JDBC

Veerle Ongenae

Overzicht

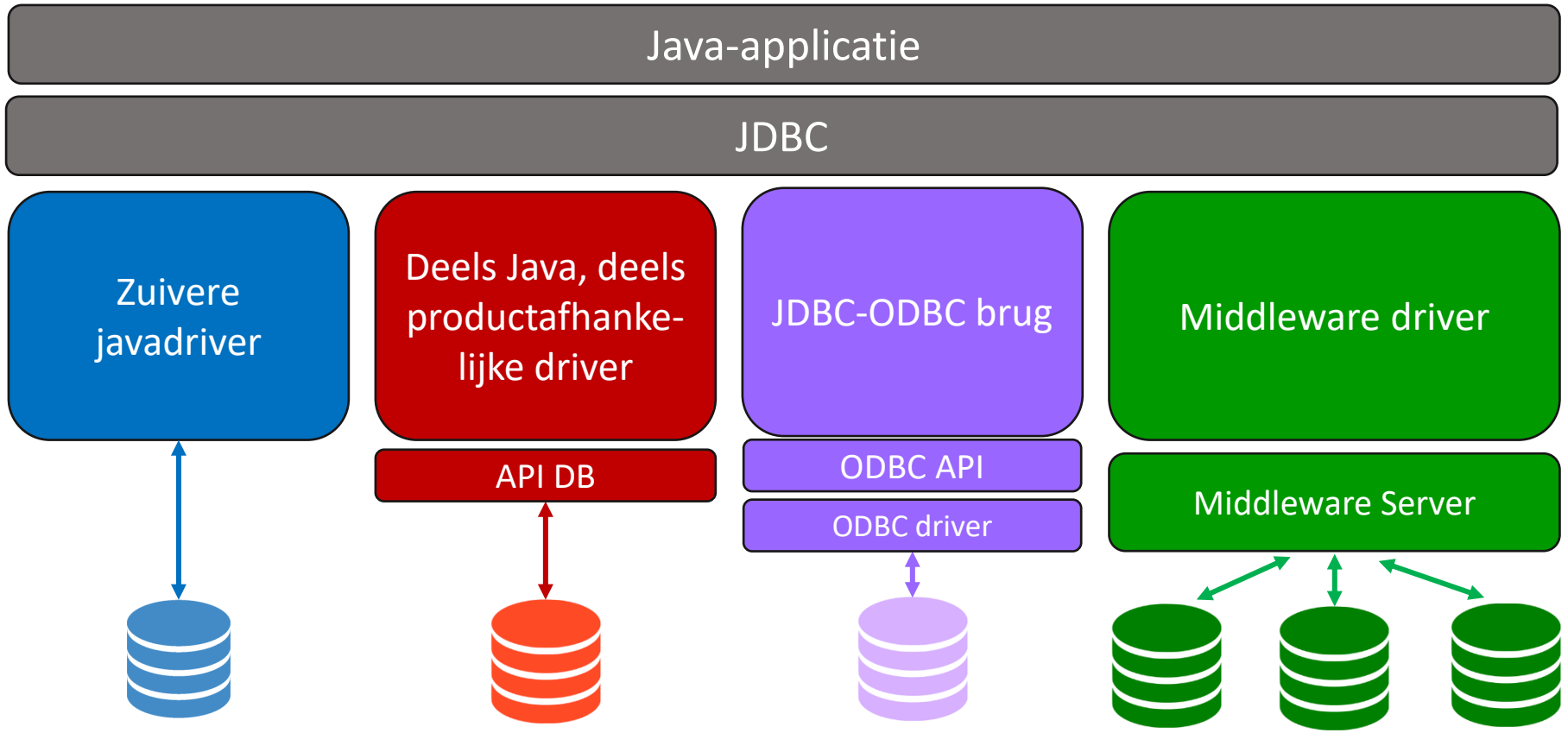
UNIVERSITEIT GENT

- Wat is JDBC?
- Basiswerking
  - Statements
  - PreparedStatements
  - CallableStatements
  - Transacties
- JDBC uitgebreid
  - DataSource
  - Scrollable en aanpasbare ResultSet
  - RowSets
  - SQL3-gegevenstypes

Java Database Connectivity (JDBC) API

- Toegang tot relationele data vanuit Java
- SQL-compatibele relationele gegevensbanken
- Een API om SQL-statements uit te voeren en de verkregen resultaten te verwerken
- Abstractie van productafhankelijke details
  - Veralgemening meest voorkomende gegevensbankfuncties
  - Applicatie bruikbaar met verschillende gegevensbanken

JDBC drivers - types



abstractie

implementatie

JDBC

UNIVERSITEIT GENT

- JDBC Core API: **`java.sql`**
  - Gegevensbankverbinding via JDBC-drivers
  - Basis SQL-operaties uitvoeren
  - Bij alle versies van Java
- JDBC Optional Package API: **`javax.sql`**
  - Voor application servers
    - ➢ Datasource ipv driver
  - Beheer en pooling van JDBC-connecties, gedistribueerde transacties, rowsets, ...
  - Standaard bij JSDK vanaf versie 1.4

