

Vorbemerkungen

Das Internet und mit ihm das World Wide Web haben sich in kurzer Zeit zu einer relevanten betrieblichen und kommerziellen Plattform entwickelt. Vor allem die netzgestützten elektronischen Verfahren oder E-Techniken wie

- E-Shopping elektronisches Einkaufen
- E-Banking elektronischer Zahlungsverkehr
- E-Cash elektronisches Geld
- E-Commerce elektronischer Handel

erfahren im Internet eine rasante Entwicklung, auch infolge ihrer überaus breiten Akzeptanz. In enger Wechselwirkung damit entwickelte sich JAVA™ zu einer unentbehrlichen Grundlage für solche Techniken. Für die Anwendungsentwicklung ergibt sich daraus als einer der wichtigsten Teilaspekte die Internettauglichkeit der verwendeten Mittel, insbesondere auch die bereits vorhandener Datenbankressourcen. Ein geeignetes Hilfsmittel, Internettauglichkeit herzustellen, ist *Java DataBase Connectivity* bzw. *JDBC™*, ein *Java-Framework* für die Anbindung besonders von relationalen Datenbanken an Java-Programme. (Mit *Frameworks* werden *generische Programmgerüste* bezeichnet, d.h. Klassen-/Komponentenbibliotheken, die auf ganz bestimmte Fallgruppen von Anwendungen zugeschnitten sind und Problemlösungen auf einem höheren Abstraktionsniveau zulassen.)

Entwicklungstendenzen der genannten Art wurden von SUN MICROSYSTEMS, dem Urheber von JAVA, nicht unerheblich dadurch gefördert, daß das JDBC-Framework frühzeitig verfügbar war, und zwar zusammen mit einer Treiberimplementierung, die dem Programmierer die Anbindung von Datenbanken mittels *ODBC* erlaubt, sowie eines Tests, mit dem der Entwickler seine Treiberprodukte auf JDBC-Konformität überprüfen kann. (ODBC oder Open DataBase

Connectivity ist ein Standard der SQL Access Group, der vor allem bei der Anbindung von Windows 9x- bzw. Windows NT-basierten Datenbanksystemen verwendet wird und dementsprechend weit verbreitet ist.)

JDBC ist beinahe ebenso alt wie Java. Mit Java Version 1.1 wurde JDBC in der Version 1 kanonisiert, indem seine Klassen und Interfaces in den Kernbestand der Java-Klassen aufgenommen wurden. (Zum Kernbestand des JDK 1.2 bzw. von Java Version 2 gehört JDBC in der neueren Version 2).

Der Erfolg von JDBC beruht auf seinem normativen Charakter *und* auf der Universalität der Java-Plattform. Java-Plattform und JDBC machen es möglich, in Problemlösungen *bewährte* Mittel wie relationale Datenbank-Management-Systeme (RDBMS) und *innovative* Client/Server-Techniken zu verbinden. In den JDBC-Klassen und Interfaces selbst ist ein Schnittstellenkonzept verwirklicht, das einheitliches, im Ansatz *datenbankunabhängiges* Programmieren erlaubt.

Die JDBC-*Programmierschnittstellen* sind der Rahmen für die Programmierung des Zugangs zu Datenbanksystemen. Die Kernfunktionalität ist recht einfach: JDBC gibt eine *beliebige* Zeichenkette über sog. *Treiber* als Anweisung weiter an das Datenbanksystem. Die Antwort wird in Tabellenform an das rufende Programm zurückgereicht.

JDBC ist auf SQL und relationale Datenbanken zugeschnitten. Dementsprechend sind die Zeichenketten SQL-Anweisungen und die Ergebnistabellen Relationen oder Views der angesprochenen relationalen Datenbanken.

Zielsetzungen

JDBC handelt unmittelbar im Bereich moderner verteilter Software-Architekturen, die zusammen mit relationalen Datenbanken sowie mit Java Themenschwerpunkte dieses Buches sind. Über JDBC *und* die Themenvielfalt des Umfeldes soll in angemessener Breite informiert und dabei in die Einzelthemen mit der nötigen Tiefe eingeführt werden. Arbeiten mit JDBC setzt die Kenntnis des Umfeldes voraus.

Angesichts des Umfangs der Themen sind Grenzziehungen unvermeidlich. So werden nur solche Themen behandelt, die in direktem Bezug zu JDBC stehen. Ausgewählt wurden neben *relationalen Datenbanken* und *SQL* insbesondere elementare *Client/Server-Strukturen*. Im Rahmen der letzteren werden die Kommunikation zwischen Client und Server im *Internet*, die *Ebenen (tiers)* der Client/Server-Interaktion und sog. *Webdatenbanken* thematisiert. Dabei wird

getrennt zwischen dem Grundlegenden, das man kennen muß, und Details, die man sich oft ohne größere Mühen selbst erschließen kann. So ergibt sich ein methodisches Netz, mit dem auch komplexe Probleme analysiert und elementarisiert werden können.

