Supplementum zum Thema Webanwendungen über XML-Spezifikationen

– Verdeutlicht am Beispiel eines holländischen Auktionssystems –

Daniela Stoykova

April, 2005

Inhaltsverzeichnis

Anhang A

Installation der Software

A.1 JDK

Die Installationsdatei *j2sdk-1_4_2_01-windows-i586-iftw.exe* wird von der Webadresse *http://java.sun.com/j2se/downloads.html* heruntergeladen und unter dem Verzeichnis *C:* der lokalen Festplatte installiert. Nach der erfolgreichen Installation werden die Umgebungsvariablen *JAVA_HOME* auf dem *<inst dir>* (hier auf *C:\j2sdk1.4.2_01*) und *PATH* auf dem *<inst dir>\bin* gesetzt (hier auf *C:\j2sdk1.4.2_01\bin*). Der Erfolg der Installation kann geprüft werden, indem im DOS-Fenster *java -version* eingegeben wird. Bei erfolgreicher Installation wird die Version der Java Installation ausgegeben.

Die Dokumentation von Java ist unter *http://java.sun.com/j2se/1.4/docs/-index.html* erreichbar und steht auch auf *http://java.sun.com/j2se/1.4/download.html* als Download bereit. Diese kann z.B. unter *C:\j2sdk1.4.2* entpackt und damit installiert werden. Die Installation erstellt unter dem Installationsverzeichnis auch ein *docs*-Verzeichnis. Zum Ansehen der Dokumentation sollte die *docs/index.html* geöffnet werden.

A.2 Tomcat Server

Die Installationsdatei *tomcat-4.1.27.zip* für die Tomcat Version 4.1.27 wird von der Adresse *http://jakarta.apache.org/site/binindex.cgi* heruntergeladen und unter dem Verzeichnis *C:* der lokalen Festplatte entpackt. Hiermit ist die Installation von Tomcat abgeschlossen. Es muss noch die Umgebungsvariable *CATALINA_HOME* auf *<inst dir>* (hier *C:\tomcat4*) gesetzt werden. Die Lauffähigkeit des Tomcat Servers wird durch Ausführen der *startup.bat* unter Windows im Verzeichnis *%CATALINA_HOME%\bin* getestet. Der Server startet und ist im Webbrowser unter *http://localhost:8080* oder *http://127.0.0.1:8080* erreichbar. *8080* ist die Standardportnummer, auf dem der Tomcat Server läuft. Bei erfolgreicher Installation des Tomcat Servers wird

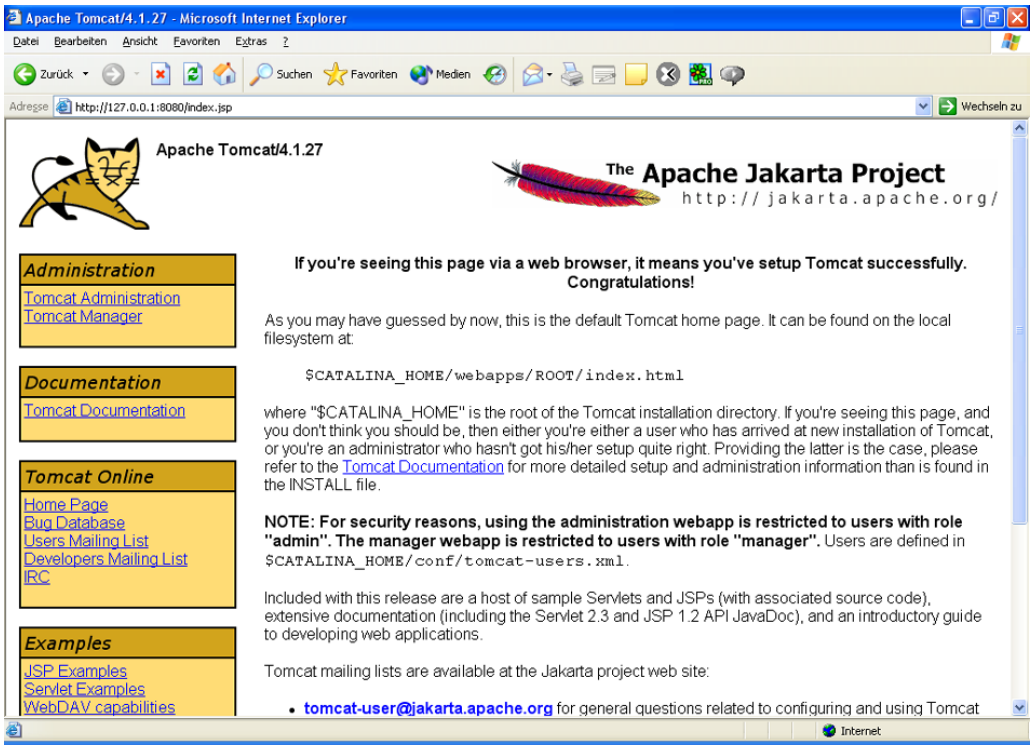


Abbildung A.1: *Index*-Seite des Tomcat Servers

beim Aufruf der obigen Adressen das Bild im Browser wie auf Abbildung A.1 angezeigt. Die Dokumentation von Tomcat ist unter der Adresse *http://localhost:8080/tomcat-docs/index.html* erreichbar.

A.3 Borland Interbase

Die Installationsdatei *InterBase_WI-V6.0.1-server.ZIP* für den Open Sourse InterbaseServer in der Version 6.01 wurde unter *http://info.borland.com/devsupport/interbase/opensource/* heruntergeladen. Diese sollte auf einem temporären Verzeichnis der Festplatte entpackt und durch ausführen der *setup.exe*-Datei installiert werden. Der Installationsverzeichnis lautet hier *C:\Programme*. Die Installation erstellt das Verzeichnis *Borland\InterBase*, in welches sich die eigentliche Installation befinden wird.

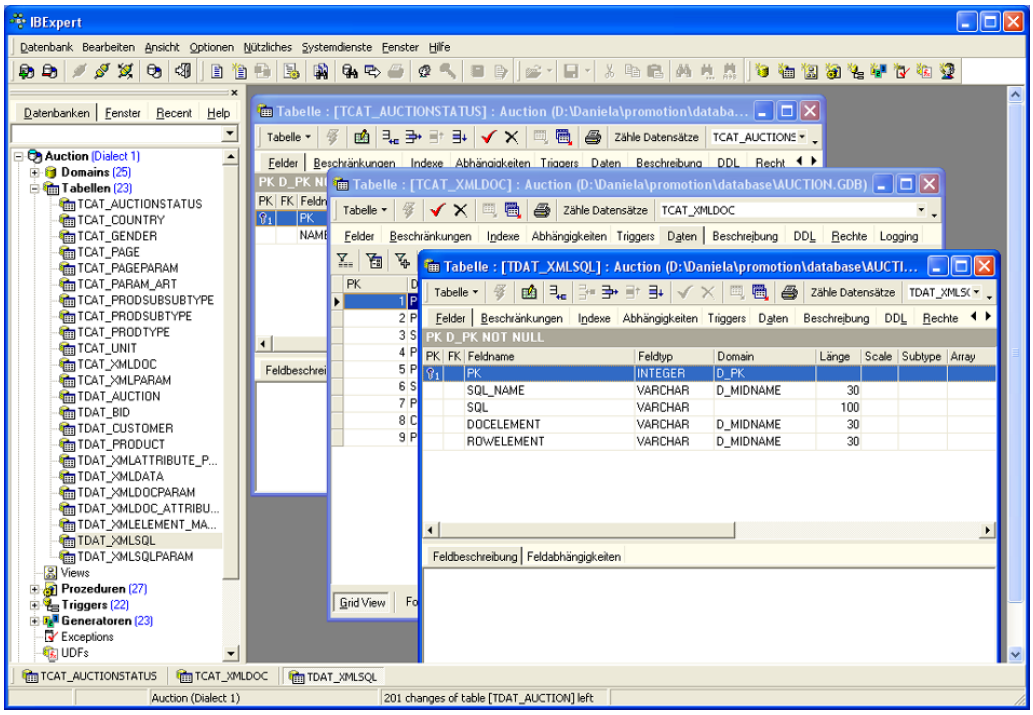


Abbildung A.2: IBExpert Screenshot

A.4 IB Expert für Interbase

Als Clientsoftware zum Editieren der Interbase Datenbank wird IBExpert eingesetzt. Die Installationsdatei *ibep_2.5.0.6_full.EXE* kann unter *http://www.hk-software.net/download/* heruntergeladen werden. Nach der Installation kann mit der Software eine neue Datenbank erstellt und editiert oder die Verbindung zu einer bereits existierenden Datenbank eingerichtet werden. Ein Screenshot ist aus der Abbildung A.2 ersichtlich.

A.5 Entwicklungsumgebung Eclipse

Als Entwicklungsumgebung wird in dieser Arbeit die Open Source Software Eclipse in der Version 2.1.3 eingesetzt. Die Installationsdatei ist auf der Adresse *http://www.eclipse.org/downloads/index.php* zu finden. Nach dem Herunterladen sollte diese lokal z.B. auf *C:* entpackt werden. Diese Aktion erstellt an dieser Stelle den Pfad *C:\eclipse* und schreibt die notwendigen Dateien in das Verzeichnis. Hiermit ist die Installation der Entwicklungsumgebung abge-

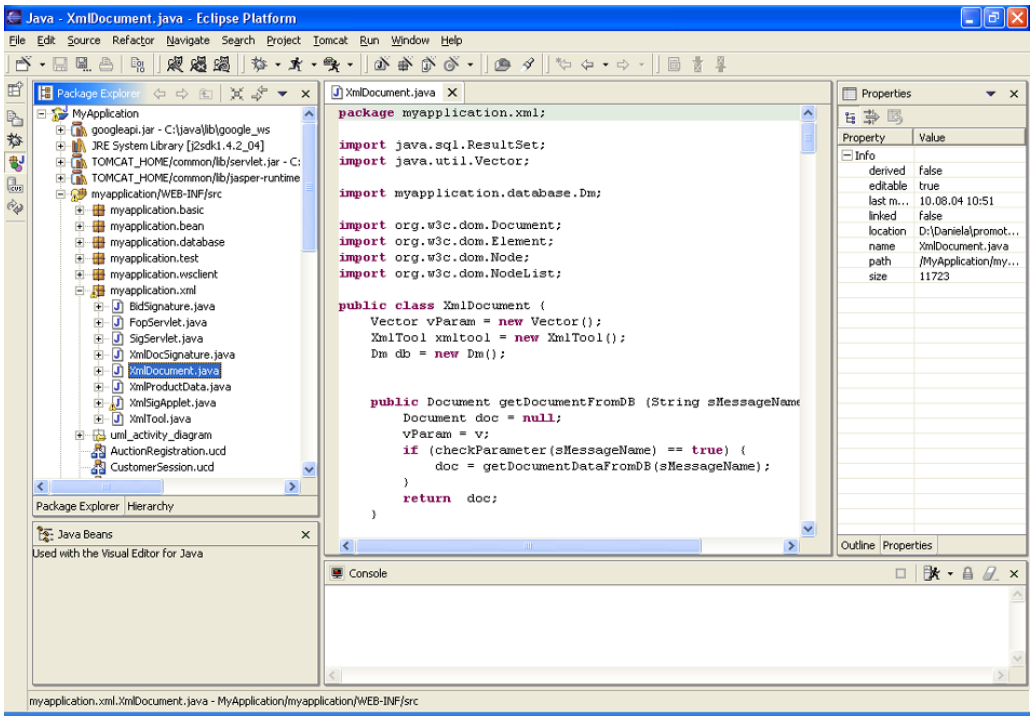


Abbildung A.3: Die Eclipse Entwicklungsumgebung

schlossen. Zum Testen der Funktionalität der Software sollte das JDK bereits installiert sein und die Umgebungsvariable *%JAVA_HOME%* auf den Installationsverzeichnis (hier *C:\j2sdk1.4.2_01*) gesetzt sein. Beim Starten von Eclipse wird das JDK automatisch erkannt. Die Java VM (Java Virtual Machine) kann auch manuell über die Option *-vm <javaw.exe Ort>* spezifiziert werden. Ein Screenshot der Eclipse-Entwicklungsumgebung kann aus der Abbildung A.3 entnommen werden.

A.5.1 Sysdeo Eclipse Tomcat Plugin

Die Binärdatei des Plugins *tomcatPluginV096.zip* im Zip-Format kann unter http://www.sysdeo.com/eclipse/tomcatPlugin.html heruntergeladen werden. Die Installation des Plugins besteht aus dem Entpacken der *zip*-Datei unter dem *plugins*-Ordner der Eclipse Installationsverzeichnis. An dieser Stelle wird das Sysdeo Plugin-Verzeichnis *com.sysdeo.eclipse.tomcat* erstellt. Die Konfiguration des Plugins erfolgt innerhalb der Entwicklungsumgebung Eclipse und besteht aus folgenden Schritten:

- Konfiguration des Plugins:
  Wählen des Menüpunktes *Window* - Unterpunkt *Preferences* und danach die Option *Tomcat* im linken Bereich des neuen Fensters. (siehe Abbildung A.4 a) Rechts wird die Tomcat Version (z.B. *4.1.x*) und das Tomcat Home-Verzeichnis (z.B. *C:\tomcat4*) ausgewählt. Unter der Suboption *JVM Settings* der Option *Tomcat* soll unter Classpath die *tools.jar*-Pfad angegeben werden (siehe Abbildung A.4 b). Der Pfad kann wie folgt aussehen: C:\j2sdk1.4.2_01\lib\tools.jar. Dieser wird zum Kompilieren von JSPs gebraucht.

