JDBC (Java Database Connectivity)

JDBC is a Java API for connecting to relational databases, executing SQL queries, and retrieving results. It provides a standard way for Java applications to interact with various databases.

Key Components

1. DriverManager: Manages JDBC drivers for different databases.

2. Connection: Represents a connection to a database.

3. Statement: Used to execute SQL queries.

4. ResultSet: Holds the results of a query.

How JDBC Works

1. Load JDBC driver: Load the JDBC driver for the specific database.

2. Establish connection: Use DriverManager to establish a connection to the database.

3. Create statement: Create a Statement object to execute SQL queries.

4. Execute query: Execute a SQL query using the Statement object.

5. Process results: Process the results of the query using a ResultSet object.

Example

import java.sql.\*;

public class JdbcExample {

public static void main(String[] args) {

try {

// Load JDBC driver

Class.forName("com.mysql.cj.jdbc.Driver");

// Establish connection

Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/mydb", "username", "password");

// Create statement

Statement stmt = conn.createStatement();

// Execute query

ResultSet rs = stmt.executeQuery("SELECT FROM customers");

// Process results

while (rs.next()) {

System.out.println(rs.getString("name") + "," + rs.getInt("age"));

}

// Close resources

rs.close();

stmt.close();

conn.close();

} catch (SQLException e) {

System.out.println("Error: " + e.getMessage());

} catch (ClassNotFoundException e) {

System.out.println("Driver not found: " + e.getMessage());

}

}

}

executeQuery for select

ExecuteUpdate for insert update and delete

Execute for ddl like create

Benefits

1. Database independence: JDBC provides a standard way to interact with different databases.

2. SQL support: JDBC supports SQL queries, allowing for complex database operations.

3. Java integration: JDBC is designed for Java applications, making it easy to integrate database functionality

Common Use Cases

1. Web applications: JDBC is often used in web applications to interact with databases.

2. Enterprise applications: JDBC is used in enterprise applications to access and manipulate large datasets.

3. Data analysis: JDBC can be used to retrieve data from databases for analysis and reporting.