Dipl.-Inform. Thomas Ziemer
www.tfh-berlin.de/~ziemer
Stand: 18. Februar 2008

2 Kapitel

WIR BAUEN UNS EINE WEB-APPLIKATION (ERSTER TEIL)

„Dreißig Speichen treffen die Nabe
– die Leere dazwischen macht das Rad.
Lehm formt der Töpfer zu Gefäßen
– die Leere darinnen macht das Gefäß.
Fenster und Türen bricht man in Mauern
– die Leere damitten macht die Behausung.

Das Sichtbare bildet die Form eines Werkes.
Das Nicht-sichtbare macht seinen Wert aus.“

(Lao-tse)

Show

Dieses Manuskript wird Sie durch die grundsätzlichen Schritte der Entwicklung, des Paketierens und des Veröffentlichens einer Web-Applikation leiten. Als Beispiel dienen zwei Fingerübungen, die sich mit dem generellen Zugriff auf eine relationale Datenbank und mit einer einfachen Benutzungsoberfläche beschäftigen (erste und dritte Fingerübung), sowie eine kleine Web-Applikation von Oracle, die Sie in die Einzelheiten der persistenten Datenspeicherung führt (zweite Fingerübung). Als Benutzungsoberfläche (*presentation layer*) wird Java-Server Faces (*JSF*) aus einem Enterprise JavaBeans-Container (*EJB*) heraus zum Einsatz kommen. Für die Datenhaltung wird die Java Persistence API (die EJB 3.0 JPA) verwendet.

Alle verwendeten Internet-Adressen waren freilich bei der Erstellung dieses Manuskripts gültig. Leider ist das Internet ein sehr dynamisches Gebilde. Im Problemfall können die benötigten Dateien dann jedoch mit hoher Sicherheit ohne großes Suchen an einer anderen Stelle gefunden werden.

2.1 Technische Voraussetzungen

Show

Bevor mit der Durchführung des „Baus“ unserer Web-Applikation begonnen werden kann, müssen verschiedene technische Voraussetzungen erfüllt sein. Im Rahmen dessen sind einige Software-Komponenten auf dem Fingerübungsrechner zu installieren.

Installation des Java Development Kits

Da unsere Web-Applikation in der Programmiersprache Java entwickelt werden soll, muss zunächst das JDK (*J2SE Development Kit*) der Java 2 Platform Standard Edition installiert werden. Es kann unter folgender Adresse im Internet heruntergeladen werden:

http://javashoplm.sun.com/ECom/docs/Welcome.jsp?StoreId=22&PartDetailId=jdk-1.5.0_06-oth-JPR&SiteId=JSC&TransactionId=noreg

Oder nehmen Sie einfach die aktuellste Version der JDK Standard Edition, die Ihnen angeboten wird. Installieren Sie während der Installation möglichst alle angebotenen Komponenten:

- Development Tools (notwendig)
- Source Code (notwendig)
- Public JRE (notwendig, beispielsweise für *Apache Tomcat*)
- Demos (empfohlen)

Ergänzend müssen zwei wichtige Umgebungsvariablen erstellt werden:

- die *JAVA_HOME*-Umgebungsvariable zeigt auf den soeben erstellten Java-Installationsordner, also beispielsweise auf *C:\Programme\Java\jdk1.5.0_06*,

und

- die *PATH*-Umgebungsvariable muss um den Ordner *bin* der Java-Installation ergänzt werden, also um *%JAVA_HOME%\bin*. Damit werden alle ausführbaren Java-Anwendungen, beispielsweise *javac* und *java* ohne Pfadangabe aufrufbar.

Installation von Ant

Apache Ant (*Another neat Tool*) ist ein in Java geschriebenes Werkzeug, das zum automatischen Erzeugen von Programmen aus Quelltext dient. Es hilft uns in diesem Fall bei der Übersetzung (*compiling*), Paketierung und Veröffentlichung (*deployment*) unserer Web-Applikation. Für Microsoft® Windows™ kann die auf der Internetseite angebotene binäre Version gewählt werden. Sie enthält bereits *Apache Xerces2*, den benötigten XML-Parser. Auf anderen Betriebssystemen muss ebenfalls dafür gesorgt werden, dass ein geeigneter XML-Parser zur Verfügung steht. Apache Ant kann unter folgender Adresse heruntergeladen werden:

http://ant.apache.org/manual/install.html#installing

Ant besteht aus einem Zip-Archiv, in dem (unter anderem) die zwei einzigen für dieses Tutorial benötigten Ordner *bin* und *lib* enthalten sind. Sie werden in einen Ordner unterhalb des Java-Installationsordners kopiert. Es bietet sich dazu an, unterhalb des Ordners *%JAVA_HOME%* einen neuen Ordner *ant* anzulegen:

- Eine zusätzliche Umgebungsvariable *ANT_HOME* muss auf den Ordner *ant* zeigen, also auf *%JAVA_HOME%\ant*.