- Setzen der Umgebungsvariable TOMCAT_HOME:
  Wählen des Menüpunktes *Window* - Unterpunkt *Preferences*. Im linken Bereich des neu geöffneten Fensters ist die Option *Java* auszuwählen. Unter dem Unterpunkt *Classpath Variables* sollte auf der rechten Seite des Fensters nach der Auswahl des Buttons *New...* der neue Variablenname TOMCAT_HOME und über den *File* Button dem Tomcat-Installationsverzeichnis ausgewählt werden (hier C:\tomcat4).

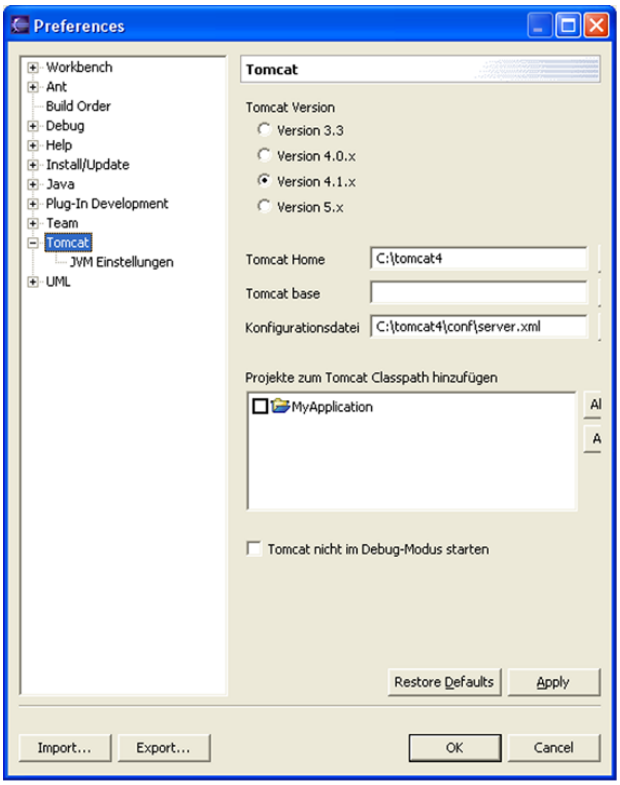
- Bereitstellung des Tomcat-Menüs:
  Wählen des Menüpunktes *Window - Open Perspective - Java*. Wenn das Tomcat Menü nicht zur Verfügung steht, sollte über das Menü *Window - Customize Perspective...* ausgewählt werden. Im linken Bereich sollte hier im Baumverzeichnis der Punkt *Other - Tomcat* ausgewählt werden. Diese Option stellt das Tomcat Menü zum Starten, Stoppen und Wiederstarten des Tomcat Servers zur Verfügung.

A.5.2 Installation des Tomcat Patches zum Debuggen von JSP-Seiten

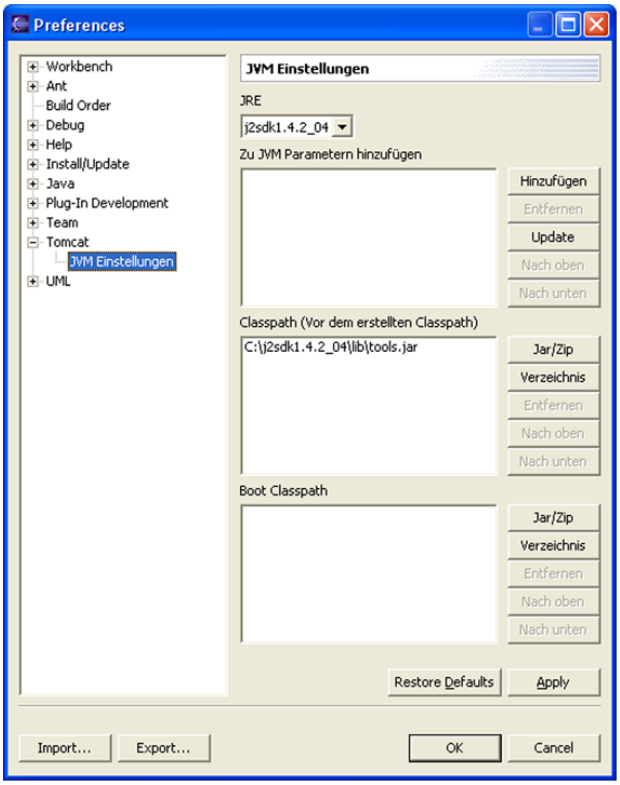
Der Patch wird unter der Webseite *http://www.sysdeo.com/eclipse/tomcatPlugin.html*. Hier sollte die *jasperDebugPatchV4.1.24.zip*-Datei heruntergeladen werden. Die Installation besteht aus dem Entpacken der zip-Datei unter *%TOMCAT_HOME%* classes für Tomcat 4.0.x oder unter *%TOMCAT_HOME%\common \classes* für Tomcat 4.1.x.

A.6 JDBC-Treiber FirebirdSQL

Der JDBC-Treiber steht auf der Webseite *http://sourceforge.net/project/showfiles.php?group_id=9028&release_id=172623* als *zip*-Datei zum Download (*FirebirdSQL-1.0.1.zip*). Nach dem Herunterladen sollte der Inhalt der Datei auf



a)



b)

Abbildung A.4: *Preferences*-Fenster für die Einstellungen des *Sysdeo Eclipse Tomcat Plugin*

Abbildung A.5: Menü-Einträge des Tomcat-Plugins im Eclipse

der lokalen Festplatte geschrieben werden (z.B. C:\java \lib). Zum Schluss sollten die Dateien *firebirdsql.jar*, *mini-concurrent.jar* und *mini-j2ee.jar* zum Klassenpfad hinzugefügt werden. Damit ist die Installation des Treibers beendet.

A.7 Apache Axis

Apache *Axis* ist die Implementierungssoftware für SOAP und Web Services. Als Web Container wird der Apache Tomcat Server eingesetzt. Dieser sollte durch ausführen der *startup.bat*-Datei im *bin*-Verzeichnis des Tomcat Installationsverzeichnisses zuerst gestartet werden. Die korrekte Arbeitsweise des Tomcat Servers wird durch den Aufruf der Adresse *http://127.0.0.1:8080* oder *http://localhost:8080* geprüft. Das *Axis* Paket kann unter *http://ws.apache.org/axis/releases.html* heruntergeladen und entpackt werden. Der Inhalt vom Ordner *Webapps* sollte unter dem Tomcat Installationsverzeichnis im Ordner *Webapps* kopiert werden. Der Name der *Axis*-Ordner unter dem *Webapps*-Ordner kann einen beliebigen Namen tragen. Dieser ist für den Aufruf vom Browser aus wichtig. Die Funktion von Apache *Axis* kann durch Aufruf der Adresse *http://localhost:8080/axis*, nachdem der Tomcat Server erfolgreich gestartet wurde, überprüft werden. Wenn die Installation erfolgreich war erscheint eine Begrüßungswebseite wie auf Abbildung A.6 zu sehen ist. Von dieser kann die Validierung der lokalen Installation vorgenommen werden oder eine Liste der bereits veröffentlichten (deployed) Services angezeigt werden. Weiterhin sollte der Klassenpfad auf die folgenden Dateien vom *lib*-Ordner des *Axis* Pakets gesetzt werden: *axis.jar*, *jaxrpc.jar*, *saaj.jar*, *commons-logging.jar*, *commons-discovery.jar*, *wsdl4j.jar*. Die *XmlParser*-Klassen müssen im Klassenpfad gesetzt werden. Die einzubindenden Klassen von *Xerces* sind *xercesImpl.jar* und *xmlParserAPIs.jar*.

Integration von Apache Axis in der Web Anwendung

Die Integration von Apache *Axis* in eine Web Anwendung enthält folgende Schritte:

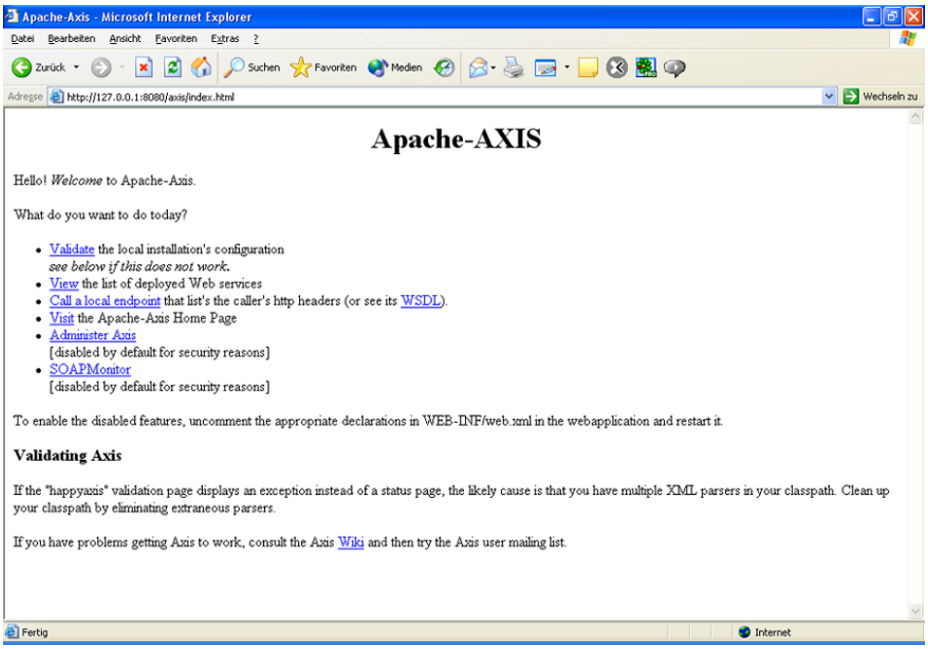


Abbildung A.6: Screenshot des Aufrufs von Apache *Axis*

- Folgenden *jar*-Dateien werden im Verzeichnis *WEB-INF/lib* der Web Anwendung eingefügt: *axis.jar*, *wsdl.jar*, *saaj.jar*, *jaxrpc.jar*, *commons-logging.jar*, *commons-discovery.jar* und alle abhängigen.
- Alle Servlet Deklarationen und Mappings werden von *axis/WEB-INF/web.xml* in die *web.xml*-Datei der Web Anwendung kopiert
- Nach dem Deployment der Web Anwendung können folgende Adressen aufgerufen werden:
  - http://127.0.0.1:8080/myapplication/services/AdminService
  - http://127.0.0.1:8080/myapplication/servlet/AxisServlet
  - http://127.0.0.1:8080/myapplication/axis/Calculator.jws
  - http://127.0.0.1:8080/myapplication/happyaxis.jsp

Anhang B

Ausgewählte Quelltexte

B.1 Ausgewählte JSP-Seiten

Die JSP-Seite complete_registration.jsp

Listing B.1: *complete_registration.jsp*-Seite

```
<%@ page errorPage="error.jsp" %>
<jsp:useBean id = "regSession" class =
"myapplication.bean.NewRegSession" scope = "request" />
<jsp:setProperty name="regSession" property="*"/>

<%
String show = "login.jsp"; boolean areFieldsValid = true;
  try {
      if(regSession.checkForEmptyFields() == false){
         show = "register.jsp";
         request.setAttribute("message", "Alle obligatorische
         Felder muessen ausgef&uuml;llt werden!");
         areFieldsValid = false;
      }
      if(areFieldsValid){
         if(regSession.isEmailValid() == false){
             show = "register.jsp";
             request.setAttribute("message", "Die eingegebene
             Email-Adresse ist nicht gueltig");
             areFieldsValid = false;
         }
      }
      if(areFieldsValid){
         if(regSession.isUserValid() == false){
             show = "register.jsp";
             request.setAttribute("message", "Der Usernamen " +
             regSession.getUserid() + " existiert bereits. Bitte
             waehlen Sie einen anderen Usernamen!");
             areFieldsValid = false;
         }
      }
```

```
32      if(areFieldsValid){
33         if(regSession.arePassFieldsValid() == false){
34             show = "register.jsp";
35             request.setAttribute("message", "Das Passwort und
36             die Passwortbestaetigung sind nicht identisch oder
37             sind kuerzer als 8 Zeichen!");
38             areFieldsValid = false;
39         }
40      }
41      if(areFieldsValid){
42         regSession.addUser();
43         request.setAttribute("text", "Herzlichen üGlckwunsch!
44         Sie wurden im System erfogreich angemeldet.");
45      }
46  } catch (Exception e){
47     e.printStackTrace();
48  }
49  regSession.closeDb();
50 %>
51 <jsp:forward page="<%= show %>"/>
```

