öS 62,– sfr 7,50 **DM 7,50** H 10554

August 1997

8

NT-5.0-Standard:
Active Directory Services

iX

MAGAZIN FÜR PROFESSIONELLE INFORMATIONSTECHNIK

Programmieren für heterogene Umgebungen:

CORBA vs. DCOM

DCOM für Unix im Test

Die aktuellen Web-Browser:

Netscape Communicator 4
MS Internet Explorer 4

Dynamic HTML, JavaScript 1.2

Internet:

Softwareverteilung im Netz

Online-Shopping-Systeme

HyperWave-Einsatz

DNS-Spoofing

Multiprozessorsysteme:

300-MHz-UltraSPARC 2

6xPentiumPro ALR

PowerPC 604

Marktübersicht:

Fileserver für alle Systeme

Datenbanken:

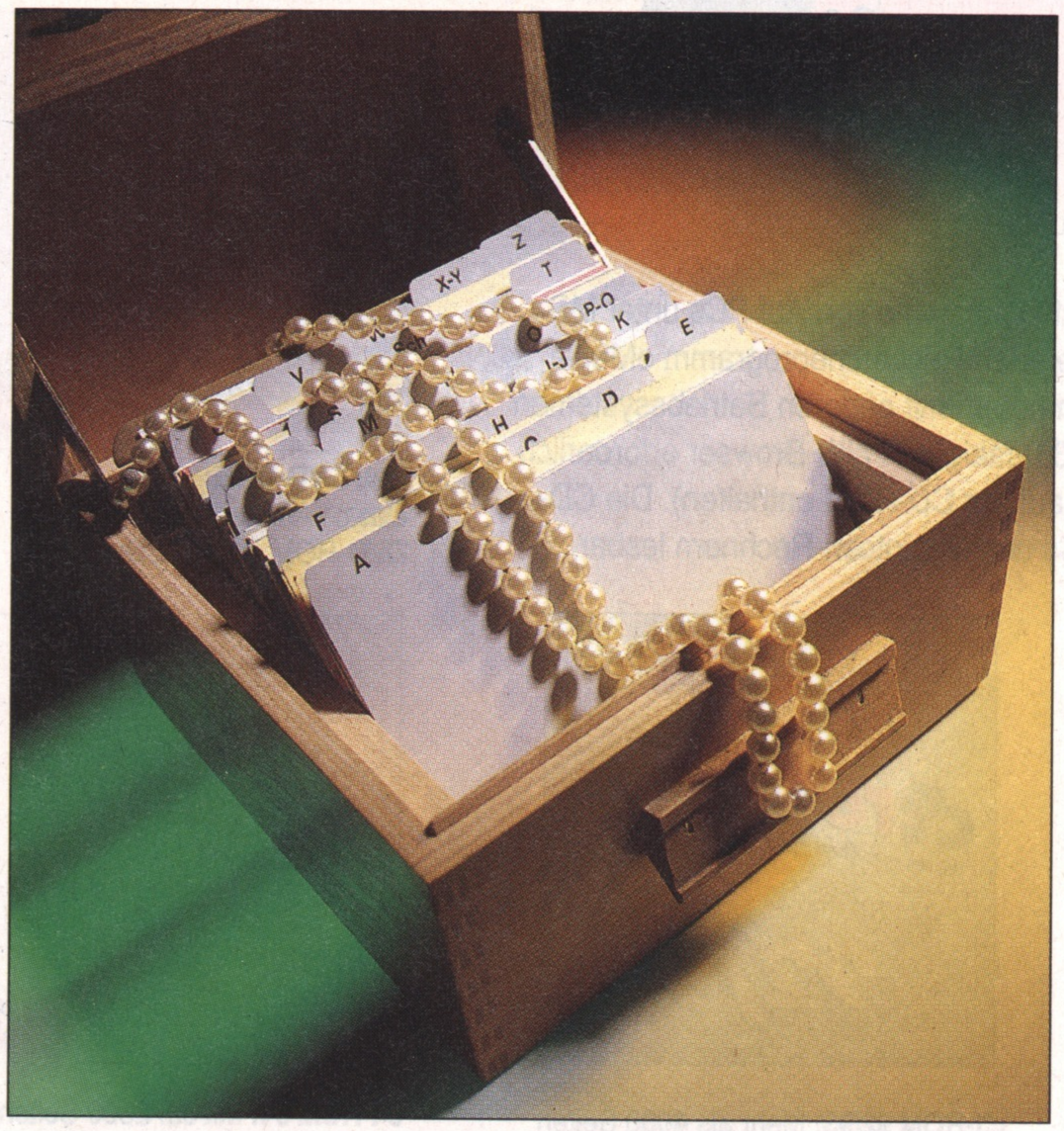
Wie Data Mining funktioniert

Kostenloser GUI-Builder:

wxWindows für X11 und MS Windows

August 1997 iX Magazin für professionelle Informationstechnik http://www.heise.de/ix/ HEISE

4 399148 507509 08



Datenbankbedienung mit Perl und CGI

Gebunkert

Michael Schilli

Web-Formulare können mit Online-Datenbanken zusammenarbeiten, um Informationen anzuzeigen oder abzulegen. Mit dem Modul 'DBI' spielt sogar die Datenbankmarke keine Rolle mehr: Ob Oracle, Informix, Sybase, Ingres, Postgres oder das preisgünstige msql – der Zugriff bleibt der gleiche.

Das CGI-Skript *inventur.pl* arbeitet mit einer msql-Datenbank, die das relationale Modell in Abbildung 1 implementiert. Es besteht aus zwei Tabellen, einer Hardware- und einer Personenliste, die untereinander verknüpft sind und so eine Zuordnung zwischen dem Hardwarebestand einer Firma und deren Mitarbeitern erlauben.

