Estratégia Concursos

Aula Extra

Banco do Brasil (Escriturário - Agente de Tecnologia) Desenvolvimento de Software - 2023 (Pós-Edital)

Autor:
Raphael Henrique Lacerda, Paolla Ramos e Silva

16 de Janeiro de 2023

Índice

JDBC

De acordo com a documentação oficial: trata-se de uma API que fornece acesso universal a dados a partir da linguagem de programação Java. É possível acessar praticamente qualquer fonte de dados, desde bancos de dados relacionais a planilhas ou arquivos simples. O JDBC também fornece uma base comum sobre a quais ferramentas e interfaces alternativas podem ser construídas.

Ele estabelece conexões com um banco de dados, envia comandos SQL e processa os resultados. Aliás, com a utilização de JDBC, torna-se fácil enviar um comando SQL para diferentes bancos de dados relacionais. Em outras palavras, não é necessário escrever um programa para acessar um banco de dados Oracle, outro para acessar um banco de dados SQL Server, e assim por diante.

Essa tecnologia permite invocar comandos SQL a partir de métodos em classes Java. Ela fornece uma API do tipo call-level para acesso a bancos de dados baseado em SQL. Os componentes dessa API estão localizados nos pacotes java.sql e javax.sql. Para conseguir acesso a diversos bancos de dados diferentes de maneira uniforme e padronizada, utilizam-se Drivers.

Como assim, professor? O Driver faz a interface entre a Aplicação Web e o SGBD! Para eu me conectar a um banco de dados individual, eu preciso ter os drivers para aquele banco de dados específico. O JDBC permite a conexão com praticamente qualquer SGBD (Oracle, DB2, SQL Server, MySQL, PostgreSQL, etc). Existem, atualmente, quatro tipos de Drivers:

- JDBC-ODBC Bridge: É o tipo mais simples. Utiliza ODBC para conectar-se com o banco de dados, convertendo métodos JDBC em chamadas às funções do ODBC. Esta ponte é normalmente usada quando não há um driver puro-Java para determinado banco de dados, pois seu uso é desencorajado devido à dependência de plataforma.
- Native API: converte chamadas JDBC na API do cliente em chamadas para o SGBD Oracle, Sybase, Informix, etc. São escritos parcialmente em Java e parcialmente em código nativo. Requer que algum código binário específico do SO seja carregado em cada máquina cliente e usa uma biblioteca nativa específica no cliente para a fonte de dados para que eles se conectem.
- Network Protocol: traduz chamadas JDBC em um protocolo de rede independente do SGBD, que é então convertido para um protocolo específico do SGBD por um middleware. Em geral, esta é a alternativa JDBC mais flexível de todas, em que fornecedores disponibilizam produtos adequados para uso na internet.
- Database Protocol: permite chamada direta da máquina do cliente para o Servidor SGBD. Converte as chamadas JDBC diretamente no protocolo do banco de dados. Implementado em Java, normalmente é independente de plataforma e escrito pelos próprios desenvolvedores. É o tipo mais recomendado para ser usado.

Open Database Connectivity (ODBC) é um padrão aberto desenvolvido para que linguagens de programação ou sistemas operacionais se comuniquem com bancos de dados de maneira independente de plataforma. O JDBC utiliza um driver específico para banco de dados; o ODBC utiliza sempre o mesmo Driver e depois configuram-se as propriedades do sistema para acessar determinado banco.

Dado que foi utilizado o driver adequado para acesso ao banco de dados, pode-se utilizar a classe Connection para estabelecer a conexão, de fato, com o banco. Essa classe se encontra no pacote java.sql – que contém uma biblioteca para acesso e processamento de dados em uma fonte de dados. As classes, métodos e interfaces dele mais importantes são:

- Driver: interface pública abstrata que todo driver deve implementar para executar instruções SQL.
- DriverManager: classe que contém o registro de cada driver, oferecendo métodos estáticos para gerenciá-los.
- Connection: interface que representa uma conexão com o banco de dados – possui diversos métodos importantes.
- Statement: interface pública abstrata que é utilizada para executar instruções SQL estáticas e obter os resultados de sua execução.
- ResultSet: essa interface é uma tabela de dados que representa o resultado de uma instrução SQL em um banco de dados.
- PreparedStatement: interface pública abstrata utilizada para estender a interface Statement e criar um objeto que represente uma instrução SQL.
- CallableStatement: interface que permite executar funções ou procedimentos armazenados no banco.

```
try

{

   Class.forName("sun.jdbc.odbc.JdbcOdbcDriver").getInstance();



   Connection con =

   DriverManager.getConnection("jdbc:odbc:meusCdsDb","conta","senha");

}

catch(SQLException e)
```

```
{   // Se o carregador não localizar o driver do banco para conexão, lança a

    // exceção java.lang.ClassNotFoundExeption

    e.printStackTrace();

}
```

