Hochschule Kempten
University of Applied Sciences

Praktikumsguide 5

Entwicklung eines Informationssystems (mit Enterprise Java Beans)

Teil I

Lernziele / Lernerfolg:

- Grundkenntnisse SQL und MySQL-Administration
- JAVA, JSP
- MVC
- EJB3
- Tomcat / JBoss

Vorbereitung Praktikum:

Wechseln Sie in das Laufwerk X:/ auf Ihrer Entwicklungsumgebung. Dort finden Sie die **Vorlagen** die Sie für Ihr Praktikum benötigen. Machen Sie sich Bewusst welche Informationen für Sie in den Vorlagen bereitgestellt wurden und welche Sie für die Erreichung des Aufgabenziels benötigen. Folgende **Vorlagen** stehen Ihnen zur Verfügung:

Quellcode-Beispiele sämtlicher benötigter Komponenten:

- EJB-Implementierung
- Webclient
- alternativer Standalone-Java-Client

Als Hilfestellung wurde Ihnen ein **Schaubild** bereit gestellt, das Sie sich selbst vollständig erschließen sollten.

Hintergrundinformationen:
Das bestehende **IKS** aus Aufgabe 4 soll nun weiterentwickelt werden. Dazu ist die Model-Komponente Ihrer Lösung aus Aufgabe 4 durch eine entsprechende Implementierung mit **Enterprise Java Beans** (EJB3) zu ersetzen.

Der Rest der Ihrer bestehenden Implementierung (also View- und Controller-Klassen) soll **weitgehend unverändert** wiederverwendet werden.

In der **Vorlage** sind funktionstüchtige Codebeispiele enthalten, die die IKS-Businessfunktionen in einer Entity Bean, einer Session Bean und einer zugehörigen Schnittstelle implementieren. Außerdem erhalten Sie einen voll funktionstüchtigen **Client**, der zum Test Ihrer EJB-Implementierung dienen soll (und als Vorlage für Teil B ☺).

Die Codestellen, an denen von Ihnen Erweiterungen vorgenommen werden, sind mit **TODO**-Kommentaren bzw. Dummy-Strings gekennzeichnet.

Tipp: **Vermeiden Sie das bloße Copy&Paste von Quellcode aus der Vorlage!**
Sie müssen jede Zeile Ihres **kompletten** IKS verstehen und in der Lage sein, deren Sinn zu erklären. Außerdem fällt dann das Troubleshooting leichter ☺.

Tipp: Eine der Intentionen hinter den Java Beans ist die **Wiederverwendbarkeit** von Komponenten. Wenn Sie geschickt implementieren, können Sie Ihre Ergebnisse von Aufgabe 4 direkt weiterverwenden.

Nachdem Aufgabe 5 auf die vorige Aufgabe 4 aufbaut, gelten natürlich weiterhin alle Hinweise und Tools aus Aufgabe 4. Sie werden weiterhin den JBoss-Applikationsserver verwenden, und können weiterhin die Datenbank mit phpmyadmin beobachten.

Folgenden **Links** könnten für Sie hilfreich sein:
http://www.torsten-horn.de/techdocs/index.htm

Teil II

Variable Informationen

Verbindung zum Datenbankbackend (vgl. Aufgabe 4)

Allgemeine Vorbereitungen

Machen Sie sich mit der Infrastruktur vertraut. Hinweise dazu finden Sie im Praktikumsguide für Aufgabe 4. Auch Ihre Erfahrungen während der Bearbeitung von Aufgabe 4 werden weiterhin von Nutzen sein.

Sie werden auf dem XP-Entwicklungsrechner die Java-Installation zum Kompilieren Ihres Quellcodes und zum Ausführen des Testclients verwenden. Außerdem läuft dort ja der JBoss-Applikationsserver, auf dem Sie Ihr EJB-Paket sowie die Webanwendung mit dem Frontend deployen werden.

Das Datenbank-Backend stellt weiterhin die MySQL-Datenbank auf dem Liveserver dar, die Zugangsdaten und der Zugriff via phpmyadmin sind identisch mit Aufgabe 4.