- Die *PATH*-Umgebungsvariable muss um den Ordner *%ANT_HOME%\bin* ergänzt werden. Damit wird die Ant-Anwendung ohne Pfadangabe aufrufbar.

Sie sehen, dass sich alle Pfadangaben relativ auf die im vorangegangenen eingeführten Umgebungsvariablen beziehen. Dieses Vorgehen wird auch weiterhin in diesem Manuskript angewandt werden, um absolute Pfadangaben zu vermeiden.

Installation der Tutorial- und Beispiel-Ordner

Erstellen Sie nun einen Ordner namens *tutorial* unterhalb von *%JAVA_HOME%* (dort haben Sie im vorangegangenen auch bereits den Ordner *ant* erstellt) und kopieren Sie das ausgepackte **T u t o r i a l**-Archiv hinein. Er dient uns dazu, im folgenden die in diesem Manuskript dokumentierten Schritte zur Erstellung einer Web-Applikation auszuführen. Das Tutorial-Archiv kann unter folgender Adresse heruntergeladen werden:

http://www.oracle.com/technology/products/ias/toplink/jpa/tutorials/order-jsf-jpa-tutorial.zip

Ergänzend muss auch hier eine Umgebungsvariable erstellt werden:

- *TUTORIAL_HOME* zeigt auf den Tutorial-Ordner, also beispielsweise auf *%JAVA_HOME%\tutorial*.

Erstellen Sie sodann einen Ordner *example* unterhalb von %JAVA_HOME% (dort haben Sie eben bereits den Ordner *tutorial* erstellt) und kopieren Sie das ausgepackte Beispielarchiv hinein. Das **B e i s p i e l**-Archiv enthält die vollständigen Beispieldateien. Es dient am Ende aller Installationen, um diese zu überprüfen. Das Beispielarchiv kann unter folgender Adresse heruntergeladen werden:

http://www.oracle.com/technology/products/ias/toplink/jpa/tutorials/order-jsf-jpa-example.zip

Ergänzend muss eine weitere Umgebungsvariable erstellt werden:

- *EXAMPLE_HOME* zeigt auf den Beispielordner, also beispielsweise auf *%JAVA_HOME%\example*.

Installation des relationalen Datenbank-Management-Systems

Ein relationales Datenbank-Management-System (*RDBMS*) dient uns zur persistenten (dauerhaften) Speicherung der erzeugten Datenbestände unserer Web-Applikation. Es gibt eine Vielzahl unterschiedlicher Datenbanksysteme, einige kostenfrei, andere kostenpflichtig. Für unser Beispiel verwenden wir die für den nicht-kommerziellen Einsatz kostenfreie Express Edition der *Oracle 10g*-Datenbank (einmal etwas anderes als immer nur *MySQL*…). Sie kann unter folgender Adresse heruntergeladen werden:

http://www.oracle.com/technology/software/htdocs/xe_lic_prod.html?url=http://www.oracle.com/technology/software/products/database/xe/htdocs/102xewinsoft.html

Beachten Sie bei dem Computer, auf dem Sie dieses Tutorial ausführen (vor allem dann, wenn es sich dabei um einen virtuellen Rechner handelt), dass dieser über mindestens 256 MB Hauptspeicher verfügt. Ist dies nicht der Fall, schlägt die Installation fehl!

Merken Sie sich das während der Installation eingegebene Kennwort des Benutzers *SYSTEM* gut!

Folgende TCP-Ports werden standardmäßig verwendet. Sie werden später noch von Bedeutung sein:

1521: Oracle database listener
2030: Oracle Services for Microsoft Transaction Server
8080: HTTP port for the Oracle Database XE graphical user interface

Die Umgebungsvariable *ORACLE_HOME* wird während der Installation automatisch erstellt. Falls nicht, holen Sie dies bitte jetzt nach. Die Variable sollte auf den Ordner *C:\oraclexe\app\oracle\product\10.2.0\server* zeigen, bzw. dorthin, wo Sie Oracle installiert haben.