Overzicht

UNIVERSITEIT GENT

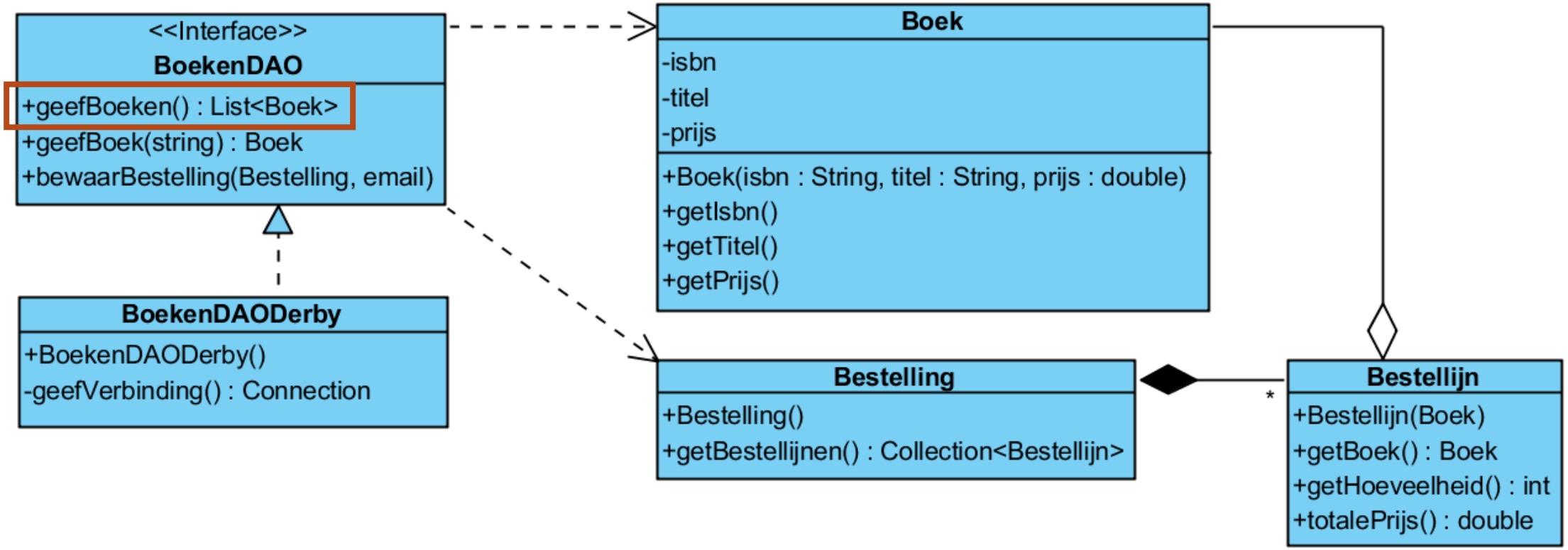
- Wat is JDBC?
- Basiswerking
  - Statements
  - PreparedStatements
  - CallableStatements
  - Transacties
- JDBC uitgebreid
  - DataSource
  - Scrollable en aanpasbare ResultSet
  - RowSets
  - SQL3-gegevenstypes

Voorbeeld

Tabel BOEKEN

ISBN	TITEL	PRIJS
isbn19789043025256	jQuery - de basis	12
isbn2	aan de slag met HTML5	15
isbn3	Head First Design Patterns	45
isbn1	Thinking in Java	58
isbn4	Java Servlet Programming	36
isbn5	Java Server Programming	81
isbn6	The Java Tutorial	65
isbn7	Java Swing	55

Voorbeeld


<<Interface>>
BoekenDAO
+geefBoeken() : List<Boek>
+geefBoek(string) : Boek
+bewaarBestelling(Bestelling, email)
Boek
-isbn
-titel
-prijs
+Boek(isbn : String, titel : String, prijs : double)
+getIsbn()
+getTitel()
+getPrijs()
BoekenDAODerby
+BoekenDAODerby()
-geefVerbinding() : Connection
Bestelling
+Bestelling()
+getBestellijnen() : Collection<Bestellijn>
*
Bestellijn
+Bestellijn(Boek)
+getBoek() : Boek
+getHoeveelheid() : int
+totalePrijs() : double

Verschillende stappen

- Configuratiegegevens inlezen
- Driver laden (optioneel vanaf JDK 1.6)
- Verbinding aanmaken
- Statement aanmaken
  - “gewoon”
  - PreparedStatement (parameters)
  - CallableStatement (stored procedure)
- Statement uitvoeren → eventueel ResultSet overlopen

Configuratie – driver laden

```
public class BoekenDAODerby implements BoekenDAO {

  private final ResourceBundle sqlOpdrachten;
  private final ResourceBundle databaseConfig;

  public BoekenDAODerby() throws ClassNotFoundException {
    sqlOpdrachten = ResourceBundle.getBundle("be.tiwi.boekenwinkel.implDb.sql");
    databaseConfig = ResourceBundle.getBundle("be.tiwi.boekenwinkel.implDb.database");

   // enkel voor oudere versies van Java (voor JDK 1.6)
   Class.forName(databaseConfig.getString("DRIVER"));

  }
  …
}
```

Configuratie

- .properties-bestand

sql.properties

```
# boek opvragen
Q_BOEKEN = select * from boeken
BOEK_ISBN = isbn
TITEL = titel
PRIJS = prijs
```

commentaar

sleutel-waarde paren

Configuratie

UNIVERSITEIT GENT

- Bestand gebruiken in programma

```
ResourceBundle databaseConfig =
  ResourceBundle.getBundle("be.tiwi.boekenwinkel.implDb.database");
```

- Naam klasse of naam bestand
- PropertyResourceBundle

```
String klasseDriver = databaseConfig.getString("DRIVER")
```

database.properties

```
# info databank
DRIVER = org.apache.derby.jdbc.ClientDriver
…
```