Vom Leser wird dreierlei erwartet:

1. Java-Kenntnisse: Sie sollten nicht nur die prozedurale Syntax umfassen, sondern auch Begriffe wie Klassen, Interfaces und Klassenbibliotheken, Vererbung und Polymorphie. Programmieranfänger sollten sich zuvor entsprechende Java-Kenntnisse aneignen, z.B. unter Zuhilfenahme von [Jobst 99].
2. Wissen um Datenbanken, Internet und World Wide Web: Begrifflichkeit, Bedeutung und Funktionalität sollten nicht gänzlich unbekannt sein, sehr einfache Grundlagenkenntnisse sollten aber ausreichen.
3. Die Bereitschaft und auch das Vergnügen, sich gleichsam simultan eines Netzes heterogener und differenzierter Methoden bei der Lösung komplexer DBMS-Probleme etwa in verteilten Umgebungen zu bedienen.

Übersicht

Das Buch ist inhaltlich so angelegt, daß es möglichst vielen Lesern Interessantes und Nützliches über JDBC und das JDBC-Umfeld bietet. Es richtet sich nicht nur an den Leser mit professionellem Interesse, sondern auch an alle, die sich nach dem Motto „learning by doing“ den Themen nähern wollen oder ganz einfach nur beabsichtigen, ihrer Desktop-Datenbank zur Internettauglichkeit zu verhelfen. Erforderliche Datenbankkenntnisse können mittels der Kapitel 2 und 3 erarbeitet werden.

Zum allgemeinen Vorgehen ist noch anzumerken, daß die einfachsten Eigenschaften vor allem von JDBC und SQL sehr frühzeitig eingeführt und angewandt werden. Das gibt die Möglichkeit, Sachverhalte unmittelbar auch aus deren Perspektive darzustellen, wie sich also z.B. Datenbankeigenschaften in JDBC-Metadaten oder kartesische Produkte und Tabellenverknüpfungen in SQL widerspiegeln.

Zur Gliederung des Buches im einzelnen:

Kapitel 1: Einleitung nennt Themen und Zielsetzungen des Buches. Dabei wird eine einfache Arbeitsgrundlage geschaffen, um bereits in den Kapiteln 2 und 3 mittels JDBC Treiber- und Datenbankeigenschaften erkunden und einfache SQL-Abfragen ausführen zu können.

Kapitel 2: Das Kapitel **Relationale Datenbanken** führt in dieses Thema über eine Vielzahl von Beispielen ein. In einigen der Beispiele werden vorbereitend die einfachsten SQL-Anweisungen demonstriert und teilweise ihre Umsetzung in Java/JDBC-Programmen gezeigt. Ein Exkurs dient dazu, eine Java/JDBC-Datenbankanwendung mit graphischer Benutzeroberfläche und Datenbankanbindung zu erstellen, mit deren Hilfe `SELECT-SQL`-Befehle eingeübt werden können.

Kapitel 3: SQL ist der Datenbanksprache SQL gewidmet, in deren einfachste und wichtigste Sprachelemente eingeführt wird. Auch hier werden Java/JDBC-Programme exemplarisch eingesetzt.

Kapitel 4: JDBC-Grundlagen ist ganz der Erläuterung des Klassenpakets `java.sql` und der grundsätzlichen Verwendung der JDBC-Klassen und -Interfaces vorbehalten. Nach Klärung einiger prinzipieller Fragen sowie zwei vollständigen Programmbeispielen zur Einstimmung wird jede der fünf JDBC-Programmierschritte eingehend behandelt. Das Kapitel schließt mit den wichtigen Themen „Metadaten“ und „Treiber und Treiberkategorien“.

Kapitel 5: JDBC und Client/Server führt in Client/Server-Architekturen ein und zeigt ihre Verwendung in sehr einfach gehaltenen, aber voll internet-tauglichen Java-Programmbeispielen. Internet- und Intranet-Grundlagen sind, soweit thematisch relevant, in einem eigenen Unterkapitel gesammelt. Den Abschluß bilden einige Anmerkungen zu Middleware-Protokollen, insbesondere im Zusammenhang mit CORBA.

Kapitel 6: Ein Anwendungsbeispiel zeigt, wie eine Anwendung bei der Konzeption beginnend systematisch zu einem lauffähigen Produkt entwickelt wird. Die Anwendung nutzt ORACLE8 als Datenbank und MVC-Komponenten aus dem Java-Swing-Paket für die Gestaltung der graphischen Benutzeroberfläche.