Der benötigte JDBC-Treiber (*Java Database Connectivity*) muss aus dem Oracle-Installationsordner *%ORACLE_HOME%\jdbc\lib* zusätzlich in den Ordner unserere Beispielanwendung nach *%JAVA_HOME%\tutorial\lib* **und** nach *%JAVA_HOME%\example\lib* kopiert werden, damit sich unsere Web-Applikation mit der Datenbank verbinden kann.

Installation des Servlet-Containers

Apache Tomcat stellt eine Umgebung zur Ausführung von Java-Code auf Webservern bereit. Es handelt sich um einen in Java geschriebenen Servlet-Container. Dazu kommt ein kompletter Webserver. Der HTTP-Server des Tomcat wird vor allem zur Anwendungsentwicklung eingesetzt, während in der Produktion zumeist ein *Apache Webserver* vor den Tomcat geschaltet wird. Tomcat kann unter folgender Adresse heruntergeladen werden:

http://tomcat.apache.org/download-55.cgi

Bitte installieren Sie möglichst alle während der Installation angebotenen Komponenten:

- Tomcat Core (notwendig)
- Webapps (notwendig für unsere erste Fingerübung)
- Tomcat Service Startup (empfohlen unter Windows)
- Tomcat Native (empfohlen)
- Start Menu Items (empfohlen)
- Documentation (empfohlen)
- Examples (wenn gewünscht)

Da der HTTP-Port des Oracle-Web-GUI (siehe oben) bereits auf Port 8080 liegt, sollten Sie während der Tomcat-Installation für den *HTTP/1.1 Connector Port* einen alternativen Port wählen, beispielsweise *8081*.

Merken Sie sich das während der Installation eingegebene Kennwort des Benutzers *ADMIN* gut!

Wählen Sie während der Installation für die *Java Virtual Machine* den Pfad der Java Laufzeit-Umgebung (*JRE*), also *%JAVA_HOME%\jre*. JRE wurde während der Installation des Java Development Kits zusätzlich installiert, da Sie die Komponente Public JRE ausgewählt haben.

Ergänzend muss einmal mehr eine sehr wichtige Umgebungsvariable erstellt werden:

- *CATALINA_HOME* zeigt auf den Tomcat-Installationsordner, also beispielsweise auf *C:\Programme\Apache Software Foundation\Tomcat 5.5*, sofern Sie keinen anderen Installationsordner gewählt haben.

Bitte kopieren Sie die beiden Oracle JDBC-Dateien *ojdbc14.jar* und *ojdbc14_g.jar* zusätzlich noch in den Ordner *%CATALINA_HOME%\common\lib*. Die Klassen werden dort für unsere erste Fingerübung (siehe weiter unten) benötigt.

Dies ist der Ordner, an dem Tomcat standardmäßig nach Java-Bibliotheken sucht.

Installation der TopLink Java Persistence API

Als Persistence Layer verwenden wir die JPA (*Java Persistence API*), eine konsequente Weiterentwicklung des bekannten Persistence Frameworks *Hibernate*. Die Installationsdateien können unter folgender Adresse heruntergeladen werden:

http://www.oracle.com/technology/products/ias/toplink/jpa/download.html

Geben Sie in einer Kommando-Shell folgenden Befehl zur Installation dieser API ein:

```
java -jar glassfish-persistence-installer-v2-b41.jar
```

Kopieren Sie die beiden entstandenen Dateien *toplink-essentials.jar* und *toplink-essentials-agent.jar* in die beiden Bibliotheksordner *%TUTORIAL_HOME%\lib* (siehe oben) **und** *%EXAMPLE_HOME%\lib*.

Bearbeiten Sie die Datei *persistence.xml* in den Ordnern *%EXAMPLE_HOME%\persistence-unit\src\META-INF* **und** *%TUTORIAL_HOME%\persistence-unit\src\META-INF*. Stellen Sie sicher, dass darin in den Zeilen der Eigenschaften *toplink.jdbc.user* und *toplink.jdbc.password* die während der Installation der Oracle-Datenbank angebenen Zugangsdaten eingetragen sind. Der JDBC-Treiber und die JDBC-URL sind bereits korrekt eingetragen – sofern Sie nicht mit MySQL arbeiten.

2.2 Überprüfung der Installationen

Es ist überstanden! Sie haben alle notwendigen Installationen durchgeführt.

Show

Um das Zusammenspiel aller Komponenten überprüfen zu können, haben Sie zusätzlich zum Oracle-Tutorial noch ein anderes Oracle-Beispiel installiert. Dieses ist bereits vollständig lauffähig, und dient uns zur Verifikation der Installation.