Driverklasse laden

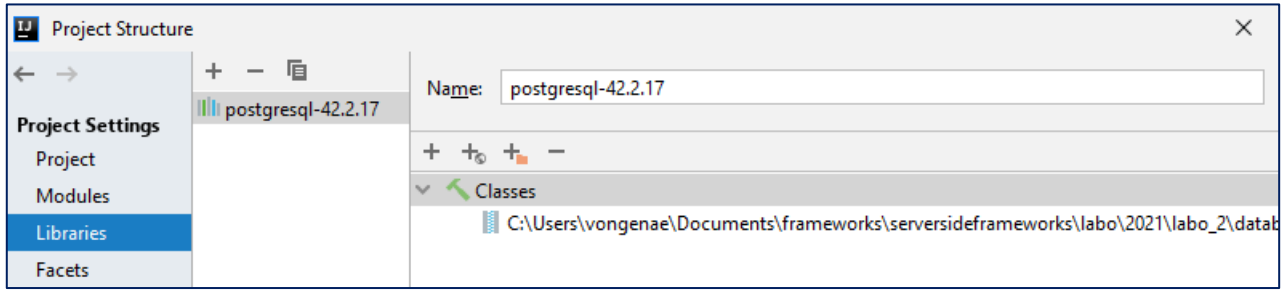
- Laden in JVM en registreren bij `DriverManager`
- Niet meer nodig vanaf JDK1.6
- Driverklasse implementeert de interface `java.sql.Driver`

```
try {
  Class.forName("org.apache.derby.jdbc.ClientDriver");
} catch (ClassNotFoundException e) {
  …
}
```

Driverklasse toevoegen aan CLASSPATH

UNIVERSITEIT GENT

- De driverklasse behoort niet tot de standaard Java API
- Meegeleverd met de gegevensbank
  - jar-bestand
- Toevoegen aan CLASSPATH
  - Als omgevingsvariabele
  - Als optie bij het `java`-commando
  - IntelliJ: External Libraries
- CLASSPATH
  - Waar zoeken naar klassen?
  - Standaard: huidige map



Verschillende stappen

- Configuratiegegevens inlezen
- Driver laden (optioneel vanaf JDK 1.6)
- Verbinding aanmaken
- Statement aanmaken
  - “gewoon”
  - PreparedStatement (parameters)
  - CallableStatement (stored procedure)
- Statement uitvoeren → eventueel ResultSet overlopen

Verbinding aanmaken

UNIVERSITEIT GENT

- DriverManager
  - Toegang tot verschillende beschikbare drivers

```
private Connection geefVerbinding() throws SQLException {
  return DriverManager.getConnection(
    databaseConfig.getString("URL"),
    databaseConfig.getString("LOGIN"),
    databaseConfig.getString("PASWOORD"));
}
```

database.properties

```
URL = jdbc:derby://localhost:1527/boekenwinkel
LOGIN = iii
PASWOORD = iiipwd
```


JDBC URL

- Vorm: `jdbc:<subprotocol>://<subname>`
- Subprotocol: identificeert gegevensbankdriver
  - MySql: `mysql`
  - Derby: `derby`
  - PostgreSQL: `postgresql`
- Subname: Afhankelijk van driver
  - MySql, Derby, PostgreSQL:
    - `host:poort/database`

Verschillende stappen

- Configuratiegegevens inlezen
- Driver laden (optioneel vanaf JDK 1.6)
- Verbinding aanmaken
- Statement aanmaken
  - “gewoon”
  - PreparedStatement (parameters)
  - CallableStatement (stored procedure)
- Statement uitvoeren → eventueel ResultSet overlopen

Opdracht aanmaken

```
public List<Boek> geefBoeken() throws BoekenNietBeschikbaarExceptie {
  ArrayList<Boek> boeken = new ArrayList<>();

  try {

    try (Connection conn = geefVerbinding();
      Statement stmt = conn.createStatement()) {
      … // opdracht uitvoeren
    }

  } catch (SQLException e) {
    throw new BoekenNietBeschikbaarExceptie(e);
  }
  return boeken;
}
```

Voorbeeld

- Voorspel uitvoer?

```
try (Connection conn = …;
     Statement stmt = conn.createStatement()) {

  System.out.println(conn.getClass());
  System.out.println(stmt.getClass());

} catch (SQLException e) {
  Logger.getLogger(KlassenDB.class.getName())
    .log(Level.SEVERE, null,
       "probleem met ophalen info uit databank: " + e.getMessage());
}
```

```
class org.postgresql.jdbc.PgConnection
class org.postgresql.jdbc.PgStatement
```

Select-opdracht uitvoeren

```
try (Connection conn = geefVerbinding();
     Statement stmt = conn.createStatement()) {

  // opdracht uitvoeren
  ResultSet rs = stmt.executeQuery(sqlOpdrachten.getString("Q_BOEKEN"));
  while (rs.next()) {
    boeken.add(new Boek(
      rs.getString(sqlOpdrachten.getString("BOEK_ISBN")),
      rs.getString(sqlOpdrachten.getString("TITEL")),
      rs.getDouble(sqlOpdrachten.getString("PRIJS"))));
  }
}
```

Resultaten ophalen

UNIVERSITEIT GENT

- Interface ResultSet
  - Resultaat één keer van boven naar beneden overlopen
  - Kolommen tellen vanaf 1 !!!!
  - Resultaten worden mogelijks niet allemaal tegelijk opgehaald
  - Conversie SQL-type naar Java-type

```
public boolean next() throws SQLException;
public xxx getXxx(int columnIndex) throws SQLException
public xxx getXxx(String columnName) throws SQLException
```


JDBC Types

UNIVERSITEIT GENT

- Java ↔ SQL
  - String ↔ CHAR, VARCHAR
  - double ↔ DOUBLE
  - int ↔ INTEGER
  - float ↔ REAL
  - ...