Para acessar um banco de dados, o primeiro passo é estabelecer uma **conexão!** É possível fazer isso em duas etapas: primeiro, carrega-se o driver do banco de dados. Uma vez carregado, o driver se registra para o DriverManager e está disponível para a aplicação. Então usa-se o DriverManager para abrir uma conexão com o banco de dados. A interface Connection designa um objeto con para receber a conexão.

```
Statement stm = con.createStatement();

String SQL = "Select coluna1, coluna2, coluna3 from TabelaX";
```

Estabelecida a conexão, podemos executar comandos **SQL n**o banco de dados. Para realizar uma operação, podemos usar **três interfaces.** A primeira delas é a interface Statement, que permite a execução dos comandos fundamentais de SQL (SELECT, INSERT, UPDATE ou DELETE). A interface PreparedStatement nos permite usufruir de SQL armazenado ou pré-compilado no banco.

```
ResultSet rs = stm.executeQuery(SQL);
```

A terceira interface é **CallableStatement,** e permite executar procedimentos e funções armazenados no banco (quando o banco suportar este recurso). A interface ResultSet permite colher os resultados da execução de nossa query no banco de dados. Esta interface apresenta uma série de métodos para prover o acesso aos dados. Depois é só encerrar o acesso, liberando recursos.

```
finally

{

  try

  {

    con.close();

  }

  catch(SQLException onConClose)
```

```
  {
     System.out.println("ERRO NO FECHAMENTO DA CONEXÃO");
     onConClose.printStackTrace();
  }
}
```

Podemos fazer isso fechando o Statement, que libera os recursos associados à execução desta consulta, mas deixa a conexão aberta para a execução de uma próxima consulta; ou fechando diretamente a conexão, que encerra a comunicação com o banco de dados. Para termos certeza de que vamos encerrar esta conexão mesmo que uma exceção ocorra, reservamos o fechamento para a cláusula finally().

QUESTÕES COMENTADAS - JDBC - MULTIBANCAS

1. (FCC – 2006 – BACEN – Analista de Sistemas) O estabelecimento de conexão entre um aplicativo Java e um banco de dados, para processar instruções SQL de consulta e atualização, é possibilitado por meio do padrão aberto, desenvolvido pela Microsoft, denominado:

   a) API.

   b) ODBC.

   c) SGDB.

   d) JDBC.

   e) OLE.

Comentários:

Open Database Connectivity (ODBC) é um padrão aberto desenvolvido para que linguagens de programação ou sistemas operacionais se comuniquem com bancos de dados de maneira independente de plataforma. O JDBC utiliza um driver específico para banco de dados; o ODBC utiliza sempre o mesmo Driver e depois configuram-se as propriedades do sistema para acessar determinado banco.

Conforme vimos em aula, trata-se do ODBC. Gabarito: B

2. (CESPE – 2008 – IPEA – Analista de Sistemas) O JDBC é usado, entre outras coisas, para acesso a bancos de dados sem SQL, por meio de Java.

Comentários:

De acordo com a documentação oficial: trata-se de uma API que fornece acesso universal a dados a partir da linguagem de programação Java. É possível acessar praticamente qualquer fonte de dados, desde bancos de dados relacionais a planilhas ou arquivos simples. O JDBC também fornece uma base comum sobre a quais ferramentas e interfaces alternativas podem ser construídas.

Conforme vimos em aula, ele pode acessar praticamente qualquer coisa. Gabarito: C

3. (FCC - 2008 - TRT - 2ª REGIÃO (SP) - Analista Judiciário - Tecnologia da Informação) A utilização de JDBC, em um programa Java, inicia com a indicação do pacote que contém a JDBC API pela declaração:

   a) import java.awt.*;

   b) import java.util.*;

c) import java.sql.*;

d) import java.swing.*;

e) import java.jdbc.*;

Comentários:

Essa tecnologia permite invocar comandos SQL a partir de métodos em classes Java. Ela fornece uma API do tipo call-level para acesso a bancos de dados baseado em SQL. Os componentes dessa API estão localizados no pacote java.sql e javax.sql. Para conseguir acesso a diversos bancos de dados diferentes de maneira uniforme padronizada, utilizam-se Drivers.