Tipp: Machen Sie sich zuerst mit dem kompletten Deployment-Prozess des EJB-Paketes vertraut, und kompilieren Sie den Testclient aus der Vorlage.

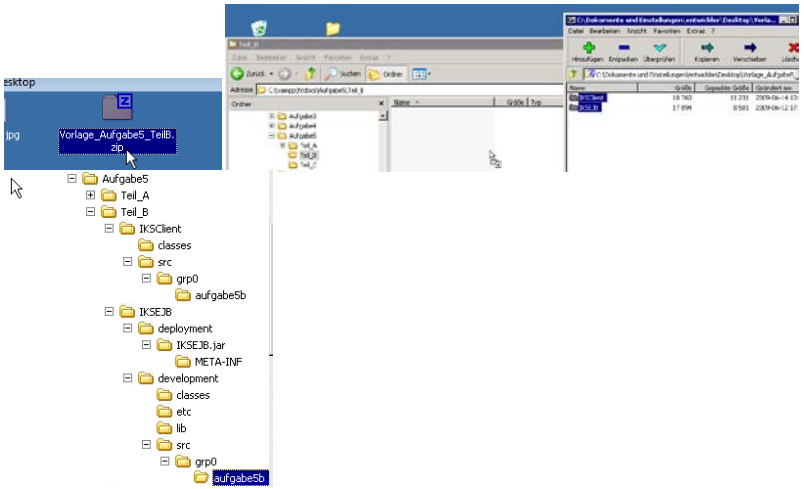
Danach können Sie mit der eigentlichen Implementierungsarbeit in den Java-Quelltexten beginnen.

Der vorliegende Praktikumsguide ist ebenfalls in dieser Reihenfolge aufgebaut.

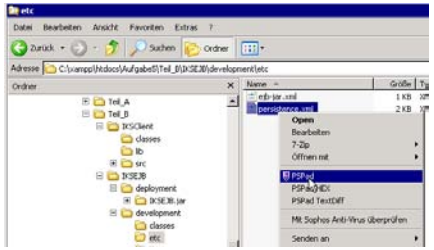
Deploymentprozess der EJB

In Vorlage sind Quelltextbeispiele enthalten, mit denen Sie den Deploymentprozess komplett durchspielen können:

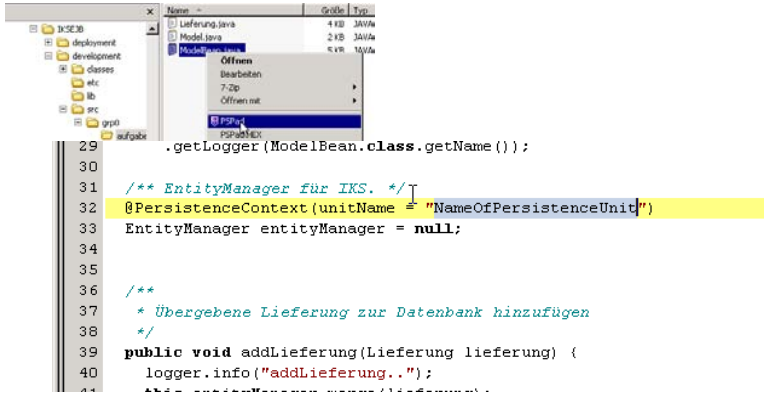
1. Stellen Sie die Verbindung zu Ihrem persönlichen Virtual Lab her. Informationen dazu finden Sie im Praktikumsguide, allgemeiner Teil.
2. Entpacken Sie die Vorlage. Sie finden darin zwei Verzeichnisse, **IKSEJB** für die Enterprise Java Beans, und **IKSClient** mit dem Testclient. Im IKSEJB-Verzeichnis gibt es (wie in Aufgabe 4) zwei Unterverzeichnisse für **development** und **deployment**.