Geben Sie in einer Kommando-Shell im Ordner *%EXAMPLE_HOME%* folgende Befehle ein:

```
ant -f build.xml generate-tables
```

Es werden die für das Beispiel notwendigen relationalen Datenbanktabellen erzeugt. Achten Sie auf eventuelle Fehlermeldungen. Meldungen wie *Table or view not existing* können jedoch ignoriert werden.

```
ant –f build.xml populate-data
```

Es werden die Beispielinhalte in die obigen Tabellen geschrieben.

```
ant -f build.xml package.webapp
```

Die Web-Applikation wird veröffentlicht.

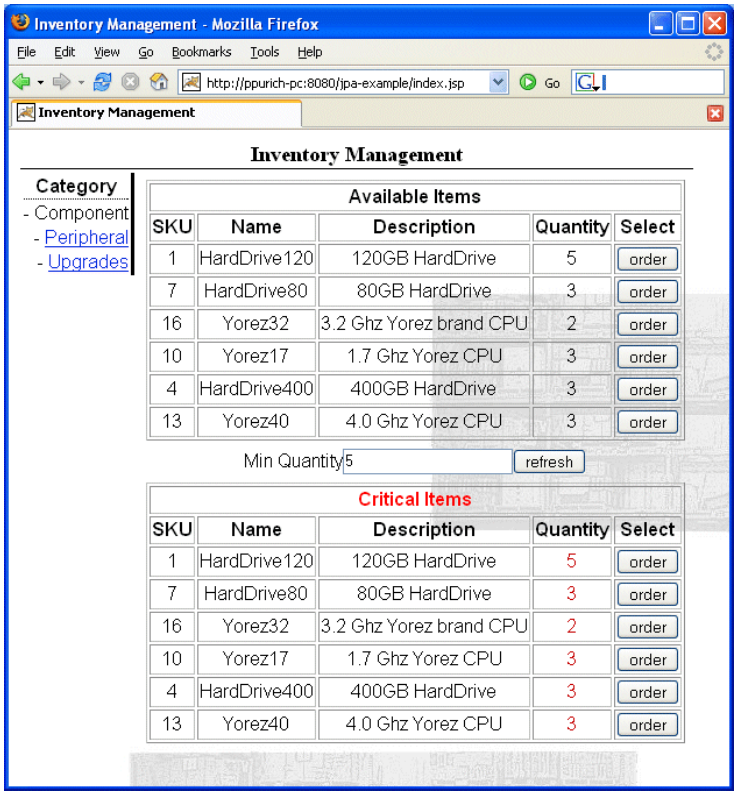
Kopieren Sie die durch den Veröffentlichungsvorgang entstandene, ausführbare Datei *jpa-example.war* aus dem Ordner *%EXAMPLE_HOME%\web-application\deploy* in den Ordner *%CATALINA_HOME%\webapps*.

Show

Rufen Sie jetzt die Adresse

```
http://<hostname>:8081/jpa-example/index.jsp
```

auf. Ersetzen Sie *<hostname>* durch den Namen des Tomcat-Servers. Sie sollten folgende (oder zumindest eine sehr ähnliche) Webseite sehen:



Wenn das der Fall ist, haben Sie alle Installationen vollständig und korrekt durchgeführt und sind bereit, die erste Fingerübung durchzuführen.

2.3 Zugriff auf die Oracle-Datenbank (Erste Fingerübung)

Um ein Gefühl für *Apache Tomcat*, den Servlet-Container, und die Datenbank *Oracle 10g* Express Edition zu bekommen, soll es uns in einer ersten Fingerübung genügen, den Zugriff auf die Datensätze einer einzelnen Tabelle zu realisieren. Wir verwenden dazu die Tabelle *Item*.

Die Servlet-Datei *index.jsp*

Show

Wir benötigen zwei Dateien. Zum einen das Servlet *index.jsp*, das uns eine minimale Benutzungsoberfläche bietet (, die wir in der dritten Fingerübung noch erweitern werden). Erstellen Sie diese Datei im Ordner *%CATALINA_HOME%\webapps\fingeruebung*. Den Ordner müssen Sie vorher anlegen. Er existiert noch nicht.

