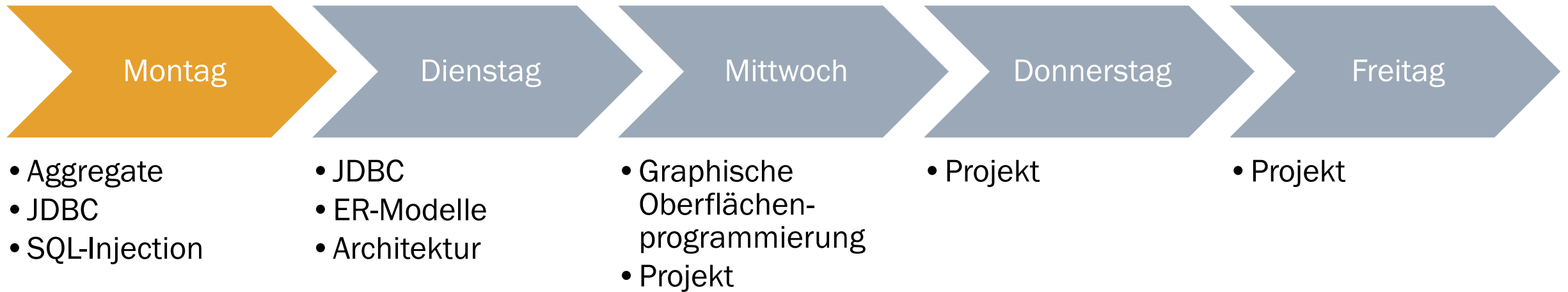


JDBC

März 2025

Plan für die Woche



Plan für heute

- Was sind Transaktionen?
- Was ist JDBC?
- notwendige Klassen und Methoden von JDBC
- SQL-Injection
- Aufsetzen eines Projektes in IntelliJ mittels Maven

Transaktionen

Einleitung

- ist eine Reihe an Datenbankoperationen (Queries), die als logische Einheit betrachtet werden
- eine Transaktion wird nach dem Prinzip „alles oder nichts“ ausgeführt:
 - Entweder wird sie vollständig abgeschlossen, oder alle Änderungen werden zurückgesetzt
- Schritte von Transaktion:
 1. Beginn einer Transaktion:
 - Änderungen werden temporär gespeichert
 - Es erfolgt keine dauerhafte Übernahme in die Datenbank!**

Einleitung

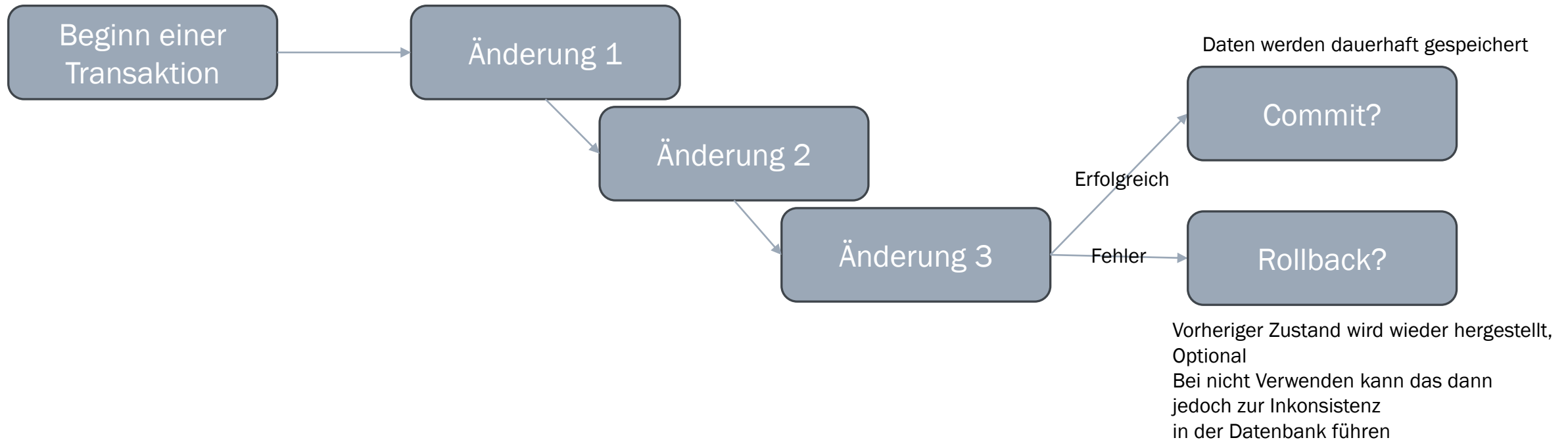
- ist eine Reihe an Datenbankoperationen (Queries), die als logische Einheit betrachtet werden
- eine Transaktion wird nach dem Prinzip „alles oder nichts“ ausgeführt:
 - Entweder wird sie vollständig abgeschlossen, oder alle Änderungen werden zurückgesetzt
- Schritte von Transaktion:
 1. Beginn einer Transaktion:
 - Änderungen werden temporär gespeichert
 - Es erfolgt keine dauerhafte Übernahme in die Datenbank!**
 2. Commit:
 - Änderungen werden endgültig gespeichert
 - Erst nach diesem Schritt sind die Änderungen dauerhaft und für andere sichtbar