Die JSP-Seite prodtype.jsp

Listing B.2: *prodtype.jsp*-Seite

```
1 <%@ page contentType="text/html"%>
2 <%@ page errorPage="error.jsp" %>
3 <%@ page import="org.w3c.dom.Document" %>
4 <%@ page import="myapplication.xml.XmlProductData" %>
5 <%@ page import="myapplication.xml.XmlTool" %>
6 <% out.clear(); %>
7 <jsp:useBean id="currentpage" class="myapplication.bean.Page"
8                               scope="session" />
9 <%
10     currentpage.setPageNr(2);
11 %>
12 <%@ include file="include_files/head.html" %>
13 <%@ include file="include_files/body_begin.html" %>
14 <%@ include file="menue.jsp" %>
15 <!-- ************ B E G I N  C O N T E N T ************ -->
16
17 <%@ include file="include_files/xmldoc.html" %>
18
19 <!-- ************ E N D  C O N T E N T ************ -->
20 <%@ include file="copyright.html" %>
21 <%@ include file="include_files/body_end.html" %>
```

Die JSP-Seite prodsubtype.jsp

Listing B.3: *prodsubtype.jsp*-Seite

```
1 <%@ page contentType="text/html"%>
2 <%@ page errorPage="error.jsp" %>
3 <%@ page import="org.w3c.dom.Document" %>
4 <%@ page import="myapplication.xml.*" %>
5 <jsp:useBean id = "category" class = "myapplication.bean.Category"
6                                scope = "session" />
7 <jsp:setProperty name="category" property="*"/>
8 <jsp:useBean id="currentpage" class="myapplication.bean.Page"
9                                scope="session" />
10 <%
11     currentpage.setPageNr(7);
12 %>
13 <%@ include file="include_files/head.html" %>
14 <%@ include file="include_files/body_begin.html" %>
15 <%@ include file="menue.jsp" %>
16 <!-- ************ B E G I N  C O N T E N T *************** -->
17
18 <%@ include file="include_files/xmldoc_withprms.html" %>
19
20 <!-- ************** E N D  C O N T E N T **************** -->
21 <%@ include file="copyright.html" %>
22 <%@ include file="include_files/body_end.html" %>
```

Die JSP-Seite sell.jsp

Listing B.4: *sell.jsp*-Seite

```
1 <%@ page errorPage="error.jsp" %>
2 <%@ page import="myapplication.bean.CustomerSession" %>
3
4 <%
5 CustomerSession customer_session =
6 (CustomerSession)session.getAttribute("user");
7 String show = "login.jsp";
8 if(customer_session == null){
9   request.setAttribute("text", "Sie muessen sich erst anmelden
10                              bevor Sie eine Auktion anmelden!");
11   request.setAttribute("forward_page", "sellcategory.jsp");
12 }else{
13     if(!customer_session.isUserValid()){
14         request.setAttribute("text", "Ihr Angaben sind nicht
15                 gueltig.Sie muessen sich erst anmelden bevor
16                                     Sie Ihr Gebot abgeben!");
```

```
17          request.setAttribute("forward_page", "sellcategory.jsp");
18      }else{
19          show = "sellcategory.jsp";
20      }
21 }
22 %>
23 <jsp:forward page="<%= show %>"/>
```

Die JSP-Seite sellcategory.jsp

Listing B.5: *sellcategory.jsp*-Seite

```
1 <%@ page contentType="text/html"%>
2 <%@ page errorPage="error.jsp" %>
3 <%@ page import="org.w3c.dom.Document" %>
4 <%@ page import="myapplication.xml.*" %>
5 <% out.clear(); %>
6 <jsp:useBean id="currentpage" class="myapplication.bean.Page"
7                                scope="session" />
8 <%
9     currentpage.setPageNr(6);
10 %>
11 <%@ include file="include_files/head.html" %>
12 <%@ include file="include_files/body_begin.html" %>
13 <%@ include file="menue.jsp" %>
14 <!-- ************** BEGIN CONTENT **************-->
15
16 <%@ include file="include_files/xmldoc.html" %>
17
18
19 <!-- ************** END CONTENT ****************-->
20 <%@ include file="copyright.html" %>
21 <%@ include file="include_files/body_end.html" %>
```

Die JSP-Seite complete_sell.jsp

Listing B.6: *complete_sell.jsp*-Seite

```
1 <%@ page errorPage="error.jsp" %>
2 <%@ page import="myapplication.bean.CustomerSession" %>
3 <jsp:useBean id = "auction" class = "myapplication.bean.Auction"
4                              scope = "request" />
5 <jsp:setProperty name="auction" property="*"/>
6
7 <%
```

```
8  String show = "prodtype.jsp";
9  boolean areFieldsValid = true;
10 if(request.getParameter("run_time_days").compareTo("")==0){
11     auction.setRun_time_days("0");
12 } if(request.getParameter("run_time_hours").compareTo("")==0){
13     auction.setRun_time_hours("0");
14 }
15
16 if(auction.checkForEmptyFields() == false){
17     show = "sell_form.jsp?SUBPK=" + request.getParameter("SUBPK");
18     request.setAttribute("message", "Alle Obligatorische Felder
19                                     muessen ausgefuellt werden!");
20     areFieldsValid = false;
21 } if(areFieldsValid){
22     if(auction.areNumberFieldsValid()== false){
23         show = "sell_form.jsp?SUBPK=" + request.
24                                 getParameter("SUBPK");
25         request.setAttribute("message", "Zahlen-Felder
26                     m&uuml;ssen Zahlen-Werte enthalten!");
27         areFieldsValid = false;
28     }
29 } if(areFieldsValid){
30     if(auction.areStartEndPriceValid()== false){
31         show = "sell_form.jsp?SUBPK=" + request.
32                                 getParameter("SUBPK");
33         request.setAttribute("message", "Der Startpreis darf
34                     nicht niedriger als der Endpreis sein!");
35         areFieldsValid = false;
36     }
37 } if(areFieldsValid){
38     if(auction.isPriceRangeValid()== false){
39         show = "sell_form.jsp?SUBPK=" + request.
40                                 getParameter("SUBPK");
41         request.setAttribute("message", "Der Preisrutsch darf
42                     nicht groesser als der Startpreis sein!");
43         areFieldsValid = false;
44     }
45 } if(areFieldsValid){
46     if(auction.areAucDurationFieldsValid() == false){
47         show = "sell_form.jsp?SUBPK=" + request.
48                                     getParameter("SUBPK");
49         request.setAttribute("message", "Das Feld üfr die
50                         Stundendauer der Auktion darf nicht
51                         den Wert 23 überschreiten!");
52         areFieldsValid = false;
53     }
54 } if(areFieldsValid){
55     if(auction.isAuctionDurationValid() == false){
```

```
56          show = "sell_form.jsp?SUBPK=" + request.
57                                      getParameter("SUBPK");
58          request.setAttribute("message", "Die Auktionsdauer darf
59                                  nicht ükrzer als 1 Stunde sein!");
60          areFieldsValid = false;
61      }
62 } if(areFieldsValid){
63  try {
64      CustomerSession customer_session = (CustomerSession)session.
65                                      getAttribute("user");
66      auction.addAuction(customer_session.getUserCode());
67  } catch (Exception e){
68      e.printStackTrace();
69 } }
70 %>
71 <jsp:forward page="<%= show %>"/>
```

Die JSP-Seite xmldoc.jsp

Listing B.7: *xmldoc.jsp*-Seite

```
1 <%
2 XmlProductData xmldata = new XmlProductData();
3 XmlTool xmltool = new XmlTool();
4 Document xmldoc = xmldata.getDocumentFromDBData(currentpage.
5                                          getXmlDocName());
6 String path = "http://" + request.getRemoteAddr()+ ":" +
7                 String.valueOf(request.getServerPort()) +
8                 request.getContextPath()+
9                 currentpage.getTransformFilePath();
10 String html = xmltool.transform(path,xmldoc);
11 out.println(html);
12 %>
```

Die JSP-Seite complete_login.jsp

Listing B.8: *complete_login.jsp*-Seite

```
1 <%@ page errorPage="error.jsp" %>
2 <jsp:useBean id = "customer_session" class =
3 "myapplication.bean.CustomerSession" scope = "page"/>
4 <jsp:setProperty name = "customer_session" property = "*" />
5 <jsp:useBean  id = "monitor" scope = "application" class =
6 "java.util.HashMap" />
7
```

```
8  <%
9  customer_session = customer_session.validateUser();
10 String show = "";
11 if(request.getParameter("forward_page") == null){
12     show = "buy.jsp";
13 }else{
14     show = request.getParameter("forward_page");
15 }
16 if(customer_session.isUserValid()){
17     customer_session.setIpAddr(request.getRemoteHost());
18     if(monitor.containsKey(customer_session.getUserid())){
19       //wenn der User angemeldet, aber nicht abgemeldet,
20       //wird die Alte Session vernichtet
21       HttpSession oldSession = (HttpSession)monitor.
22                          get(customer_session.getUserid());
23       oldSession.invalidate();
24     }
25     session = request.getSession(true);
26     session.setAttribute("user", customer_session);
27     boolean user_logged = true;
28     monitor.put(customer_session, session);
29     session.setMaxInactiveInterval(900);
30 }else{
31     show = "login.jsp";
32     request.setAttribute("forward_page", request.
33                          getParameter("forward_page"));
34     request.setAttribute("text", "Ihre Daten sind nicht gueltig!
35                  Bitte versuchen Sie das Anmelden noch einmal.");
36 } customer_session.closeDb();
37 %>
38
39 <jsp:forward page="<%= show %>"/>
```

Die JSP-Seite bid.jsp

Listing B.9: *bid.jsp*

```
1  <%@ page errorPage="error.jsp" %>
2  <%@ page import="myapplication.bean.CustomerSession" %>
3
4  <%
5  CustomerSession customer_session =
6                 (CustomerSession)session.getAttribute("user");
7  String show = "login.jsp";
8  if(customer_session == null){
9      request.setAttribute("text", "Sie muessen sich zuerst
10                  anmelden bevor sie Ihr Gebot abgeben!");
```

```
11      request.setAttribute("forward_page", "bid_form.jsp");
12  }else{
13      if(!customer_session.isUserValid()){
14          request.setAttribute("text", "Ihre Angaben sind
15              nicht ügltig. Sie muessen sich zuerst anmelden
16              bevor Sie Ihr Gebot abgeben!");
17          request.setAttribute("forward_page", "bid_form.jsp");
18      }else{
19          show = "bid_form.jsp";
20      }
21  }
22  %>
23  <jsp:forward page="<%= show %>"/>
```

Die JSP-Seite complete_bid.jsp

Listing B.10: *complete_bid.jsp*

```
1  <%@ page errorPage="error.jsp" %>
2  <%@ page import="myapplication.bean.CustomerSession" %>
3  <%@ page import="myapplication.basic.Number" %>
4  <jsp:useBean id = "auction" class = "myapplication.bean.Auction"
5                               scope = "page" />
6  <jsp:useBean id = "product" class =
7                      "myapplication.bean.Product" scope = "request" />
8  <jsp:useBean id = "bid" class = "myapplication.bean.Bid"
9                           scope = "request" />
10 <jsp:setProperty name ="bid" property="*" />
11
12 <%
13 CustomerSession customer_session =
14                  (CustomerSession)session.getAttribute("user");
15 String name = customer_session.toString();
16
17 String show = "bid_form.jsp";
18 Number nr = new Number();
19 String bid_value = nr.getNumberForDb(request.
20                     getParameter("maxbid")); //form value
21 String bid_quantity =
22                     request.getParameter("bidquantity");
23 String bid_signature = "";
24 if(request.getParameter("bidsignature") == null){
25     bid_signature = "nein";
26 }else{
27     bid_signature = request.getParameter("bidsignature");
28 }
29 String min_bid_value = request.getParameter("min_price");
```

```
30 String available_quantity =
31 product.getAvailableQuantity(request.getParameter("productId"));
32 boolean areFieldsValid = true;
33 bid.setMaxBid(request.getParameter("maxbid"));
34 bid.setQuantity(request.getParameter("bidquantity"));
35 if(bid.areFieldsFilledOut() == false){
36     request.setAttribute("text",
37                 "Obligatorische Felder wurden leer gelassen!");
38     areFieldsValid = false;
39 } if(bid.areFieldValuesValid() == false){
40     request.setAttribute("text",
41     "Die obligatorische Felder muessen positive Werte enthalten!");
42     areFieldsValid = false;
43 } if(areFieldsValid == true){
44     if(Double.valueOf(bid_quantity).compareTo(Double.
45                             valueOf(available_quantity)) > 0 ){
46         String available_quantity_formated = nr.
47                             getFormatedDouble(available_quantity);
48         request.setAttribute("text", "Es sind " +
49          available_quantity_formated + " Einheiten verfuegbar!");
50         areFieldsValid = false;
51     }
52 } if(areFieldsValid == true){
53     if(Double.valueOf(min_bid_value).compareTo(Double.
54                                     valueOf(bid_value)) > 0){
55         request.setAttribute("text", "Sie muessen mindestens " +
56                                 min_bid_value + " Euro anbieten!");
57         areFieldsValid = false;
58     }
59 }
60
61 if(areFieldsValid == true){
62     if(bid_signature.compareTo("nein") == 0){
63         boolean bidInDb = bid.insertBidInDb(request.
64             getParameter("productId"), customer_session.
65                     getUserCode(), bid_value, bid_quantity);
66         show = "bid_success.jsp";
67     }else{
68         show ="applet.jsp";
69     }
70 } product.closeDb();
71 %>
72
73 <jsp:forward page="<%= show %>"/>
```

