

UNIVERSITÄT LEIPZIG  
LPZ E-BUSINESS  
*applied telematics*

## Übungen zum Java-Praxiskurs für Fortgeschrittene

Matthias Book  
Malte Hülder

Lehrstuhl für Angewandte Telematik / e-Business WS 2005/2006

UNIVERSITÄT LEIPZIG  
LPZ E-BUSINESS  
*applied telematics*

### Wiederholung: JDBC

- Verbindung zur Datenbank aufbauen  
`java.sql.Connection con = DriverManager.getConnection(url, user, pwd);`
- SQL-Anweisung erzeugen  
`Statement stmt = con.createStatement();  
String sql = "SELECT * FROM tabelle";`
- SQL-Anweisung ausführen  
`ResultSet rs = stmt.executeQuery(sql);`
- Ergebnis der SQL-Anweisung verarbeiten  
`while (rs.next()) {  
    System.out.println(rs.getString("spalte"));  
}`
- Datenbankverbindung schließen  
`con.close();`

Der Java-Praxiskurs II 2

UNIVERSITÄT LEIPZIG  
LPZ E-BUSINESS  
*applied telematics*

### Wiederholung: SQL

- `CREATE TABLE name ( columnDefinition [, ...] [, constraintDefinition...]);`
- `INSERT INTO table [(column[,...])] { VALUES (Expression[,...]) | SelectStatement};`
- `SELECT [{LIMIT n m | TOP m}][DISTINCT] { selectExpression | table.* | * } [, ... ] FROM tableList [ WHERE Expression ];`
- `UPDATE table SET column = Expression [, ...] [WHERE Expression];`
- `DELETE FROM table [ WHERE Expression ] ;`

Der Java-Praxiskurs II 3

UNIVERSITÄT LEIPZIG  
LPZ E-BUSINESS  
*applied telematics*

### Wiederholung: OR-Mapping

Problem:

- Wie lege ich die Daten meiner Objekte in der Datenbank ab?

Buch

Autoren  
Titel  
Verlag  
Jahr

Tabelle: buch

autoren	titel	verlag	jahr
Sami Beydeda	The STECC Method	mpress Verlag	2004
Ernst-Erich Doberkat, Stefan Dissman	Einführung in die OOP mit Beta		1996

Der Java-Praxiskurs II 4

UNIVERSITÄT LEIPZIG  
LPZ E-BUSINESS  
*applied telematics*

### Wiederholung: OR-Mapping

Tabelle: buch				
benutzer_id	autoren	titel	verlag	jahr
	Sami Beydeda	The STECC Method	mpress Verlag	2004
2	Ernst-Erich Doberkat, Stefan Dissman	Einführung in die OOP mit Beta		1996

Tabelle: benutzer		
id	name	vorname
2	Schöpe	Lothar

- Durch den Fremdschlüssel `benutzer_id` kann eindeutig referenziert werden, welcher Benutzer (2) welches Buch bzw. welche Bücher ausgeliehen hat
- `SELECT * FROM buch WHERE benutzer_id=2`

Der Java-Praxiskurs II 5

UNIVERSITÄT LEIPZIG  
LPZ E-BUSINESS  
*applied telematics*

### Wiederholung: OR-Mapping

Tabelle: buch				
id	autoren	titel	verlag	jahr
1	Sami Beydeda	The STECC Method	mpress Verlag	2004
2	Ernst-Erich Doberkat, Stefan Dissman	Einführung in die OOP mit Beta		1996

Tabelle: vormerkung			
datum	buch_id	benutzer_id	
29.09.2004	1	2	

Tabelle: benutzer		
id	name	vorname
2	Schöpe	Lothar

- Durch die Fremdschlüssel `benutzer_id` und `buch_id` kann eindeutig referenziert werden, welcher Benutzer (2) welches Buch (1) vorgemerkt hat
- `SELECT * FROM vormerkung WHERE benutzer_id=2 AND buch_id=1`

Der Java-Praxiskurs II 6

## Wiederholung: Transaktionen

- Transaktionen müssen ACID-Eigenschaften haben:
  - Atomicity: Sie sind eine untrennbare Einheit
  - Consistency: Sie führen zu gültigen Zuständen
  - Independency: Sie sind unabhängig von anderen Transaktionen
  - Durability: Das Ergebnis einer Transaktion ist dauerhaft
- Transaktionssteuerung mit JDBC:
  - Mit `setAutoCommit(false)` AutoCommit abschalten
  - Transaktion entweder mit `commit()` bestätigen oder
  - mit `rollback()` Änderungen rückgängig machen, weil ein Fehler aufgetreten ist.