Liegt *inventur.pl* ausführbar im *cgi-bin*-Verzeichnis eines Web-Servers und ist die msql-Datenbank installiert, gestartet und initialisiert (siehe Kasten 'Vorbereitungen'), zeigt ein Browser bei Aufruf des URL die Oberfläche (Abbildung 2) an.

Bereits eingegebene Mitarbeiter sind in der Spalte 'Person' zu sehen, neue lassen sich unter Angabe einer eindeutigen Personalnummer, eines Namens-String und der Raumnummer durch Drücken von 'Einfügen' unten links hinzufügen. Der 'Löschen'-Knopf entfernt markierte Personen aus der Datenbank. Analog zeigt und manipuliert die mittlere Spalte ('Hardware') den Gerätebestand: Ein Posten besteht aus einer eindeutigen Inventurnummer (Asset) und einer Gerätebezeichnung.

Nach Raumnummern sortiert, zeigt die rechte Spalte ('Inventurliste') an, welcher Person ein bestimmtes Gerät zugeordnet ist. Neue Zuordnungen entstehen, indem man aus der 'Person'- und der 'Hardware'-Liste jeweils ein Element auswählt und den 'Verknüpfen'-Knopf drückt. Eine Zuordnung verschwindet durch Wählen eines Elements der 'Inventurliste' und Anklicken von 'Trennen'.

Details des CGI-Skripts zeigt das Listing. Die Zeilen 8 bis 10 legen Datenbankparameter fest. Es kommt ein msql-Treiber zum Einsatz, die Datenbank soll *inventur* heißen, und der Datenbank-Server läuft auf dem lokalen Rechner. Zeile 13 nutzt Methoden des CGI-Pakets, um den HTTP-Header und die HTML-Startsequenz auszugeben. Der in Zeile 16 definierte Error-Handler veranlaßt den Perl-Interpreter bei Fehlern, die in eine *die*-Anweisung münden, nicht einfach abzubrechen, sondern vorher den Fehlertext HTML-formatiert auszugeben und die Datenbankverbindung zu schließen.