Conforme vimos em aula, trata-se do java.sql. Gabarito: C

4. (ESAF - 2009 - ANA - Analista Administrativo - Tecnologia da Informação - Desenvolvimento) Em uma aplicação Java, se o carregador de classes não conseguir localizar a classe do driver de banco de dados para uma conexão JDBC, é lançada a exceção

a) java.lang.ClassNotFoundException.

b) java.io.FileNotFoundException.

c) java.lang.SecurityException.

d) java.io.IOException.

e) java.util.InputMismatchException.

Comentários:

```
try
{
   Class.forName("sun.jdbc.odbc.JdbcOdbcDriver").getInstance();

   Connection con =
   DriverManager.getConnection("jdbc:odbc:meusCdsDb","conta","senha");
}
catch(SQLException e)
{  // Se o carregador não localizar o driver do banco para conexão, lança a
   // exceção java.lang.ClassNotFoundExeption
   e.printStackTrace();
```

}

Conforme vimos em aula, trata-se do java.lang.ClassNotFoundException. Gabarito: A

5. (FGV - 2009 - MEC - Analista de Sistemas - Especialista) Observe o código abaixo, que se refere á implementação de Java no acesso a Banco de Dados em JDBC.

```
package wlss.jdbcTT;
   import java.sql.Connection;
   import java.sql.DriverManager;
   import java.sql.ResultSet;
   import java.sql.SQLException;
   import java.sql.Statement;

   class T1 {
        public static void main(String args[]) {
          Connection con = null;
          try {

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver").getInstance(
);
            con =
DriverManager.getConnection("jdbc:odbc:meusCdsDb",
             " conta ", " senha ");
            Statement stm = con.createStatement();
            String SQL = " Select titulo, autor,
total_faixas from MeusCDs ";
            ResultSet rs = stm.executeQuery(SQL);
            while (rs.next()) {
              String tit = rs.getString("titulo");
           String aut = rs.getString("autor");
           int totalFaixas = rs.getInt("total_faixas");
           System.out.println(48 + " Titulo: " + tit + "
Autor: " + aut
                + " 49: Tot. Faixas: " + totalFaixas);
         }

       } catch (SQLException e) {
       } finally {
         try {
           con.close();
         } catch (SQLException onConClose) {
           System.out.println(" Houve erro no fechamento
da conexão ");
           onConClose.printStackTrace();
         }
       }
     }

   }
```

Analise a instrução a seguir: con = DriverManager.getConnection("jdbc:odbc:meusCdsDb", " conta ", " senha "); Assinale a alternativa que indique corretamente o significado da instrução acima.

a) Abrir as tabelas do Banco de Dados.

b) Fechar a conexão com o Banco de Dados.

c) Liberar a conexão com o Banco de Dados.

d) Estabelecer a conexão com o Banco de Dados.

e) Criar uma variável para logon do Banco de Dados.

Comentários:

Para acessar um banco de dados, o primeiro passo é estabelecer uma conexão! É possível fazer isso em duas etapas: primeiro, carrega-se o driver do banco de dados. Uma vez carregado, o driver se registra para o DriverManager e está disponível para a aplicação. Então usa-se o DriverManager para abrir uma conexão com o banco de dados. A interface Connection designa um objeto con para receber a conexão.

Conforme vimos em aula, trata-se do estabelecimento de uma conexão. Gabarito: D

6. (FCC - 2012 - MPE-AP - Técnico Ministerial - Informática) Analise as linhas a seguir presentes em um programa Java que não apresenta erros.

```
a = DriverManager.getConnection("jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};DBQ=E:\\bd.mdb", "", "");
b = a.createStatement( );
c = b.executeQuery("select * from cliente where id = "+ valor +"");
```

Considere que os objetos a, b e c são de interfaces contidas no pacote java.sql. Pode-se concluir que esses objetos são, respectivamente, das interfaces

a) Connection, SessionStatement e Result.

b) DriverManager, PreparedStatement e RecordSet.

c) ConnectionStatement, PreparedStatement e RecordSet.

d) Connection, Statement e ResultSet.

e) DaoConnection, Statement e ResultSet.

Comentários:

Conforme vimos em aula: a = Connection; b = Statement; c = ResultSet. Gabarito: D

7. (CESPE - 2012 - TJ-RO - Analista Judiciário - Análise de Sistemas – Desenvolvimento – B) JDBC, uma biblioteca vinculada a API da arquitetura JEE, define como um cliente pode acessar bancos de dados OO exclusivamente.

Comentários:

De acordo com a documentação oficial: trata-se de uma API que fornece acesso universal a dados a partir da linguagem de programação Java. É possível acessar praticamente qualquer fonte de

dados, desde bancos de dados relacionais a planilhas ou arquivos simples. O JDBC também fornece uma base comum sobre a quais ferramentas e interfaces alternativas podem ser construídas.