B.2 Java Beans

Die Bean Auction

Listing B.11: *Auction.java*

```
1 package myapplication.bean;
2
3 import java.io.Serializable;
4 import java.sql.Statement;
5 import java.util.Vector;
6 import myapplication.basic.Date;
7 import myapplication.basic.Number;
8 import myapplication.basic.Time;
9 import myapplication.database.Dm;
10
11 public class Auction implements Serializable {
12       private int subsubcat_pk;
13       private int subcat_pk;
14       private String article_name;
15       private String quantity;
16       private int unit;
17       private String highest_price;
18       private String price_range;
19       private String lowest_price;
20       private String run_time_days;
21       private String run_time_hours;
22
23       public Statement statement;
24
25       Time time = new Time();
26       Date date = new Date();
27
28       public void   setSubsubcat_pk (int sscpk) {
29         subsubcat_pk = sscpk;
30       }
31       public int getSubsubcat_pk ( ) {
32         return subsubcat_pk;
33       }
34
35       public void   setSubcat_pk (int scpk) {
36         subcat_pk = scpk;
37       }
38       public int getSubcat_pk ( ) {
39         return subcat_pk;
40       }
41
42       public void   setArticle_name (String an) {
```

```
43          article_name = an;
44        }
45        public String getArticle_name ( ) {
46          return article_name;
47        }
48
49        public void   setQuantity (String qn) {
50          quantity = qn;
51        }
52        public String getQuantity () {
53          return quantity;
54        }
55
56        public void   setUnit (int un) {
57          unit = un;
58        }
59        public int getUnit () {
60          return unit;
61        }
62
63        public void   setHighest_price (String hp) {
64          highest_price = hp;
65        }
66        public String getHighest_price () {
67          return highest_price;
68        }
69
70        public void   setPrice_range (String pr) {
71          price_range = pr;
72        }
73        public String getPrice_range () {
74          return price_range;
75        }
76
77        public void   setLowest_price (String lp) {
78          lowest_price = lp;
79        }
80        public String getLowest_price () {
81          return lowest_price;
82        }
83
84        public void   setRun_time_days (String rtd) {
85          if(rtd.compareTo("")==0){
86              rtd = "0";
87          }
88          run_time_days = rtd;
89        }
90        public String getRun_time_days () {
```

```
91          return run_time_days;
92        }
93
94        public void setRun_time_hours (String rth) {
95          if(rth.compareTo("")==0){
96              rth = "0";
97          }
98          run_time_hours = rth;
99        }
100       public String getRun_time_hours() {
101         return run_time_hours;
102       }
103
104       // Get rem-Values
105       public String getArticle_name_rem() {
106         return article_name_rem;
107       }
108       public String getQuantity_rem() {
109         return quantity_rem;
110       }
111       public String getHighest_price_rem() {
112         return highest_price_rem;
113       }
114       public String getPrice_range_rem() {
115         return price_range_rem;
116       }
117       public String getLowest_price_rem() {
118         return lowest_price_rem;
119       }
120
121     public String article_name_rem = "";
122     public String quantity_rem = "";
123     public String highest_price_rem = "";
124     public String price_range_rem = "";
125     public String lowest_price_rem = "";
126
127     public boolean checkForEmptyFields(){
128       boolean areFieldsValid = true;
129       if(this.article_name == null){
130           article_name_rem = "Obligatorisches Feld";
131           areFieldsValid = false;
132       }
133       if(this.quantity == null){
134           quantity_rem = "Obligatorisches Feld";
135           areFieldsValid = false;
136       }
137       if(this.highest_price == null){
138           highest_price_rem = "Obligatorisches Feld";
```

```
139             areFieldsValid = false;
140         }
141         if(this.price_range == null){
142             price_range_rem = "Obligatorisches Feld";
143             areFieldsValid = false;
144         }
145         if(this.lowest_price == null){
146             lowest_price_rem = "Obligatorisches Feld";
147             areFieldsValid = false;
148         }
149         return areFieldsValid;
150     }
151 
152     public boolean areNumberFieldsValid(){
153         boolean areNumberFieldsValid = true;
154         Number number = new Number();
155         if(number.isNumberValid(this.quantity) == false){
156             quantity = null;
157             quantity_rem =
158                 "G&uuml;ltiger Zahlen-Wert erforderlich";
159             areNumberFieldsValid = false;
160         }
161         if(number.isNumberValid(this.highest_price) == false){
162             highest_price = null;
163             highest_price_rem =
164                 "G&uuml;ltiger Zahlen-Wert erforderlich";
165             areNumberFieldsValid = false;
166         }
167         if(number.isNumberValid(this.price_range) == false){
168             price_range = null;
169             price_range_rem =
170                 "G&uuml;ltiger Zahlen-Wert erforderlich";
171             areNumberFieldsValid = false;
172         }
173         if(number.isNumberValid(this.lowest_price) == false){
174             lowest_price = null;
175             lowest_price_rem =
176                 "G&uuml;ltiger Zahlen-Wert erforderlich";
177             areNumberFieldsValid = false;
178         }
179         return areNumberFieldsValid;
180     }
181 
182     private int getAuctionDuration(){
183         int duration = time.
184             getTimeInMinutes(this.getRun_time_days(),
185                              this.getRun_time_hours());
186         return duration;
```

```
187     }
188
189     public boolean isAuctionDurationValid(){
190         boolean isDurationValid = false;
191         if(this.getAuctionDuration()>=60){
192             isDurationValid = true;
193         }
194         return isDurationValid;
195     }
196
197     public boolean areStartEndPriceValid(){
198         boolean areStartEndPriceValid = false;
199         if((Integer.valueOf(this.highest_price).intValue()-
200            Integer.valueOf(this.lowest_price).intValue()) > 0){
201             areStartEndPriceValid = true;
202         }
203         return areStartEndPriceValid;
204     }
205
206     public boolean isPriceRangeValid(){
207         boolean isPriceRangeValid = false;
208         if((Integer.valueOf(this.highest_price).intValue()-
209            Integer.valueOf(this.price_range).intValue()) > 0){
210             isPriceRangeValid = true;
211         }
212         return isPriceRangeValid;
213     }
214
215     public boolean areAucDurationFieldsValid(){
216         boolean areAucDurationFieldsValid = true;
217         if(Integer.valueOf(this.getRun_time_hours()).
218                                             intValue()>23){
219             areAucDurationFieldsValid = false;
220         }
221         return areAucDurationFieldsValid;
222     }
223
224       public void addAuction (String userCode){
225         try {
226         String opening = date.getCurrentTime();
227         int duration = this.getAuctionDuration();
228         String termination = date.
229             getAuctionTerminationTime(opening, duration);
230         //Dauer fuer die Preissenkung:
231         int totaltime_pricerange = duration - 30;
232         //Differenz zwischen hoechster und niedrigster Preis:
233         double diff_highlow_price =
234             Double.valueOf(this.getHighest_price()).doubleValue() -
```

```
235             Double.valueOf(this.getLowest_price()).doubleValue();
236         //Anzahl der Preissenkungen:
237         int number_price_reductions = (int)diff_highlow_price /
238          (int)Double.valueOf(this.getPrice_range()).doubleValue();
239         //Zeitabstaende fuer die Preissenkung:
240         int timelag_pricereduc = totaltime_pricerange /
241                                     number_price_reductions;
242         String sql = "execute procedure p_insert_newauction("+
243         this.getSubsubcat_pk() + "," + "'" +
244         this.getArticle_name() + "'" + "," + userCode + "," +
245         "1" + "," + this.getQuantity() + "," + this.getUnit() +
246         "," + this.getHighest_price() + "," +
247         this.getPrice_range()+ "," + this.getLowest_price() +
248         "," + duration + "," + "'" + opening + "'" + "," +
249         "'" + termination + "'" + "," + totaltime_pricerange +
250         "," + diff_highlow_price + "," + number_price_reductions +
251         "," + timelag_pricereduc + "," + "1" + ")";
252           Dm db = new Dm();
253           db.executeStatement(sql);
254           db.closeDatabase();
255         }
256         catch(Exception e) {
257           e.printStackTrace();
258         }
259       }
260
261       public void updateAuctionData(){
262         Dm db = new Dm();
263         Vector vProduct = null;
264         String sql = "select product_code, current_quantity
265                       from tdat_auction where status_code = 1";
266         vProduct = db.selectDbDataInVector(sql, "PRODUCT_CODE",
267                     "CURRENT_QUANTITY", "PRODUCT_CODE",
268                      db.getQueryRowCount(sql));
269         Product prod = new Product();
270         String[] sPara = new String[2];
271         String pk = "";
272         String sqlUpdate = "";
273         String current_price = "";
274         String current_bid_number = "";
275         int days = 0;
276         int rest = 0;
277         int hours = 0;
278         int mins = 0;
279         for(int i = 0; i < vProduct.size(); i++){
280             sPara = (String[])vProduct.elementAt(i);
281             pk = sPara[0];
282             String sLeftMins = date.
```

```
                    getLeftAuctionDuration_Mins(pk);
            if((Integer.valueOf(sLeftMins).intValue()<=0) ||
                                (sPara[1].compareTo("0.0")==0)){
                sqlUpdate = "update tdat_auction set status_code=2,
                    current_left_days=0, current_left_hrs=0,
                    current_left_mins=0 where product_code=" + pk;
                db.executeStatement(sqlUpdate);
            }else{
                current_price = String.valueOf(
                    prod.getActualPrice(pk,
                        Integer.valueOf(sLeftMins).intValue()));
                current_bid_number = prod.getCurrentBidNumber(pk);
                prod.closeDb();
                days = Integer.valueOf(sLeftMins).intValue() /
                                                    (60*24);
                rest = Integer.valueOf(sLeftMins).intValue() %
                                                  (60 * 24);
                hours = rest / 60;
                mins = rest % 60;
                sqlUpdate =  "update tdat_auction set
                    current_price=" + current_price +
                    ", current_left_days=" +
                    String.valueOf(days) +
                    ", current_left_hrs=" +
                    String.valueOf(hours) +
                    ", current_left_mins=" +
                    String.valueOf(mins) +
                    ", current_bid_number=" +
                    current_bid_number +
                    " where product_code=" + pk;
                db.executeStatement(sqlUpdate);
            }
        }
        try{
            db.closeDatabase();
        }catch(Exception e){
            e.printStackTrace();
        }
    }
}
```

Die Bean Bid

Listing B.12: *Bid.java*

```
package myapplication.bean;

