JDBC

Jar-Datei:

Es handelt sich um ein Dateiformat, das auf dem Zip-Dateiformat basiert und zum Komprimieren vieler Dateien in eine einzige verwendet wird einzelne Datei.

Inhalt der JAR-Datei:

ÿ .java
ÿ .Klasse
ÿ Konfigurationsdateien

ÿ .Xml-Datei
ÿ .Eigenschaften

.xml:

Wird verwendet, um die Ressourcen mithilfe benutzerdefinierter oder benutzerdefinierter Tags zu konfigurieren.

.Eigenschaften:

Wird verwendet, um eine Reihe von Eigenschaften in Form eines Schlüssel-Wert-Paares bereitzustellen.

Bedarf an JAR-Datei:

Wird benötigt, um die Eigenschaften entsprechend der Anforderung zu importieren.

API (Anwendungsprogrammierschnittstelle):

ÿ Es wird für die Kommunikation zwischen Anwendungen
verwendet. ÿ Eine Anwendung kann mithilfe der Abstraktion mit einer anderen Anwendung kommunizieren
API zur Erzielung einer losen Kopplung.
ÿ Das Rückgrat der API ist ABSTRAKTION. ÿ
Ausgabe der Abstraktion ist LOOSE COUPLING.

Beispiel für API:

ÿ Apache-POI
ÿ Jaxcel – ***Java Excel zum Erstellen einer Excel-Tabelle.*** ÿ
JDBC
ÿ Servlet

Inhalt der API:

ÿ Schnittstellen
ÿ Hilfsklassen
ÿ Implementierungsklassen

Zwei Formen von API:

Ich bilde eine API:

Einzelne JAR-Datei

II-Form der API:

Zwei JAR-Dateien. Eine enthält Schnittstellen und Hilfsklassen.
Eine davon enthält Implementierungsklassen.

JDBC-API:

Enthält,

Schnittstellen – Treiber, Verbindung, Anweisung, vorbereitete Anweisung, aufrufbare Anweisung, ResultSet.

Hilfsklasse – DriverManager

Implementierungsklassen – werden vom jeweiligen Datenbankserver in Form einer JAR-Datei bereitgestellt.

Der JDBC-Treiber ist die Implementierung der JDBC-API, da er die Implementierung enthält Klassen für die in der JDBC-API vorhandenen Schnittstellen.

Alle Treiberklassen müssen zwingend die java.sql.Driver-Schnittstelle implementieren.

JDBC:

JAVA-DATENBANK-KONNEKTIVITÄT.

ÿ Es handelt sich um die Spezifikation, die in Form einer Abstraktions-API bereitgestellt wird, um Loose zu erreichen Kopplung zwischen der Java-Anwendung und dem Datenbankserver.

VORTEILE:

ÿ Plattformunabhängig. ÿ Wir
können eine lose Kopplung zwischen der Anwendung und dem Datenbankserver erreichen.

KOSTBARE RESSOURCEN:

Die Ressourcen, die Systemeigenschaften in Form von Stream nutzen, werden als kostspielig bezeichnet Ressourcen.

Alle kostspieligen Ressourcen müssen innerhalb des endgültigen Blocks mithilfe von IF CONDITION geschlossen werden Vermeiden Sie NULLPOINTEREXCEPTION.

In JDBC sind alle Schnittstellen der JDBC-API kostspielige Ressourcen.

SPEZIFIKATIONEN VON JDBC:

ÿ Alle Treiberklassen müssen einen statischen Block enthalten. ÿ Alle
Treiberklassen müssen die in vorhandene java.sql.Driver-Schnittstelle zwingend implementieren JDBC-API.

ÿ Alle Fahrerklassen müssen zwingend mit dem DriverManager registriert werden die Methode registerDriver().

SCHRITTE VON JDBC:

TREIBER LADEN UND REGISTRIEREN:

Es gibt zwei Möglichkeiten:

ÿ Manuell (Keine ***gute Vorgehensweise, da es zu einer engen Kopplung zwischen Java kommt Anwendung und Datenbankserver).*** ÿ
Automatisierter Weg

AUTOMATISIERTER WEG:

ÿ Verwenden einer statischen Methode namens ***forName() ,*** die in der Klasse vorhanden ist mit Namen als ***Klasse (im java.lang-Paket vorhanden)*** selbst. ÿ Es wird ein
vollständig qualifizierter Klassenname benötigt. ÿ
Die nur zum Laden verwendet wird. ÿ Immer
wenn wir diese Methode verwenden, wird eine Ausnahme namens ausgelöst ***ClassNotFoundException(Geprüfte Ausnahme).***
ÿ Die Klasse „Object of Driver“ wird erstellt, um den Code im zu registrieren ***RegisterManager-*** Klasse mit Hilfe einer statischen Methode namens registerDriver ***()*** vorhanden in der DriverManager()-Klasse.
ÿ Innerhalb der Driver-Klasse, die in der vom Datenbankserver bereitgestellten JAR-Datei vorhanden ist enthält einen ***statischen Block.*** ÿ
Dieser statische Block ist für die Registrierung verantwortlich. ÿ
Dieser statische Block enthält ***DriverManager.registerDriver(new Driver).***

Stellen Sie eine Verbindung zwischen der Java-Anwendung und dem Datenbankserver her:

ÿ Hierfür verwenden wir eine statische Factory/Helper-Methode namens ***getConnection()*** -Methode, die ist in der ***DriverManager-*** Klasse vorhanden , die zum Erstellen und Zurückgeben der verwendet wird Implementierungsobjekt der in der JDBC-API vorhandenen Verbindungsschnittstelle basierend auf URL.
ÿ Der ***Rückgabetyp der getConnection()-Methode ist Connection Interface.***

Es gibt drei überladene Varianten von getConnection-Methoden:

ÿ getConnection(String URL)
ÿ getConnection(String URL,Properties Info)
ÿ getConnection(String URL,String Benutzername,String Passwort)

Immer wenn wir diese Methode verwenden, wird eine Ausnahme namens ***SQLException(Checked)*** ausgelöst ***Ausnahme).***

ERSTELLEN SIE EINE ERKLÄRUNG ODER PLATTFORM:

ÿ Zum Erstellen einer Anweisung verwenden wir die aufgerufene nicht-statische Factory/Helper-Methode Methode createStatement(), die im Connection Interface vorhanden ist.
ÿ Der Rückgabetyp dieser Methode ist Statement Interface.

Führen Sie die SQL-Abfragen aus:

Um die SQL-Abfragen auszuführen, haben wir drei Methoden:

1. ausführen()
2.executeUpdate()
3.executeQuery()
4.

Alle diese Methoden sind in der Statement-Schnittstelle vorhanden, auf die durch Prepared zugegriffen werden kann
Anweisung sowie aufrufbare Anweisung auch durch INHERITENCE(Multi-Level
Vererbung).

ausführen():

ÿ Es handelt sich um eine generische Methode, die zum Ausführen aller Arten von SQL-Abfragen
verwendet wird. ÿ Es gibt True für DQL, False für DDL und DML zurück.
ÿ Der Rückgabetyp der Methodeexecute() ist BOOLEAN.

WAS IST DAS ERGEBNIS VON DML-ABFRAGEN/ DML-ERKLÄRUNGEN?

Das Ergebnis von DML-Abfragen ist ein ganzzahliger Wert von 0 bis n, der die Gesamtzahl angibt
Betroffene Datensätze im Datenbankserver.

ausführenUpdate():

ÿ Es handelt sich um eine spezielle Methode, die nur zur Ausführung von DML-Abfragen verwendet
wird. ÿ Der Rückgabetyp dieser Methode ist Integer, da das Ergebnis von DML-Abfragen 0 bis n ist
Integer Wert , Deshalb ist der Rückgabetyp Integer. ÿ Immer
wenn wir diese Methode verwenden, wird eine Ausnahme namens SQLException ausgelöst.

Wir können die Daten nicht direkt aus der Tabelle abrufen, sondern müssen die Daten stattdessen abrufen aus dem Cursor-/Pufferspeicher.

executeQuery():

ÿ Es handelt sich um eine spezielle Methode, die nur zur Ausführung von DQL-Abfragen
verwendet wird. ÿ Das Ergebnis von DQL-Abfragen sind die verarbeiteten/resultierenden Daten, die im gespeichert werden
Cursor-/Pufferspeicher, der mit Hilfe der ResultSet-Schnittstelle abgerufen werden kann
Warum der Rückgabetyp der MethodeexecuteQuery() ResultSet Interface ist.