Conforme vimos em aula, ele pode acessar bancos de dados de quaisquer paradigmas (OO, Relacional, etc). Gabarito: E

8. (COPEVE-UFAL - 2012 - ALGÁS - Analista de Tecnologia da Informação – I) Na arquitetura do JDBC, a diferença entre os tipos Statement e PreparedStatement é o fato do PreparedStatement manter os dados criptografados durante o tráfego entre o cliente e o servidor do SGBD.

Comentários:

Não existe isso! De fato, o PreparedStatement é mais seguro, visto que ajuda a impedir SQL Injection. No entanto, ele não mantém dados criptografados. Gabarito: E

9. (FGV - 2009 - MEC - Administrador de Banco de Dados) O pacote "java.sql" da API Java consiste de um conjunto de classes e interfaces que permitem embutir código SQL em métodos Java para por meio de drivers JDBC acessar diversos SGBDs. As alternativas a seguir apresentam interfaces do pacote "java.sql", à exceção de uma. Assinale-a.

a) SQLData

b) ResultSet

c) Statement

d) DriverManager

e) Connection

Comentários:

Pegadinha de banca que não sabe avaliar conhecimento! DriverManager não é uma interface, é uma classe! Gabarito: D

10. (CESPE - 2008 - HEMOBRÁS - Analista de Gestão Corporativa - Administrador de Banco de Dados) O JDBC fornece a classe CallableStatementSQL, que permite que procedimentos ou funções SQL armazenados sejam chamados.

Comentários:

A terceira interface é CallableStatement, e permite executar procedimentos e funções armazenados no banco (quando o banco suportar este recurso). A interface ResultSet permite

colher os resultados da execução de nossa query no banco de dados. Esta interface apresenta uma série de métodos para prover o acesso aos dados. Depois é só encerrar o acesso, liberando recursos.

Conforme vimos em aula, o nome correto é CallableStatement. Gabarito: E

LISTA DE QUESTÕES - JDBC - MULTIBANCAS

1. (FCC – 2006 – BACEN – Analista de Sistemas) O estabelecimento de conexão entre um aplicativo Java e um banco de dados, para processar instruções SQL de consulta e atualização, é possibilitado por meio do padrão aberto, desenvolvido pela Microsoft, denominado:

   a) API.

   b) ODBC.

   c) SGDB.

   d) JDBC.

   e) OLE.

2. (CESPE – 2008 – IPEA – Analista de Sistemas) O JDBC é usado, entre outras coisas, para acesso a bancos de dados sem SQL, por meio de Java.

3. (FCC - 2008 - TRT - 2ª REGIÃO (SP) - Analista Judiciário - Tecnologia da Informação) A utilização de JDBC, em um programa Java, inicia com a indicação do pacote que contém a JDBC API pela declaração:

   a) import java.awt.*;

   b) import java.util.*;

   c) import java.sql.*;

   d) import java.swing.*;

   e) import java.jdbc.*;

4. (ESAF - 2009 - ANA - Analista Administrativo - Tecnologia da Informação - Desenvolvimento) Em uma aplicação Java, se o carregador de classes não conseguir localizar a classe do driver de banco de dados para uma conexão JDBC, é lançada a exceção

   a) java.lang.ClassNotFoundException.

   b) java.io.FileNotFoundException.

   c) java.lang.SecurityException.

   d) java.io.IOException.

   e) java.util.InputMismatchException.

5. (FGV - 2009 - MEC - Analista de Sistemas - Especialista) Observe o código abaixo, que se refere á implementação de Java no acesso a Banco de Dados em JDBC.

```
package wlss.jdbcTT;
   import java.sql.Connection;
   import java.sql.DriverManager;
   import java.sql.ResultSet;
   import java.sql.SQLException;
   import java.sql.Statement;

   class T1 {
        public static void main(String args[]) {
          Connection con = null;
          try {

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver").getInstance(
);
            con =
DriverManager.getConnection("jdbc:odbc:meusCdsDb",
             " conta ", " senha ");
            Statement stm = con.createStatement();
            String SQL = " Select titulo, autor,
total_faixas from MeusCDs ";
            ResultSet rs = stm.executeQuery(SQL);
            while (rs.next()) {
              String tit = rs.getString("titulo");
           String aut = rs.getString("autor");
           int totalFaixas = rs.getInt("total_faixas");
           System.out.println(48 + " Titulo: " + tit + "
Autor: " + aut
                + " 49: Tot. Faixas: " + totalFaixas);
         }

       } catch (SQLException e) {
       } finally {
         try {
           con.close();
         } catch (SQLException onConClose) {
           System.out.println(" Houve erro no fechamento
da conexão ");
           onConClose.printStackTrace();
         }
       }
     }

   }
```

