Lösungsvorschlag Reifeprüfung 2009

Inhalt

Analyse des Systems mit Annahmen

Problembezogene Annahmen

Es wird angenommen dass es an die hundert Techniker der Telefongesellschaft gibt, welche ihrerseits auf alle Kontaktdaten Zugriff haben.

In der Aufgabenstellung wird mehrmals von Mitgliedern und Gruppen gesprochen. Darunter sind alle Techniker gemeint, welche mit der Webanwendung arbeiten. Techniker selbst sind nicht in Gruppen eingeteilt.

In den Kontaktdaten werden beispielsweise die anagrafischen Daten von Kunden welche von den Technikern betreut werden aber auch Lieferanten, Arbeitskollegen usw. abgespeichert.

Es wird davon ausgegangen, dass es einige zehntausend Kontaktdaten gibt, welche ständig aktualisiert, gelöscht und erweitert werden.

Als anagrafische Daten werden Nach- und Vorname, Straße, Hausnummer, Postleitzahl und Ort sowie Provinz, Geschlecht und Geburtsdatum herangezogen. Weiters werden Telefon und E-Mail-Adresse abgelegt.

Der Punkt „…den Stand der Rubik vor der Änderung zu behalten, bis die Verwaltung des Systems die Änderungen für gültig erklärt und sie veröffentlich“ wird so interpretiert, dass alle Operationen auf die Kontakte in Transaktionen ablaufen. Die Operation wird automatisch vom System – d.h. Datenbanksystem – als gültig erklärt, wenn die Transaktion mit COMMIT bestätigt wurde. Entsprechend wird auch die Programmierung der Datenbankzugriffe ausgelegt. Führt beispielsweise ein Techniker eine Änderung an einem Kontakt durch, so wird zuerst eine Transaktion gestartet, dann wird der Kontakt geändert und mit protokolliert, welche Operation der Techniker auf welchen Kontakt durchgeführt hat. Dann wird die Transaktion abgeschlossen und somit „für gültig erklärt“.

Ein Grund der für die Interpretation des obigen Punktes in der angeführten Art und Weise spricht ist der, dass die große Anzahl von Technikern im Laufe eines Tages viele Operationen auf die große Anzahl von Kontakten durchführen. Würde hier die Verwaltung all diese Änderung erst händisch gutheißen müssen, wäre dies mit einem sehr großen organisatorischen Aufwand verbunden, der kaum zu bewältigen wäre.

Technische Annahmen

Zur technischen Realisierung der Aufgabenstellung werden folgende Server-Komponenten verwendet:

MySQL 5.0.15 als Datenbanksystem
Tomcat 5.5.9 mit JSTL und EL als Webserver
JDBC-Realms zur Realisierung der Benutzerauthentifizierung

Es wird angenommen, dass die Dateien server.xml und web.xml so konfiguriert wurden, dass Realms sowie JSTL und EL einwandfrei funktionieren. Besondere Einstellungen werden bei Bedarf nachfolgend angeführt. Auch alle benötigten JAR-Archive zur korrekten Funktion von JDBC und JSTL wurden bereitgestellt.