Zeile 19 verbindet das Skript mit der Datenbank; im Fehlerfall enthält *$DBI::errstr* den Text der Fehlermeldung. Bei Verwendung einer anderen Datenbank weicht die Signatur der *connect*-Methode eventuell leicht von der verwendeten ab. Auf der DBI-Manual-Seite finden sich Beispiele, eine Liste verfügbarer Datenbanktreiber enthält der Kasten 'Vorbereitungen'. Zeile 23 prüft mit einer *SELECT*-Anweisung, ob die Datenbank schon die notwendigen Tabellen enthält; falls nicht, legt sie *db_init()* ab Zeile 32 an. Das in Zeile 25 aufgerufene *switch_action()* analysiert die Eingabeparameter des CGI-Skripts und führt notwendige Aktionen durch, wie das Einfügen oder Löschen von Einträgen. Das anschließende *draw_form()* erzeugt das HTML-Formular, indem es Daten aus der Datenbank grafisch aufbereitet. Als letzter ausgeführter Befehl schließt Zeile 30 die Datenbankverbindung.

SQL-Anweisungen als Here-Dokument

db_init() setzt in den Zeilen 38 und 39 zwei *DROP*-Anweisungen ab und löscht so sicherheitshalber Tabellen, die zu diesem Zeitpunkt noch gar nicht vorhanden sein sollten. Die *do*-Methode eines Datenbankobjekts *$dbh* sendet eine SQL-Anweisung an den Server.

Zeilen 41 bis 47 senden eine mehrzeilige SQL-Anweisung und bedient sich dazu eines Here-Dokuments. Die Tabelle 'person' enthält als erste Spalte die Personalnummer (*pn*), die als Integer-Wert und als Primärschlüssel ('pri-

Tabelle hw

asset	name	person
8000	Pentium 100, 32 MB	1000
8001	Monitor Zapp 2000	1000
8002	Pentium 100, 32 MB	1001
8003	Monitor Zapp 2000	1001

Tabelle person

pn	name	raum
1000	Schuster, Max	100
1001	Wenger, Thomas	100

Dieses relationale Modell stellt die Beziehung zwischen Personen und Geräten her (Abb. 1).