Kapitel 7: Ausblicke beschäftigt sich mit der Version 2.0 des JDBC. Weitere wichtige Themen aus dem Umfeld von JDBC sind Embedded SQL und die Transformation von Objekten in Tabellen und umgekehrt. Der Abschnitt

Objektrelationale Transformation ist ein Beitrag von *Dr. Börries Ludwig* von debis Systemhaus.

Die **Anhänge** umfassen Treiberlisten, JDBC/SQL-Typumwandlungstabellen, Angaben zur Struktur der Musterdatenbank sowie Informationen zu Mini-SQL und zu ODBC.

Am Ende des Buches findet sich eine *JDBC-Kurzanleitung*, die auch herausgetrennt werden kann.

Beispiele und verwendete Werkzeuge (JDK, Datenbanksysteme, Webserver, Datenbanken, DB-Server etc.) beziehen sich ohne Ausnahme auf Windows 9x/NT. Als Werkzeuge wurden vorzugsweise solche ausgewählt, die für nicht-kommerzielle oder Evaluierungszwecke kostenlos zur Verfügung stehen.

Die Beispiele

Alle Programmbeispiele sind möglichst unkompliziert geschrieben, d.h. erklärungsbedürftige Kodierungstechniken werden weitgehend vermieden. Fast alle Beispiele sind vollständig und, bei entsprechend vorbereiteten Datenbanken und JDBC-Treibern, unmittelbar ablauffähig. Um die Beispiele übersichtlich zu halten, sind sie möglichst knapp formuliert und auf das Grundsätzliche konzentriert. Ihre Überschaubarkeit wird auch dadurch gefördert, daß Leer- und Kommentarzeilen sowie Fehlerbehandlungen auf ein Minimum reduziert sind. Überhaupt wird mit Codezeilen, die nicht direkt zur Lösung des jeweiligen Problems beitragen, sparsam umgegangen. Die erforderlichen Erläuterungen der Programme sind grundsätzlich im laufenden Text vorgenommen, bei kürzeren meist im Anschluß an das Programm und bei längeren vor bzw. nach zusammenhängenden Programmteilen.

Alle Beispielprogramme wurden mittels des Java Development Kits in der Version 1.1.7 (JDK 1.1.7) unter Windows 9x/NT entwickelt und sind alle mit JDBC in der Version 1.2 und aufwärts kompatibel.

Auch den Applets ist JDK 1.1.7 zugrunde gelegt. Getestet wurden sie entsprechend mit einem Java-1.1-fähigen Web-Browser, nämlich dem Netscape Navigator in der Version 4.5.

Neben den erforderlichen JDBC-Treibern für die verwendeten Datenbanksysteme ist in Kapitel 6 zusätzlich das Swing-Klassenpaket erforderlich.

In beinahe allen Beispielen wurde eines der folgenden Datenbanksysteme eingesetzt:

1. *MICROSOFT ACCESS* in der Version 8 wird als sogenannte *Desktop*-Datenbank verwendet. Die Musterdatenbank *Kurse* ist mittels des Windows-ODBC-Managers unter dem Namen „Kurse“ registriert und mit Java-Programmen über eine *JDBC-ODBC-Brücke* verbunden.
2. *MINISQL* des australischen Herstellers Hughes Technologies wird in der Version 1.016 als einfacher, aber voll funktionsfähiger *SQL-Datenbankserver* für verteilte Anwendungen eingesetzt. Die Verbindung mit Java-Programmen geschieht mithilfe eines 100% Java-Treibers über ein MiniSQL-eigenes Netzprotokoll. Auf seiner Basis sind vor allem Programmbeispiele in Kapitel 5 realisiert.
3. *ORACLE8 PERSONAL EDITION* wird stellvertretend für den Oracle-Datenbankserver verwendet. Von diesem weit verbreiteten Datenbanksystem wird insbesondere in Kapitel 6 Gebrauch gemacht.

Die Client/Server-Beispiele erfordern außerdem ein funktionierendes TCP/IP- bzw. Internet-Umfeld. Einige wenige der verwendeten Werkzeuge wie die Administrationshilfen des SQL-Datenbankservers MiniSQL senden Anfragen an DNS-Server im Internet, brauchen also Verbindung ins Internet. (Allerdings kann die Nutzung der Administrationshilfen meist ohne größere Nachteile umgangen werden).

In fast allen Beispielen wird die Musterdatenbank *Kurse* verwendet (siehe Anhang C). Diese Datenbank ist für alle verwendeten DBMS mit gleichen Strukturen und Inhalten realisiert, so daß durchgehend gleichsam dieselbe Datenbank Anwendung findet.