Sicherheitsaspekte

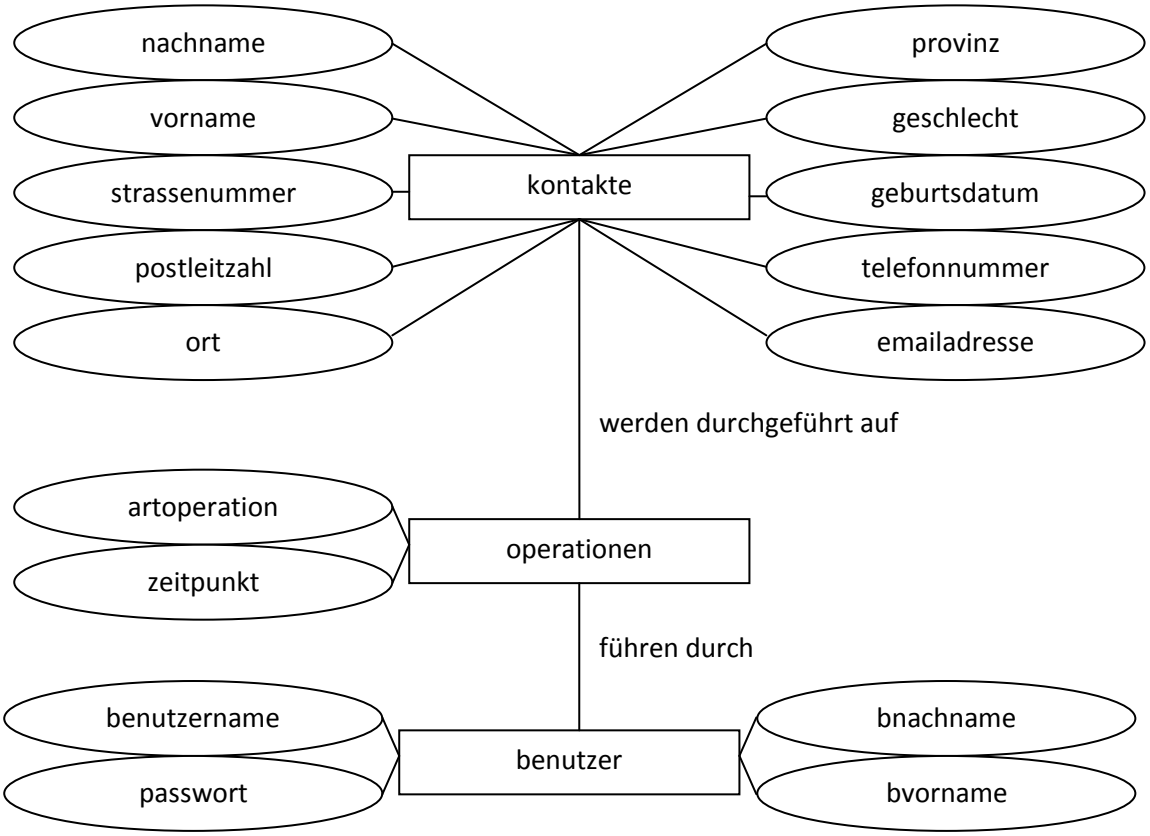
JDBC-Realms ermöglichen es, Benutzernamen und Passwörter in einer MySQL-Datenbank abzulegen, gesicherte Bereiche in Web-Anwendungen zu definieren, so dass automatisch beim Betreten eines gesicherten Bereiches der Benutzer nach Benutzername und Passwort gefragt wird.

Die Passwörter selbst werden in der Datenbank MD5-verschlüsselt als 32 Zeichen langer Text abgelegt, so dass auch bei Kenntnis dieser verschlüsselten Zeichenfolge trotzdem nicht ein Login erfolgen kann.

Zudem unterstützt Tomcat die SSL-Verschlüsselung, so dass über das gesicherte HTTPS-Protokoll der Zugriff vom Web aus auf die Daten erfolgen muss.

Um das Sessionmanagement von Tomcat zu ermöglichen, müssen am Browser des Benutzers die Cookies aktiviert werden.

Konzeptschema, Grobentwurf, Entity-Relationship-Modell



Bemerkungen

Unter Bezeichnung des Kontaktes wird Zu- und Vorname verstanden.

Aus der Aufgabenstellung geht nur hervor, dass der Zeitpunkt der Abmeldung des Technikers gespeichert werden soll. Der Anmeldezeitpunkt wird also nicht gespeichert. Um zuverlässig den Abmeldezeitpunkt erfassen zu können muss eine Bean programmiert werden, welche das Interface HttpSessionListener implementiert und im Deployment-Descriptor registriert ist. Diese Bean wird vom Tomcat immer dann benachrichtigt, wenn eine Session erstellt oder zerstört wird. Im zweiten Fall sorgt die Bean dafür, dass der Abmeldezeitpunkt für den Benutzer in die Datenbank aufgenommen wird.

Es gibt folgende Arten von Operationen welche ein Benutzer auf einen Kontakt durchführen kann:

1 bedeutet Kontakt aufrufen
2 bedeutet neuer Kontakt
3 bedeutet Kontakt ändern
4 bedeutet Kontakt löschen
5 bedeutet Benutzer meldet sich ab

Für jede dieser Operationen wird gemerkt, welcher Benutzer sie durchgeführt hat, wann die Operation stattgefunden hat, welche Art von Operation durchgeführt wurde und auf welchen Kontakt die Operation durchgeführt wurde.

Meldet sich ein Techniker ab, so wird der Kontakt der Operation leer gelassen.

Wenn ein Techniker sich eine Liste von Kontakten anzeigen lässt, so wird diese Operation nicht vermerkt. Nur wenn er einen speziellen Kontakt auswählt und sich nur diesen anzeigen lässt, dann wird diese als eigene Operation mit protokolliert. Somit muss beim Erstellen einer Liste nicht für

jeden Kontakt der in der Liste aufgenommen wird, dies mit protokolliert werden, was einen beträchtlichen Aufwand verursachen würde.

Um die nachfolgenden Abfragen übersichtlicher gestalten zu können, wird für jeden Benutzer auch noch sein Nach- und Vorname abgespeichert obwohl dies nicht in der Aufgabenstellung gefordert wird.