3. Editieren Sie die Descriptor-Dateien im etc-Verzeichnis. Tragen Sie in der Datei **persistence.xml** einen bezeichnenden Namen für die **Persistence-Unit** ein, und ändern Sie den Wert für **die jta-data-source** auf die richtige Datasource, die auf die MySQL-Datenbank auf dem Liveserver zeigt.
   (Tipp: Hinweise dazu finden Sie im Praktikumsguide Aufgabe 4, „Konfiguration von JBoss“ bzw. in der Model-Klasse von Aufgabe 4, in der Funktion, die die Datenbankverbindung herstellt)



4. Wechseln Sie ins Unterverzeichnis mit den java-Quellcode Dateien. Editieren Sie die Datei **ModelBean.java**, indem Sie in der Konfiguration der **PersistenceUnit** (ca. Zeile 32) den entsprechenden Namen Persistence-Unit des Persistence-Descriptors aus dem vorigen Schritt eintragen.

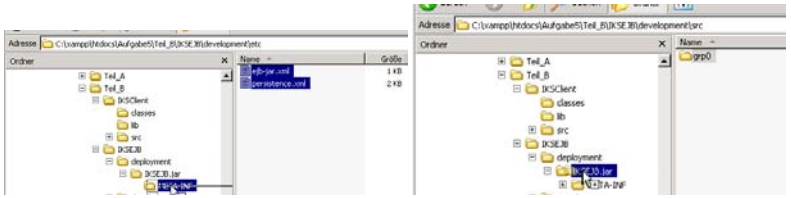


5. Öffnen Sie ein Konsolenfenster, und wechseln Sie ins src-Verzeichnis der development-Umgebung von IKSEJB.
6. Kompilieren Sie die java-Dateien der Vorlage. Dabei sind die Bibliotheken von JBoss mit einzubinden, die kompilierten class-Dateien sollen ins classes-Verzeichnis der development-Umgebung.
   Der entsprechende Befehl muss also lauten:
   `javac –cp c:\jboss-5.0.1.GA\common\lib\*;c:\jboss-5.0.1.GA\server\default\lib\* -d ..\classes grp0\aufgabe5\*.java`
   Tipp: Achtung, der Package-Name der Vorlage lautet nun grp0.aufgabe5! (Denken Sie im weiteren Verlauf Ihrer Arbeiten daran, evtl. müssen Sie später an

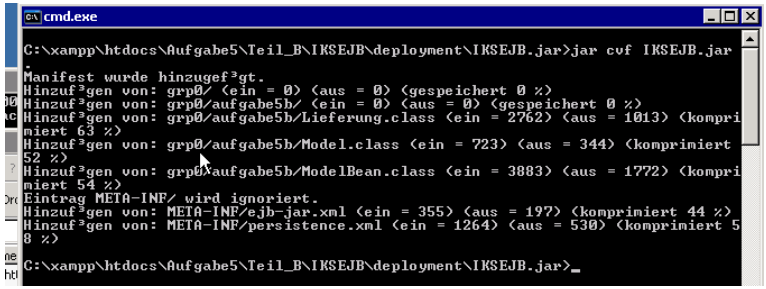
anderer Stelle entsprechende Änderungen vornehmen!)

Evtl. Warnungen des Compilers können Sie (ausnahmsweise ☺) vorerst ignorieren.

7. Kopieren Sie die xml-Dateien und die kompilierten class-Dateien an die richtige Stelle im **deployment**-Verzeichnis (vgl. entsprechender Schritt in Teil A)

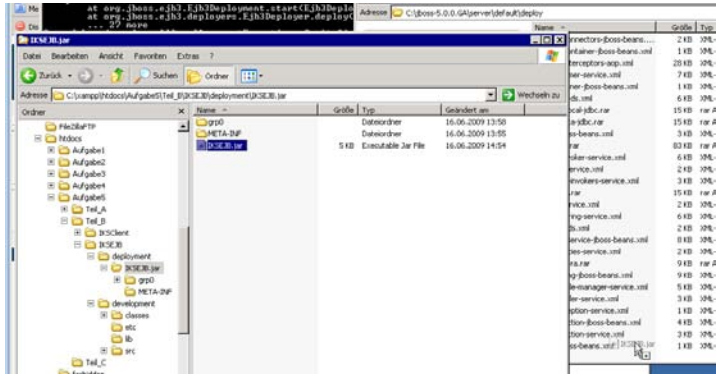


8. Die komplette Deployment-Umgebung wird nun in ein jar-Archiv gepackt, welches später in das JBoss-Deployment-Verzeichnis kopiert wird.
   Wechseln Sie dazu in der Konsole ins Verzeichnis deployment\IKSEJB.jar. Packen Sie das Verzeichnis zu einem Archiv:
   `jar cvf IKSEJB.jar .`
   (Tipp: Vergessen sie den Punkt am Ende nicht!)
   Im Verzeichnis sollte nun eine neue **Datei** names IKSEJB.jar existieren.