Andere opdrachten uitvoeren

UNIVERSITEIT GENT

- delete, update, insert, DDL (Data Definition Language)

```
try (Connection conn = geefVerbinding();
      Statement stmt = conn.createStatement()) {

  // opdracht uitvoeren
  String opdr = “create table temp (nr numeric)”;
  stmt.executeUpdate(opdr);

  String insertSql = “insert into temp values (1)”;
  String updateSql = “update temp set nr=2 where nr=1”;
  stmt.executeUpdate(insertSql);
  int updateCnt = stmt.executeUpdate(updateSql);
}
```


Overzicht Statement

```
try (
    Connection connection = DriverManager.getConnection(…,…,… );
    Statement stmt = conn.createStatement()
  ) {

  // delete-, insert-, update-opdracht of DDL
  int aantRijenAangepast = stmt.executeUpdate(…);

  // select-opdracht
  ResultSet rs = stmt.executeQuery(…);
  while (rs.next()) {
    … = rs.getXXX(kolomNaam);
    … = rs.getXXX(kolomNummer);
  }
}
```

Verschillende stappen

- Configuratiegegevens inlezen
- Driver laden (optioneel vanaf JDK 1.6)
- Verbinding aanmaken
- Statement aanmaken
  - “gewoon”
  - PreparedStatement (parameters)
  - CallableStatement (stored procedure)
- Statement uitvoeren → eventueel ResultSet overlopen

Overzicht

UNIVERSITEIT GENT

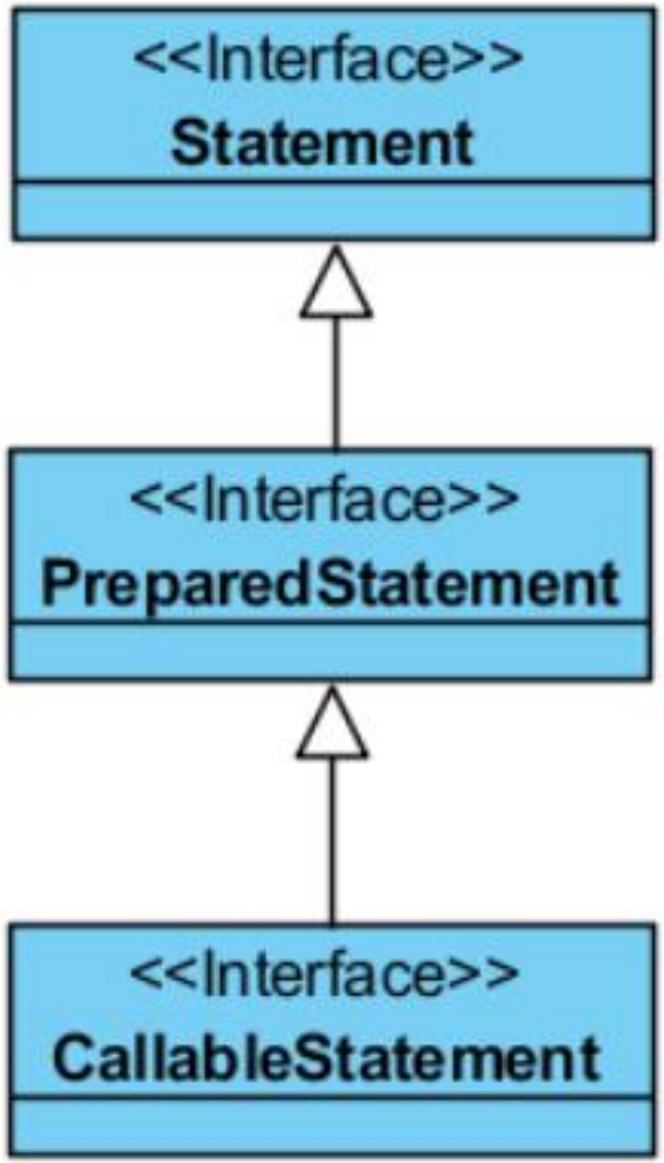
- Wat is JDBC?
- Basiswerking
  - Statements
  - PreparedStatements
  - CallableStatements
  - Transacties
- JDBC uitgebreid
  - DataSource
  - Scrollable en aanpasbare ResultSet
  - RowSets
  - SQL3-gegevenstypes