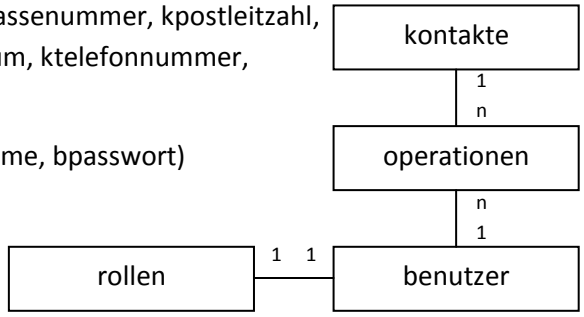
Logikschema, Feinentwurf, exakte Beziehungsdefinition, PS und FS

kontakte(knummer, knachname, kvorname, kstrassenummer, kpostleitzahl, kort, kprovinz, kgeschlecht, kgeburtsdatum, ktelefonnummer, kemailadresse, geloescht)

benutzer(bnummer, bnachname, bvorname, bname, bpasswort)

rollen(bname, rname)

operationen(onummer, bnummer, knummer, oartoperation, ozeitpunkt)



Bemerkungen

Realms setzen voraus, dass neben der Benutzertabelle auch eine Rollentabelle existiert. Diese ordnet dem Benutzer seine Rolle zu. Als Rollenname (rname) wird der Text Administrator oder Techniker eingetragen. Der Benutzer darf nur einer Rolle zugewiesen werden.

Die Tabelle operationen verbindet die Kontakte mit den Benutzern. Es entsteht eine n:m-Beziehung zwischen benutzer und kontakte.

Wenn ein Benutzer einen Kontakt löscht, dann wird dieser nicht tatsächlich von der Tabelle gelöscht. Die ist deshalb, damit nachvollzogen werden kann, welcher Kontakt gelöscht wurde. Würde der Kontakt tatsächlich gelöscht werden, dann dürft die referentielle Integrität zwischen operationen und benutzer nicht eingestellt werden, da die Kontaktnummer welche beim Löschen in die Operationentabelle eingetragen werden müsste, nicht mehr existiert. Aus diesem Grunde wird ein gelöschter Kontakt nur als gelöscht markiert.

Definition der Relationen, Feinstentwurf, exakte Datenfelddefinition

```
CREATE TABLE kontakte(
        knummer INTEGER NOT NULL AUTO_INCREMENT,
        knachname VARCHAR(100) NOT NULL,
        kvorname VARCHAR(100) NOT NULL,
        kstrassenummer VARCHAR(100),
        kpostleitzahl VARCHAR(20),
        kort VARCHAR(100),
        kprovinz VARCHAR(100) NOT NULL,
        kgeschlecht BOOL,
        kgeburtsdatum DATE,
        ktelefonnummer VARCHAR(50),
        kemailadresse VARCHAR(100),
        geloescht BOOL NOT NULL DEFAULT 0,
        PRIMARY KEY (knummer)) ENGINE = InnoDB;
```

Ein Kontakt ist ein gültiger Kontakt wenn dessen Nach- und Vorname und die Provinz (diese wird zur Durchführung der Abfrage benötigt) eingegeben werden. Die anderen Datenfelder müssen nicht eingegeben werden.

```
CREATE TABLE benutzer(
        bnummer INTEGER NOT NULL AUTO_INCREMENT,
        bnachname VARCHAR(100) NOT NULL,
        bvorname VARCHAR(100) NOT NULL,
        bname VARCHAR(30) NOT NULL,
        bpasswort VARCHAR(32) NOT NULL,
        PRIMARY KEY (bnummer),
        UNIQUE (bname)) ENGINE = InnoDB;
```

Der eingegebene Benutzername muss eindeutig sein. Weiters muss das Passwort ebenfalls eingegeben werden. Das Datenfeld des Passwortes muss 32 Zeichen lang sein, damit das MD5-codierte Passwort in diesem Datenfeld Platz findet.

```
CREATE TABLE rollen(
        bname VARCHAR(30) NOT NULL,
        rname VARCHAR(100) NOT NULL,
        PRIMARY KEY (bname),
        FOREIGN KEY (bname) REFERENCES benutzer(bname)
                ON DELETE CASCADE ON UPDATE CASCADE) ENGINE = InnoDB;
```