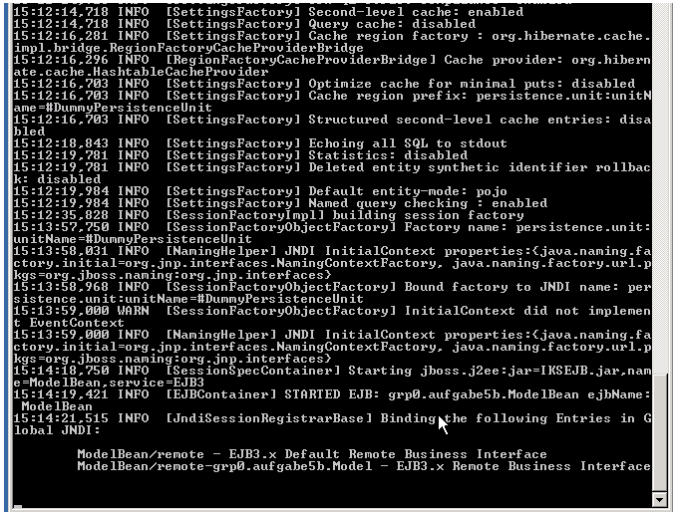
9. Öffnen Sie ein neues Konsolenfenster und starten Sie JBoss. Hinweise dazu finden Sie im Praktikumsguide zu Teil A.

10. Kopieren Sie erstellte **Datei** IKSEJB.jar ins Deployment-Verzeichnis von JBoss unter c:\jboss-5.0.1.GA\server\default\deploy.



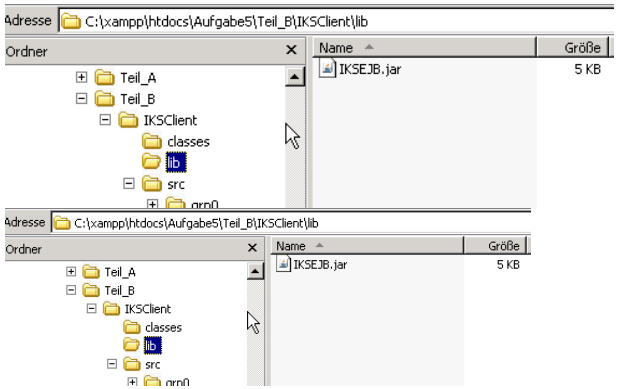
Der Autodeployment-Mechanismus von JBoss wird automatisch das Paket laden. Beachten Sie dazu die Konsolenausgabe von JBoss. Wenn alles klappt,

sollten am Ende die Beans deployt sein, und entsprechend im Nameservice von JBoss registriert sein (was dann in der Konsolenausgabe zu sehen ist).

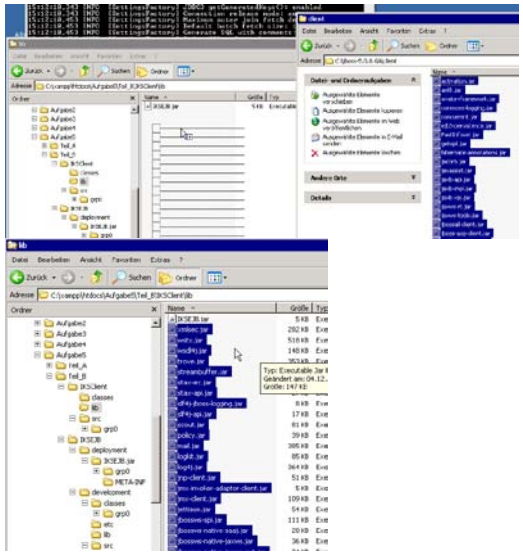


Tipp: Wenn Fehlermeldungen auftauchen, überprüfen Sie noch mal die Einstellungen für die Persistence-Unit in der xml-Datei und der ModelBean.java.

