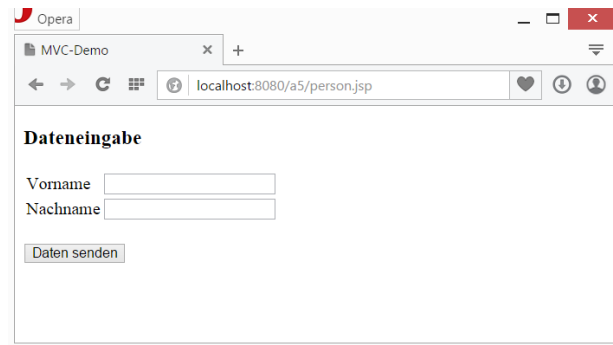


Internetprogrammierung

Übungsblatt 11

1) Eine einfache Anwendung nach dem MVC-Prinzip

Auf der Web-Seite finden Sie eine kleine Anwendung, die das MVC-Pattern realisiert. Die Anwendung besteht nur aus einer JSP und einem Servlet. Der Einstiegspunkt in die Anwendung ist die JSP, die direkt im Browser aufgerufen werden kann.



The screenshot shows a web browser window with the title 'MVC-Demo'. The address bar displays 'localhost:8080/a5/person.jsp'. The main content area contains a form titled 'Dateneingabe'. Inside the form, there are two text input fields labeled 'Vorname' and 'Nachname'. Below these fields is a button labeled 'Daten senden'.

Das Formular sendet die Eingabedaten an ein Servlet, das ein Personenobjekt erzeugt. Das erzeugte Objekt wird dann mit Hilfe derselben JSP wieder angezeigt, wobei alle Eingaben in Großbuchstaben umgewandelt werden.

Analysieren Sie das aus der JSP erzeugte Servlet. Der generierte Code finden Sie im Eclipse-Worspace im Verzeichnis:

```
\Workspace\.metadata\.plugins\org.eclipse.wst.server.core\tmp0\work\Catalina
```

Wie wird das `useBean`-Tag im Code umgesetzt? Wie wird der erzeugte Code geändert, wenn Sie `class` durch `type` ersetzen?

2) Refactoring der Notiz-Anwendung

Überführen Sie nun Ihre Notizanwendung in eine MVC-Architektur, wobei lediglich das `useBean`- und `getProperty`-Tag auf der JSP benutzt werden soll. Damit alle Notizen ausgegeben werden können, kann man dem NotizContainer eine Methode `getAsHTML` hinzufügen, der eine HTML-codierte Tabelle liefert. Dieses Property kann dann auf der JSP zur Anzeige benutzt werden.

Internetprogrammierung Übungsblatt 11

3) Einrichten einer MySQL-Datenbak

Falls Sie xampp installiert haben, starten Sie über das Control-Panel den Apache-Web-Server und die MySQL-Datenbank (vgl. Abb. 1).

Danach können Sie über den Link

<http://127.0.0.1/phpmyadmin/>

die Administrationsanwendung aufrufen (vgl. Abb. 2).

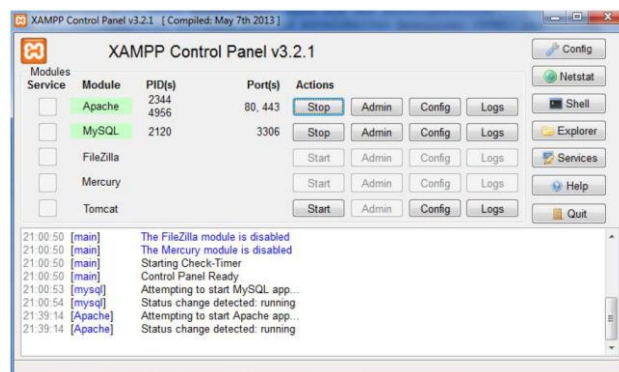


Abbildung 1

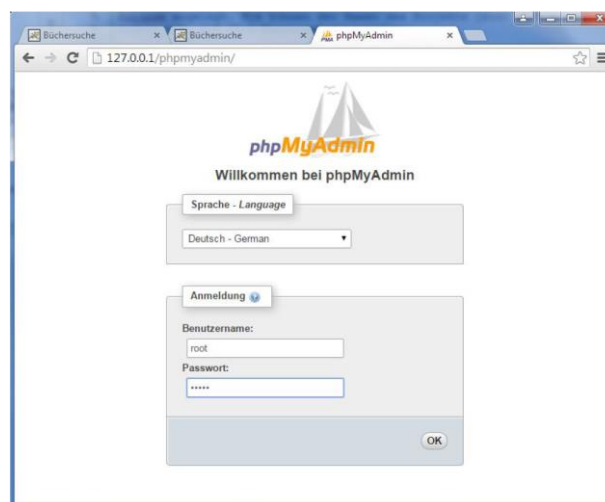


Abbildung 2

Nach Eingabe von User und Passwort (hier root und admin) gelangt man zur Administrationsanwendung. Legen Sie nun ein Datenbankschema mit dem Namen *buchlager* an.