Weil Realms dies so vorschreiben, muss in diese Tabelle der Benutzername und nicht die Benutzernummer eingetragen werden. Sollte ein Benutzer aus der Benutzertabelle gelöscht werden oder sollte ein Benutzername geändert werden, so muss dies auch in der Rollentabelle zur Löschung bzw. Änderung führen.

```
CREATE TABLE operationen(
        onummer INTEGER NOT NULL AUTO_INCREMENT,
        bnummer INTEGER NOT NULL,
        knummer INTEGER,
        oartoperation TINYINT NOT NULL,
        ozeitpunkt TIMESTAMP NOT NULL,
        PRIMARY KEY (onummer),
        KEY (bnummer),
        KEY (knummer),
        FOREIGN KEY (bnummer) REFERENCES benutzer(bnummer)
                ON DELETE CASCADE ON UPDATE RESTRICT,
        FOREIGN KEY (knummer) REFERENCES kontakte(knummer)
                ON DELETE CASCADE ON UPDATE RESTRICT) ENGINE = InnoDB;
```

Die Kontaktnummer muss nicht zwingend eingegeben werden, denn meldet sich ein Benutzer ab, so wird dies in dieser Tabelle vermerkt, ohne dass diese Operation einen Kontakt betrifft.

Mögliche Werte für oartoperation sind 1, 2, 3, 4 oder 5. Deshalb wird hier der kleinste Integer-Datentyp verwendet.

ozeitpunkt ist vom Typ TIMESTAMP, so dass automatisch bei jedem Erstellen eines neuen Datensatzes der aktuelle Zeitstempel mit eingetragen wird und somit der Zeitpunkt der Durchführung der Operation automatisch eingetragen wird.

Wird ein Benutzer gelöscht, so werden automatisch alle von ihm durchgeführten Operationen mit gelöscht. Dies ist auch beim Löschen eines Kontaktes der Fall.

Realisierung der geforderten Abfragen in SQL

Aufgabe 1: Parameter ist die Provinz nach der gesucht wird:

```
SELECT *
        FROM kontakte
        WHERE kprovinz = ? AND geloescht = FALSE
        ORDER BY knachname, kvorname;
```

Aufgabe 2: Parameter ist der Nach- und Vorname des Benutzers:

```
SELECT o.oartoperation, o.ozeitpunkt
        FROM operationen o, benutzer b
        WHERE o.bnummer = b.bnummer
                AND b.bnachname = ? AND b.bvorname = ?
        ORDER BY o.ozeitpunkt;
```

Aufgabe 3: Parameter ist der Nach- und Vorname des Benutzers sowie die Zeitspanne. Dabei werden die beiden Zeitspannen als Datumsangaben übergeben:

```
SELECT COUNT(*) / DATEDIFF(?, ?)
        FROM operationen o, benutzer b
        WHERE o.bnummer = b.bnummer
                AND b.bnachname = ? AND b.bvorname = ?
                AND DATE(o.ozeitpunkt) BETWEEN ? AND ?;
```

Aufgabe 4: Hier wird kein Parameter übergeben:

```
SELECT b.bname, b.bnachname, b.bvorname, COUNT(*)
        FROM benutzer b, operationen o
        WHERE b.bnummer = o.bnummer
                AND o.oartoperation = 2
        GROUP BY b.bname, b.bnachname, b.bvorname
        (ORDER BY 2, 3);
```

Aufgabe 5: Als Parameter wird das Datum übergeben:

```
SELECT b.bnummer, b.bnachname, b.bvorname, o.oartoperation, o.ozeitpunkt
        FROM operationen o, benutzer b
        WHERE o.bnummer = b.bnummer
                AND DATE(o.ozeitpunkt) = ?
        ORDER BY 2, 3;
```

Aufgabe 6: Unter „aufgerufen“ wird nur das Ansehen des Kontaktes verstanden. Als Parameter wird das Beginndatum der Woche übergeben. Auch gelöschte Kontakte werden ausgegeben:

```
SELECT k.knummer, knachname , kvorname, kstrassenummer, kpostleitzahl,
        kort, kprovinz, kgeschlecht, kgeburtsdatum, ktelefonnummer, kemailadresse,
        COUNT(*)
    FROM kontakte k, operationen o
    WHERE k.knummer = o.knummer
        AND o.oartoperation = 2
        AND DATE(o.ozeitpunkt) BETWEEN ? AND ADDDATE(?,6)
    GROUP BY k.knummer, knachname , kvorname, kstrassenummer, kpostleitzahl,
        kort, kprovinz, kgeschlecht, kgeburtsdatum, ktelefonnummer, kemailadresse
    ORDER BY 12 DESC
    LIMIT 1;
```

