Java – JDBC

Fabiano Baldo

Tópicos

- Introdução a JDBC
- Modo de utilização do JDBC
- *Biblioteca Java.sql*

JDBC

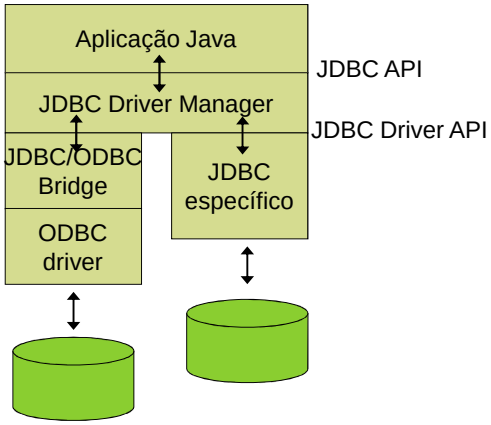
- JDBC significa *Java Database Connectivity*.
- Sua primeira versão foi lançada em 1996.
- Ele auxilia programadores a se conectar a banco de dados para fazer consultas e atualizações, usando SQL.
- Sua maior vantagem é que programas desenvolvidos utilizando Java e JDBC são independentes de plataforma.

JDBC

- O intuito do JDBC é fornecer uma única API Java para acesso a banco de dados.
- Esta API dá acesso ao JDBC Driver Manager.
- O Driver Manager permite que drivers de acesso a BD sejam conectados a ele.
- Ou seja, o vendedor do BD pode prover seus próprios drivers e plugá-lo no Driver Manager.

JDBC

- O JDBC tem duas interfaces:
  - Programadores de aplicação usam a JDBC API,
  - Vendedores de BD usam a JDBC Driver API.



Passos para a utilização do JDBC

- Você precisa ter um BD compatível com o JDBC. Ex.: IBM DB2, Microsoft SQL Server, MySQL, Oracle, and PostgreSQL.
- Você precisa criar um BD para armazenar os dados da sua aplicação.
- Você precisa instalar o cliente do BD que você está utilizando.
- Você precisa ter acesso a biblioteca JDBC do BD utilizado.

Como conectar o BD usando JDBC

- Primeiramente você precisa incluir em sua aplicação a biblioteca de drivers JDBC para o BD utilizado.
- Na maioria dos casos os drivers JDBC são providos pelos próprios fabricantes do BD.
- O estabelecimento da conexão com o BD é feito em dois passos:
  1. Carregamento em memória dos driver JDBC
  2. Estabelecimento da conexão usando o DriverManager;

Exemplo

```
public class JdbcTest1 {
    public static void main (String[] args) {
        try {

            // Step 1: Carrega o driver JDBC.
            Class.forName("com.imaginary.sql.msql.MsqlDriver");

            // Step 2: Estabelece a conexão com o BD.
            String url = "jdbc:msql://www.myserver.com:1114/my_db";
            Connection conn =
                DriverManager.getConnection(url,"user1","password");

        } catch (ClassNotFoundException c) {
            System.err.println(“Driver não encontrado");
        }
         catch (SQLException e) {
            System.err.println("Não foi possível conectar");
        }
    }
}
```

DriverManager – getConnection()

- Assinatura do método:

```
public static Connection getConnection(String url, String user, String password)
```

- Parâmentros:
  - url: um caminho que identifique o BD, no formato "jdbc:*subprotocolo*:*subnome*"
  - user: nome de um usuário válido para o BD
  - password: selha do usuário informado
- getConnection é um método de classe, da classe DriverManager

Java.sql

- O java.sql é uma API que prove acesso e processamento aos dados armazenados em fontes tais como BD relacional, usando a linguagem de programação Java.
- Principais classes da API:
  - **Connection**: Mantém a conexão com um BD específico.
  - **Statement**: É o objeto que executa uma declaração SQL.
  - **PreparedStatement**: Objeto que representa uma declaração SQL pré-compilada.
  - **ResultSet**: Objeto que contém o resultado de uma consulta SQL.
  - **SQLException**: Objeto mais geral que trata as exceções SQL.

Exemplo de consulta

```
try {
    // Cria um ResultSet contendo todos os dados de my_table
    Statement stmt = connection.createStatement();
    ResultSet rs = stmt.executeQuery("SELECT * FROM my_table");

    // Percorre todo o ResultSet
    while (rs.next()) {
        //Pega os dados de um campo de um registro usando o índice da coluna
        String s = rs.getString(1);

       //Pega os dados de um campo de um registro usando o nome da coluna
        s = rs.getString(“nome_coluna");
    }
} catch (SQLException e) {
     ...
}
```


Exemplo de Inserção

- Usando statement

```
try {
    Statement stmt = connection.createStatement();

    // Prepara um statement para inserir um registro
    String sql = "INSERT INTO my_table (nome_campo) VALUES('valor_campo')";

    // Executa o statement de inserção
    stmt.executeUpdate(sql);
} catch (SQLException e) {
     ...
}
```

Exemplo de Inserção

- Usando preparedStatement

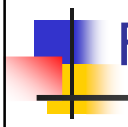
```
try {
    // Prepara um statement para inserir um registro
    String sql = "INSERT INTO my_table (nome_campo) VALUES(?)";
    PreparedStatement pstmt = connection.prepareStatement(sql);

    // Insere 10 registros
    for (int i=0; i<10; i++) {
        // Define os valores
        pstmt.setString(1, “linha "+i);

        // Insere um registro
        pstmt.executeUpdate();
    }
} catch (SQLException e) {
    ...
}
```


Exemplo de Atualização

```
try {
    Statement stmt = connection.createStatement();

    // Prepara um statement para atualizar um registro
    String sql = "UPDATE my_table SET nome_campo='novo_valor' WHERE
             campo_chave = 'valor'";

    // Executa o statement de atualização
    int updateCont = stmt.executeUpdate(sql);
    // updateCont contém o número de registros alterados
} catch (SQLException e) {
    ...
}
```

Exemplo de Exclusão

```
    try {
        // Cria um statement
       Statement stmt = connection.createStatement();

       // Prepara um statement para excluir um registro
       String sql = "DELETE FROM my_table WHERE campo_chave='valor'";

       // Executa o statement de exclução
       int deleteCont = stmt.executeUpdate(sql);
       // deleteCont contém o número de registros excluídos
    } catch (SQLException e) {
        ...
   }
```

Fim