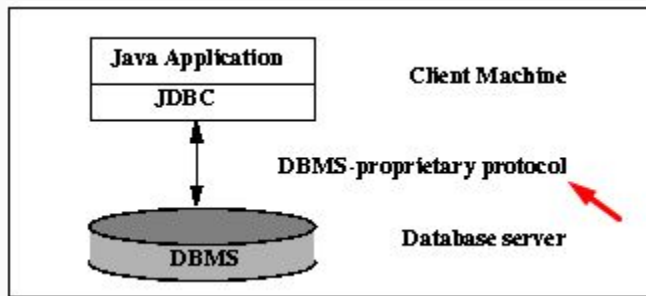


# AfroDev

Java Básico - Aula 9 - Parte 2

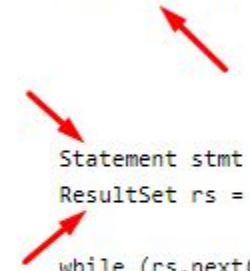
# JDBC - Conceito

- **J**ava **D**ata**B**ase **C**onnectivity
- API Java para tornar “fácil” a interação com o banco de dados
- Fácil? Sim, porém trabalhoso.
- Pode acessar quaisquer dados tabulares (bancos relacionais)



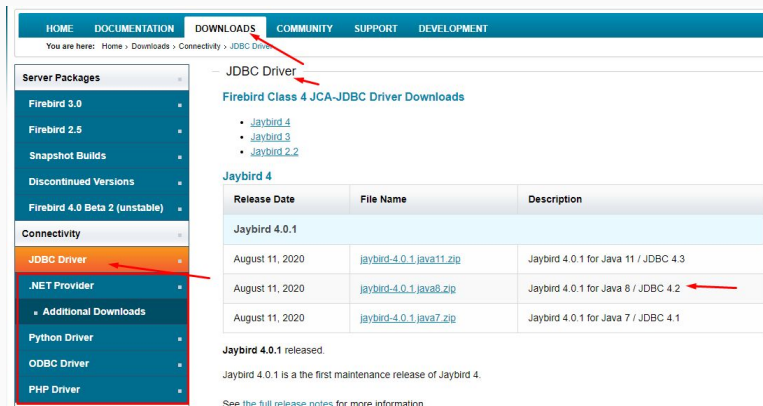
# JDBC - Interface

```
public void connectToAndQueryDatabase(String username, String password) {  
  
    Connection con = DriverManager.getConnection(  
        "jdbc:myDriver:myDatabase",  
        username,  
        password);  
  
    Statement stmt = con.createStatement();  
    ResultSet rs = stmt.executeQuery("SELECT a, b, c FROM Table1");  
  
    while (rs.next()) {  
        int x = rs.getInt("a");  
        String s = rs.getString("b");  
        float f = rs.getFloat("c");  
    }  
}
```



# JDBC - Implementações

- Os elementos destacados são interfaces. Onde estão as implementações?
- Quem implementa essas interfaces?
- Vamos ver um exemplo: Firebird -> <https://firebirdsql.org/en/start/>