Die blauen Zeilennummern vor den einzelnen Zeilen des Dateiinhalts sind natürlich **kein** Bestandteil der Datei, sondern dienen allein der besseren Referenzierung innerhalb dieses Manuskripts:

```
01 <jsp:useBean id="orderDb" class="beans.DbBean" scope="request"/>
02 <html>
03   <body>
04     <%
05       orderDb.open("system", "<password>");
06       orderDb.query("SELECT name FROM Item");
07       while(orderDb.next()) {
08         out.println(orderDb.getString(1) + "<br/>");
09       }
10       orderDb.close();
11     %>
12   </body>
13 </html>
```

In der Zeile 01 wird eine JavaBean referenziert. JavaBeans sind Softwarekomponenten für die Programmiersprache Java. Sie werden als Container zur Datenübertragung verwendet. Wir verwenden eine einfache JavaBean zum Zugriff auf die Oracle-Datenbank. Die JavaBean erhält eine *id="orderDb"*, über die sie künftig in diesem Servlet angesprochen wird. Es wird die Klasse *beans.DbBean* von ihr referenziert. Wir schauen uns diese Klasse in Kürze an.

Die Zeilen 04 bis 11 greifen mittels Java-Code, der in das Servlet durch die Zeichen *<%* bzw. *%>* eingebettet ist, auf *DbBean* zu. In der Zeile 05 wird die Oracle-Datenbank geöffnet (ersetzen Sie *<password>* durch das tatsächliche Kennwort, das Sie während der Oracle-Installation angegeben haben) und in der Zeile 06 ein SQL-Statement abgesetzt. In den Zeilen 07 bis 09 wird auf das zurück gelieferte ResultSet zugegriffen. In der Zeile 10 wird die Datenbank schließlich wieder geschlossen.

Show

Die JavaBean *DbBean.java*

Schauen wir uns nun den Inhalt von *DbBean.java* an. Diese Datei muss im Ordner *%CATALINA_HOME%\webapps\fingeruebung\WEB-INF\classes\beans* abgelegt werden. Der Ordner *classes* ist standardmäßig derjenige Ort, an dem durch den Tomcat nach ausführbaren Klassen, Resourcen etc. gesucht wird. Erstellen Sie in diesem Zusammenhang bitte auch alle noch nicht existierenden Ordner:

```
01 package beans;
02
03 import java.sql.*;
04
05 public class DbBean {
06
07   // Datenbankverbindung
08   private Connection con = null;
09   private Statement st = null;
10   private ResultSet rs = null;
[…]
```

Zeile 01: die Klasse befindet sich im Namensraum *beans*. Die Angabe des Namensraumes ist sehr wichtig! Die Klasse importiert in Zeile 03 alle SQL-Bibliotheken, die zum Java-Zugriff auf Datenbanken benötigt werden.

Show

```
[…]
11   public DbBean() {}
[…]
```

In jeder JavaBean ist ein öffentlich zugänglicher Konstruktor zwingend vorgeschrieben. Im einfachsten Fall, wie hier in der Zeile 11, ist er leer.

```
[…]
12   public void open(String user, String passwd) {
13     final String URL = "jdbc:oracle:thin:@<hostname>:1521:<sid>";
14     final String DRIVERCLASS = "oracle.jdbc.OracleDriver";
15
16     try {
17       Class.forName(DRIVERCLASS);
18       this.con = DriverManager.getConnection(URL, user, passwd);
19     }
20     catch (Exception e) {
21       System.out.println("Db-Connect gescheitert");
22     }
23   }
[…]
```

Die Operation *open()* öffnet eine Datenbankverbindung. Ersetzen Sie *<hostname>* in *URL* in der Zeile 13 durch den Namen Ihres Tomcat-Servers. Ersetzen Sie die *<sid>* durch *XE*.

Zur Laufzeit der Anwendung wird in Zeile 17 ein Objekt der Klasse *DRIVERCLASS* erstellt und in Zeile 18 mit ihr eine Verbindung zur Datenbank hergestellt. Enthält das zugrunde liegende RDBMS mehrere Datenbanken (was in unserem Beispiel jedoch **nicht** der Fall ist), muss in der Verbindung mittels *getConnection()* zusätzlich die gewünschte Datenbank angegeben werden. Dazu würde der Code *getConnection(URL + database, user, passwd);* dienen.

Greifen Sie auf eine *MySQL*-Datenbank zu, verwenden Sie die Driver Class *org.gjt.mm.mysql.Driver* und den URL *jdbc:mysql://<hostname>/*. Vergessen Sie dann auch nicht, den JDBC-Treiber für MySQL zu installieren.