Der Java-Praxiskurs II

7

## Ü3.1 Anlegen der Datenbanktabellen

- Legen Sie die notwendigen Tabellen für die Klassen der Bibliotheksverwaltung an. Implementieren Sie dazu in den vorgegebenen Klassen die `createTable()`-Methoden, sodass diese eine Verbindung zur Datenbank aufbauen und die nötigen SQL-Statements absetzen.
- Schreiben Sie eine Klasse, die die `createTable()`-Methoden der anderen Klassen aufruft, um die Datenbanktabellen zu erzeugen.
- Sollten Sie Tabellen nicht ganz korrekt angelegt haben und/oder es erneut versuchen wollen, sollten Sie die Tabellen zunächst mit `DROP TABLE` löschen, bevor Sie sie erneut anlegen können. Implementieren Sie diese Funktionalität am besten in den jeweiligen `dropTable()`-Methoden.

Der Java-Praxiskurs II

8

## Buch.createTable()

```
public static void createTable(){
    String sql = "CREATE TABLE "+TAB_BUCH+" (" + COL_SIGNATUR+
    "VARCHAR(100) NOT NULL PRIMARY KEY, ";
    try {
        Datenbank hsql = new Datenbank();
        Connection con = hsql.getConnection();
        Statement stmt = con.createStatement();
        sql+= COL_TITEL+" VARCHAR(1000) NOT NULL, ";
        sql+= COL_ANZAHLEXEMPLARE+" INT DEFAULT 1 NOT NULL, ";
        sql+= COL_AUSLEIHERID+" INT, ";
        sql+= COL_AUSLEIHDATUM+" DATE, ";
        sql+= COL RUECKGABEDATUM+" DATE, ";
        sql+= COL_AUTOR+" VARCHAR(255), ";
        sql+= COL_VERLAG+" VARCHAR(255), ";
        sql+= COL_ERSCHEINUNGSJAHR+" INT, ";
        sql+= COL_ISBN+" VARCHAR(10)";
        stmt.execute(sql);
        con.close();
    } catch (Exception e) {
        System.err.println("Fehler mit Statement: "+sql);
        e.printStackTrace();
    }
}
```

Der Java-Praxiskurs II

9

## Benutzer.createTable()

```
public static void createTable(){
    String sql = "CREATE TABLE "+TAB_BENUTZER+" (" +
    COL_ID+" INT NOT NULL PRIMARY KEY, ";
    try {
        Datenbank hsql = new Datenbank();
        Connection con = hsql.getConnection();
        Statement stmt = con.createStatement();
        sql+= COL_NACHNAME+" VARCHAR(200), ";
        sql+= COL_VORNAME+" VARCHAR(200), ";
        sql+= COL_PASSWORT+" VARCHAR(200)";
        stmt.execute(sql);
        con.close();
    } catch (Exception e) {
        System.err.println("Fehler mit Statement: "+sql);
        e.printStackTrace();
    }
}
```

Der Java-Praxiskurs II

10

## dropTable() in Buch und Benutzer

```
public static void dropTable(){
    String sql = "DROP TABLE "+TAB_BUCH;
    // String sql = "DROP TABLE "+TAB_BENUTZER;
    try{
        Datenbank hsql = new Datenbank();
        Connection con = hsql.getConnection();
        Statement stmt = con.createStatement();
        stmt.execute(sql);
        con.close();
    } catch (Exception e) {
        System.err.println("Fehler mit Statement: "+sql);
        e.printStackTrace();
    }
}
```

Der Java-Praxiskurs II

11

## Die Klasse DBSetup

```
public class DBSetup {
    public static void main(String[] args) {
        try {
            //Benutzer.dropTable();
            Benutzer.createTable();
            //Buch.dropTable();
            Buch.createTable();
            //Vormerkung.dropTable();
            Vormerkung.createTable();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Der Java-Praxiskurs II

12

### Ü3.2 Anlegen von Objekten

1. Erzeugen Sie zunächst ein neues Benutzer-Objekt mit allen Attributen und speichern Sie es durch Aufruf der `insert()`-Methode ab. Merken Sie sich das Passwort für spätere Aufgaben!