Einleitung

- ist eine Reihe an Datenbankoperationen (Queries), die als logische Einheit betrachtet werden
- eine Transaktion wird nach dem Prinzip „alles oder nichts“ ausgeführt:
 - Entweder wird sie vollständig abgeschlossen, oder alle Änderungen werden zurückgesetzt
- Schritte von Transaktion:
 1. Beginn einer Transaktion:
 - Änderungen werden temporär gespeichert
 - Es erfolgt keine dauerhafte Übernahme in die Datenbank!**
 - 2a. Commit:
 - Änderungen werden endgültig gespeichert
 - Erst nach diesem Schritt sind die Änderungen dauerhaft und für andere sichtbar
 - 2b. Rollback:
 - Es werden alle temporären Änderungen verworfen
 - Datenbank wird in den Zustand vor der Transaktion zurückversetzt

Einleitung

– Schritte von Transaktion



JDBC

Einleitung

- JDBC steht für Java Database Connectivity
- ist eine API (Schnittstelle), die es Java-Anwendungen ermöglicht, mit relationalen Datenbanken zu kommunizieren

Wichtige Klassen und Methoden

- JDBC besteht aus mehreren Klassen und Schnittstellen
- folgende Klassen und Methoden sind wichtig, damit Java sich mit einer Datenbank verbinden und mit ihr kommunizieren kann

DriverManager

- Stellt eine Verbindung zur Datenbank her
- wichtige Methode:

```
getConnection(String url, String user, String password)
```

- Erzeugt eine Verbindung zur Datenbank
- URL zur Datenbank `jdbc:mysql://localhost:3306/myDatabaseName`
- User der Datenbank: `root`
- Passwort der Datenbank `übergebenes Passwort`

Connection

- repräsentiert eine Verbindung zur Datenbank
- wird in Kombination mit dem DriverManager genutzt, da der DriverManager eine Connection erzeugt
- Beispiel:

```
private static final String URL = "jdbc:mysql://localhost:3306/hospital";
private static final String USER = "root";
private static final String PASSWORD = "password";

public static Connection getConnection() {
    try {
        return DriverManager.getConnection(URL, USER, PASSWORD);
    } catch (SQLException ex) {
        System.out.println("Verbindung konnte nicht hergestellt werden");
    }
    return null;
}
```

Connection

– wichtige Methoden:

```
setAutoCommit(boolean autoCommit);  
commit();
```

- Commit: Vorgang, bei dem alle Anfragen zu Änderungen, die in einer Transaktion gemacht wurden, dauerhaft in einer Datenbank gespeichert werden
- Das automatische Commiten einer Query kann ausgeschaltet werden
- Ist das automatische Commiten ausgeschaltet, muss man manuell für das Beenden einer Transaktion die Methode `commit()` aufrufen