Internetprogrammierung

Übungsblatt 11

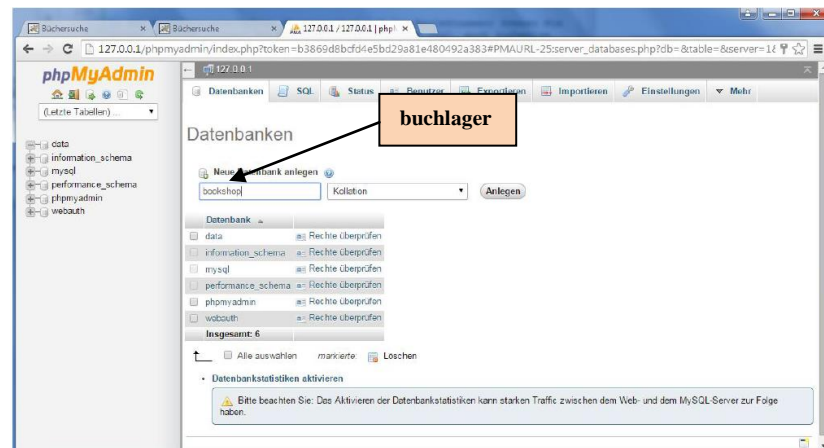
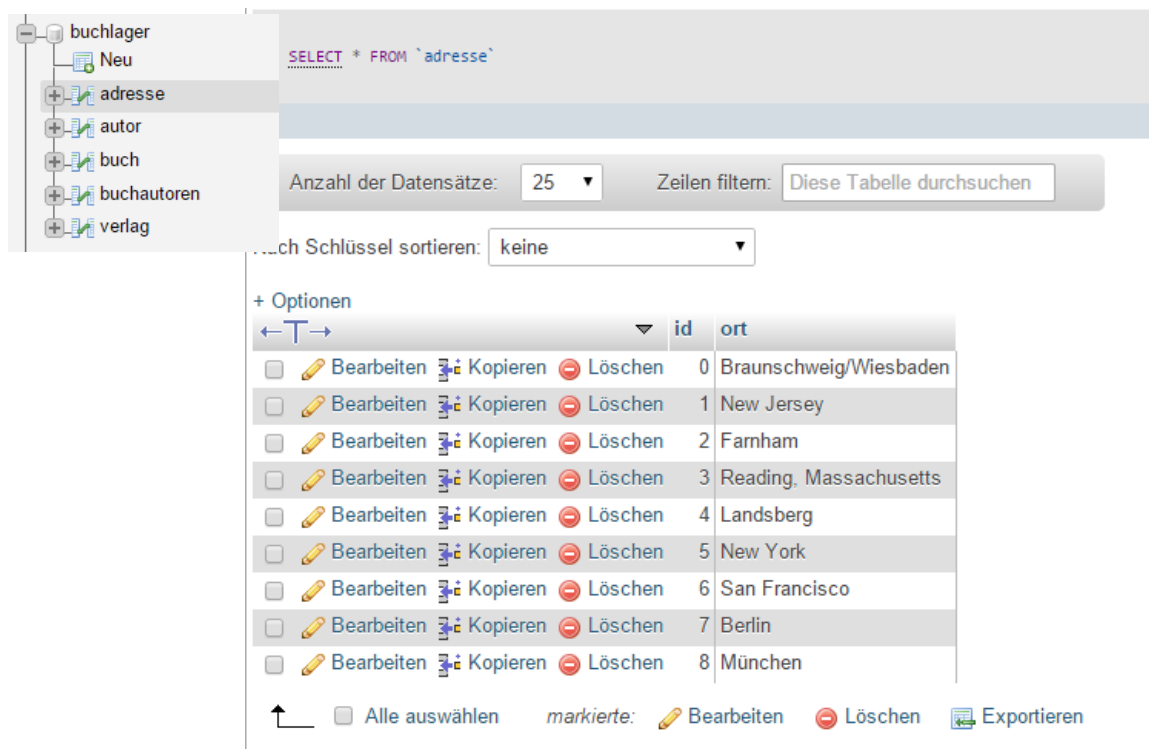


Abbildung 3

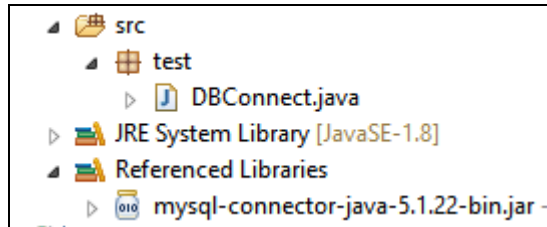
Wählen Sie danach auf der linken Seite die erzeugte Datenbank *buchlager* aus und danach den Tab SQL. Kopieren Sie nun die Create-Statements für die Tabellen in das Textfeld (ein entsprechendes Skript finden Sie auf der Web-Seite) und bestätigen Sie mit OK. Hierdurch werden die benötigten Tabellen angelegt und mit entsprechenden Daten befüllt.



Internetprogrammierung Übungsblatt 11

4) Ein erstes JDBC-Programm

Erzeugen Sie ein gewöhnliches Java-Programm und fügen Sie dem Build-Path den JDBC-Treiber der MySQL-Datenbank hinzu.



