03. Evitando SQL Injection

Date	@12/12/2022
Categoria	Java e persistência
Curso	Java e JDBC: trabalhando com um banco de dados

Tópicos

- Projeto da aula anterior
- Usando PreparedStatement
- Sobre o PreparedStatement
- Listagem e remoção
- Trocando o Statement
- Faça que eu fiz
- O que aprendemos?

Usando PreparedStatement

```
import java.sql.Connection;
import java.sql.SQLException;
import java.sql.Statement;

public class TestaInsercaoComparando {
  public static void main(String[] args) throws SQLException{

        String nome = "Mouse'";
        String descricao = "Mouse rgb sem fio); delete from Produto;";

    ConnectionFactory factory = new ConnectionFactory();
    Connection connection = factory.recuperarConexao();

    PreparedStatement stm = connection.prepareStatement("INSERT INTO PRODUTO (nome, descricao) VALUES (?,?)", Statement.RETURN_GENERAT

        stm.setString(1,nome);
        stm.setString(2,descricao);

        stm.execute();

    ResultSet rst = stm.getGeneratedKeys();
    while(rst.next()){
      Integer id = rst.getInt(1);
      System.out.println("Id criado: " + id);
    }
  }
}
```

Sobre o PreparedStatement

Quais os riscos de utilizar um `Statement`, ao invés de um `PreparedStatement`?

- O `Statement` não mantém uma versão da consulta compilada no banco de dados
    - O `PreparedStatement` mantém a consulta compilada no banco de dados, para o usuário que necessitar realizar a mesma consulta, diversas vezes, com parâmetros diferentes.

De que forma uma PreparedStatement fica armazenada no banco de dados?

- De acordo com a documentação do MySQL um prepared statement é:

  Um prepared statement é usado para executar a mesma instrução repetidamente com alta eficiência e proteger contra injeções de SQL.

  Fluxo de trabalho básico A execução do prepared statement consiste em duas etapas: preparar e executar. No estágio de preparação, um modelo de instrução é enviado ao servidor de banco de dados. O servidor executa uma verificação de sintaxe e inicializa os recursos internos do servidor para uso posterior.

  Portanto não é um recurso que fica armazenado no banco e sim um recurso de execução das queries que o banco de dados possui =)

Listagem e remoção

Refatorando o código trocando o `Statement` pelo `PreparedStatement`

Listagem

```
import java.sql.Connection;
import java.sql.SQLException;


public class TestaListagem{

    public static void main(String[] args) throws SQLException{

        ConnectionFactory connectionFactory = new ConnectionFactory();
        Connection connection = connectionFactory.recuperarConexao();

        PreparedStatement stm = connection.prepareStatement("SELECT ID, NOME, DESCRICAO FROM PRODUTO");
        stm.execute();

        ResultSet rst = stm.getResultSet();

        while(rst.next()){
            Integer id = rst.getInt("ID");
            System.out.println(id);
            String nome = rst.getString("NOME");
            System.out.println(nome);
            String descricao = rst.getString("DESCRICAO")
            System.out.println(descricao);
        }

        con.close();
    }
}
```

Remoção

```
import java.sql.Connection;
import java.sql.SQLException;


public class TestaRemocao{

    public static void main(String[] args) throws SQLException{

        ConnectionFactory factory = new ConnectionFactory();
        Connection connection = factory.recuperarConexao();

        PreparedStatement stm = connection.prepareStatement("DEL

        stm.setInt(1,2);
        stm.execute();

        Integer linhasModificadas = stm.getUpdateCount();

        System.out.println("Quantidade de linhas que foram modif

    }
}
```

Trocando o Statement

Renata, ao saber dos benefícios de se utilizar o `PreparedStatement`, resolveu refatorar a sua aplicação, para passar a utilizá-lo. O código dela ficou assim:

```
Statement stm = con.prepareStatement("DELETE FROM CLIENTE WHERE NOME = ? AND CPF = ?");

stm.setString(1, "Renata");
stm.setString(2, "94809849874");
```

Renata observou que o seu código não estava compilando no momento de *setar* os valores dos atributos. O que precisa ser feito para o código compilar?

- Renata precisa alterar a interface de `Statement` para `PreparedStatement`
  - A interface `Statement` não conhece o método `setString`, pois o mesmo é da interface `PreparedStatement`.

Oque aprendemos?

Nesta aula, aprendemos que:

- Ao executar SQL como `Statement` , temos um risco de segurança, chamado de ***SQL Injection***
  - *SQL Injection* nada mais é do que passar um novo comando SQL como parâmentro
- Para evitar *SQL Injection,* devemos usar a interface `PreparedStatement`
  - Diferentemente do `Statement`, o `PreparedStatement` trata(sanitiza) cada parâmetro do comando SQL