Connection

– Beispiel:

```
public static Connection getConnection() {  
    try {  
        Connection connection = DriverManager.getConnection(URL, USER, PASSWORD);  
        connection.setAutoCommit(false);  
        return connection;  
    } catch (SQLException ex) {  
        System.out.println("Verbindung konnte nicht hergestellt werden");  
    }  
    return null;  
}
```

Connection

- wichtige Methoden:

`prepareStatement(String query)`

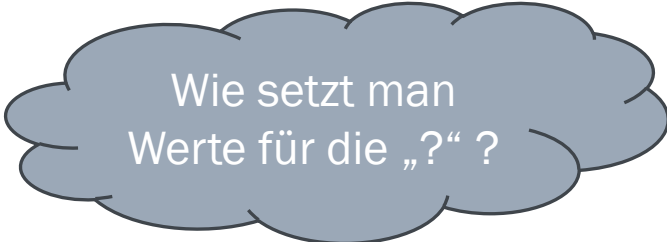
`statement(String query)`

//eher vermeiden, da unsicher

- Bereitet eine Query vor, um sie an die Datenbank zu schicken
- Gibt einen PreparedStatement bzw. Statement zurück

PreparedStatement

- Wird für vorbereitete SQL-Abfragen genutzt
- sicherer gegen SQL-Injections (vs. Statement)
- Aufbau der Query muss abgewandelt werden:
 - Es werden Platzhalter benötigt – sogenannte Parameter-Marker
 - Diese Platzhalter werden mit „?“ dargestellt, in die später tatsächliche Werte eingesetzt werden
 - Pro Wert wird ein „?“ benötigt
- Beispiel:



Wie setzt man
Werte für die „?“ ?

```
PreparedStatement p2 = connection.prepareStatement("INSERT INTO Patient(firstname, lastname, email, dayOfBirth, Age, Weight, DoctorID) VALUES (?, ?, ?, ?, ?, ?, ?)");
```

PreparedStatement

– wichtige Methoden:

`setString(int parameterIndex, String value)`

`setInt(int parameterIndex, int value)`

– Wird genutzt, um die gewünschten Werte in die Query statt den „?“ einzusetzen

PreparedStatement

– Beispiel:

```
PreparedStatement p2 = c.prepareStatement("INSERT INTO Patient(firstname, lastname, email, dayOfBirth, Age, Weight, DoctorID) VALUES (?, ?, ?, ?, ?, ?, ?)");  
p2.setString(1, "Lars");  
p2.setString(2, "Müller");  
p2.setString(3, "Lars.Mueller@web.de");  
p2.setDate(4, Date.valueOf(LocalDate.parse("1999-03-14")));  
p2.setInt(5, 26);  
p2.setInt(6, 26);  
p2.setInt(7, 1);
```

PreparedStatement

- wichtige Methoden:

`execute()`

- Wird genutzt, um beliebige SQL-Anweisungen auszuführen
- Gibt einen boolean zurück:
 - true – wenn ein ResultSet zurück geliefert wird
 - false – wenn kein ResultSet oder die Anzahl an geänderten Zeilen zurückgeschickt wird

PreparedStatement

– wichtige Methoden:

`executeQuery()`

- Wird genutzt, um das SQL-Anweisungen, wie SELECT, auszuführen
- Gibt einen ResultSet-Objekt zurück, das die durch die Abfrage erzeugten Daten enthält

PreparedStatement

– wichtige Methoden:

`executeUpdate()`

- Wird genutzt, um SQL-Anweisungen, wie INSERT, UPDATE, DELETE, auszuführen
- Gibt einen int zurück, der die Anzahl der durch die Anweisung betroffenen Zeilen angibt
- Wird oft im Kontext von Transaktionen genutzt

PreparedStatement

Methode	Verwendungszweck	Rückgabewert	Beispiel-SQL-Befehl
execute()	Für beliebige SQL-Anfragen, die entweder ein ResultSet oder eine Anzahl der betroffenen Zeilen zurückgeben.	boolean (true/false)	SELECT, UPDATE, DELETE (je nach Rückgabewert)
executeQuery()	Nur für SELECT -Abfragen, die ein ResultSet zurückgeben.	ResultSet	SELECT
executeUpdate()	Für INSERT , UPDATE , und DELETE -Befehle.	int (Anzahl der betroffenen Zeilen)	INSERT, UPDATE, DELETE