```

```
3 import myapplication.basic.Date;
4 import myapplication.database.Dm;
5
6 public class Bid {
7   private int productId = 0;
8   private String maxBid = "";
9   private String quantity = "";
10
11  private String maxBid_rem = "";
12  private String quantity_rem = "";
13  public boolean maxBidValid = true;
14  public boolean quantityValid = true;
15
16  public boolean isMaxBidValid (){
17    return maxBidValid;
18  }
19
20  public boolean isQuantityValid(){
21    return quantityValid;
22  }
23
24  public void setProductId(int pId) {
25      productId = pId;
26  }
27  public int getProductId() {
28    return productId;
29  }
30
31  public void setMaxBid(String mb) {
32    maxBid = mb;
33  }
34  public String getMaxBid() {
35    return maxBid;
36  }
37
38  public void setQuantity(String qy) {
39    quantity = qy;
40  }
41  public String getQuantity() {
42    return quantity;
43  }
44
45  public void setMaxBid_rem(String mb_rem) {
46    maxBid_rem = mb_rem;
47  }
48  public String getMaxBid_rem() {
49    return maxBid_rem;
50  }
```

```
51
52    public void setQuantity_rem(String q_rem) {
53      quantity_rem = q_rem;
54    }
55
56    public String getQuantity_rem() {
57      return quantity_rem;
58    }
59
60    public boolean areFieldsFilledOut(){
61      boolean areFieldsValid = true;
62      if(this.getMaxBid().compareTo("") ==0){
63          this.setMaxBid_rem("Obligatorisches Feld");
64          areFieldsValid = false;
65          this.maxBidValid = false;
66      }
67      if(this.getQuantity().compareTo("") ==0){
68          this.setQuantity_rem("Obligatorisches Feld");
69          areFieldsValid = false;
70          this.quantityValid = false;
71      }
72      return areFieldsValid;
73    }
74
75    public boolean areFieldValuesValid(){
76      boolean areFieldValuesValid = true;
77      if((this.getMaxBid().compareTo("0.0") == 0) ||
78              (this.getMaxBid().compareTo("0") == 0)){
79          this.setMaxBid_rem("Positiver Wert erforderlich!");
80          areFieldValuesValid = false;
81          this.maxBidValid = false;
82      }
83      if((this.getQuantity().compareTo("0.0") == 0) ||
84              (this.getQuantity().compareTo("0") == 0)){
85          this.setQuantity_rem("Positiver Wert erforderlich!");
86          areFieldValuesValid = false;
87          this.quantityValid = false;
88      }
89      return areFieldValuesValid;
90    }
91
92
93    public String getPrice(String prodId){
94      String price = "";
95      Date date = new Date();
96      String sTotal_mins = date.getLeftAuctionDuration_Mins(prodId);
97      int total_mins = Integer.valueOf(sTotal_mins).intValue();
98      Product prod = new Product();
```

```
99      String min_price = String.valueOf(prod.getActualPrice(prodId,
100                                                            total_mins));
101     prod.closeDb();
102     return min_price;
103   }
104
105   public boolean insertBidInDb (String sProdId, String sUserCode,
106                                 String sBidValue, String sBidQuantity){
107     Dm db = new Dm();
108     Date date = new Date();
109     String time = date.getCurrentTime();
110     boolean bidInDb = false;
111     String sql = "execute procedure p_complete_bid(" + sProdId +
112         "," + sUserCode + "," + "'" + time + "'" + "," +
113         sBidValue + "," + sBidQuantity + "," + null + ")";
114     bidInDb = db.executeStatement(sql);
115     try{
116         db.closeDatabase();
117     }catch(Exception e){
118         e.printStackTrace();
119     }
120     return bidInDb;
121   }
122 }
```

Die Bean CustomerSession

Listing B.13: *CustomerSession.java*

```
1  package myapplication.bean;
2
3  import java.util.Vector;
4
5  import myapplication.database.Dm;
6
7  public class CustomerSession implements Comparable{
8    private String userid = "";
9    private String pass = "";
10   private String pk;
11   private String ipaddr;
12   private String forward_page = "";
13   private CustomerSession CustomerSession = null;
14   Dm db = new Dm();
15   Vector v = null;
16   private boolean isAccountValid = false;
17   public CustomerSession(){}
18
```

```
19   public CustomerSession(String id, boolean accountValid,
20                                          String user_code){
21     this.userid = id;
22     this.isAccountValid = accountValid;
23     this.pk = user_code;
24   }
25
26   public void  setUserid (String id) {
27     userid = id;
28   }
29   public String getUserid () {
30     return userid;
31   }
32
33   public String getUserCode () {
34     return pk;
35   }
36
37   public void   setPass (String p1) {
38    pass = p1;
39   }
40   private String getPass () {
41    return pass;
42   }
43
44   public void   setIpAddr (String ip) {
45    ipaddr = ip;
46   }
47   public String getIpAddr () {
48     return ipaddr;
49    }
50
51   public void   setForward_Page (String fp) {
52     forward_page = fp;
53   }
54
55   public String getForward_Page () {
56     return forward_page;
57    }
58
59   private void setIsAccountValid (boolean status) {
60     isAccountValid = status;
61   }
62   private boolean getIsAccountValid () {
63     return isAccountValid;
64   }
65
66   public boolean equals (Object user){
```

```
67      return compareTo(user) == 0;
68    }
69
70    public int compareTo(Object CustomerSession){
71      int result = userid.
72       compareTo(((CustomerSession)CustomerSession).getUserid());
73      return result == 0 ? ipaddr.
74          compareTo(((CustomerSession)CustomerSession).
75                                        getIpAddr()):result;
76    }
77
78    private Vector getTcatCustomerData(String sql, String svName,
79                                  String svValue, String matchValue){
80      v = db.selectDbDataInVector(sql, svName, svValue, matchValue,
81                                          db.getQueryRowCount(sql));
82      return v;
83    }
84
85    private Vector getDbUserIDData(){
86      Vector vek = new Vector(db.getQueryRowCount("select pk,
87                                    userid from tdat_customer"));
88      vek.clear();
89      vek = this.getTcatCustomerData("select pk, userid
90                     from tdat_customer", "PK", "USERID", "PK");
91      return vek;
92    }
93
94    private Vector getDbUserPassData(){
95     Vector passvek = new Vector(db.getQueryRowCount("select
96                                  pk, pass from tdat_customer"));
97     passvek = this.getTcatCustomerData("select pk, pass from
98                           tdat_customer", "PK", "PASS", "PK");
99     return passvek;
100   }
101
102   private boolean validateUserid(){
103     boolean isUserValid = false;
104     String[] sUser = null;
105     Vector vUid = null;
106     vUid = this.getDbUserIDData();
107      for (int i=0; i<vUid.size(); i++) {
108        sUser = (String[])vUid.elementAt(i);
109            if (sUser[1].compareTo(this.getUserid()) == 0) {
110              pk = sUser[0];
111              isUserValid = true;
112            }
113      }
114      return isUserValid;
```

```
115   }
116
117   private boolean validatePassword(){
118     boolean isPasswordValid = false;
119     String[] sUserPass = null;
120     Vector vUserPass = null;
121     vUserPass = getDbUserPassData();
122      for (int i=0; i<vUserPass.size(); i++) {
123        sUserPass = (String[])vUserPass.elementAt(i);
124        if ((sUserPass[0].compareTo(this.pk)) == 0) {
125          if (sUserPass[1].compareTo(this.getPass()) == 0) {
126            isPasswordValid = true;
127          }
128        }
129      }
130      return isPasswordValid;
131   }
132
133   public CustomerSession validateUser(){
134     setIsAccountValid(false);
135     if(validateUserid()){
136      if(validatePassword()){
137        setIsAccountValid(true);
138      }
139     }
140     return new CustomerSession(this.getUserid(),
141                     getIsAccountValid(), getUserCode());
142   }
143
144   public String toString(){
145     return userid;
146   }
147
148   public boolean isUserValid(){
149     return getIsAccountValid();
150   }
151
152   public void closeDb(){
153     try{
154         db.closeDatabase();
155     }catch(Exception e){
156         e.printStackTrace();
157     }
158   }
159 }
```

Die Bean NewRegSession

Listing B.14: *NewRegSession.java*

```
1 package myapplication.bean;
2
3 import java.io.Serializable;
4 import java.sql.ResultSet;
5 import java.sql.Statement;
6
7 import myapplication.database.Dm;
8
9 public class NewRegSession implements Serializable {
10   private int gender;
11   private String last_name;
12   private String first_name;
13   private String adr1;
14   private String adr2;
15   private String plz;
16   private String city;
17   private int country;
18   private String phone;
19   private String email;
20   private String userid;
21   private String pass1;
22   private String pass2;
23
24   public Statement statement;
25
26   Dm db = new Dm();
27
28   public String last_name_rem = "";
29   public String first_name_rem = "";
30   public String address1_rem = "";
31   public String address2_rem = "";
32   public String pocode_rem = "";
33   public String city_rem = "";
34   public String phone_rem = "";
35   public String email_rem = "";
36   public String userid_rem = "";
37   public String pass1_rem = "";
38   public String pass2_rem = "";
39
40   public void   setGender (int gn) {
41     gender = gn;
42   }
43   public int getGender ( ) {
44     return gender;
```

```
45    }
46
47    public void   setFirst_name (String fn) {
48      first_name = fn;
49    }
50    public String getFirst_name ( ) {
51      return first_name;
52    }
53
54    public void   setLast_name (String ln ) {
55      last_name =ln;
56    }
57    public String getLast_name () {
58      return last_name;
59    }
60
61    public void   setAdr1 (String ad1 ) {
62      adr1 =ad1;
63    }
64    public String getAdr1 () {
65      return adr1;
66    }
67
68    public void   setAdr2 (String ad2) {
69      adr2 =ad2;
70    }
71    public String getAdr2 () {
72      return adr2;
73    }
74
75    public void   setPlz (String pl) {
76      plz = pl;
77    }
78    public String getPlz () {
79      return plz;
80    }
81
82    public void   setCity (String cy) {
83      city = cy;
84    }
85    public String getCity () {
86      return city;
87    }
88
89    public void   setCountry (int ct) {
90      country = ct;
91    }
92    public int getCountry () {
```

```
93      return country;
94    }
95
96    public void   setPhone (String phn) {
97      phone = phn;
98    }
99    public String getPhone () {
100     return phone;
101   }
102
103   public void   setEmail (String em) {
104     email = em;
105   }
106   public String getEmail () {
107     return email;
108   }
109
110   public void   setUserid (String id) {
111     userid = id;
112   }
113   public String getUserid () {
114     return userid;
115   }
116
117   public void   setPass1 (String p1) {
118    pass1 = p1;
119  }
120  public String getPass1 () {
121    return pass1;
122  }
123  public void   setPass2 (String p2) {
124   pass2 = p2;
125  }
126  public String getPass2 () {
127    return pass2;
128  }
129
130  // Rem Field-Values
131   public String getLast_name_rem(){
132     return last_name_rem;
133   }
134   public String getFirst_name_rem(){
135     return first_name_rem;
136   }
137   public String getAdr1_rem(){
138     return address1_rem;
139   }
140   public String getPocode_rem(){
```

```
141     return pocode_rem;
142   }
143   public String getCity_rem(){
144     return city_rem;
145   }
146   public String getPhone_rem(){
147     return phone_rem;
148   }
149   public String getEmail_rem(){
150     return email_rem;
151   }
152   public String getUserid_rem(){
153     return userid_rem;
154   }
155   public String getPass1_rem(){
156     return pass1_rem;
157   }
158   public String getPass2_rem(){
159     return pass2_rem;
160   }
161
162   public boolean checkForEmptyFields(){
163     boolean areFieldsValid = true;
164     if(this.last_name == null){
165         last_name_rem = "Obligatorisches Feld";
166         areFieldsValid = false;
167     }
168     if(this.first_name == null){
169         first_name_rem = "Obligatorisches Feld";
170         areFieldsValid = false;
171     }
172     if(this.adr1 == null){
173         address1_rem = "Obligatorisches Feld";
174         areFieldsValid = false;
175     }
176     if(this.plz == null){
177         pocode_rem = "Obligatorisches Feld";
178         areFieldsValid = false;
179     }
180     if(this.city == null){
181         city_rem = "Obligatorisches Feld";
182         areFieldsValid = false;
183     }
184     if(this.phone == null){
185         phone_rem = "Obligatorisches Feld";
186         areFieldsValid = false;
187     }
188     if(this.email == null){
```

```
189         email_rem = "Obligatorisches Feld";
190         areFieldsValid = false;
191     }
192     if(this.userid == null){
193         userid_rem = "Obligatorisches Feld";
194         areFieldsValid = false;
195     }
196     if(this.pass1 == null){
197         pass1_rem = "Obligatorisches Feld";
198         areFieldsValid = false;
199     }
200     if(this.pass2 == null){
201         pass2_rem = "Obligatorisches Feld";
202         areFieldsValid = false;
203     }
204     return areFieldsValid;
205   }
206
207   public boolean isUserValid(){
208     boolean isUserValid = true;
209     try{
210         String sqlUser = "select count(c.userid) from
211         tdat_customer c where c.userid='" + this.userid + "'";
212         ResultSet rsUser = db.executeQuery(sqlUser);
213         rsUser.next();
214         if(rsUser.getObject("count").toString().
215                                     compareTo("0") !=0){
216             isUserValid = false;
217         }
218     }catch(Exception e){
219         e.printStackTrace();
220     }
221
222     return isUserValid;
223   }
224
225   public boolean isEmailValid(){
226     boolean isEmailValid = false;
227     int numberAtChar = 0;
228     for(int i=0; i<=this.email.length()-1; i++){
229         if(this.email.charAt(i) == '@'){
230             numberAtChar++;
231         }
232     }
233     if(numberAtChar == 1){
234         isEmailValid = true;
235     }
236     return isEmailValid;
```

```
237   }
238
239   public boolean arePassFieldsValid(){
240     boolean arePassFieldsValid = true;
241     if(this.pass1.compareTo(this.pass2) != 0){
242         arePassFieldsValid = false;
243     }
244     if(this.pass1.length() < 6 || this.pass2.length() < 6){
245         arePassFieldsValid = false;
246     }
247     return arePassFieldsValid;
248   }
249
250   public boolean addUser (){
251     boolean userInserted = false;
252         try {
253           String sql = "insert into tdat_customer (lastname,
254                 firstname, gender_code, address1, address2,
255                 pocode, city, country_code, phonenumber,
256                 email, userid, pass) values ('"+
257                 this.getLast_name() + "', '" +
258                 this.getFirst_name() + "', " +
259                 String.valueOf(this.gender) + ", '" +
260                 this.getAdr1() + "' , '" + this.getAdr2() +
261                 "' , '" + this.getPlz() + " ', '" +
262                 this.getCity() + " ',' +
263                 String.valueOf(this.getCountry()) + ", '" +
264                 this.getPhone() + "' , '" + this.getEmail() +
265                 "' , '" + this.getUserid() + "' , '" +
266                 this.getPass1() + "')";
267           db.executeStatement(sql);
268
269           userInserted = true;
270         }
271         catch(Exception e) {
272           e.printStackTrace();
273         }
274     return userInserted;
275   }
276   public void closeDb(){
277     try{
278         db.closeDatabase();
279     }catch(Exception e){
280         e.printStackTrace();
281     }
282   }
283 }
```