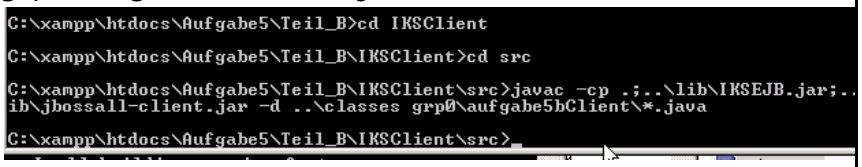
11. Wenn das Bean-Paket erfolgreich deployt wurde können Sie den TestClient kompilieren und den Zugriff auf die Beans testen. Damit die Kompilierung und das Ausführen des Client erfolgreich ist, benötigt dieser noch die entsprechenden Bibliotheken.
Kopieren Sie also die in Schritt 8 erstellte **Datei** IKSEJB.jar ins lib-Verzeichnis des IKSClient.



Außerdem benötigt der Client natürlich noch die Client-Bibliotheken von JBoss. Kopieren Sie dazu den kompletten Verzeichnis-Inhalt von c:\jboss-5.0.1.GA\client (jede Menge jar-Dateien ☺) ins lib-Verzeichnis des IKSClient.

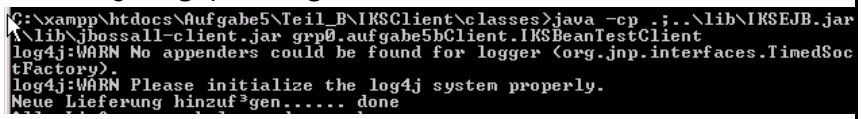


12. Nachdem Bereitstellen der Bibliotheken kann der Client kompiliert werden. Wechseln Sie dazu in der Konsole ins src-Verzeichnis des IKSClient. Kompilieren Sie den Client mit javac, geben Sie dabei im Classpath das aktuelle Verzeichnis . und die Dateien IKSEJB.jar und jbossall-client.jar im lib-Verzeichnis an:
`javac -cp .;..\lib\IKSEJB.jar;..\lib\jbossall-client.jar –d ..\classes grp0\aufgabe5Client\*.java`



Tipp: Achtung, der Package-Name der Client-Vorlage lautet nun grp0.aufgabe5**Client**! (Denken Sie im weiteren Verlauf Ihrer Arbeiten daran, evtl. müssen Sie später an anderer Stelle entsprechende Änderungen vornehmen!)

13. Führen Sie den TestClient aus, wechseln Sie dazu ins classes-Verzeichnis des TestClient. Starten Sie den Client mit
`java –cp .;..\lib\IKSEJB.jar;..\lib\jbossall-client.jar grp0.aufgabe5Client.IKSBeanTestClient.`

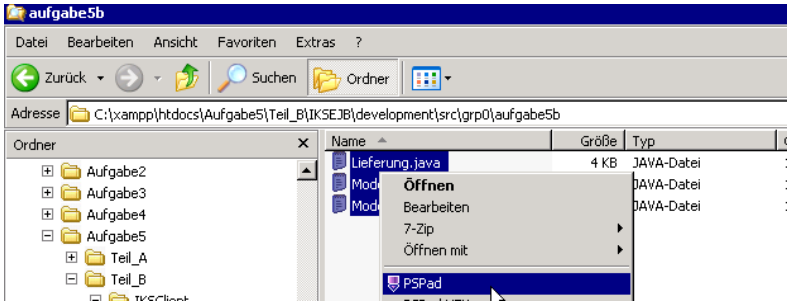


Beobachten Sie die Ausgabe, und verfolgen Sie die Ausgabe des JBoss-Servers.