```
public class BenutzerSetup {
    public static void main(String[] args) {
        try {
            Benutzer benutzer = new Benutzer();
            benutzer.setId(1);
            benutzer.setNachname("Hülder");
            benutzer.setVorname("Malte");
            benutzer.setPasswort("javakurs");
            benutzer.insert();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

### Ü3.2 Anlegen von Objekten

2. Ergänzen Sie die `insert()`-Methode der Klasse `Medium` so, dass sie vor dem Aufruf von `constructKeysValues` (`String keys`, `String vals`) in den Strings `keys` und `values` die notwendigen Spaltennamen bzw. Werte der Attribute der `Medium`-Klasse zusammenstellt. Nach dem Aufruf der Methode `constructKeysValues` (`String keys`, `String vals`) soll das Rückgabe-Array an Position 0 die `keys` enthalten und an Position 1 die `values`. Lesen Sie diese Variablen also nach dem Aufruf der Methode wieder ein und bauen Sie aus den beiden Strings ein SQL-Statement, das Sie über eine Datenbankverbindung absetzen.

### Ü3.2 Anlegen von Objekten

```
public Medium insert(){
    String sql = "";
    try {
        String keys = "INSERT INTO "+getTable()+" (";
        String values = "VALUES(";
        if (this.signatur!=null) {
            keys+= COL_SIGNATUR+", ";
            values+= ""+this.signatur+", ";
        }
        if (this.titel!=null) {
            keys+= COL_TITEL+", ";
            values+= ""+this.titel+", ";
        }
        keys+= COL_ANZAHEXEMPLARE+", ";
        values+= this.anzahlExemplare+", ";
        String[] kv = constructKeysValues(keys, values);
        // Fortsetzung nächste Folie
    }
}
```

### Ü3.2 Anlegen von Objekten

```
keys = kv[0];
values = kv[1];
if (keys.endsWith(", ")){
    keys = keys.substring(0,keys.length()-2);
}
if (values.endsWith(", ")){
    values = values.substring(0,values.length()-2);
}
sql = keys+") "+values+");";
Datenbank hsql = new Datenbank();
Connection con = hsql.getConnection();
Statement stmt = con.createStatement();
stmt.execute(sql);
con.close();
} catch (Exception e) {
    System.err.println("Fehler mit Statement: "+sql);
    e.printStackTrace();
}
return this; }
```

### Ü3.2 Anlegen von Objekten

- Nun müssen Sie in den Unterklassen die Methode `constructKeysValues(String keys, String vals)` überschreiben.

```
protected String[] constructKeysValues(String keys, String vals) {
    if (this.getAusleiher()!=null){
        keys+= COL_AUSLEIHERID+", "; //Fremdschlüssel
        vals+= this.ausleiherId+", ";
    }
    if (this.ausleihDatum!=null){
        keys+= COL_AUSLEIHDATUM+", ";
        vals+= ""+dateFormat.format(ausleihDatum)+", ";
    }
    if (this.rueckgabeDatum!=null){
        keys+= COL_RUECKGABEDATUM+", ";
        vals+= ""+dateFormat.format(rueckgabeDatum)+", ";
    }
    return new String[] { keys, vals };
}
```

### Ü3.2 Anlegen von Objekten

```
• Buch:
protected String[] constructKeysValues(String keys, String vals) {
    String[] kv = super.constructKeysValues(keys, vals);
    keys = kv[0];
    vals = kv[1];
    if (this.autor!=null){
        keys+= COL_AUTOR+", ";
        vals+= ""+this.autor+", ";
    }
    if (this.verlag!=null){
        keys+= COL_VERLAG+", ";
        vals+= ""+this.verlag+", ";
    }
    keys+= COL_ERSCHEINUNGSJAHR+", ";
    vals+= this.erscheinungsjahr+", ";
    if (this.isbn!=null){
        keys+= COL_ISBN;
        vals+= ""+this.isbn+"";
    }
    return new String[] { keys, vals };
}
```

**Ü3.2 Anlegen von Objekten** UNIVERSITÄT LEIPZIG  
LPZ E-BUSINESS

3. Implementieren Sie die Methode `update()` in der Klasse `Medium`.

➤ Lösung: siehe Quelltext

Der Java-Praxiskurs II 19

**Ü3.2 Anlegen von Objekten** UNIVERSITÄT LEIPZIG  
LPZ E-BUSINESS

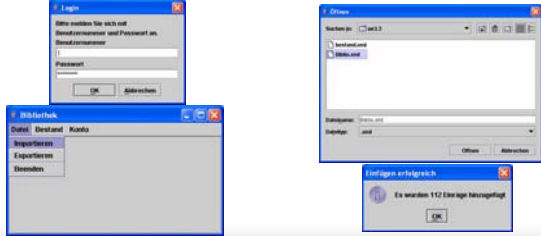
- Implementieren Sie die Methode `String constructUpdatesql(String sql)` in den Unterklassen
- Prüfen, ob Attribut `null` ist → wenn ja, `NULL` in die DB schreiben