Prepared Statements

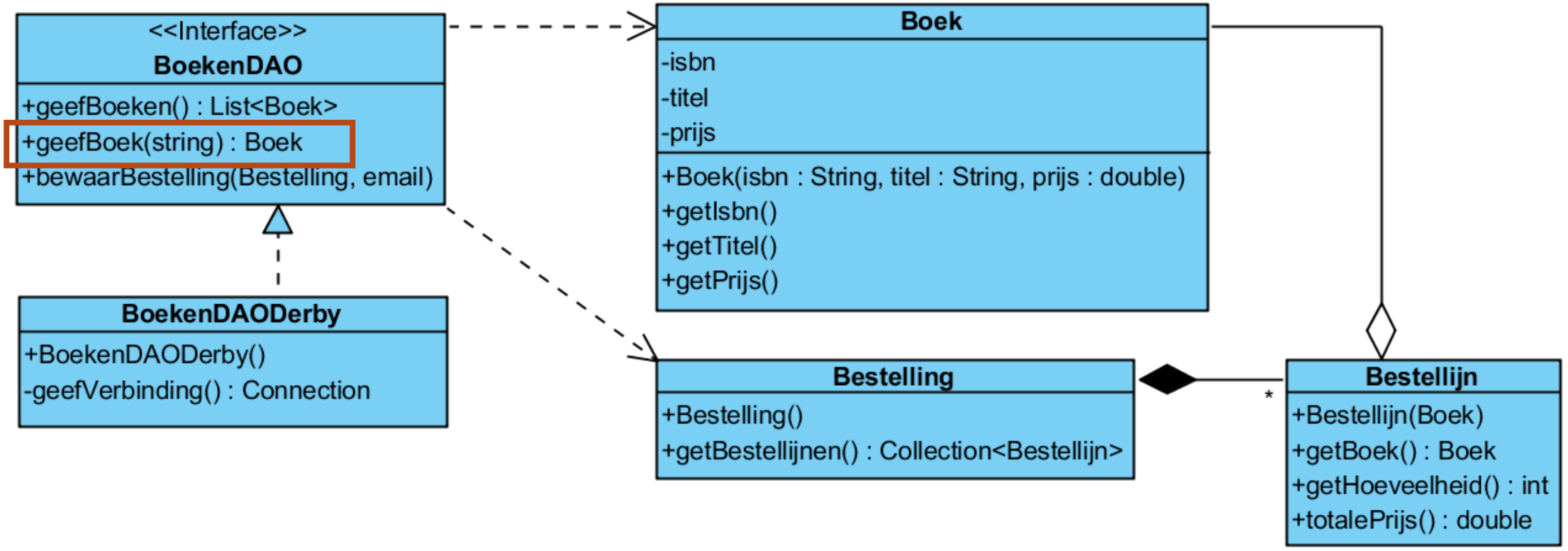
- SQL-opdrachten met parameters
- Voorgecompileerd in gegevensbank
- Opdrachten die verschillende keren uitgevoerd moeten worden
- Reductie uitvoertijd
- Veiligheid
  - Gegevensconversie
  - Voorkomen SQL-injectie

Prepared Statements

- Werkwijze
  - SQL-opdracht: parameters = ?
  - PreparedStatement aanmaken
  - Parameters invullen
  - Uitvoeren, eventueel resultaat ophalen



Voorbeeld


<<Interface>>
BoekenDAO
+geefBoeken() : List<Boek>
+geefBoek(string) : Boek
+bewaarBestelling(Bestelling, email)
Boek
-isbn
-titel
-prijs
+Boek(isbn : String, titel : String, prijs : double)
+getIsbn()
+getTitel()
+getPrijs()
BoekenDAODerby
+BoekenDAODerby()
-geefVerbinding() : Connection
Bestelling
+Bestelling()
+getBestellijnen() : Collection<Bestellijn>
*
Bestellijn
+Bestellijn(Boek)
+getBoek() : Boek
+getHoeveelheid() : int
+totalePrijs() : double

Voorbeeld

UNIVERSITEIT GENT

Tabel BOEKEN

ISBN	TITEL	PRIJS
isbn19789043025256	jQuery - de basis	12
isbn2	aan de slag met HTML5	15
isbn3	Head First Design Patterns	45
isbn1	Thinking in Java	58
isbn4	Java Servlet Programming	36
isbn5	Java Server Programming	81
isbn6	The Java Tutorial	65
isbn7	Java Swing	55

Voorbeeld PreparedStatement

UNIVERSITEIT GENT

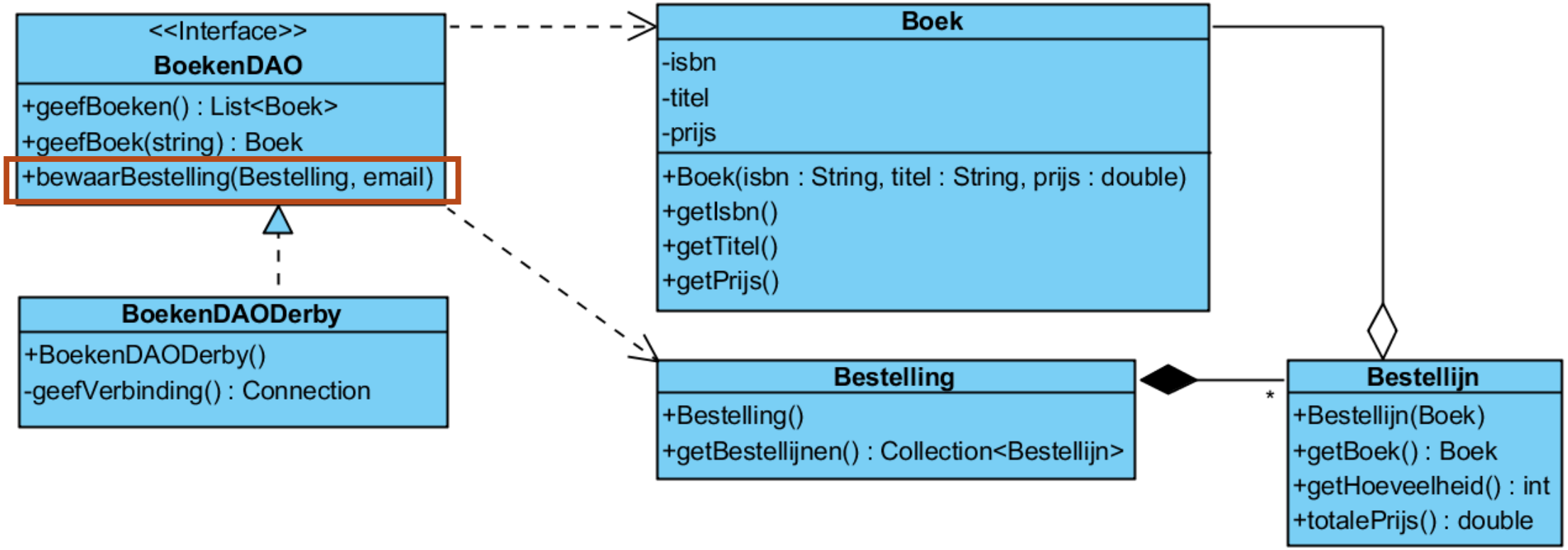
```
public Boek geefBoek(String isbn) throws BoekenNietBeschikbaarExceptie {
  Boek boek = null;
  try {
    try (Connection conn = geefVerbinding();
      PreparedStatement stmt = conn.prepareStatement(sqlOpdrachten.getString("Q_BOEK"))) {

        stmt.setString(1, isbn);
        ResultSet rs = stmt.executeQuery();
        if (rs.next()) {
          boek = new Boek(
            rs.getString(sqlOpdrachten.getString("BOEK_ISBN")),
            rs.getString(sqlOpdrachten.getString("TITEL")),
            rs.getDouble(sqlOpdrachten.getString("PRIJS")));
        }
    }
  } catch (SQLException e) { … }
  return boek;
}
```