Vorschlag zur Realisierung der Webanwendung

Die Webanwendung besteht aus einem öffentlichen Bereich, der über das Web für alle zugänglich ist. In diesem öffentlichen Bereich gibt es einen Link auf den geschützten Bereich, der – wenn er verfolgt wird – den Benutzer auffordert, sich mit Benutzername und Passwort zu authentifizieren. Wird die Authentifizierung erfolgreich durchgeführt, gelangt der Benutzer in den geschützten Bereich, der folgendermaßen aufgebaut ist:

<table>
<tr><td colspan="2">Kopfbereich (bleibt unverändert) und enthält:
Überschrift Kontaktverwaltung
evtl. Logininformationen wie Name des Technikers, Datum letzte Abmeldung usw.</td></tr>
<tr><td>Menübereich
(bleibt unverändert)
Befehle:

Suchen Kontakt
Neuer Kontakt
Logout

Statistiken
(nur Administrator)</td><td>Detailbereich:
Dieser wird je nach ausgewählten Menüpunkt ständig angepasst und enthält die nachfolgend beschriebenen Inhalte.

Hat sich der Benutzer eingeloggt und noch nicht gesucht, so wird sofort das Suchenformular angezeigt. Immer dann wenn Suchbegriffe eingegeben wurden, werden die Kontakte welche den Suchbegriffe entsprechen angezeigt</td></tr>
</table>

Unter Statistiken können die unter der vorigen Überschrift realisierten Abfragen dem Administrator des Systems zur Verfügung gestellt werden.

Suchen Kontakt

Sofort nach dem Einloggen oder wenn auf den Link Suchen Kontakt geklickt wird, erscheint im Detailbereich folgende Seite:

<table>
<tr><td colspan="2">Suchen Kontakt

Provinz: _____

Nachname: _____</td></tr>
<tr><td>Suche starten</td><td>Abbrechen</td></tr>
</table>

Der Benutzer kann die Provinz oder den Nachnamen des zu suchenden Kontaktes angeben. Sollte bereits vorher eine Kontaktsuche durchgeführt worden sein, so werden die vorher eingegeben Provinz und/oder der Nachname nochmals ausgegeben.

Der Link Abbrechen wird nur dann angezeigt, wenn vorher schon eine Kontaktsuche durchgeführt wurde, Provinz und/oder Nachname bekannt sind und die Kontaktliste schon einmal angezeigt wurde. In diesem Fall wird die vorher schon angezeigt Kontaktliste nochmals angezeigt. Mit dem Link Suche starten wird die Suche gestartet und das nachfolgende Formular Kontaktliste angezeigt.

Kontaktliste

Kontaktliste

Ausgabe der eingestellten Suchkriterien

Nachname1, Vorname1, Ort1, Telefonnummer1, E-Mail-Adresse1	Bearbeiten Löschen
…	
NachnameN, VornameN, OrtN, TelefonnummerN, E-Mail-AdresseN	Bearbeiten Löschen

Sollten keine Kontakte die den Suchkriterien entsprechen gefunden werden, so wird eine Meldung angezeigt und mit dem Link Weiter wird nochmals das Suchenformular sichtbar gemacht.

Am Beginn der Kontaktliste werden die bei der Suche eingestellten Suchkriterien nochmals ausgegeben.

Es wird aus Gründen der Einfachheit keine seitenweise Ausgabe der gefundenen Kontaktdaten realisiert, sondern alle Kontakte welche dem Suchkriterium entsprechen werden untereinander ausgegeben.

Vom gefundenen Kontakt werden nur die oben angezeigten Daten sichtbar gemacht. Kontakte können aus Gründen der Einfachheit nur einzeln gelöscht werden. Dabei erfolgt eine Sicherheitsabfrage ob der Kontakt wirklich gelöscht werden soll.

Wird auf den Link Bearbeiten geklickt, so wird das nachfolgende Formular zur Bearbeitung des Kontaktes angezeigt.

Kontakt bearbeiten/Neuer Kontakt

Dieses Formular wird sowohl beim Bearbeiten eines Kontaktes als auch beim Erstellen eines neuen Kontaktes angezeigt:

Kontakt bearbeiten/Neuer Kontakt

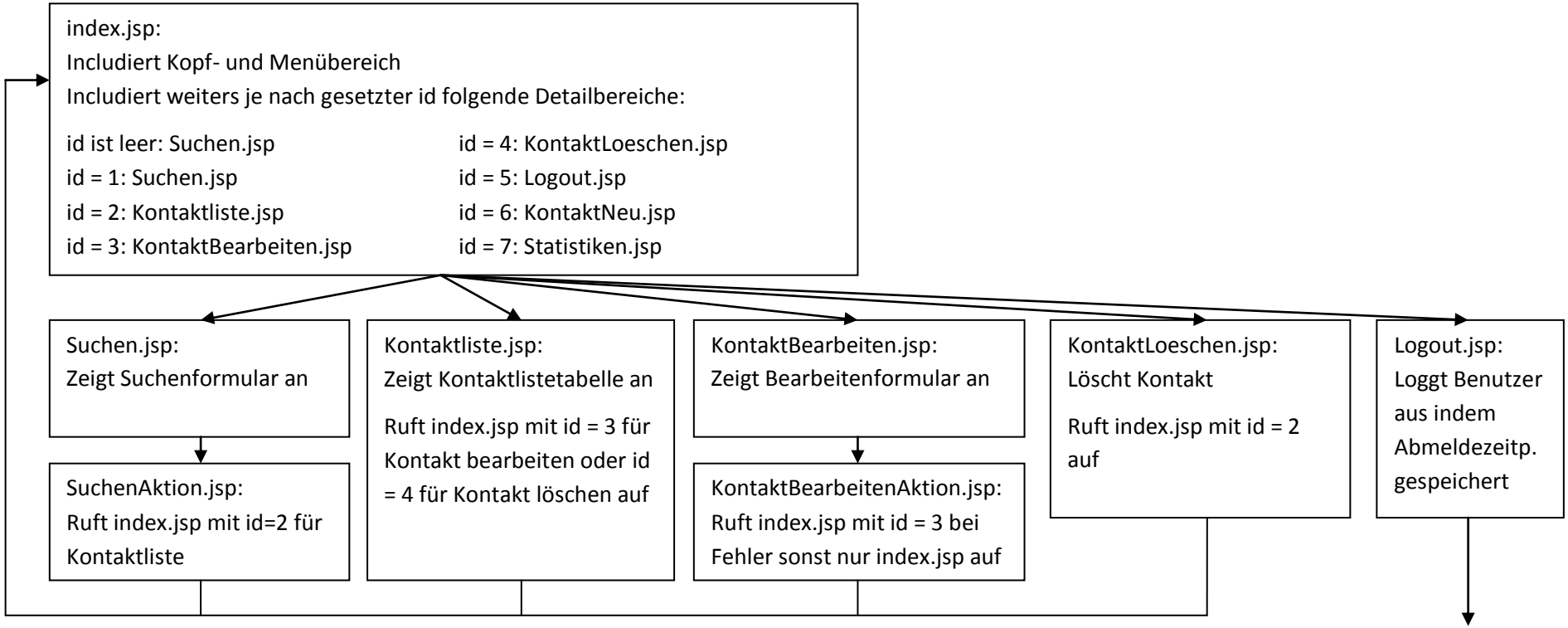
Nachname: _____
Vorname: _____
Straße, Nummer: _____
Postleitzahl: _____
Ort: _____
Provinz: _____
Männlich: ☐
Geburtsdatum: _____
Telefonnummer: _____
E-Mail-Adresse: _____

Kontakt ändern/Kontakt aufnehmen	Abbrechen

Wenn auf Abbruch geklickt wird, wird die Liste der gefundenen Kontakte wieder angezeigt. Wenn noch keine Kontaktsuche durchgeführt wurde – die Suchkriterien noch nie festgelegt wurden, so wird das Suchenformular angezeigt.

Werden die Knöpfe Kontakt ändern/Kontakt aufnehmen gedrückt, so werden die Eingaben kontrolliert. Bei Eingabefehlern wird zurück in das Formular gesprungen und die Fehlermeldungen rechts neben den Eingabefeldern angezeigt. Bei erfolgreicher Änderung/Neuaufnahme wird zurück in die Liste der gefundenen Kontakte gesprungen bzw. das Suchenformular angezeigt, wenn noch keine Kontaktsuche erfolgt ist.