Wie bereits angemerkt, sind, von wenigen Ausnahmen abgesehen, in den Beispielen vollständige Programme wiedergegeben. Um Fehler im Programmtext möglichst gering zu halten, sind die abschließenden Tests auf der Basis des Buchmanuskripts durchgeführt worden: Die Programme wurden aus dem Manuskript per „Kopieren-und-Einfügen“ in einen Editor kopiert, gespeichert, kompiliert und ausgeführt. Eventuell noch vorhandene Fehler wurden direkt im Manuskript korrigiert; danach wurden die Programme wie eben beschrieben erneut getestet.

Eine Aufzählung aller Programmbeispiele ist als „Programmverzeichnis“ im Anschluß an das Inhaltsverzeichnis zu finden. Die Adresse, unter der diese Beispiele als Quellcode im Web verfügbar sind, ist unter „Webware“ am Ende des Buches angegeben.

Die Programmbeispiele haben in der Regel die folgende Form:

```
// Programm 1-1: ./Einfuehrung/JdbcKonform.java
public class JdbcKonform {
    ...
    Programmcode
    ...
} // Ende class JdbcKonform
```

Hinter der Programmnummer steht das relative Verzeichnis, in dem das Quellprogramm im Web zu finden ist. An die Stelle des Punktes muß lediglich noch die passende Webadresse aus dem Abschnitt „Webware“ gesetzt werden.

Konventionen

Durch unterschiedliche Schriftarten und Schriftschnitte werden Textteile in diesem Buch wie folgt gegeneinander abgehoben:

Courier wird als Schrifttyp für *Programmcode* (Java, SQL etc.), Programm- und Dateinamen, URLs und Internetadressen usw. verwendet.

Courier kursiv bezeichnet *Platzhalter* in Programmen, das heißt, an ihrer Stelle müssen bei Umsetzung noch gültige Werte eingesetzt werden.

Courier fett / **Courier fett kursiv** lenkt die Aufmerksamkeit auf Teile in Programmbeispielen, die meist im laufenden Text näher erläutert werden.

Times kursiv dient der Hervorhebung, weist also beispielsweise auf einen *neuen* oder *wichtigen Begriff* hin oder signalisiert einfach „Obacht!“

KAPITÄLCHEN werden gelegentlich verwendet, um Produkt- und Firmennamen hervorzuheben.

`name()` in *Courier* und mit *leerer* Klammer bezeichnet den Namen einer Methode im laufenden Text, z.B. „Die `name()`-Methode ist ...“. Die leere Klammer `()` dient lediglich der Kennzeichnung von `name` als Methode und sagt *nichts* über den Klammerinhalt, d.h. die Parameter der Methode, aus.

Fachsprache: oder ?

Java und SQL sind englischsprachig, das Internet ist englischsprachig, und auch die Primärliteratur ist fast immer englischsprachig. *Fachtermini* werden daher nur dann eingedeutscht, wenn sie in der Übersetzung geläufig sind. Das ist etwa bei der Datenbankterminologie mit wenigen Ausnahmen wie *Join* der Fall. Sonst werden überwiegend die englischsprachigen Begriffe verwendet.

Englischsprachige Beispiele sind etwa: *Join, Compiler, Computer, Web-Browser, Firewall, Host, Client, Server, Interface, Webserver* etc.

Deutschsprachige sind: *Anweisung, Dreiebenenarchitektur, Domäne, Anforderung* und *Antwort* (nicht: *Statement, Three-Tier Architecture, Domain, Request* und *Answer*).

Dank

Dem Carl Hanser Verlag, insbesondere Frau Margarete Metzger und Frau Irene Weilhart, bin ich für die effiziente und angenehme Zusammenarbeit verpflichtet.

Herrn Börries Ludwig verdanke ich neben vielen Anregungen insbesondere zum Themenkreis Middleware den Beitrag „Objektrelationale Transformation“ in Kapitel 7.

Sofern das Buch nicht allzu viele Fehler beinhaltet, ist das vor allem auf das sorgfältige Korrekturlesen von Frau Stefanie Schulz (Universität Hohenheim) und Herrn Manfred Sommer (Carl Hanser Verlag) zurückzuführen, denen ich hiermit meinen Dank ausspreche.

Besonderen Dank schulde ich Frau Kollegin Brigitte Hagenmeyer, die mir nicht nur zu diesem Buchprojekt riet, sondern mir auch beim Verfassen des Buches mit fachlichem Rat zur Seite stand.

Hohenheim, im Juni 1999

Wolfgang Dehnhardt