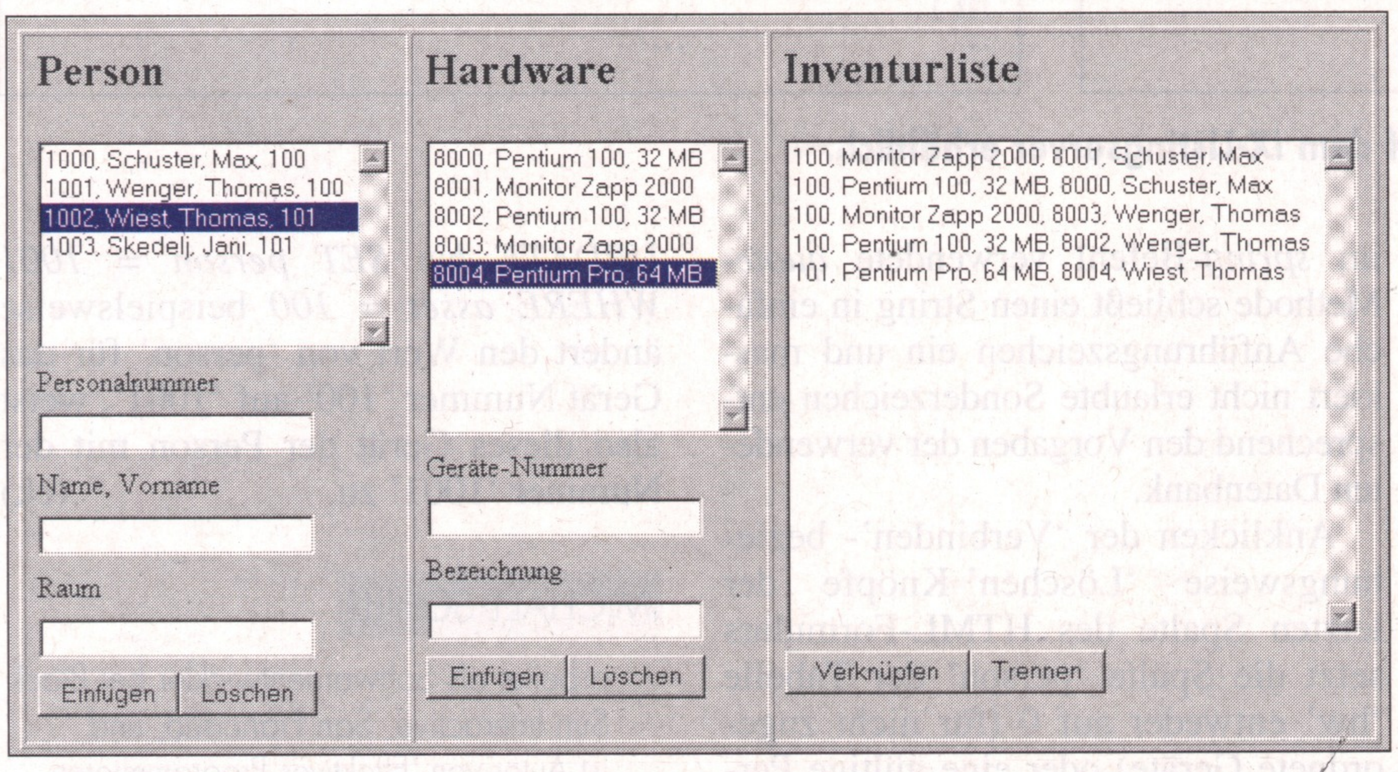
mary key') definiert ist. Doppelt auftretende Personalnummern führen daher bereits beim Einfügen zu einem Fehler.

Den Namen einer Person speichert ein 80 Zeichen breites *char*-Feld, der Raum, in dem die Person sich aufhält, ist ein Integer-Wert. Die Tabelle 'hw' führt als eindeutige Kennung eines Geräts dessen Inventurnummer (*asset*) sowie zur Beschreibung ein *char*-Feld. Die letzte Spalte dieser Tabelle ist ein sogenannter Fremdschlüssel: *person* ist ein Integer-Wert, der auf einen Primärschlüsseleintrag der 'person'-Tabelle verweist. Dieser Trick ordnet Hardwareteile letztlich Personen zu.

switch_action() 'weiß' durch das hereingereichte CGI-Objekt *$q*, welche Felder des CGI-Formulars ausgefüllt beziehungsweise selektiert wurden und entscheidet, was der Benutzer unternehmen will. Beim ersten Aufruf des Skripts tut *switch_action()* gar nichts, da keinerlei Formularinformation vorliegt. Erst das nachfolgende *draw_form()* zeichnet das Formular und gibt dem Benutzer die Möglichkeit, etwas auszuwählen oder auszufüllen.

Diese Funktion holt in Zeile 114 alle Personen nach Personalnummern sortiert aus der Datenbank. Die *sql()*-Funktion ist ab Zeile 189 definiert und ruft zwei Methoden der DBI-Schnittstelle auf: *prepare()* zum Vorbereiten eines SQL-Kommandos und *execute()* für seine Ausführung. Anschließend iteriert eine *while*-Schleife über jede Zeile der Tabelle. Wegen der etwas eigenwilligen Parameterübergabe an die in Zeile 155 aufgerufene *scrolling_list()*-Methode erfolgt die Speicherung der extrahierten Tabellendaten in zwei Datenstrukturen: einem Array *@person*, das lediglich die gefundenen Personalnummern enthält, und einem Hash *%lperson*, der jede dieser Nummern einem *Label* zuweist, das den später angezeigten Listeneintrag ausmacht. Analog gelangen die Daten der Tabelle 'hw' in das Array *@hw* und den Hash *%lhw*.