Zusammenwirken der JSP-Dateien



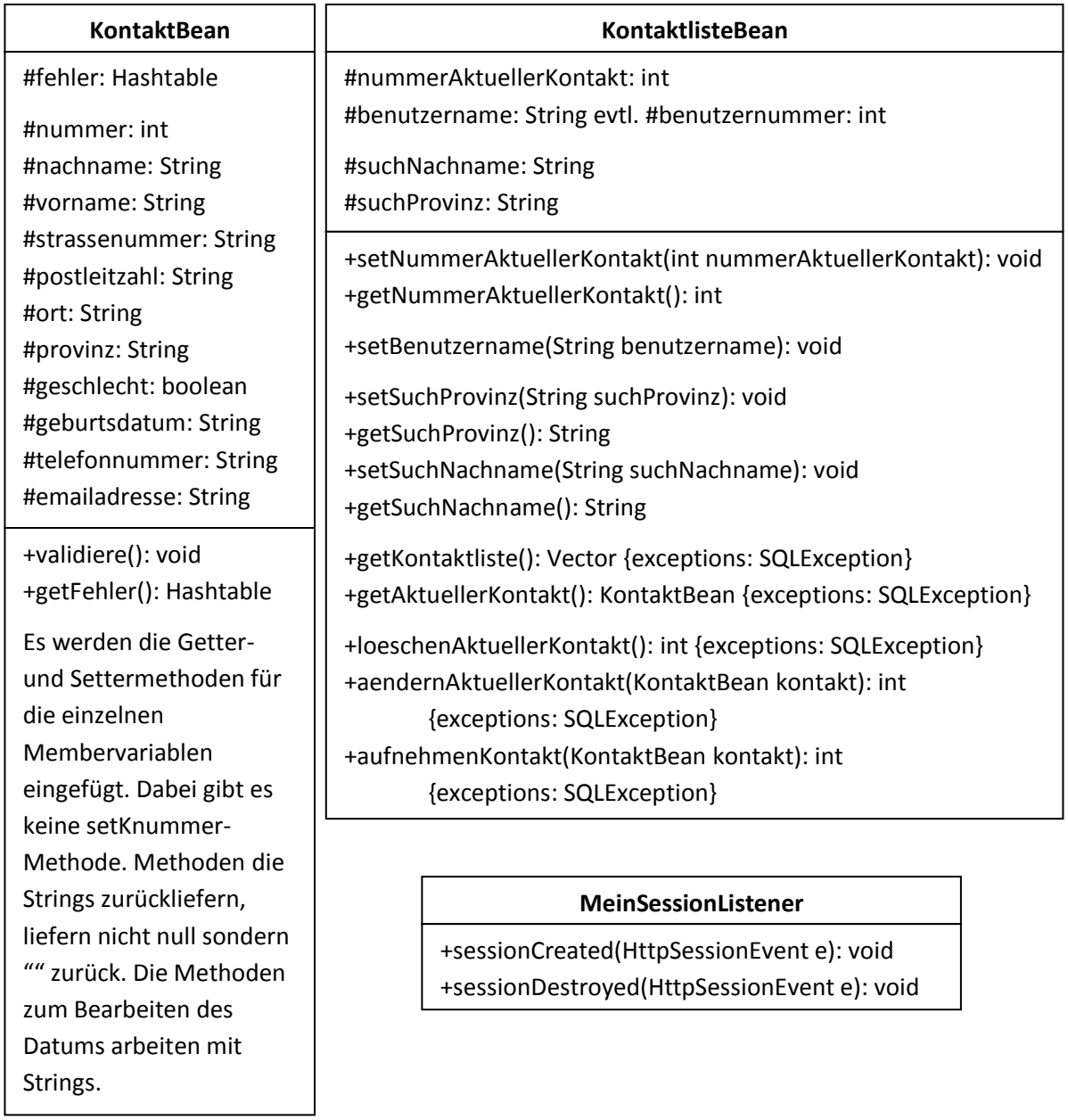
Da KontaktBearbeiten.jsp und KontaktNeu.jsp ähnlich funktionieren, wird hier zwischen den beiden Seiten nicht unterschieden.

Weiters wird beim Bearbeiten und Löschen von Kontaktdaten nicht nur die id sondern auch noch die Nummer des Kontaktes übergeben.

Loggt sich der Benutzer aus, wird in die Datei index.jsp des öffentlichen Bereiches zurück gesprungen.

Die Seite der Statistiken wird nur dem Administrator zur Verfügung gestellt. In dieser Seite kann der Administrator sich die gewünschte Statistik auswählen und anzeigen lassen. Auf diese Seite wird hier nicht genauer eingegangen.

Beans und ihre Aufgaben



Die Bean KontaktlisteBean erhält Session-Scope und wird sofort nach deren Erstellung mit dem Benutzernamen des angemeldeten Benutzers versehen. Dieser Benutzername wird verwendet um in der Operationentabelle die Operation dem Benutzer zuzuordnen. Ist der Benutzername in der Bean nicht gesetzt, so werden keine Operationen durchgeführt.

Die Methode getKontaktliste funktioniert nur dann, wenn die Provinz und/oder der Nachname der zu suchenden Kontakte festgelegt wurden. Wenn diese nicht festgelegt wurden, dann werden alle Kontakte gesucht. Die Liste der gefundenen Kontakte wird nach Provinz und bei gleicher Provinz nach Nach- und Vorname sortiert.

Die Methoden getAktuellerKontakt, loeschenAktuellerKontakt, aendernAktuellerKontakt und einfuegenKontakt werden jeweils in einer Transaktion durchgeführt. In der Transaktion wird beispielsweise der Kontakt gelöscht und diese Operation für den Benutzer in die Operationentabelle eingetragen.

