }
    }else {

    }
    return script;
}
```

Imagem 4 (Fonte autor)

Na classe Tabela há um método onde e criado a tabela sendo necessário informar o nome da tabela retornando uma String, neste método ele chama outro método chamado script() onde ele busca a lista da classe ScriptBD onde ela faz algumas verificações e monta a tabela.

```
public String criarTabela() {
    String tabela = "CREATE TABLE "+  getNomeTabela() + "(" + "\n";
    tabela +=  script();
    ScriptBD scpt = new ScriptBD();

    for (int x = 0; x < lista.size(); x++) {
        scpt = lista.get(x);
        if (scpt.getUnique() ==  true) {
            tabela += " UNIQUE(" + scpt.getNome() + ")";
        }
    }
    return tabela + ")ENGINE = innodb;";


}
```

Imagem 5 (Fonte autor)

```
public String script() {
    String st = "";
    boolean A = false;
    for (int x = 0; x < lista.size(); x ++) {
        if (x == lista.size() - 1) {
            ScriptBD scpt = new ScriptBD();
            scpt = lista.get(x);
            if (scpt.getUnique() == true) {
                A = true;
            }
            if (A == true) {
                st += scpt.CriarScript() + "," + "\n";
            }else {
                st += scpt.CriarScript() + "\n";
            }

        }else {
            ScriptBD scpt = new ScriptBD();
            scpt = lista.get(x);
            st += scpt.CriarScript() + "," + "\n";
            if (scpt.getUnique() == true) {
                A = true;
            }
        }
    }
    return st;
}
```

Imagem 6 (Fonte autor)

Existe outro método que pode ser usado para a criação de chaves estrangeiras, mas neste caso as chaves estrangeiras não podem ser nulas, pois na hora de inserção no banco de dados ele apresenta erro caso as chaves estrangeiras forem nulas. O método utiliza o formato parecido com o método CriarTabela(), porém há a necessidade de informar qual a chave estrangeira e qual a tabela que irá ser referenciada.

```
public String criarTabelaForeignKey(String foreignKey, String nomeTabela, String referencia) {
    String tabela = "CREATE TABLE "+  getNomeTabela() + "(" + "\n";
    tabela += script();
    ScriptBD scpt = new ScriptBD();
    int A = 0;
    for (int x = 0; x < lista.size(); x++) {
        scpt = lista.get(x);
        if (scpt.getUnique() ==  true) {
            tabela += "CONSTRAINT id_" + foreignKey + " FOREIGN KEY (" + foreignKey + ")\n";
            tabela += "REFERENCES " + nomeTabela + "(" + referencia + "),\n";
            tabela += " UNIQUE(" + scpt.getNome() + ")";
            A ++;
        }
    }
    if (A <= 0) {
        tabela += ",CONSTRAINT id_" + foreignKey + " FOREIGN KEY (" + foreignKey + ")\n";
        tabela += "REFERENCES " + nomeTabela + "(" + referencia + ")\n";
    }
    return tabela + ")ENGINE = innodb;";
}
```

Imagem 7 (Fonte autor)

Na classe ConnectarBD podemos de forma simplificada fazer a conexão do banco informando apenas através do construtor o classForName, url, usuário e senha do banco de dados, após inserir esses dados apenas chamar o método conectar() que ele ira fazer a conexão com o banco de dados.

```
public ConectarBD(String classForName, String url, String usuario, String senha) {
    this.classForName = classForName;
    this.url = url;
    this.usuario = usuario;
    this.senha = senha;
}
```

Imagem 8 (Fonte autor)

```
public Connection conectar() {
    try {
        Class.forName(getClassForName());
        String url = getUrl();
        return DriverManager.getConnection(url,getUsuario(),getSenha());
    } catch (Exception e) {
        System.err.println("Erro: " + e.toString());
        e.printStackTrace();
        return null;
    }
}
```

Imagem 9 (Fonte autor)

Através da interface Persistencia podemos gravar e ler as tabelas criadas para determinado banco de dados, onde e criado uma lista de tabelas através da classe ListAbstratTabela e então gravadas em Json ou Csv.

```java
public interface Persistencia {
    /**
     * Interface usada para gravar os Scripts criados
     * @param lista
     * @return
     * @throws Exception
     */
    public boolean gravar(List<AbstratTabela> lista)  throws Exception;
    /**
     * Interface usada para ler os Scripts criados
     * @param lista
     * @return
     * @throws Exception
     */
    public List<AbstratTabela> ler(List<AbstratTabela> lista) throws Exception;
}
```

Imagem 10 (Fonte autor)

Conclusão

Após a conclusão do framework foram feitos alguns testes através da biblioteca JUnit de criação de script e tabela, onde não foi verificado nenhum erro. Com isso o framework pode ser usado para facilitar a criação de bancos de dados no MySQL criando assim Scripts e caso necessário a gravação desses Scripts para documentação.

Referências

SHVETS, ALEXANDER. **Mergulho nos PADRÕES de PROJETO** e-Book.