□□□□ (□□□□ □□□□□□□□ □□□□□□□□□□□□□□) ड्राइवर का उपयोग

एक □□□□ (□□□□ □□□□□□□□ □□□□□□□□□□□□□□) ड्राइवर को □□□□ में उपयोग करने के लिए, आपको डेटाबेस से कनेक्शन स्थापित करने, क्वेरी को एग्जेक्यूट करने और परिणामों को हैंडल करने के लिए एक श्रृंखला के कदमों का पालन करना होगा। नीचे, मैं आपको एक स्पष्ट और प्रैक्टिकल तरीके से कदम-दर-कदम प्रक्रिया के बारे में बताता हूँ।

कदम 1: प्रोजेक्ट में □□□□ ड्राइवर जोड़ें

पहले, आपको आपका काम कर रहे डेटाबेस के लिए सही □□□□ ड्राइवर चाहिए (जैसे कि □□□□□, □□□□□□□□□□, □□□□□□, आदि)। प्रत्येक डेटाबेस वेंडर अपने ड्राइवर प्रदान करता है।

- □ □□□□□ **प्रोजेक्टों के लिए**: `pom.xml` में निर्भरता जोड़ें। उदाहरण के लिए, □□□□□ के लिए:

  ```
  <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
      <version>8.0.33</version> <!--                -->
  </dependency>
  ```

  □□□□□□□□□□ के लिए:

  ```
  <dependency>
      <groupId>org.postgresql</groupId>
      <artifactId>postgresql</artifactId>
      <version>42.7.3</version> <!--                -->
  </dependency>
  ```

- □ □□□-□□□□□ **प्रोजेक्टों के लिए**: वेंडर की वेबसाइट से □□□□ ड्राइवर □□□ फ़ाइल डाउनलोड करें (जैसे कि □□□□□ □□□□□□□□□/□ से □□□□□.□□□) और इसे प्रोजेक्ट के क्लासपाथ में मैन्युअल रूप से जोड़ें।

कदम 2: □□□□ ड्राइवर को लोड करें

पुराने संस्करणों में □□□□ (□□□□ 4.0 से पहले), आपको `Class.forName()` का उपयोग करके ड्राइवर को स्पष्ट रूप से लोड करना पड़ा था। आधुनिक □□□□ ड्राइवरों (□□□□ 4.0+) के साथ, यह कदम विकल्प है क्योंकि ड्राइवर सेवा प्रोवाइडर मैकेनिज्म के माध्यम से स्वचालित रूप से पंजीकृत होता है। फिर भी, स्पष्टता के लिए, यह कैसे काम करता है:

```
try {
    Class.forName("com.mysql.cj.jdbc.Driver"); // MySQL
    // PostgreSQL     : Class.forName("org.postgresql.Driver");
} catch (ClassNotFoundException e) {
    e.printStackTrace();
}
```

कदम 3: कनेक्शन स्थापित करें

`DriverManager` क्लास का उपयोग करके कनेक्शन □□□, उपयोगकर्ता नाम और पासवर्ड प्रदान करके डेटाबेस से कनेक्शन स्थापित करें। □□□ प्रारूप डेटाबेस पर निर्भर करता है।

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class JdbcExample {
    public static void main(String[] args) {
        String url = "jdbc:mysql://localhost:3306/mydatabase"; // MySQL
        String username = "your_username";
        String password = "your_password";

        try (Connection connection = DriverManager.getConnection(url, username, password)) {
            System.out.println("              !");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

□ □□□ **उदाहरण**:

- □□□□□: `jdbc:mysql://localhost:3306/mydatabase?useSSL=false`
- □□□□□□□□□□: `jdbc:postgresql://localhost:5432/mydatabase`
- □□□□□□: `jdbc:oracle:thin:@localhost:1521:xe`