Voorbeeld PreparedStatement

sql.properties

```
# boek opvragen aan de hand van isbnnummer
Q_BOEK = select * from boeken where isbn = ?
# kolomnamen
BOEK_ISBN = isbn
TITEL = titel
PRIJS = prijs
```

Voorbeeld


<<Interface>>
BoekenDAO
+geefBoeken() : List<Boek>
+geefBoek(string) : Boek
+bewaarBestelling(Bestelling, email)
Boek
-isbn
-titel
-prijs
+Boek(isbn : String, titel : String, prijs : double)
+getIsbn()
+getTitel()
+getPrijs()
BoekenDAODerby
+BoekenDAODerby()
-geefVerbinding() : Connection
Bestelling
+Bestelling()
+getBestellijnen() : Collection<Bestellijn>
*
Bestellijn
+Bestellijn(Boek)
+getBoek() : Boek
+getHoeveelheid() : int
+totalePrijs() : double

Voorbeeld PreparedStatement

Tabel BESTELLINGEN

AANTAL	KLANT	ISBNBOEK
4	veerle@hogent.be	isbn2
1	veerle@hogent.be	isbn3
1	test@test.be	isbn2
1	vo@gent.be	isbn5
2	vo@gent.be	isbn3
2	veerle@tiwi.be	isbn4
1	veerle@tiwi.be	isbn7

SQL-opdrachten

```
Q_BESTELLING = select aantal from bestellingen where klant = ? and isbnboek = ?
AANTAL = aantal
I_BESTELLING = insert into bestellingen (klant, isbnboek, aantal) values (?,?,?)
U_BESTELLING = update bestellingen set aantal = ? where klant = ? and isbnboek = ?
```

```
public void bewaarBestelling(Bestelling bestelling, String email) throws BestellingMisluktExceptie {
  try (Connection conn = geefVerbinding();
       PreparedStatement bestaatBestel = conn.prepareStatement(sqlOpdrachten.getString("Q_BESTELLING")))
{
  PreparedStatement voegBestelToe = conn.prepareStatement(sqlOpdrachten.getString("I_BESTELLING"));
  PreparedStatement pasBestelAan = conn.prepareStatement(sqlOpdrachten.getString("U_BESTELLING"));

  bestaatBestel.setString(1, email);
  voegBestelToe.setString(1, email);
  pasBestelAan.setString(2, email);
  for (Bestellijn lijn : bestelling.getBestellijnen()) {
    bestaatBestel.setString(2, lijn.getBoek().getIsbn());
    ResultSet bestaandeBestelling = bestaatBestel.executeQuery();
    if (bestaandeBestelling.next()) {
      int aantal = bestaandeBestelling.getInt(sqlOpdrachten.getString("AANTAL"));
      pasBestelAan.setString(3, lijn.getBoek().getIsbn());
      pasBestelAan.setInt(1, aantal + lijn.getHoeveelheid());
      pasBestelAan.executeUpdate();
    } else {
      voegBestelToe.setString(2, lijn.getBoek().getIsbn());
      voegBestelToe.setInt(3, lijn.getHoeveelheid());
      voegBestelToe.executeUpdate();
    }
  }
} catch (Exception e) {
  System.out.println(e.getMessage()); throw new BestellingMisluktExceptie(e);
}
}
```

Parameter

- SQL-opdracht met parameter uitvoeren
- Herhaaldelijk uitvoer van dezelfde opdracht met andere waarden
- Gegevensconversie
- Voorkomen SQL-injectie

Gegevensconversie

- Aanhalingsteken in naam

```
String naam = “D’Haese”;
```

- SQL-opdracht met knippen en plakken

```
String opdracht = “select * from studenten where naam = \’” + naam + “\’”;
```

- SQL-opdracht met parameter

```
String opdracht = “select * from studenten where naam = ?”;
```