In der Methode sessionDestroyed der Bean MeinSessionListener wird der Benutzername des auszuloggenden Benutzers ermittelt und für diesen in der Datenbank sein Logoutzeitpunkt eingetragen.

Codierung eines wichtigen Abschnittes

Es wird die Methode loeschenAktuellerKontakt der Bean KontaktlisteBean sowie die JSP-Datei KontaktLoeschen.jsp nachfolgen abgebildet.

Methode loeschenAktuellerKontakt

```
/**
 * Löscht den Kontakt dessen Nummer in nummerAktuellerKontakt eingetragen wurde.
 * Der Kontakt kann nur gelöscht werden, wenn diese Nummer gesetzt ist und
 * wenn der Benutzername ebenfalls bekannt ist. Weiters wird in der Tabelle
 * Operationen das Löschen mitprotokolliert
 * @return 0 falls Kontakt erfolgreich gelöscht werden konnte<br>
 * -1 falls Benutzername nicht gesetzt wurde<br>
 * -2 falls nummerAktuellerKontakt nicht gesetzt wurde<br>
 * -3 falls Kontakt nicht als gelöscht markiert werden konnte<br>
 * -4 falls die Löschoperation nicht mitprotokolliert werden konnte
 * @throws SQLException wenn ein Datenbankfehler aufgetreten ist
 */
public int loeschenAktuellerKontakt() throws SQLException {
   int ret = 0;
   if (this.benutzername == null || this.benutzername.length() == 0)
      ret = -1;
   if (ret == 0 && this.nummerAktuellerKontakt < 0)
      ret = -2;
   if (ret == 0) {
      Connection con = null;
      PreparedStatement pstmt1 = null;
      PreparedStatement pstmt2 = null;
      try {
         con = ConnectionManager.getConnection();
         con.setAutoCommit(false);
         // Lösche zuerst den Kontakt
         pstmt1 = con.prepareStatement(
            "UPDATE kontakte " +
            "  SET geloescht = TRUE " +
            "  WHERE knummer = ?;");
         pstmt1.setInt(1,this.nummerAktuellerKontakt);
         if (pstmt1.executeUpdate() == 0)
            ret = -3;
         if (ret == 0) {
            // Protokolliere die Löschung. 4 bedeutet Löschen
            pstmt2 = con.prepareStatement(
               "INSERT INTO operationen(bnummer, knummer, oartoperation) " +
               "  VALUES (?, ?, 4);");
            pstmt2.setInt(1,this.benutzernummer);
            pstmt2.setInt(2,this.nummerAktuellerKontakt);
            if (pstmt2.executeUpdate() == 0)
               ret = -4;
         }
         if (ret == 0)
            con.commit();
         else
            con.rollback();
      } catch (SQLException e) {
         throw new SQLException("loeschenAktuellerKontakt(): " + e.getMessage());
      } finally {
         try { pstmt1.close(); } catch (Exception e1) { ; }
         try { pstmt2.close(); } catch (Exception e1) { ; }
         try { con.close(); } catch (Exception e1) { ; }
      }
   }
```

```
   return ret;
}
```

JSP-Datei KontaktLoeschen.jsp

```
<%-- Definition der Fehlerseite --%>
<%@ page errorPage="Error.jsp"%>

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>

<jsp:useBean id="kontaktliste" class="kontaktverwaltung.KontaktlisteBean"
   scope="session"/>

<c:choose>
   <c:when test="${not empty param.nummer}">
      <%-- Nummer des aktuellen Kontaktes wird in Kontaktliste geschrieben --%>
      <c:set target="${kontaktliste}" property="nummerAktuellerKontakt"
         value="${param.nummer}"/>

      <c:set var="ergebnis"
         value="<%=new Integer(kontaktliste.loeschenAktuellerKontakt())%>"/>

      <c:choose>
         <c:when test="${ergebnis eq 0}">
            <%-- Löschen war erfolgreich --%>
            <jsp:forward page="index.jsp?id=2"/>
         </c:when>
         <c:otherwise>
            <%-- Beim Löschen ist ein Fehler aufgetreten --%>
            <h2>Kontakt löschen: Fehler</h2>
            <p>
               Beim Löschen des Kontaktes ist ein Fehler aufgetreten
            </p>
            <p align="right">
               <a href="index.jsp?id=2">Zurück</a>
            </p>
         </c:otherwise>
      </c:choose>
   </c:when>
   <c:otherwise>
      <jsp:forward page="index.jsp?id=2"/>
   </c:otherwise>
</c:choose>
```