Analise a instrução a seguir: con = DriverManager.getConnection("jdbc:odbc:meusCdsDb", " conta ", " senha "); Assinale a alternativa que indique corretamente o significado da instrução acima.

a) Abrir as tabelas do Banco de Dados.

b) Fechar a conexão com o Banco de Dados.

c) Liberar a conexão com o Banco de Dados.

d) Estabelecer a conexão com o Banco de Dados.

e) Criar uma variável para logon do Banco de Dados.

6. (FCC - 2012 - MPE-AP - Técnico Ministerial - Informática) Analise as linhas a seguir presentes em um programa Java que não apresenta erros.

a = DriverManager.getConnection("jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};DBQ=E:\\bd.mdb", "", "");

```
b = a.createStatement( );
c = b.executeQuery("select * from cliente where id = "+ valor +"");
```

Considere que os objetos a, b e c são de interfaces contidas no pacote java.sql. Pode-se concluir que esses objetos são, respectivamente, das interfaces

a) Connection, SessionStatement e Result.

b) DriverManager, PreparedStatement e RecordSet.

c) ConnectionStatement, PreparedStatement e RecordSet.

d) Connection, Statement e ResultSet.

e) DaoConnection, Statement e ResultSet.

7. (CESPE - 2012 - TJ-RO - Analista Judiciário - Análise de Sistemas – Desenvolvimento – B) JDBC, uma biblioteca vinculada a API da arquitetura JEE, define como um cliente pode acessar bancos de dados OO exclusivamente.

8. (COPEVE-UFAL - 2012 - ALGÁS - Analista de Tecnologia da Informação – I) Na arquitetura do JDBC, a diferença entre os tipos Statement e PreparedStatement é o fato do PreparedStatement manter os dados criptografados durante o tráfego entre o cliente e o servidor do SGBD.

9. (FGV - 2009 - MEC - Administrador de Banco de Dados) O pacote "java.sql" da API Java consiste de um conjunto de classes e interfaces que permitem embutir código SQL em métodos Java para por meio de drivers JDBC acessar diversos SGBDs. As alternativas a seguir apresentam interfaces do pacote "java.sql", à exceção de uma. Assinale-a.

a) SQLData

b) ResultSet

c) Statement

d) DriverManager

e) Connection

10. (CESPE - 2008 - HEMOBRÁS - Analista de Gestão Corporativa - Administrador de Banco de Dados) O JDBC fornece a classe CallableStatementSQL, que permite que procedimentos ou funções SQL armazenados sejam chamados.

GABARITO

1. B
2. C
3. C
4. A
5. D
6. D
7. E
8. E
9. D
10. E

ESSA LEI TODO MUNDO CONHECE:
PIRATARIA É CRIME.

Mas é sempre bom revisar o porquê e como você pode ser prejudicado com essa prática.

1. 

Professor investe seu tempo para elaborar os cursos e o site os coloca à venda.

2. 

Pirata divulga ilicitamente (grupos de rateio), utilizando-se do anonimato, nomes falsos ou laranjas (geralmente o pirata se anuncia como formador de "grupos solidários" de rateio que não visam lucro).

3. 

Pirata cria alunos fake praticando falsidade ideológica, comprando cursos do site em nome de pessoas aleatórias (usando nome, CPF, endereço e telefone de terceiros sem autorização).

4. 

Pirata compra, muitas vezes, clonando cartões de crédito (por vezes o sistema anti-fraude não consegue identificar o golpe a tempo).

5. 

Pirata fere os Termos de Uso, adultera as aulas e retira a identificação dos arquivos PDF (justamente porque a atividade é ilegal e ele não quer que seus fakes sejam identificados).

6. 

Pirata revende as aulas protegidas por direitos autorais, praticando concorrência desleal e em flagrante desrespeito à Lei de Direitos Autorais (Lei 9.610/98).

7. 

Concurseiro(a) desinformado participa de rateio, achando que nada disso está acontecendo e esperando se tornar servidor público para exigir o cumprimento das leis.

8. 

O professor que elaborou o curso não ganha nada, o site não recebe nada, e a pessoa que praticou todos os ilícitos anteriores (pirata) fica com o lucro.

Deixando de lado esse mar de sujeira, aproveitamos para agradecer a todos que adquirem os cursos honestamente e permitem que o site continue existindo.