Gegevensconversie

- Aanhalingsteken in naam
  - Knippen en plakken geeft fout
- Gebruik Parameter
  - Opdracht wordt gecompileerd met parameter
  - Bij uitvoeren
    - Parameters doorsturen
    - Parameters worden geconverteerd naar juiste type
    - Parameters zijn argumenten voor gecompileerde functie

SQL-injectie

- Meestal in een webapplicatie
- Gebruiker
  - Parameter ingeven
  - Combineert parameter met SQL-opdracht
    - ➢ Injecteert SQL-opdrachten
      - Niet-toegankelijke gegevens bekijken
      - Gegevens verwijderen
      - …

SQL-injectie: voorbeeld

UNIVERSITEIT GENT

- Ingave gebruiker: studentennummer
- SQL-opdracht

```
String opdracht =  “select * from studenten where studnr = ” + nummer;
```

- Malafide ingave

```
•  200200234 OR 1=1
•  200200234; DROP TABLE studenten
•  0 UNION SELECT username, password FROM users
```


SQL-injectie voorkomen → gebruik parameters!

UNIVERSITEIT GENT

- Opdracht wordt gecompileerd

```
select * from studenten where studnr = ?
```

- Instellen parameter converteert invoer naar een VARCHAR
- Deze VARCHAR is een parameter voor de voorgecompileerde SQL-opdracht

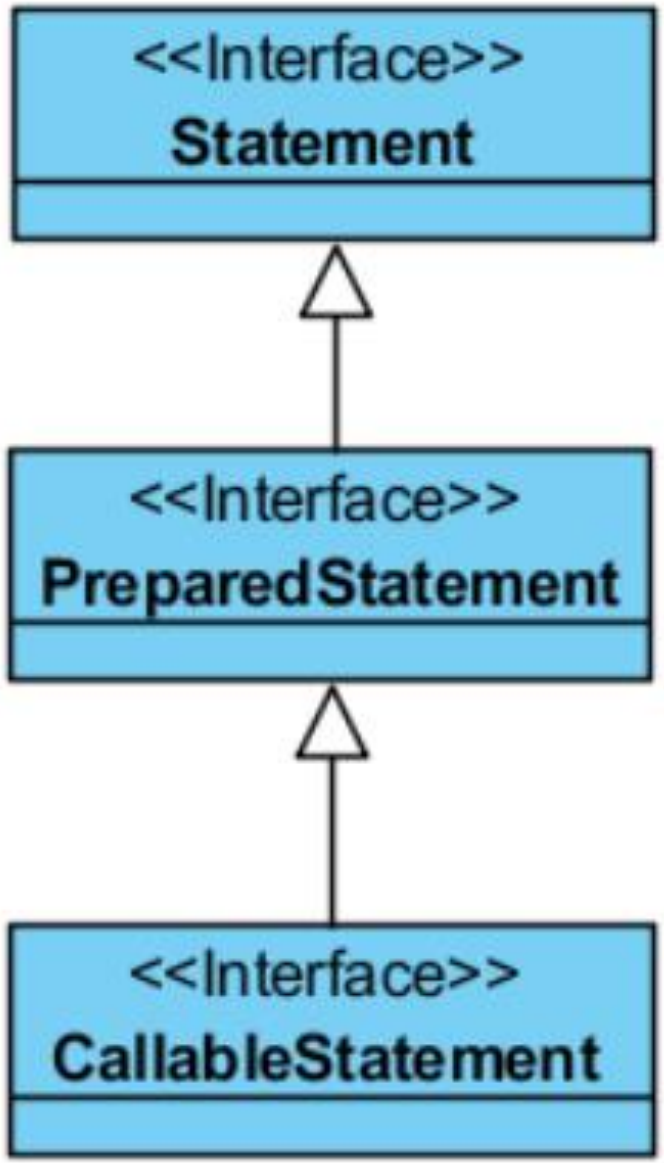
Overzicht

UNIVERSITEIT GENT

- Wat is JDBC?
- Basiswerking
  - Statements
  - PreparedStatements
  - CallableStatements
  - Transacties
- JDBC uitgebreid
  - DataSource
  - Scrollable en aanpasbare ResultSet
  - RowSets
  - SQL3-gegevenstypes

Prepared Statements

- Uitvoeren stored procedure
- Stored procedure
  - Functie of procedure
  - In gegevensbank
  - Geen standaardsyntax
  - Niet door alle systemen ondersteund
- JDBC API
  - Uniforme manier



Callable Statement

UNIVERSITEIT GENT

- Nodig
  - Naam procedure (bv. SELECT_BOEKEN)
  - Parameters
- Aanmaken CallableStatement

```
String opdracht = “{call SELECT_BOEKEN}”;
CallableStatement cstmt = conn.prepareCall(opdracht);
```

- Escape-syntax
  - Geen standaard SQL
  - Driver moet interpreteren of omzetten
- Verwijzing naar procedure in de databank

Voorbeeld CallableStatement

```
String opdracht = “{call BEWAAR_BESTEL(?,?,?)}”;

try (Connection conn = geefVerbinding();
      CallableStatement cstmt = conn.prepareCall(opdracht)) {

  // parameters instellen
  cstmt.setString(1,…);
  cstmt.setString(2,…);
  cstmt.setInt(3,…);
  cstmt.executeUpdate();

}
```