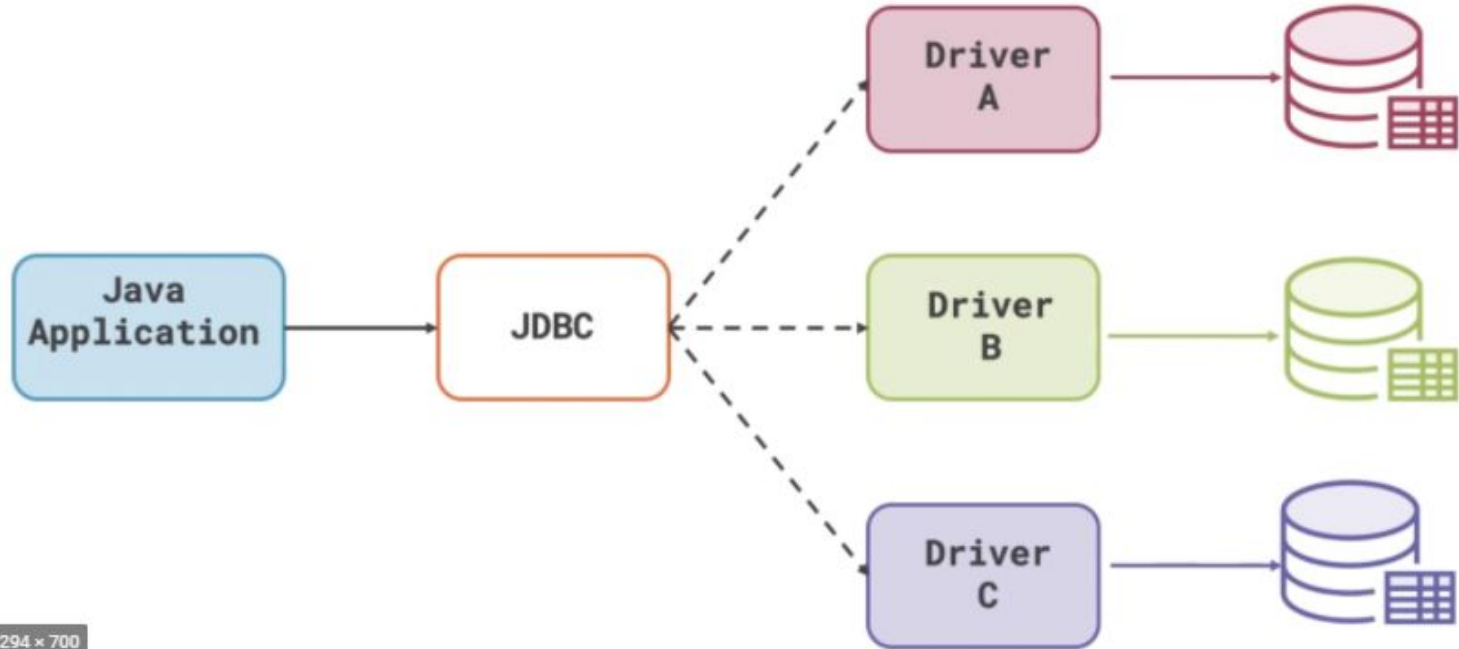


The screenshot shows the Firebird website's 'Downloads' section. The 'JDBC Driver' option is highlighted in the left sidebar. The main content area shows 'Firebird Class 4 JCA-JDBC Driver Downloads' with links to 'Jaybird 4', 'Jaybird 3', and 'Jaybird 2.2'. Below this is a table for 'Jaybird 4' releases.

Release Date	File Name	Description
Jaybird 4.0.1		
August 11, 2020	<a href="#">jaybird-4.0.1.java11.zip</a>	Jaybird 4.0.1 for Java 11 / JDBC 4.3
August 11, 2020	<a href="#">jaybird-4.0.1.java8.zip</a>	Jaybird 4.0.1 for Java 8 / JDBC 4.2
August 11, 2020	<a href="#">jaybird-4.0.1.java7.zip</a>	Jaybird 4.0.1 for Java 7 / JDBC 4.1

Below the table, it states: 'Jaybird 4.0.1 released. Jaybird 4.0.1 is the first maintenance release of Jaybird 4. See the [full release notes](#) for more information.'

# JDBC - Interface + Implementação



# JDBC - Vamos pensar

- Vamos pensar um pouco. Quais vantagens do JDBC?
- Para o *vendor* é vantagem?
- Se eu trocar de vendor, o que eu preciso fazer no meu código?

# JDBC - Exemplo

```
import java.sql.*;

public class UpdateCar {

    public static void UpdateCarNum(int carNo, int empNo)
        throws SQLException {

        Connection con = null;
        PreparedStatement pstmt = null;

        try {
            con = DriverManager.getConnection(
                "jdbc:default:connection");

            pstmt = con.prepareStatement(
                "UPDATE EMPLOYEES " +
                "SET CAR_NUMBER = ? " +
                "WHERE EMPLOYEE_NUMBER = ?");

            pstmt.setInt(1, carNo);
            pstmt.setInt(2, empNo);
            pstmt.executeUpdate();
        }
        finally {
            if (pstmt != null) pstmt.close();
        }
    }
}
```

# JDBC - Considerações

- Trabalhoso no sentido de que é muito verboso
- Perigoso deixar “coisas” abertas
- Código *boilerplate*
- Boa prática: não misturar código SQL com código JAVA. Comumente é utilizado um arquivo texto externa a classe, mas no mesmo JAR
- Utilitários para facilitar a vida:
  - [Spring JdbcTemplate](#) e [Using JdbcTemplate](#)
  - [MyBatis](#) e [SQL Builder](#)



# JDBC - Dicas

- Mesmo JPA usa JDBC
- Mesmo Spring usa JDBC
- JDBC é uma tecnologia base e poderosa, mas o seu nível de abstração é muito baixo
- Ficar de olho na performance
- Visível somente a partir dos teste de integração

# Revisão

- JDBC é a API do JAVA para comunicação com sistemas de banco de dados
- Cada vendor provê a sua própria implementação
- Você deve sempre se basear na API e não em coisas “vendor specific”
- JDBC tem utilitários que facilitam a vida, como MyBatis e Spring JDBC