PreparedStatement

– Beispiel:

```
public static void main(String[] args) {  
    try (Connection c = getConnection()) {  
        if (c != null) {  
            PreparedStatement p2 = c.prepareStatement("INSERT INTO Patient(firstname, lastname, email, dayOfBirth, Age, Weight, DoctorID) VALUES (?, ?, ?, ?, ?, ?, ?)");  
            p2.setString(1, "Lars");  
            p2.setString(2, "Müller");  
            p2.setString(3, "Lars.Mueller@web.de");  
            p2.setDate(4, Date.valueOf(LocalDate.parse("1999-03-14")));  
            p2.setInt(5, 26);  
            p2.setInt(6, 26);  
            p2.setInt(7, 1);  
            p2.execute();  
            c.commit(); //setzen, wenn AutoCommit ausgeschaltet  
        }  
    } catch (SQLException e) {  
        System.out.println(e.getMessage());  
    }  
}
```


ResultSet

- speichert die Ergebnisse einer SQL-Abfrage
- wichtige Methoden:

`next()`

- Verschiebt den Zeiger auf die nächste Zeile aus dem Ergebnis
- Gibt einen boolean zurück, falls es ein nächstes Element im Set gibt
- Für Schleifen wichtig!

ResultSet

- speichert die Ergebnisse einer SQL-Abfrage
- wichtige Methoden:

`getString(String columnLabel) / getString(int columnIndex)`
`getInt(String columnLabel) / getInt(int columnIndex)`

- Werden verwendet, um den Wert einer bestimmten Spalte abzurufen

ResultSet

– Beispiel:

```
try (Connection c = getConnection()) {  
    if (c != null) {  
        PreparedStatement p = c.prepareStatement("SELECT * FROM Patient");  
        ResultSet resultSet = p.executeQuery();  
  
        while (resultSet.next()) {  
            System.out.println(resultSet.getString("firstname"));  
            System.out.println(resultSet.getInt("Age"));  
        }  
    }  
} catch (SQLException e) {  
    System.out.println(e.getMessage());  
}
```

SQL-Injection

Rechtlicher Hinweis

Das Ausführen von SQL-Injection-Angriffen ist **illegal** und kann strafrechtliche Konsequenzen nach sich ziehen. In vielen Ländern wird dies als **Computerkriminalität** betrachtet und kann zu hohen Geldstrafen oder sogar Gefängnisstrafen führen.

Die Teilnehmer dieser Vorlesung sind dazu angehalten, die erlernten Konzepte ausschließlich für legale und sicherheitsrelevante Zwecke zu verwenden. Der Missbrauch dieser Techniken kann ernste rechtliche Folgen haben.

Was ist eine SQL-Injection?

- ist eine Sicherheitslücke, bei der ein Angreifer bösartigen SQL-Code in eine Anwendung einschleust
- entsteht durch unsichere (= nicht überprüfte) Benutzereingaben
- Beispiel:
 - Falls die Benutzereingabe den Wert `" OR '1'='1"` beinhaltet, resultiert die Abfrage in einem Zugriff auf alle Datensätze
- durch PreparedStatements wird die Benutzereingabe sicher eingebunden, wodurch eine Manipulation der Query verhindert wird