ERKLÄRUNGSSCHNITTSTELLE:

ÿ Es handelt sich um eine Schnittstelle, die in der JDBC-API vorhanden ist, wo sich die Implementierungsklassen befinden
wird vom jeweiligen Datenbankserver bereitgestellt.
ÿ Im Falle einer Statement-Schnittstelle erfolgt die Kompilierung jedes Mal zusammen mit dem
Ausführung.
ÿ Es verbraucht mehr Speicher. ÿ Deshalb
entscheiden wir uns für das Prepared Statement Interface (PLACE HOLDER CONCEPT).

Standardmäßig zeigt die ResultSet-Schnittstelle nicht auf einen Datensatz im Cursor/ Puffer Speicher, dh >>> Es zeigte auf NULL AUFZEICHNUNGSBEREICH.

RESULTSET-SCHNITTSTELLE:

ÿ Es handelt sich um eine Schnittstelle, die in der JDBC-API vorhanden ist, wo sich die Implementierungsklassen befinden wird vom jeweiligen Datenbankserver bereitgestellt. ÿ Die ResultSet-Schnittstelle enthält eine Methode namens getXXX()-Methode, mit der diese Methode verwendet wird Rufen Sie die Daten aus dem Cursor/Pufferspeicher ab. ÿ Es enthält auch zwei weitere Methoden, ÿ die Methoden next() und absolute().

Wir können ein Implementierungsobjekt der ResultSet-Schnittstelle auf zwei Arten erstellen:

1) Bei Verwendung der getResultSet()-Methode ist der Rückgabetyp die ResultSet-Schnittstelle.
2) Eine andere verwendet die MethodeexecuteQuery().

GetXXX():

Diese Methode ist von zwei überladenen Typen:

ÿ getXXX(int Spaltennummer)
ÿ getXXX(String-Spaltenname)

Der Rückgabetyp dieser Methode ist der jeweilige Datentyp.

Nächste():

Mit dieser Methode wird überprüft, ob der nächste Datensatz vorhanden ist oder nicht. Es wird nicht zurückgegeben Wert.
Der Rückgabetyp der next()-Methode ist Boolean.
Diese Methode wird nur verwendet, wenn wir eine Mindestanzahl an Datensätzen haben, da eine höhere Anzahl erforderlich ist Die Anzahl der Datensätze beginnt bei jedem Prüfvorgang bei 0. Es ist ein zeitaufwändiger Prozess.

Absolut():

Diese Methode wird auch verwendet, um zu überprüfen, ob der jeweilige Datensatz vorhanden ist oder nicht.
Als Parameter wird eine ganzzahlige Zeilennummer verwendet.
Der Rückgabetyp ist Boolean.

PLATZHALTER:

Platzhalter ist ein Parameter, der dynamische Werte zur Laufzeit durch den Benutzer speichert.

REGELN ZUM FESTLEGEN VON WERTEN FÜR PLATZHALTER:

ÿ Wir müssen vor der Ausführung die Werte für den Platzhalter festlegen. ÿ Die Anzahl der Daten sollte mit der Anzahl der vorhandenen Platzhalter übereinstimmen. ÿ Wir müssen die Daten für den Platzhalter mithilfe der setXXX()-Methode festlegen.

setXXX():

ÿ Diese Methode ist in Prepared Statement vorhanden, die Statement-Schnittstelle kann nicht darauf zugreifen. ÿ Daher unterstützt Statement Interface das Platzhalterkonzept nicht. ÿ Der Rückgabetyp der setXXX()-Methode ist ***Void.***

VORBEREITETE ERKLÄRUNG:

ÿ Es handelt sich um eine in der JDBC-API vorhandene Schnittstelle, über die die Implementierungsklassen bereitgestellt werden
Der Datenbankserver in Form einer JAR-Datei. ÿ In der
vorbereiteten Anweisungsschnittstelle erfolgt die Kompilierung nur einmal während des Objekts
Die Erstellung selbst und die Ausführung erfolgen viele Male ***(AUSFÜHRUNGSPLANUNG).***

Die PrepareStatement()- Methode ist eine nicht statische Methode, die in der Verbindungsschnittstelle vorhanden ist und zum Erstellen und Zurückgeben des Implementierungsobjekts der Prepared Statement-Schnittstelle verwendet wird.

Der Rückgabetyp der PrepareStatement()-Methode ist die Prepared Statement Interface.

Warum wird die Schnittstelle für vorbereitete Anweisungen als vorkompilierte Anweisung bezeichnet?

Im Prepared Statement Interface übergeben wir die Abfrage zum Zeitpunkt des Implementierungsobjekts
Schaffung. Aus diesem Grund wird es als vorkompilierte Anweisung bezeichnet.