Die Bean Page

Listing B.15: *Page.java*

```
1 package myapplication.bean;
2
3 import java.sql.ResultSet; import java.util.Vector; import
4 myapplication.database.Dm;
5
6 public class Page {
7     ResultSet rsPage = null;
8     Vector vPrm = new Vector();
9
10     private int iPageNr = 0;
11     private int iRowNr = 0;
12     private String sPageTitle = "";
13     private String sXmlDocName = "";
14     private String sTrasformFilePath = "";
15     boolean test = false;
16
17     public void setPageNr(int pnr){
18         iPageNr = pnr;
19         boolean pageFound = false;
20         try{
21             init(iPageNr);
22         }catch(Exception e){
23             e.printStackTrace();
24         }
25     }
26
27     public String getPageNr(){
28         return String.valueOf(iPageNr);
29     }
30
31     public void setTitle(String sTitle){
32         sPageTitle = sTitle;
33     }
34
35     public String getTitle(){
36         return sPageTitle;
37     }
38
39     public void setXmlDocName(String xdn){
40         sXmlDocName = xdn;
41     }
42
43     public String getXmlDocName(){
44         return sXmlDocName;
```

```
45      }
46
47      public void setTransformFilePath(String tdp){
48          sTrasformFilePath = tdp;
49      }
50
51      public String getTransformFilePath(){
52          return sTrasformFilePath;
53      }
54
55      public void setDocParams(Vector v){
56          vPrm = v;
57      }
58
59      public Vector getDocParams(){
60          return vPrm;
61      }
62
63      private void init(int page_nr) throws Exception {
64          Dm db = new Dm();
65          String sql = "select * from p_pagedata(" +
66                          String.valueOf(page_nr) + ")";
67          rsPage = db.executeQuery(sql);
68          rsPage.next();
69          this.setTitle(rsPage.getObject("TITLE").toString());
70          if(rsPage.getObject("XMLDOC_NAME").toString().
71                                          compareTo("") != 0){
72              this.setXmlDocName(rsPage.getObject("XMLDOC_NAME").
73                                                  toString());
74          }
75          if(rsPage.getObject("TRANSFORMFILE_PATH").toString().
76                                          compareTo("") != 0){
77              this.setTransformFilePath(rsPage.
78                      getObject("TRANSFORMFILE_PATH").toString());
79          }
80          if(rsPage.getObject("REQUESTPARAM_NAME").toString().
81                                          compareTo("") != 0){
82              String[] prm = new String[2];
83              Vector v = new Vector();
84              do{
85                  prm[0] = rsPage.getObject("XMLPRM_NAME").
86                                              toString();
87                  prm[1] = rsPage.getObject("REQUESTPARAM_NAME").
88                                                  toString();
89                  v.add(prm);
90              }while(rsPage.next());
91              this.setDocParams(v);
92          }
```

```
93          rsPage.close();
94          db.closeDatabase();
95      }
96  }
```

Die Bean Product

Listing B.16: *Product.java*

```
1  package myapplication.bean;
2
3  import java.sql.Connection;
4  import java.sql.ResultSet;
5  import java.util.Vector;
6  import myapplication.database.Dm;
7
8  public class Product {
9    private int productId;
10   ResultSet rs;
11   Connection con;
12   Vector v = new Vector();
13
14
15   public int getProductId() {
16     return productId;
17   }
18
19   public void setProductId(int pId) {
20      productId = pId;
21   }
22
23   private Vector getProductData(String sql, String svName,
24                          String svValue, String matchValue){
25     Dm db = new Dm();
26   v = db.selectDbDataInVector(sql, svName, svValue,
27                       matchValue, db.getQueryRowCount(sql));
28   return v;
29  }
30
31     public Vector getDbUnitData(){
32       Vector vek = null;
33       vek = this.getProductData("select * from tcat_unit",
34                                       "PK", "NAME", "PK");
35       return vek;
36     }
37
38     public double getActualPrice(String prodpk, int time_left){
```

```
39          Dm db = new Dm();
40          double actual_price = -1.00;
41          int time_expired = 0; //abgel. Min. Anfang d. Auktion
42          int timelag_pricereduction = 0;
43          int totaltime_pricerange = 0;
44          double max_price = 0.0;
45          double min_price = 0.0;
46          double price_range = 0.0;
47          int pricereductins_nr = 0;
48          String price_sql = "select run_time, totaltime_pricerange,
49          timelag_pricereduction, highest_price, lowest_price,
50          price_range, pricereductions_number from tdat_auction
51          where product_code=" + prodpk;
52          rs = db.executeQuery(price_sql);
53          double actual_auction_price = 0;
54          try{
55              rs.next();
56              time_expired = Integer.valueOf(
57                  rs.getObject("run_time").toString()).intValue() -
58                                                          time_left;
59              timelag_pricereduction =
60              Integer.valueOf(rs.getObject("timelag_pricereduction").
61                                          toString()).intValue();
62              max_price = Double.valueOf(rs.
63                      getObject("highest_price").
64                              toString()).doubleValue();
65              min_price = Double.valueOf(rs.
66                      getObject("lowest_price").
67                              toString()).doubleValue();
68              price_range = Double.valueOf(rs.
69                      getObject("price_range").
70                              toString()).doubleValue();
71              pricereductins_nr = Integer.valueOf(rs.
72                      getObject("pricereductions_number").
73                              toString()).intValue();
74              totaltime_pricerange = Integer.valueOf(rs.
75                      getObject("totaltime_pricerange").
76                              toString()).intValue();
77          }catch(Exception ex){
78              ex.printStackTrace();
79          }
80
81          int time_tmp = 0;
82          double price = max_price;
83          int i = 0;
84
85          if(time_expired < totaltime_pricerange){
86              actual_price = max_price - price_range*
```

```
87                  ((int)time_expired/timelag_pricereduction);
88          }else{
89              actual_price = min_price;
90          }
91          return actual_price;
92      }
93
94      public String getCurrentBidNumber(String pk){
95          Dm db = new Dm();
96          String bidnr = "";
97          ResultSet rs_nr = null;
98          try{
99          rs_nr = db.executeQuery("select bidnumber from
100                                  p_bidnumber(" + pk + ")");
101         rs_nr.next();
102         bidnr = rs_nr.getObject("BIDNUMBER").toString();
103         }catch(Exception e){
104             e.printStackTrace();
105         }
106
107         return bidnr;
108     }
109
110     public String getAvailableQuantity(String pk){
111         Dm db = new Dm();
112         String quantity = "";
113         ResultSet rs_nr = null;
114         try{
115         rs_nr = db.executeQuery("select current_quantity from
116                         tdat_auction where product_code=" + pk);
117         rs_nr.next();
118         quantity = rs_nr.getObject("CURRENT_QUANTITY").toString();
119         }catch(Exception e){
120             e.printStackTrace();
121         }
122         return quantity;
123     }
124
125     public void closeDb(){
126       try{
127         Dm db = new Dm();
128           db.closeDatabase();
129       }catch(Exception e){
130           e.printStackTrace();
131       }
132     }
133 }
```

Die Bean SubSubCategory

Listing B.17: *SubSubCategory.java*

```
1 package myapplication.bean;
2
3 import java.io.Serializable;
4 import java.sql.Connection;
5 import java.sql.ResultSet;
6 import java.util.Vector;
7 import myapplication.database.Dm;
8
9 public class SubSubCategory implements Serializable {
10   private int subpk;
11   private String subname;
12   private int subsubpk;
13   private String subsubname;
14   ResultSet rs;
15   Connection con;
16   Vector v = new Vector();
17
18   public void setSubPK (int nr) {
19     subpk = nr;
20   }
21   public int getSubPK () {
22     return subpk;
23   }
24
25   public void setSubName (String nm) {
26     subname = nm;
27   }
28   public String getSubName () {
29     return subname;
30   }
31   public void setSubSubPK (int nr) {
32     subsubpk = nr;
33   }
34   public int getSubSubPK () {
35     return subsubpk;
36   }
37
38   public void setSubSubName (String nm) {
39     subsubname = nm;
40   }
41   public String getSubSubName () {
42     return subsubname;
43   }
44
```

```
45      private Vector getTcatSubSubCategoryData(String sql,
46          String svName, String svValue, String matchValue){
47      Dm db = new Dm();
48      v = db.selectDbDataInVector(sql, svName, svValue,
49                          matchValue, db.getQueryRowCount(sql));
50      try{
51          db.closeDatabase();
52      }catch(Exception e){
53          e.printStackTrace();
54      }
55      return v;
56    }
57
58    public Vector getDbSubSubCategoryData(String sSubSubCatId){
59      Vector vek = null;
60      vek = this.getTcatSubSubCategoryData("select * from
61              tcat_prodsubsubtype where prodsubtype_code= " +
62                              sSubSubCatId, "PK", "NAME", "PK");
63      return vek;
64    }
65  }
```

B.3 XML-Klassen

Listing B.18: *XmlProductData.java*

```
1  package myapplication.xml;
2
3  import java.util.Vector;
4
5  import org.w3c.dom.Document; import org.w3c.dom.Element;
6
7  public class XmlProductData {
8      Vector v = new Vector();
9      XmlTool xmltool = new XmlTool();
10
11     public void addParamToProductData (String sPrmName,
12                                        String sPrmValue) {
13         String[] sPara = new String[2];
14         sPara[0] = sPrmName;
15         sPara[1] = sPrmValue;
16         v.addElement(sPara);
17     }
18
19     public Document getDocumentFromDBData (String sMessageName) {
20         Document doc = null;
```

```
21          XmlDocument xmldoc = new XmlDocument();
22          doc = xmldoc.getDocumentFromDB(sMessageName, v);
23          return  doc;
24      }
25
26      public Document getDocument(Document document) {
27          return document;
28      }
29  }
```