Testen Sie mit dem folgenden Konsolenprogramm, ob der Zugriff auf die Datenbank funktioniert:

```
public class DBConnect
{
    public static void main(String[] args) throws Exception
    {
        Class.forName("com.mysql.jdbc.Driver").newInstance();

        Connection con =
            DriverManager.getConnection("jdbc:mysql://localhost/buchlager?user=root");

        Statement stmt = con.createStatement();

        ResultSet rs = stmt.executeQuery("Select * FROM autor");
        while( rs.next() )
        {
            System.out.println( rs.getString("vorname") + " " + rs.getString("nachname"));
        }

        rs.close();
        stmt.close();
        con.close();
    }
}
```

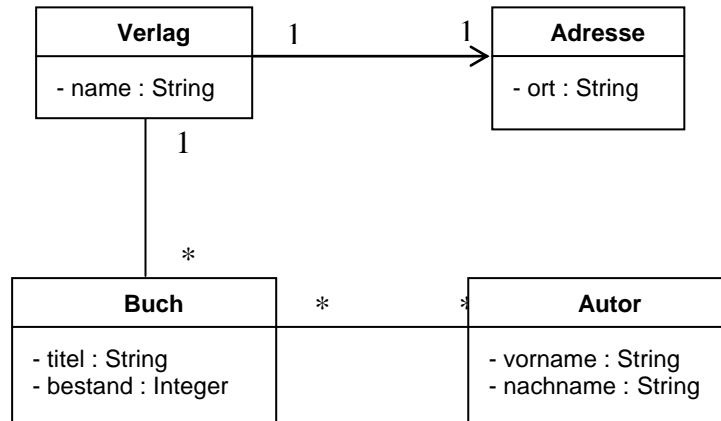
Es kann sein, dass Sie bei der DB-URL noch das Passwort mit angeben müssen. Das hängt vom Installationsstatus der MySQL ab.

Internetprogrammierung

Übungsblatt 11

5) Implementierung verschiedener Anfragen

Die Tabellen repräsentieren das folgende Datenmodell:



Schreiben Sie eine Klasse, die folgende Suchfunktionalität in der Schnittstelle bereitstellt:

```

Collection<Buch> searchBuecherMitTitel(String titel)
Adresse findAdresseVonVerlag(int verlagId)
Verlag findVerlagVonBuch(int buchId)
Collection<Buch> findBuecherVonVerlag(int verlagId)
Collection<Buch> findBuecherVonAutor(int autorId)
Collection<Autor> findAutorenVonBuch(int buchId)
  
```

Realisieren Sie die Klasse als Singleton und implementieren Sie auch die entsprechenden Modellklassen, wobei die Modellklassen hier im Wesentlichen nur die Attributwerte verwalten.

Lassen Sie im Moment die Navigationsmethoden (wie z.B. `getAdresse` von `Verlag`) noch unimplementiert. Die Implementierung erfolgt in der nächsten Aufgabe.

Hinweise und Idioms

Es bietet sich an, die Klasse mit einer Methode `getConnection` auszustatten, die eine Datenbank-Verbindung zurückliefert. (Um eine gewisse Datenbankunabhängigkeit zu erreichen, sollte die DB-URL eigentlich nicht hart codiert sein. Wir werden später sehen, dass die JEE hierzu ein geeignetes Konzept bereitstellt.)

```

Connection getConnection() throws SQLException
{
    return DriverManager.getConnection("jdbc:derby://localhost:1527/Buchlager");
}
  
```

JDBC-Syntax für einen PreparedStatement-Zugriff

```

Connection      con = null;
PreparedStatement pstmt = null;
ResultSet       rs = null;

try{
    con = this.getConnection();
    pstmt = con.prepareStatement("SELECT * FROM Buch WHERE id = ?");
    pstmt.setInt(1, buchId );
}
  
```

Internetprogrammierung Übungsblatt 11

```
ResultSet rs = pstmt.executeQuery();
while ( rs.next() )
{
    String titel = rs.getString( "titel" );
    System.out.print( "Buchtitel: " + titel );
}
catch (SQLException sqlexce)
{
    sqlexce.printStackTrace();
}
finally
{
    if( rs != null )
        try{ rs.close() }catch(SQLException e){}
    if(pstmt!= null )
        try{ pstmt.close() }catch(SQLException e){}
    if( con != null )
        try{ con.close() }catch(SQLException e){}
}
```

6) Implementierung der Navigationsmethoden (*poor man's solution*)

Implementieren Sie nun die Navigationsmethoden

Verlags-Klasse:

- `getAdresse` soll *eager* sein, d.h. wenn das Verlagsobjekt instanziiert wird, soll auch gleich immer das zugehörige Adressobjekt mit instanziiert werden.
- `getBücher` soll *lazy* implementiert werden, d.h. die Buchliste soll erst dann geladen werden, wenn die Methode aufgerufen wird.

Buch-Klasse:

- `getVerlag` soll *eager* sein, `getAutoren` *lazy*

Autor-Klasse:

- `getBuecher` soll *lazy* sein.

Testen Sie Ihr Programm.