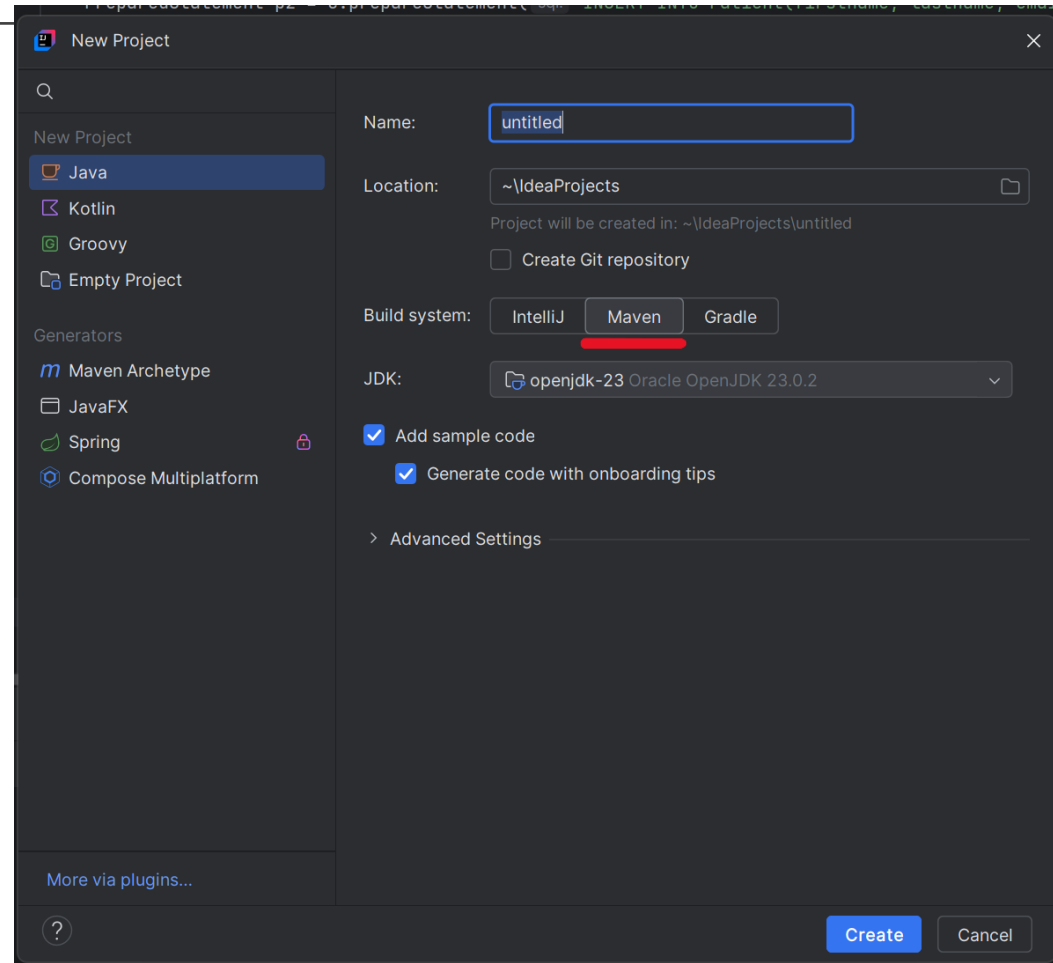
Aufsetzen eines Projektes in IntelliJ

MAVEN ALS BUILD-SYSTEM

Einrichten

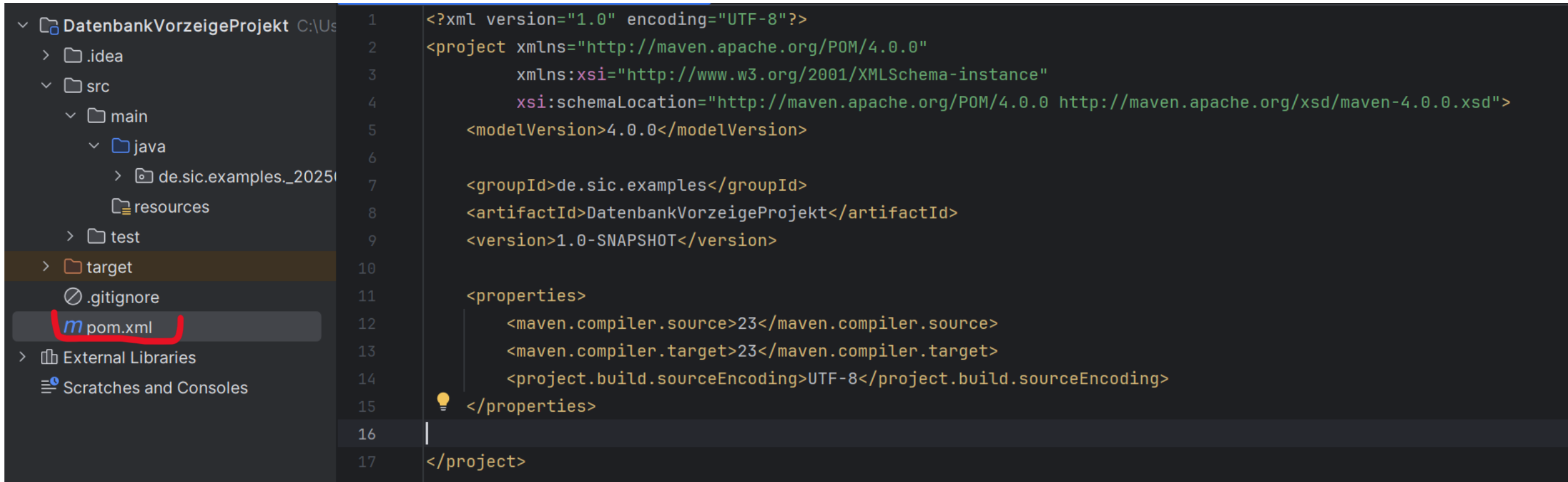
1. Neues Projekt in IntelliJ öffnen:

1. Name auswählen
2. Maven auswählen
3. Create drücken



Einrichten

2. pom.xml-Datei öffnen



Einrichte

2. Folgendes hinzufügen:


```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>de.sic.examples</groupId>
  <artifactId>DatenbankVorzeigeProjekt</artifactId>
  <version>1.0-SNAPSHOT</version>

  <properties>
    <maven.compiler.source>23</maven.compiler.source>
    <maven.compiler.target>23</maven.compiler.target>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

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

</project>
```



Einrichtu

3. Reload drücken

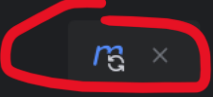
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>de.sic.examples</groupId>
  <artifactId>DatenbankVorzeigeProjekt</artifactId>
  <version>1.0-SNAPSHOT</version>

  <properties>
    <maven.compiler.source>23</maven.compiler.source>
    <maven.compiler.target>23</maven.compiler.target>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

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

</project>
```



Einrichten

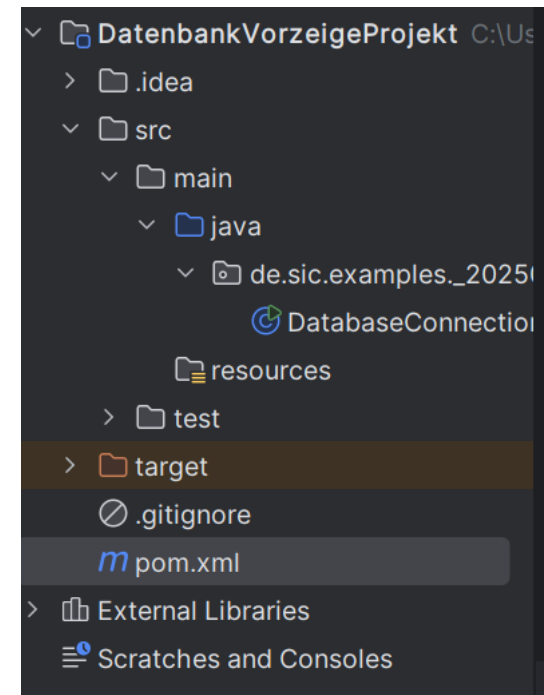
3. Was passiert?

- Wir nutzen Maven – ein Build-Management-System, um die MySQL-Connector-Bibliothek herunterzuladen, um sie in unserem Projekt nutzen zu können
- Die pom.xml-Datei enthält alle Abhängigkeiten (Bibliotheken, Frameworks,...), die das Projekt benötigt
- bei Hinzufügen einer Dependency, lädt Maven diese aus dem zentralen Maven-Repository herunter
- diese heruntergeladene Bibliothek wird in den Klassenpfad des Projekts aufgenommen und ist während der Kompilierung und Laufzeit verfügbar

Einrichten

4. Erstellen eines Projektes

- Unter dem src/main/java/org/example/ eine neue Klasse erstellen



Einrichten

4. Einfügen des Initial-Codes und auf Run drücken