Listing B.19: *XmlDocument.java*

```
1  package myapplication.xml;
2
3  import java.sql.ResultSet; import java.util.Vector;
4
5  import myapplication.database.Dm;
6
7  import org.w3c.dom.Document; import org.w3c.dom.Element; import
8  org.w3c.dom.Node; import org.w3c.dom.NodeList;
9
10 public class XmlDocument {
11     Vector vParam = new Vector();
12     XmlTool xmltool = new XmlTool();
13     Dm db = new Dm();
14
15
16     public Document getDocumentFromDB (String sMessageName,
17                                        Vector v) {
18         Document doc = null;
19         vParam = v;
20         if (checkParameter(sMessageName) == true) {
21             doc = getDocumentDataFromDB(sMessageName);
22         }
23         return  doc;
24     }
25
26     private boolean checkParameter (String sMessageName) {
27         boolean bAllParamsAvailable = true;
28         boolean bParamAvailable = false;
29         Vector vDbParam = new Vector();
30         String sParamName;
31         String[] sPara = null;
32         ResultSet rset = null;
33         String sql = "SELECT * FROM P_XMLDOC_PARAM ('" +
34                                 sMessageName + "')";
35         if(db.getQueryRowCount(sql)==0){
36             bParamAvailable = true;
```

```
37          }else{
38              rset = db.executeQuery(sql);
39              try{
40              rset.next();
41              do {
42                  sParamName = "";
43                  sParamName = rset.getObject("PARAM_NAME").
44                                                  toString();
45                  vDbParam.addElement(sParamName);
46              } while (rset.next());
47              }catch(Exception e){
48                  e.printStackTrace();
49              }
50              for (int i = 0; i < vDbParam.size(); i++) {
51                  bParamAvailable = false;
52                  String sDbParam = (String)vDbParam.elementAt(i);
53                  for (int j = 0; j < vParam.size(); j++) {
54                      sPara = (String[])vParam.elementAt(j);
55                      if (sPara[0].compareTo(sDbParam) == 0) {
56                          bParamAvailable = true;
57                      }
58                  }
59                  if (!bParamAvailable) {
60                      System.out.println("Parameter " + sDbParam +
61                                          " wurde nicht gesetzt");
62                      bAllParamsAvailable = false;
63                  }
64              }
65          }
66          return  bAllParamsAvailable;
67      }
68
69      //Die Parameternamen holen, die fuer diese sql gebraucht werden
70      //und Vektor mit den Parameterwerten üfllen
71      public Vector getSqlParamValues (String sSql) {
72          Vector vSqlParam = new Vector();
73          Vector vSqlParamValues = new Vector();
74          String[] sParamName = null;
75          ResultSet rsql = null;
76          try{
77          String sql = "select * from p_xmldoc_sqlparam('" +
78                                              sSql + "')";
79          rsql = db.executeQuery(sql);
80
81          int k = db.getQueryRowCount(sql);
82
83          if(rsql.next()){
84          do {
```

```
85              String str = rsql.getObject("PARAM_NAME").toString();
86              vSqlParam.addElement(rsql.getObject("PARAM_NAME").
87                                                      toString());
88          } while (rsql.next());
89          for (int i = 0; i < vSqlParam.size(); i++) {
90              for (int j = 0; j < vParam.size(); j++) {
91                  sParamName = (String[])vParam.elementAt(j);
92                  if (sParamName[0].compareTo((String)vSqlParam.
93                                                      elementAt(i))
94                          == 0) {
95                      vSqlParamValues.addElement(sParamName[1]);
96                  }
97              }
98          }
99          }
100         }catch(Exception e){
101            e.printStackTrace();
102         }
103         return  vSqlParamValues;
104     }
105
106     public boolean hasParamVectorValidElements (Vector v) {
107         boolean bValidParams = true;
108         for (int i = 0; i < v.size(); i++) {
109             if (((String)v.elementAt(i)).compareTo("") == 0)
110                 bValidParams = false;
111         }
112         return  bValidParams;
113     }
114
115     public Document getDocumentDataFromDB (String messageName) {
116         Document doc= null;
117         Document dDoc = null;
118         ResultSet rs = null;
119         String sDocCode = null;
120         dDoc = xmltool.getEmptyDocument();
121         try {
122             String sql = "select * from p_xmldoc_sqldata('" +
123                                         messageName + "')";
124             rs = db.executeQuery(sql);
125             boolean b = rs.next();
126             sDocCode = rs.getObject("DOC_CODE").toString();
127             dDoc.appendChild(dDoc.createElementNS("", rs.
128                   getObject("MESSAGEDOCELEMENT").toString()));
129             do {
130                 //Aus dem bereits existierenden Vektor v
131                 //einen neuen Vektor vSqlParams mit den Parametern
132                 //fuer die aktuelle Abfrage erstellen
```

```
133                 String abfrage = rs.getObject("SQL_CODE").
134                                                 toString();
135                 Vector vSqlParams = getSqlParamValues(abfrage);
136                 if (hasParamVectorValidElements(vSqlParams)) {
137                 // Gueltigkeit der Parameter pruefen
138                     if(rs.getObject("XPATH_PARAM") == null){
139                         doc = getXmlData(vSqlParams,
140                         rs.getObject("SQL").toString(),
141                         rs.getObject("DOCELEMENT").toString(),
142                         rs.getObject("ROWELEMENT").toString(),
143                         rs.getObject("PK").toString());
144                         if (doc != null) {
145                 // Wenn erstellter document nicht leer, den
146                 //TARGET_NODE an den existierenden
147                 //SOURCE_NODE üeinfgen
148                         Node target_node = xmltool.
149                         getSingleNodeFromDocument(dDoc,
150                         rs.getObject("TARGET_NODE").toString());
151                         NodeList source_nlist = doc.
152                           getElementsByTagName(rs.
153                           getObject("SOURCE_NODE").toString());
154                          for(int i=0; i<source_nlist.
155                                           getLength(); i++){
156                           target_node.appendChild(dDoc.
157                           importNode(source_nlist.item(i), true));
158                          }
159                         }
160                    }else{
161                        String prm_path = rs.
162                            getObject("XPATH_PARAM").toString();
163                        NodeList prm_nodelist = xmltool.
164                        getNodeListFromDocument(dDoc,
165                        rs.getObject("XPATH_PARAM").toString());
166                        for(int j=0; j<prm_nodelist.
167                                           getLength(); j++){
168                            String prm_wert = prm_nodelist.
169                              item(j).getFirstChild().getNodeValue();
170                            vSqlParams.addElement(prm_wert);
171                            doc = getXmlData(vSqlParams,
172                              rs.getObject("SQL").toString(),
173                              rs.getObject("DOCELEMENT").toString(),
174                              rs.getObject("ROWELEMENT").toString(),
175                              rs.getObject("PK").toString());
176                            if (doc != null) {
177                            // Wenn erstellter document nicht leer,
178                            //den TARGET_NODE an den existierenden
179                            //SOURCE_NODE einfuegen
180                                Node target_node = xmltool.
```

```
                                getSingleNodeFromDocument(dDoc,
                                rs.getObject("TARGET_NODE").
                                                toString(), j);
                                NodeList source_nlist = doc.
                                  getElementsByTagName(rs.
                                    getObject("SOURCE_NODE").
                                                    toString());
                                for(int i=0; i<source_nlist.
                                                getLength(); i++){
                                    target_node.appendChild(dDoc.
                                      importNode(source_nlist.
                                                item(i), true));
                                }
                            }
                            vSqlParams.clear();
                        }
                    }
                }
            } while (rs.next());
        } catch (Exception ex) {
            ex.printStackTrace();
        }
        dDoc = this.addDocAttributes(sDocCode, dDoc);
        xmltool.writeXmlFile(dDoc, "D:\\document.xml");
        try{
            db.closeDatabase();
        }catch(Exception e){
            e.printStackTrace();
        }
        return  dDoc;
    }

    public Document addDocAttributes(String doc_code,
                            Document doc_without_attr){
        Document doc = doc_without_attr;
        ResultSet attr_rs = null;
        ResultSet sql_rs = null;
        String[] sDocParam = null;
        String sPrmValue = null;
        Node nTarget = null;
        String attr_sql = "select * from p_xmldoc_attribute (" +
                                            doc_code + ")";
        String sPrmSql = null;
        String sPrmArt = null;

        try{
            attr_rs = db.executeQuery(attr_sql);
            if (db.getQueryRowCount(attr_sql) > 0) {
```

```
229                     attr_rs.next();
230                     do{
231                         sPrmSql = attr_rs.getObject("SQL").toString();
232                         sPrmArt = attr_rs.getObject("PARAM_ART").
233                                                         toString();
234                         if(sPrmArt.compareTo("SYSPARAM") == 0){
235                             nTarget = xmltool.
236                              getSingleNodeFromDocument(doc,
237                              attr_rs.getObject("PARENT_NODE_PATH").
238                                                         toString());
239                             Element el = (Element)nTarget;
240                             for (int i = 0; i < vParam.size(); i++) {
241                                 sDocParam = (String[])vParam.
242                                                         elementAt(i);
243                                 if (sDocParam[0].compareTo(attr_rs.
244                                   getObject("PARAM_NAME").
245                                                 toString()) == 0) {
246                                     // Den Wert des Parameters holen
247                                     sPrmValue = sDocParam[1];
248                                 }
249                             }
250                             if(sPrmSql.compareTo("param") == 0){
251                                 el.setAttribute(attr_rs.
252                                   getObject("ATTRIBUTE_NAME").
253                                         toString(), sPrmValue);
254                             }else{
255                                 sql_rs = null;
256                                 sql_rs = db.executeQuery(sPrmSql.
257                                   replaceFirst(":param", sPrmValue));
258                                 sql_rs.next();
259                                 el.setAttribute(attr_rs.
260                                 getObject("ATTRIBUTE_NAME").toString(),
261                                 sql_rs.getObject("PRMVALUE").
262                                                         toString());
263                                 sql_rs.close();
264                             }
265                         }
266                         if(sPrmArt.compareTo("XPATHPARAM") == 0){
267                             NodeList prm_nodelist = xmltool.
268                             getNodeListFromDocument(doc,
269                             attr_rs.getObject("XPATH_TO_PARAM").
270                                                         toString());
271                             for(int j=0; j<prm_nodelist.
272                                                 getLength(); j++){
273                                 String prm_wert = prm_nodelist.item(j).
274                                     getFirstChild().getNodeValue();
275                                 ResultSet rs = db.executeQuery(sPrmSql.
276                                     replaceFirst(":param", prm_wert));
```

```
277                                 rs.next();
278                                 Node target_node = xmltool.
279                                  getSingleNodeFromDocument(doc,
280                                  attr_rs.getObject("PARENT_NODE_PATH").
281                                                          toString(), j);
282                                 Element el = (Element)target_node;
283                                 el.setAttribute(attr_rs.
284                                     getObject("ATTRIBUTE_NAME").
285                                                          toString(),
286                                     rs.getObject("PRMVALUE").
287                                                          toString());
288                                 rs.close();
289                             }
290                         }
291                     }while(attr_rs.next());
292                     attr_rs.close();
293                 }
294             }catch(Exception e){
295                 e.printStackTrace();
296             }
297 
298             return doc;
299         }
300 
301         public Document getXMLSingleNode(Vector vParam,
302                                          String source_node){
303           Document doc = null;
304           doc = xmltool.getEmptyDocument();
305           Element nDateStart = doc.createElementNS("", source_node);
306           nDateStart.appendChild(doc.createTextNode((String)vParam.
307                                                      elementAt(0)));
308           doc.appendChild((Node)nDateStart);
309           return doc;
310         }
311 
312         private String getQueryString(String sSql, Vector param) {
313             String sQuery = "";
314             String sParam = "";
315             String sql = sSql;
316             if(param.size()>0){
317                 for (int i = 0; i < param.size(); i++) {
318                     if(i>0){
319                         sParam += ", ";
320                     }
321                     String paramValue = (String)param.elementAt(i);
322                     sParam += "'" + paramValue + "'";
323                 }
324                 sQuery = sql.substring(0,(sql.indexOf("(")+1)) +
```

```
325                                                         sParam + ")";
326         }
327         else{
328             sQuery = sSql;
329         }
330         return sQuery;
331     }
332 
333     public Document getXmlData (Vector vParam, String sSql,
334                 String sDocElement, String sRowElement,
335                                             String xmlsql) {
336         Document result = null;
337         Vector map = null;
338         map = getMappingData(xmlsql);
339         try {
340             Document docNeu = xmltool.getEmptyDocument();
341             Element root = docNeu.createElementNS("", sDocElement);
342             docNeu.appendChild(root);
343             String Sql = getQueryString(sSql, vParam);
344             ResultSet rsxml = db.executeQuery(Sql);
345             String[] value = null;
346             Element oColumn = null;
347             int row = 0;
348             boolean colInMap = false;
349             if (db.getQueryRowCount(Sql) > 0) {
350                 rsxml.next();
351                 do {
352                     docNeu.getFirstChild().appendChild(docNeu.
353                                     createElement(sRowElement));
354                     Node nRow = docNeu.
355                      getElementsByTagName(sRowElement).item(row);
356                     row++;
357                     for (int i = 0; i < db.
358                                 getQueryColumnCount(Sql); i++) {
359                         if(rsxml.getObject(rsxml.getMetaData().
360                                 getColumnName(i+1).toString()).
361                                   toString().compareTo("")!= 0) {
362                             String colname = rsxml.getMetaData().
363                                     getColumnName(i+1).toString();
364                             colInMap = false;
365                             if(map != null){
366                               for(int j = 0; j < map.size(); j++){
367                                   value = (String[])map.elementAt(j);
368                                       if(value[0].compareTo(rsxml.
369                                           getMetaData().
370                                           getColumnName(i+1).
371                                               toString())==0){
372                                           oColumn = docNeu.
```

```
373                                                 createElement(value[1]);
374                                             colInMap = true;
375                                         }
376                                     }
377                                 }
378                                 if(!colInMap){
379                                   oColumn = docNeu.createElement(rsxml.
380                                     getMetaData().getColumnName(i+1).
381                                                         toString());
382                                 }
383                                 nRow.appendChild(oColumn);
384                                 oColumn.appendChild(docNeu.
385                                  createTextNode(rsxml.getObject(rsxml.
386                                  getMetaData().getColumnName(i+1).
387                                             toString()).toString()));
388                             }
389                         }
390                     } while (rsxml.next());
391                 }
392                 else {
393                     docNeu = null;
394                 }
395                 result = docNeu;
396             } catch (Exception ex) {
397                 ex.printStackTrace();
398             }
399             return  result;
400         }
401
402         private Vector getMappingData(String xmldata_pk){
403             String[] sPara = null;
404             String sql = "select * from tdat_xmlelement_mapping
405                         where xmldata_code = " + xmldata_pk;
406             ResultSet rsmapping = db.executeQuery(sql);
407             Vector vek = null;
408             try{
409                 int rowcount = db.getQueryRowCount(sql);
410                 if (rowcount > 0) {
411                     rsmapping.next();
412                     vek = new Vector(rowcount);
413                     do{
414                         sPara = new String[2];
415                         sPara[0] = rsmapping.getObject("SQL_COLNAME").
416                                                         toString();
417                         sPara[1] = rsmapping.getObject("XML_TAGNAME").
418                                                         toString();
419                         vek.addElement(sPara);
420                     }while(rsmapping.next());
```

```
421                 }
422             }catch(Exception e){
423                 e.printStackTrace();
424             }
425             return vek;
426         }
427 }
```