Zeile 137 definiert ein *SELECT*-Kommando, das gleichzeitig mit zwei Tabellen arbeitet: Es verknüpft den Fremdschlüssel *person* aus Tabelle 'hw' mit dem Primärschlüssel *pn* aus Tabelle 'person' und liefert so eine nach Raumnummern, Personen und Hardware sortierte Inventurliste.



Je nach Datenbestand sieht der Inhalt des Formulars anders aus (Abb. 2).

Vorbereitungen

msql liegt zwar im Sourcecode vor, ist aber weder Public-Domain noch Freeware. Der Autor David Hughes schont nur universitäre Einrichtungen, alle anderen zahlen 65 US-$ für eine Privatlizenz oder 225 $ für die kommerzielle Nutzung.

Die msql-1.0-Distribution findet sich in den Sunsite-Verzeichnissen der Linux-Spiegelungen, so unter ftp://ftp.uni-erlangen.de/pub/Linux/MIRROR.sunsite als *apps/database/sql/msql-1.0.16.src.tgz*. Die Version 2.0 liegt als Beta-Release vor (http://hughes.com.au). msql läuft auf allen gängigen Unix-Plattformen.

Die neue Datenbank *inventur* erzeugt unter *root* der Befehl

```
msqladmin create inventur
```

Das relationale Datenbankmodell legt das Skript *inventur.pl* selbst an, indem es leere Tabellen mit den notwendigen Spalten erzeugt.

Damit DBI mit einer msql-Datenbank kommunizieren kann, ist der entsprechende Treiber notwendig, der auf dem CPAN (siehe unten) als *DBD-mSQL-0.65.tar.gz* vorliegt. Der Befehl

```
gtar -zxf DBD-mSQL-0.65.tar.gz
```

entpackt die Distribution,

```
cd DBD-mSQL-0.65
perl Makefile.PL
```

führt zum Dialog.

```
Which version of mSQL are you using [1/2]
Is this installation a root install or non-root?
[root/notroot]
```

Die zweite Frage ist abhängig von den bei der msql-Installation gewählten Parameter zu beantworten, *root* ist die vorgewählte Einstellung.

Benötigte Module: *CPAN/modules/by-module/DBD/DBI-0.83.tar.gz* und *CPAN/modules/by-module/CGI/CGI.pm-2.36.tar.gz*. In *CPAN/modules/by-module/DBD* gibt es zudem Treiber für Oracle, Sybase, Informix, DB2 und Postgresql, ein Treiber für mysql ist bei http://www.tcx.se/Contrib/DBD-mysql-1.65.tar.gz erhältlich. Deutsche Spiegel des CPAN sind unter anderem

ftp://ftp.leo.org/pub/comp/programming/languages/perl/CPAN/

ftp://ftp.rz.ruhr-uni-bochum.de/pub/CPAN/

ftp://ftp.uni-hamburg.de/pub/soft/lang/perl/CPAN/

Unter Windows NT ist DBI zur Zeit nicht installierbar, hier gibt es jedoch mit *Win32::ODBC* ein ähnliches Modul.

Die Bedingung *WHERE hw.person = person.pn* verbindet beide Tabellen.

draw_form() zeichnet die Ergebnisse dieser Queries als dreispaltige Tabelle, deren Spalten wiederum eine Auswahlbox, eventuell Eingabefelder und Submit-Buttons enthalten (siehe Abbildung 2).