➤ Lösung: siehe Quelltext

Der Java-Praxiskurs II 20

**Ü3.3 XML ⇔ DB-Konverter** UNIVERSITÄT LEIPZIG  
LPZ E-BUSINESS

1. Lassen Sie die Buchobjekte aus der XML-Datei `Biblio.xml` einlesen und in der Datenbank abspeichern. Sie können dazu den Menüpunkt „Datei | importieren“ aufrufen. Dabei werden die in Aufgabe 2 von Ihnen implementierten `insert()`-Methoden benutzt.



Der Java-Praxiskurs II 21

**Ü3.3 XML ⇔ DB-Konverter** UNIVERSITÄT LEIPZIG  
LPZ E-BUSINESS

2. Erweitern Sie dazu die entsprechenden Klassen um die notwendigen Methoden, ihre Attribute aus einem `ResultSet` der Datenbank zu lesen.

- `Medium`:
 

```
public Medium(ResultSet rs) throws
SQLException {
    this.signatur = rs.getString(COL_SIGNATUR);
    this.titel = rs.getString(COL_TITEL);
    this.anzahlExemplare =
        rs.getInt(COL_ANZAHEXEMPLARE);
}
```

Der Java-Praxiskurs II 22

**Ü3.3 XML ⇔ DB-Konverter** UNIVERSITÄT LEIPZIG  
LPZ E-BUSINESS

- `AusleihbaresMedium`:
 

```
public AusleihbaresMedium(ResultSet rs) throws
SQLException {
    super(rs);
    this.ausleiherId =
        rs.getInt(COL_AUSLEIHERID);
    this.ausleihDatum =
        rs.getDate(COL_AUSLEIHDATUM);
    this.rueckgabeDatum =
        rs.getDate(COL_RUECKGABEDATUM);
}
```

Der Java-Praxiskurs II 23

**Ü3.3 XML ⇔ DB-Konverter** UNIVERSITÄT LEIPZIG  
LPZ E-BUSINESS

- `Buch`:
 

```
public Buch(ResultSet rs) throws SQLException{
    super(rs);
    this.autor = rs.getString(COL_AUTOR);
    this.erscheinungsjahr =
        rs.getInt(COL_ERSCHEINUNGSJAHR);
    this.isbn = rs.getString(COL_ISBN);
    this.verlag = rs.getString(COL_VERLAG);
}
```

Der Java-Praxiskurs II 24

### Ü3.4 Suchfunktionalität

- Schreiben Sie eine Methode (`findByStichwort(String stichwort)`) in der Klasse `Medium`, mit der Sie nach einem Teil eines Buchtitels (Stichwort) suchen können.

```
public Collection findByStichwort(String stichwort){
    Vector al = new Vector();
    try{
        Datenbank hsql = new Datenbank();
        Connection con = hsql.getConnection();
        Statement stmt = con.createStatement();
        String sql = "SELECT * FROM "+getTable()+
            " WHERE "+COL_TITEL+" LIKE '%"+stichwort+"%'";
        ResultSet rs = stmt.executeQuery(sql);
        while (rs.next()) {
            Medium m = constructFromResultSet(rs);
            al.add(m);
        }
        con.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
    return (Collection)al;
}
```

### Ü3.5 Ausleihe und Rückgabe

- Implementieren Sie die Methoden in der Klasse `AusleihbaresMedium`, mit denen Bücher ausgeliehen, verlängert, zurückgegeben und vorgemerkt werden können. Wenn sich der Leihstatus eines Buches ändert, soll dieser auch in der Datenbank aktualisiert werden.

- Bei jeder Änderung die Methode `update()` aufrufen.

➤ Lösung: siehe Quelltext

### Ü3.6 Leihübersicht

- Implementieren Sie die Methode `findByAusleiher(Benutzer ausleiher)` in der Klasse `AusleihbaresMedium`, mit der es möglich ist, alle Bücher zu finden, die ein Benutzer ausgeliehen hat.

➤ Lösung: siehe Quelltext