B.4 Weitere Klassen

Die Klasse Date

Listing B.20: *Date.java*

```
1 package myapplication.basic;
2
3 import java.sql.ResultSet;
4 import java.util.Calendar;
5 import java.util.StringTokenizer;
6 import myapplication.database.Dm;
7
8 public class Date {
9     Time timeObj = new Time();
10    Dm db = new Dm();
11
12    public String getCurrentTime(){
13        String time = "";
14        java.util.Date date = new java.util.
15                Date(System.currentTimeMillis());
16        time = getCurrentTimeForDb(date.toString());
17        return time;
18    }
19
20    public String getCurrentTimeForDb(String str){
21        String time = "";
22        String date = "";
23        String month = "";
24        String year = "";
25        String hour_min = "";
26        time = String.valueOf(System.currentTimeMillis());
27        time = date.toString();
28        StringTokenizer tokenizer = new StringTokenizer(str, " ");
29        int i = 0;
30        while ( tokenizer.hasMoreTokens() ){
31            i++;
32            String curtoken = tokenizer.nextToken();
```

```
33                  if(i==2){
34                      month = curtoken;
35                  }
36                  if(i==3){
37                      date = curtoken;
38                  }
39                  if(i==4){
40                      hour_min = curtoken;
41                  }
42                  if(i==6){
43                      year = curtoken;
44                  }
45              }
46              time = year + "-" + this.getMonthNumber(month) +
47                                  "-" + date + " " + hour_min;
48              return time;
49          }
50
51          private String getMonthNumber(String month){
52              String monthNumber = "";
53              if(String.valueOf(month).compareTo("Jan")==0){
54                  monthNumber = "1";
55              }
56              if(String.valueOf(month).compareTo("Feb")==0){
57                  monthNumber = "2";
58              }
59              if(String.valueOf(month).compareTo("Mar")==0){
60                  monthNumber = "3";
61              }
62              if(String.valueOf(month).compareTo("Apr")==0){
63                  monthNumber = "4";
64              }
65              if(String.valueOf(month).compareTo("May")==0){
66                  monthNumber = "5";
67              }
68              if(String.valueOf(month).compareTo("Jun")==0){
69                  monthNumber = "6";
70              }
71              if(String.valueOf(month).compareTo("Jul")==0){
72                  monthNumber = "7";
73              }
74              if(String.valueOf(month).compareTo("Aug")==0){
75                  monthNumber = "8";
76              }
77              if(String.valueOf(month).compareTo("Sept")==0){
78                  monthNumber = "9";
79              }
80              if(String.valueOf(month).compareTo("Oct")==0){
```

```
81                  monthNumber = "10";
82              }
83              if(String.valueOf(month).compareTo("Nov")==0){
84                  monthNumber = "11";
85              }
86              if(String.valueOf(month).compareTo("Dec")==0){
87                  monthNumber = "12";
88              }
89              return monthNumber;
90          }
91
92          public String getAuctionTerminationTime(String sOpeningTime,
93                                                  int sMinDuration){
94              String termination = "";
95              int[] duration = new int[3];
96              String opening = sOpeningTime;
97              String duration_in_mins = String.valueOf(sMinDuration);
98              duration = timeObj.
99                          getTimeDurationFromMinutes(duration_in_mins);
100             Calendar openingDate = Calendar.getInstance();
101
102             openingDate.set(Calendar.DAY_OF_MONTH,
103                         this.getDateFromDBDateFormat(opening));
104             openingDate.set(Calendar.MONTH,
105                         this.getMonthFromDBDateFormat(opening));
106             openingDate.set(Calendar.YEAR,
107                         this.getYearFromDBDateFormat(opening));
108             openingDate.set(Calendar.HOUR_OF_DAY,
109                         this.getHourFromDBDateFormat(opening));
110             openingDate.set(Calendar.MINUTE,
111                         this.getMinuteFromDBDateFormat(opening));
112             openingDate.set(Calendar.SECOND,
113                         this.getSecondFromDBDateFormat(opening));
114
115             openingDate.add(Calendar.DATE, duration[0]);
116             openingDate.add(Calendar.HOUR, duration[1]);
117             openingDate.add(Calendar.MINUTE, duration[2]);
118             termination = getDBTimeFromGregorianCalendar(openingDate);
119             return termination;
120         }
121
122         private Calendar getAucTerminationFromDb(String prodpk){
123             Calendar aucTermination = Calendar.getInstance();
124             String sDate;
125             ResultSet rs = null;
126             String sql = "select termination from
127                     tdat_auction where product_code=" +prodpk;
128             try{
```

```
129             rs = db.executeQuery(sql);
130             rs.next();
131             sDate = rs.getObject("TERMINATION").toString();
132             aucTermination.set(Calendar.YEAR,
133                         this.getYearFromDBDateFormat(sDate));
134             aucTermination.set(Calendar.MONTH,
135                         this.getMonthFromDBDateFormat(sDate));
136             aucTermination.set(Calendar.DAY_OF_MONTH,
137                         this.getDateFromDBDateFormat(sDate));
138             aucTermination.set(Calendar.HOUR_OF_DAY,
139                         this.getHourFromDBDateFormat(sDate));
140             aucTermination.set(Calendar.MINUTE,
141                         this.getMinuteFromDBDateFormat(sDate));
142             aucTermination.set(Calendar.SECOND,
143                         this.getSecondFromDBDateFormat(sDate));
144         }catch(Exception e){
145             e.printStackTrace();
146         }
147 
148         return aucTermination;
149     }
150 
151     public String getLeftAuctionDuration_Mins(String prodpk){
152         int left_mins = 0;
153         Calendar currentDate = Calendar.getInstance();
154         java.util.Date cDate = currentDate.getTime();
155         Calendar cDateTermination = null;
156         cDateTermination = getAucTerminationFromDb(prodpk);
157         left_mins = this.minutesBetween(currentDate.getTime(),
158                                 cDateTermination.getTime());
159         return String.valueOf(left_mins);
160     }
161 
162    public int minutesBetween(java.util.Date d1, java.util.Date d2){
163       long difference = (d2.getTime() - d1.getTime())/60 * 60 * 24;
164       int diff = (int)((difference/1000)/60)/24;
165       return diff;
166    }
167 
168 
169     private String getDBTimeFromGregorianCalendar(Calendar calendar){
170         String date = "";
171         date += String.
172                 valueOf(calendar.get(Calendar.YEAR)) + "-";
173         date += String.
174                 valueOf(calendar.get(Calendar.MONTH)+1) + "-";
175         date += String.
176                 valueOf(calendar.get(Calendar.DAY_OF_MONTH)) + " ";
```

```
177         date += String.
178                 valueOf(calendar.get(Calendar.HOUR_OF_DAY)) + ":";
179         date += String.
180                 valueOf(calendar.get(Calendar.MINUTE)) + ":";
181         date += String.
182                 valueOf(calendar.get(Calendar.SECOND));
183         return date;
184     }
185 
186     private String getTimeFromDBDateFormat(String DBDate){
187         String time = "";
188         StringTokenizer dateTokenizer =
189                         new StringTokenizer(DBDate, " ");
190         dateTokenizer.nextToken();
191         time = dateTokenizer.nextToken();
192         return time;
193     }
194 
195     private int getHourFromDBDateFormat(String DBDate){
196         int hour;
197         StringTokenizer tokenizer = new StringTokenizer(this.
198                         getTimeFromDBDateFormat(DBDate), ":");
199         hour = Integer.valueOf(tokenizer.nextToken()).intValue();
200         return hour;
201     }
202 
203     private int getMinuteFromDBDateFormat(String DBDate){
204         int miunute;
205         StringTokenizer tokenizer = new StringTokenizer(this.
206                         getTimeFromDBDateFormat(DBDate), ":");
207         tokenizer.nextToken();
208         miunute = Integer.
209                 valueOf(tokenizer.nextToken()).intValue();
210         return miunute;
211     }
212 
213     private int getSecondFromDBDateFormat(String DBDate){
214         String sSecond = "";
215         int second;
216         int i = 0;
217         StringTokenizer secondtokenizer = null;
218         String time = this.
219                 getTimeFromDBDateFormat(DBDate).substring(0, 8);
220         secondtokenizer = new StringTokenizer(time, ":");
221         while ( secondtokenizer.hasMoreTokens() ){
222             i++;
223             String curtoken = secondtokenizer.nextToken();
224             if(i==3){
```

```
225                 sSecond = curtoken;
226             }
227         }
228         second = Integer.valueOf(sSecond).intValue();
229         return second;
230     }
231 
232     private String getDateMonthYearFromDBDateFormat(String DBDate){
233         String time = "";
234         StringTokenizer dateTokenizer =
235                     new StringTokenizer(DBDate, " ");
236         time = dateTokenizer.nextToken();
237         return time;
238     }
239 
240     private int getDateFromDBDateFormat(String DBDate){
241         int date;
242         StringTokenizer tokenizer = new StringTokenizer(this.
243                     getDateMonthYearFromDBDateFormat(DBDate), "-");
244         tokenizer.nextToken();
245         tokenizer.nextToken();
246         date = Integer.valueOf(tokenizer.nextToken()).intValue();
247         return date;
248     }
249 
250     private int getMonthFromDBDateFormat(String DBDate){
251         int month;
252         StringTokenizer tokenizer = new StringTokenizer(this.
253                     getDateMonthYearFromDBDateFormat(DBDate), "-");
254         tokenizer.nextToken();
255         month = Integer.valueOf(tokenizer.nextToken()).intValue();
256         return month-1;
257     }
258 
259     private int getYearFromDBDateFormat(String DBDate){
260         int year;
261         StringTokenizer tokenizer = new StringTokenizer(this.
262                     getDateMonthYearFromDBDateFormat(DBDate), "-");
263         year = Integer.valueOf(tokenizer.nextToken()).intValue();
264         return year;
265     }
266 }
```

Die Klasse Number

Listing B.21: *Number.java*

```
package myapplication.basic;

import java.text.DecimalFormat;

public class Number {

    public String getNumberForDb(String nr){
        String number = nr;
        number = number.replace(',', '.');
        return number;
    }

    public String getFormatedDouble(String sNumber){
        String sFormatedNr = "";
        DecimalFormat df = new DecimalFormat("#0.00");
        double dNumber = Double.valueOf(sNumber).doubleValue();
        sFormatedNr = df.format(dNumber);
        return sFormatedNr;
    }

    public boolean isNumberValid(String wert){
        boolean isNumberValid = true;
        String number = "";
        number = wert;
        boolean isTokenValid = false;
        for(int i = 0; i < number.length(); i++){
            isTokenValid = false;
            String token = "";
            token = number.substring(i, i+1);
            if(token.compareTo("0")==0){
                isTokenValid = true;
            }
            if(token.compareTo("1")==0){
                isTokenValid = true;
            }
            if(token.compareTo("2")==0){
                isTokenValid = true;
            }
            if(token.compareTo("3")==0){
                isTokenValid = true;
            }
            if(token.compareTo("4")==0){
                isTokenValid = true;
            }
```

```
45                if(token.compareTo("5")==0){
46                    isTokenValid = true;
47                }
48                if(token.compareTo("6")==0){
49                    isTokenValid = true;
50                }
51                if(token.compareTo("7")==0){
52                    isTokenValid = true;
53                }
54                if(token.compareTo("8")==0){
55                    isTokenValid = true;
56                }
57                if(token.compareTo("9")==0){
58                    isTokenValid = true;
59                }
60                if(token.compareTo(".")==0){
61                    isTokenValid = true;
62                }
63                if(token.compareTo(",")==0){
64                    isTokenValid = true;
65                }
66                if(isTokenValid == false){
67                    isNumberValid = false;
68                }
69            }
70            return isNumberValid;
71        }
72    }
```

Die Klasse Time

Listing B.22: *Time.java*

```
1  package myapplication.basic;
2
3  public class Time {
4
5      public int getTimeInMinutes(String days, String hours){
6          int totalminutes = 0;
7          totalminutes += Integer.valueOf(days).intValue()*24*60;
8          totalminutes += Integer.valueOf(hours).intValue()*60;
9          return totalminutes;
10     }
11
12     public int[] getTimeDurationFromMinutes(String sMin){
13         int[] time = new int[3];
14         int iMin = Integer.valueOf(sMin).intValue();
```

```
15          time[0] = iMin / 1440;
16          int rest = iMin % 1440;
17          time[1] = rest / 60;
18          time[2] = rest %60;
19          return time;
20      }
21  }
```

Die Klasse UpdateAuctionStatus

Listing B.23: *UpdateAuctionStatus.java*

```
1  package myapplication.basic;
2
3  import myapplication.bean.Auction; import
4  myapplication.database.Dm;
5
6  public class UpdateAuctionStatus extends Thread{
7      boolean bStopThread = false;
8      Dm db = new Dm();
9      Auction auction = new Auction();
10
11     public UpdateAuctionStatus(){
12     }
13
14     public void run(){
15         while(!bStopThread){
16             auction.updateAuctionData();
17         }
18     }
19
20     public void stopUpdate(){
21         bStopThread = true;
22     }
23
24 }
```