Entscheidungsfindung in *switch_action()*

Für die Felder der ersten Spalte sind die Zeilen 152 bis 163 verantwortlich. Die *map*-Funktion umschließt alle nachfolgenden grafischen Elemente mit dem *<TR><TD>*-Tag, weist ihnen also eine Tabellenreihe zu. Es stehen deshalb untereinander eine *<H2>*-Überschrift, eine Auswahlliste, der String 'Personalnummer', das zugehörige Eingabefeld und so weiter, bis in der letzten Reihe schließlich zwei Submit-Buttons mit den Beschriftungen 'Einfügen' beziehungsweise 'Löschen' warten.

switch_action() stellt fest, welcher der sechs Submit-Buttons gedrückt wurde, welche Felder der Auswahlboxen selektiert sind und welche Daten in den Eingabefeldern stehen. Dementsprechend sind Daten in die Datenbank

```
 1 #!/usr/bin/perl -w
 2
 3 use DBI;                          # Generisches Datenbank-Interface
 4 use CGI;
 5
 6 $q = CGI->new;                    # CGI-Objekt
 7
 8 my $host     = "localhost";       # Datenbank-Rechner
 9 my $database = "inventur";        # Name der Datenbank
10 my $driver   = "mSQL";            # Verwendeter Treiber
11
12 # CGI-Header ausgeben
13 print $q->header, $q->start_html('Inventur-Liste');
14
15 # Im Fehlerfall Meldung ausgeben, Datenbankverbindung lösen
16 $SIG{__DIE__} = sub { msg(@_); $dbh->disconnect if $dbh;
17                                 print $q->end_html; exit 0;};
18 # Verbindung zur Datenbank aufbauen
19 $dbh = DBI->connect($host, $database, '', $driver) ||
20        die $DBI::errstr;
21
22 # Tabelle vorhanden? Falls nicht, Datenbank initialisieren
23 $dbh->do("SELECT * FROM hw") || db_init($dbh);
24
25 switch_action($q);   # Aktion entsprechend gedrückter
26                      # Taste einleiten
27
28 draw_form($dbh, $q); # Tabellen ausgeben
29
30 $dbh->disconnect;    # Datenbankverbindung lösen
31
32 sub db_init {
...                         ...
36     my $dbh = shift;                    # Datenbank-Objekt
37
38     $dbh->do("DROP TABLE hw");          # Hardware-Tabelle entfernen
39     $dbh->do("DROP TABLE person");      # Personen-Tabelle entfernen
40
41     $dbh->do(<<EOT) || die $dbh->errstr; # Neue Personen-Tabelle
42         create table person (
43             pn        int primary key,  # Personalnummer
44             name      char(80),         # Nachname, Vorname
45             raum      int               # Raumnummer
46         )
47 EOT
...                       ...
55 }
56
57 sub switch_action {
...                       ...
61                          # 'Einfügen'-Button unter 'Hardware'-Tabelle
62     if($q->param('hw.insert')) {
63         if($q->param('hw.asset')) {
64             sql(sprintf "INSERT INTO hw VALUES (%d, %s, 0)",
65                         $q->param('hw.asset'),
66                         $dbh->quote($q->param('hw.name')));
67         }
68                          # 'Löschen'-Button unter 'Hardware'-Tabelle
69     } elsif($q->param('hw.delete')) {
70         if($q->param('hw.selected')) { # Eintrag ausgewählt?
71             sql(sprintf "DELETE FROM hw WHERE asset = %s",
72                         $q->param('hw.selected'));
73         }
...                       ...
87                          # 'Verknüpfen'-Button unter 'Raum'-Tabelle
88     } elsif($q->param('raum.connect')) {
89         if($q->param('person.selected') && $q->param('hw.selected')) {
90             sql(sprintf "UPDATE hw SET person = %d
91                          WHERE asset = %d",
92                          $q->param('person.selected'),
93                          $q->param('hw.selected'));
94         }
95                          # 'Trennen'-Button unter 'Raum'-Tabelle
96     } elsif($q->param('raum.disconnect')) {
97         if($q->param('raum.selected')) {
```

```
98             sql(sprintf "UPDATE hw SET person = 0
99                          WHERE asset = %d",
100                          $q->param('raum.selected'));
101        }
102    }
103 }
104
105 sub draw_form {
...                 ...
110
111    my (@person, @hw, @row, %lperson, %lhw);
112
113    # Person-Tabelle abfragen
114    my $sth = sql( q{ SELECT pn, name, raum
115                      FROM person
116                      ORDER BY pn
117                    });
118
119    while(@row = $sth->fetchrow) {              # Query-Ergebnisse
120        push(@person, $row[0]);                 # Personalnummer
121        $lperson{$row[0]} = join(', ', @row);   # Rest in Label-Hash
122    }
123
124    # Hardware-Tabelle abfragen
125    $sth = sql( q{ SELECT asset, name
126                   FROM hw
127                   ORDER BY asset
128                 });
129
130    while(@row = $sth->fetchrow) {              # Query-Ergebnisse
131        push(@hw, $row[0]);                     # Inventurnummer
132        $lhw{$row[0]} = join(', ', @row);       # Rest in Label-Hash
133    }
134
135    # Join über zwei Tabellen ergibt die nach Räumen sortierte
136    # Inventurliste
137    $sth = sql( q{ SELECT   person.raum, hw.name, hw.asset,
138                            person.name
139                   FROM     hw, person
140                   WHERE    hw.person = person.pn
141                   ORDER BY person.raum, person.name, hw.name
142                 });
143
...                   ...
148
149    # Tabellen als HTML ausgeben
150    print "<TABLE border=1><TR><TD valign=top>\n", $q->start_form;
151
152    print "<TABLE>",
153          map { "<TR><TD>$_\n" }     # Einträge untereinander
154              $q->h2("Person"),      # in Tabelle anordnen
155              $q->scrolling_list('person.selected', [@person],
156                                 [], 7, '', \%lperson),
157              "Personalnummer", $q->textfield("person.pn", ""),
158              "Name, Vorname\n", $q->textfield("person.name", ""),
159              "Raum", $q->textfield("person.raum", ""),
160              $q->submit('person.insert', 'Einfügen') .
161              $q->submit('person.delete', 'Löschen');
162
163    print "</TABLE><TD valign=top>\n";
164
...                       ...
185    print "</TABLE>\n", $q->end_form, "</TABLE>\n";
186        print $q->end_html;
187 }
188
189 sub sql {
...                       ...
193    my $sth = $dbh->prepare("@_") || msg("Prepare failed: @_");
194
195    $sth->execute || msg("SQL Error:", $sth->errstr);
196
197    $sth;
198 }
...                       ...
```