Show

```
[…]
24    public ResultSet query(String query) {
25      try {
26        this.st = this.con.createStatement();
27        this.rs = this.st.executeQuery(query);
32        return this.rs;
28      }
29      catch (Exception e) {
30        System.out.println("Db-Query gescheitert");
31        return null;
32      }
33    }
[…]
```

In der Operation *query()* wird in Zeile 26 mittels *createStatement()* der Statement-Kontext erstellt bzw. das Statement datenbankseitig kompiliert und in Zeile 27 durch *executeQuery()* das übergebene SQL-Statement ausgeführt.

Show

```
[…]
34    public boolean next() {
35      try {
36        return this.rs.next();
37      }
38      catch (Exception e) {
39        System.out.println("Db-Fetch gescheitert");
40        return false;
41      }
42    }
[…]
```

In der Operation *next()* wird zum nächsten Datensatz des ResultSets navigiert. Bevor auf ein ResultSet zugegriffen werden kann, muss mit *next()* zum ersten Datensatz navigiert werden. Die Operation liefert so lange *true* zurück, solange noch weitere Datensätze verfügbar sind.

Show

```
[…]
43    public String getString(int index) {
44      try {
45        return this.rs.getString(index);
46      }
47      catch (Exception e) {
48        System.out.println("Db-Select gescheitert");
49        return null;
50      }
51    }
[…]
```

Die Operation *getString()* greift auf ein bestimmtes Attribut mit dem *index* des ausgewählten Datensatzes zu. Die Attribute sind fortlaufend indiziert, beginnend bei 1.

Show

```
[…]
52    public void close() {
53      try {
54        this.st.close();
55        this.con.close();
56        this.rs = null;
57      }
58      catch (Exception e) {
59        System.out.println("Db-Disconnect gescheitert");
60      }
61    }
62 }
```

Letztlich schließen wir mit *close()* die Datenbank wieder ordnungsgemäß und geben die Objekte frei.

2.4 Überprüfung der Web-Applikation

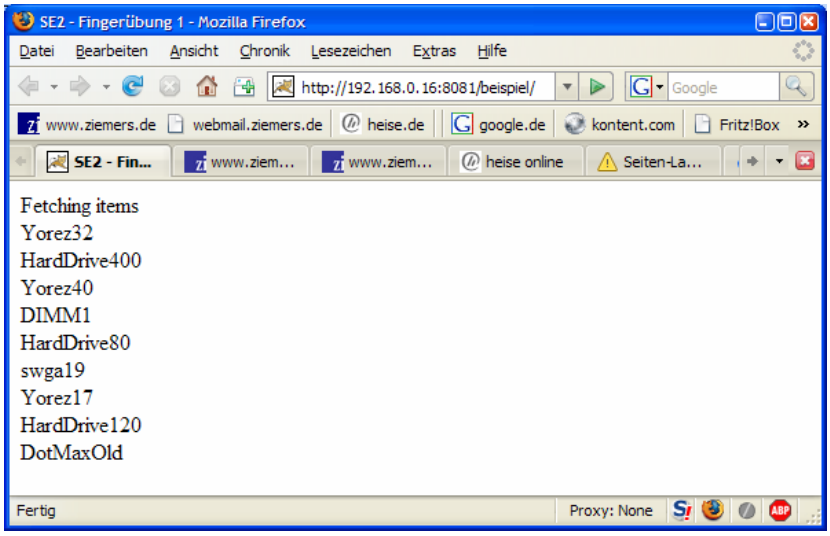
Übersetzen Sie die JavaBean. Wechseln Sie dazu in einer Kommando-Shell in den Ordner *classes\beans*, der die entprechende Java-Datei enthält, und führen Sie folgendes Kommando aus:

```
javac DbBean.java
```

Nun können Sie die Anwendung im Webbrowser starten:

```
http://<hostname>:8081/fingeruebung/index.jsp
```

Ersetzen Sie <*hostname*> durch den Namen Ihres Tomcat-Servers. Sie sollten folgenden Browser-Inhalt sehen, oder zumindest eine sehr ähnliche Webseite (starten Sie bei Bedarf den Dienst des Tomcat-Servers neu):



Sollte es zu Fehlern kommen, lohnt häufig ein Blick in die Log-Dateien des Tomcat-Servers, die sie im Ordner *%CATALINA_HOME%\logs* finden können.

Bitte führen Sie diese gesamte erste Fingerübung im Rahmen einer Übungsveranstaltung der Lehrveranstaltung *Software Engineering II* durch.