Uitvoerparameters

- Corresponderen met uitvoerparameters van de stored procedure
- Methode
  - ? in de SQL-opdracht
  - Registeren
- Voorbeeld Opdracht
  - ? : aantal boeken

```
String opdracht = “{? = call SELECT_BOEKEN}”;
String opdracht = “{call SELECT_BOEKEN(?)}”;
```

Registreren uitvoerparameter

UNIVERSITEIT GENT

- Opmerking
  - Eerst ResultSet overlopen
  - ? : bepaalt NIET het type parameter (in- of uit-)

```
CallableStatement cstmt = conn.prepareCall(opdracht);
cstmt.registerOutParameter(1, java.sql.Types.INTEGER);
ResultSet rs = cstmt.executeQuery();
… // boeken ophalen
int aantalBoeken = cstmt.getInt(1);
```


Overzicht

UNIVERSITEIT GENT

- Wat is JDBC?
- Basiswerking
  - Statements
  - PreparedStatements
  - CallableStatements
  - Transacties
- JDBC uitgebreid
  - DataSource
  - Scrollable en aanpasbare ResultSet
  - RowSets
  - SQL3-gegevenstypes

Transacties

- Verschillende rijen en/of tabellen tegelijkertijd aanpassen
- Consistente gegevens
- Data integrity
  - Andere gebruikers zien pas aangepaste waarden als ze “definitief” zijn

Transacties

- Principe
  - Auto-commit-mode uitschakelen
  - Opdrachten uitvoeren
    - Geslaagd: commit
    - Mislukt: rollback
  - Auto-commit-mode inschakelen
- Auto-commit-mode
  - Standaard aan

Transacties

UNIVERSITEIT GENT

```
Connection conn = …;
PreparedStatement stmt = …;
try {
  stmt.setString(1,email);
  conn.setAutoCommit(false);
  for (Bestellijn lijn : bestelling.geefBestellijnen()) {
    stmt.setString(2,lijn.getBoek().getId());
    stmt.setInt(3,lijn.getHoeveelheid());
    stmt.executeUpdate();
  }
  conn.commit();
} catch (SQLException e) {
  conn.rollback();
} finally {
  conn.setAutoCommit(true);
}
```

Opmerkingen

- Let op: `Connection`-object lokaal
- Verschillende transacties
  - Verschillende `Connection`-objecten

Overzicht

UNIVERSITEIT GENT

- Wat is JDBC?
- Basiswerking
  - Statements
  - PreparedStatements
  - CallableStatements
  - Transacties
- JDBC uitgebreid
  - DataSource
  - Scrollable en aanpasbare ResultSet
  - RowSets
  - SQL3-gegevenstypes

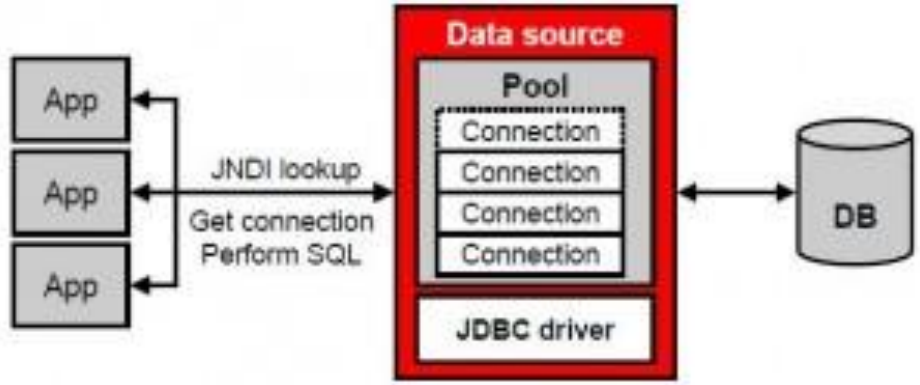
Driver versus DataSource

UNIVERSITEIT GENT

- java.sql
  - Gebruik Driver
- javax.sql
  - Gebruik DataSource
- DataSource
  - Applicatieservers of frameworks zoals Spring
  - Representeert een DBMS
  - Aangemaakt en ter beschikking gesteld via configuratie
  - Beschikbaar via JNDI (Java Naming and Directory Interface) of Dependency Injection

DataSource

- Beter alternatief voor JDBC-driver
  - Geen driverklasse, gebruikersnaam en wachtwoord in code
  - Gebruik connection pool
  - Ondersteunt gedistribueerde transacties
- Steunt op ofwel
  - JNDI: Java naming en directory services
    - ➢ Naming service: naam ~ object
    - ➢ Directory service uitbreiding naming service
      - Object heeft attributen
      - Zoeken op objecten mogelijk
  - Dependency Injection



bron: http://jianmingli.com/wp/?p=5286

Gebruik DataSource

UNIVERSITEIT GENT

- In applicatieserver bv. Glassfish
- Via framework bv. Spring

- Zorgen voor connection pool

Configuratie DataSource

UNIVERSITEIT GENT

- In configuratiebestand (Spring: application.properties)
  - JDBC-URL
  - Login
  - Wachtwoord

```
spring.datasource.url=jdbc:postgresql://localhost:5432/iii
#spring.datasource.driverClassName=org.postgresql.Driver
spring.datasource.username=iii
spring.datasource.password=iiipwd
```

- Nodige drivers ter beschikking stellen bv. via pom.xml

Gebruik DataSource

UNIVERSITEIT GENT

- Ophalen via JNDI of Dependency Injection

```
    private DataSource dataSource;

    @Autowired
    public void setDataSource(DataSource dataSource) {
        this.dataSource = dataSource;
    }
```