Auszüge aus *inventur.pl*, vollständig auf dem *iX*-Listingserver erhältlich.

einzufügen oder zu löschen. Der in Zeile 62 geprüfte CGI-Parameter *hw.insert* ist beispielsweise gesetzt, wenn der Benutzer den Knopf 'Einfügen' in der Hardwarespalte gedrückt hat, da dieser in *draw_form()* den Namen *hw.insert* erhielt. Der SQL-Befehl *INSERT INTO hw VALUES (100, 'name', 0)* fügt eine neue Zeile in die Hardwaretabelle ein: ein Element mit der Nummer '100' und der Bezeichnung 'name'. Der Wert 0 für die Spalte 'person' besagt, daß das Gerät bislang keiner Person gehört. Die im *sprintf*-Befehl verwendete *quote*-Methode schließt einen String in einfache Anführungszeichen ein und maskiert nicht erlaubte Sonderzeichen entsprechend den Vorgaben der verwendeten Datenbank.

Anklicken der 'Verbinden'- beziehungsweise 'Löschen'-Knöpfe der letzten Spalte des HTML-Formulars setzt die Spalte 'person' der Tabelle 'hw' entweder auf 0 (für nicht zugeordnete Geräte) oder eine gültige Personalnummer. Das SQL-Kommando *UPDATE hw SET person = 1001 WHERE asset = 100* beispielsweise ändert den Wert von 'person' für das Gerät Nummer '100' auf '1001', weist also dieses Gerät der Person mit der Nummer '1001' zu. (ck)

MICHAEL SCHILLI

arbeitet als Softwareentwickler bei Black Sun Interactive, San Francisco, und ist Autor von 'Effektives Programmieren mit Perl 5'.