कदम 4: एक स्टेटमेंट बनाएं और क्वेरी को एग्जेक्यूट करें

कनेक्शन स्थापित होने के बाद, आप `Statement`, `PreparedStatement` (परामेटराइज्ड क्वेरी के लिए पसंदीदा) या `CallableStatement` (स्टोर्ड प्रोसीजर्स के लिए) का उपयोग करके □□□ क्वेरी को एग्जेक्यूट कर सकते हैं।

एक सरल क्वेरी के साथ उदाहरण:

```
import java.sql.*;

public class JdbcExample {
    public static void main(String[] args) {
        String url = "jdbc:mysql://localhost:3306/mydatabase";
```

```
        String username = "your_username";
        String password = "your_password";

        try (Connection connection = DriverManager.getConnection(url, username, password)) {
            Statement statement = connection.createStatement();
            ResultSet resultSet = statement.executeQuery("SELECT * FROM users");

            while (resultSet.next()) {
                System.out.println("ID: " + resultSet.getInt("id") + ",   : " + resultSet.getString("name"));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

एक □□□□□□□□□□□□□□□□□□□ के साथ उदाहरण (उपयोगकर्ता इनपुट के लिए सुरक्षित):

```
try (Connection connection = DriverManager.getConnection(url, username, password)) {
    String sql = "SELECT * FROM users WHERE id = ?";
    PreparedStatement preparedStatement = connection.prepareStatement(sql);
    preparedStatement.setInt(1, 5); //
    ResultSet resultSet = preparedStatement.executeQuery();

    while (resultSet.next()) {
        System.out.println("ID: " + resultSet.getInt("id") + ",   : " + resultSet.getString("name"));
    }
} catch (SQLException e) {
    e.printStackTrace();
}
```

कदम 5: संसाधन बंद करें

हमें हमेशा `ResultSet`, `Statement`, और `Connection` ऑब्जेक्ट को बंद करना चाहिए ताकि संसाधन मुक्त हो जाएं। □□□-□□□□-□□□□□□□□□□ (उपरोक्त दिखाया गया है) इसे स्वचालित रूप से हैंडल करता है।

पूर्ण कार्यात्मक उदाहरण (□□□□□□)

यह एक पूर्ण उदाहरण है, जो एक `users` टेबल के साथ है जिसमें `id` (□□□) और `name` (□□□□□□□□) स्तंभ हैं:

```java
import java.sql.*;

public class JdbcExample {
    public static void main(String[] args) {
        String url = "jdbc:mysql://localhost:3306/mydatabase?useSSL=false";
        String username = "root";
        String password = "your_password";

        try (Connection connection = DriverManager.getConnection(url, username, password)) {
            String sql = "SELECT * FROM users WHERE id = ?";
            PreparedStatement preparedStatement = connection.prepareStatement(sql);
            preparedStatement.setInt(1, 1);
            ResultSet resultSet = preparedStatement.executeQuery();

            while (resultSet.next()) {
                System.out.println("ID: " + resultSet.getInt("id") + ",   : " + resultSet.getString("name"));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

मुख्य टिप्स

- **एरर हैंडलिंग**: डेटाबेस संबंधी त्रुटियों को हैंडल करने के लिए हमेशा `SQLException` को कैच करें।
- **निर्भरताएं**: □□□□ ड्राइवर को डेटाबेस संस्करण के साथ मेल खाएं।
- **सुरक्षा**: उपयोगकर्ता इनपुट के साथ क्वेरी के लिए `PreparedStatement` का उपयोग करें ताकि □□□ इंजेक्शन से बचा जा सके।
- **कनेक्शन पूलिंग**: उत्पादन एप्लिकेशन के लिए, `DriverManager` के बजाय एक कनेक्शन पूल (जैसे कि □□□□□□□□□) का उपयोग करने की सोचें।

मुझे किसी विशेष डेटाबेस या अधिक उन्नत उपयोग के मामलों में मदद चाहिए तो मुझे बताएं!