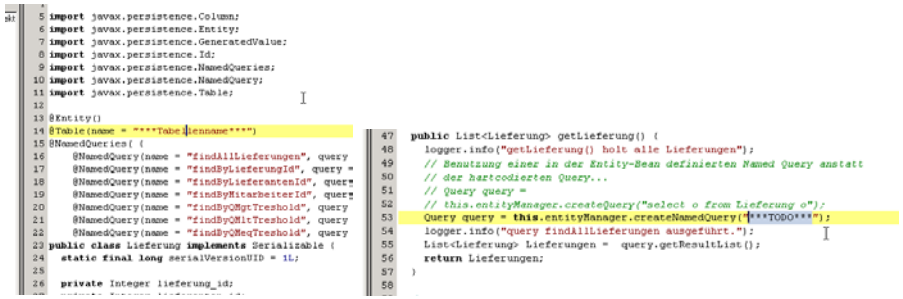
Wenn die der TestClient startet, die Warnungen bzgl. Logging anzeigt und mit der ersten Funktion addLieferung (**Neue Lieferung hinzufügen**… in der Ausgabe des Client) starten will, ist soweit alles in Ordnung. Denn dann hat der Client die ModelBean auf dem JBoss-Server gefunden und führt deren Methoden aus (auch wenn danach etliche Fehlermeldungen kommen, denn noch sind die EJB nicht vollständig korrekt).

Entwicklung der Enterprise Java Beans

Nachdem der Deployment-Prozess gesichert ist, können Sie beginnen, die EJB zu implementieren. Gehen Sie dazu zurück ins scr-Verzeichnis der **development**-Umgebung, und öffnen Sie die java-Dateien.



In der Lieferung.java müssen Sie an folgender Stelle eine Änderung vornehmen: (ca. Zeile 14) Sie müssen den korrekten Namen der **Tabelle** angeben, in der die Lieferung-Objekte gespeichert werden sollen (siehe Datenbankschema Teil A und phpymadmin).

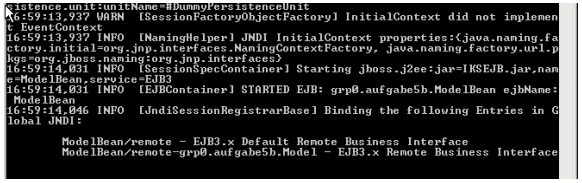


```
5  import javax.persistence.Column;
6  import javax.persistence.Entity;
7  import javax.persistence.GeneratedValue;
8  import javax.persistence.Id;
9  import javax.persistence.NamedQueries;
10 import javax.persistence.NamedQuery;
11 import javax.persistence.Table;
12
13 @Entity
14 @Table(name = "****Tabellenname****")
15 @NamedQueries( {
16     @NamedQuery(name = "findAllLieferungen", query
17     @NamedQuery(name = "findByLieferungId", query =
18     @NamedQuery(name = "findByLieferantenId", query
19     @NamedQuery(name = "findByMitarbeiterId", query
20     @NamedQuery(name = "findByQMgtTreshold", query
21     @NamedQuery(name = "findByQMltTreshold", query
22     @NamedQuery(name = "findByQMeqTreshold", query
23 public class Lieferung implements Serializable {
24   static final long serialVersionUID = 1L;
25
26   private Integer lieferung_id;
```

```
47  public List<Lieferung> getLieferung() {
48    logger.info("getLieferung() holt alle Lieferungen");
49    // Benutzung einer in der Entity-Bean definierten Named Query anstatt
50    // der hartcodierten Query...
51    // Query query =
52    // this.entityManager.createQuery("select o from Lieferung o");
53    Query query = this.entityManager.createNamedQuery("***TODO***");
54    logger.info("query findAllLieferungen ausgeführt.");
55    List<Lieferung> Lieferungen =  query.getResultList();
56    return Lieferungen;
57  }
58
```

In der ModelBean.java müssen Sie folgende Änderungen vornehmen:
In den Funktionen getLieferung() und getLieferungBy…() müssen Sie bei der Ausführung der Abfrage den richtigen Namen der **NamedQuery** angeben. Diese Stellen sind mit ***TODO*** markiert. Die entsprechenden Informationen finden Sie in der zugehörigen Entity Bean.

Nach diesen Änderungen sollten die Beans soweit korrekt funktionieren, dass der Testlauf mit dem Client erfolgreich ist. Führen Sie also den **kompletten** Deployment-Prozess noch einmal durch.

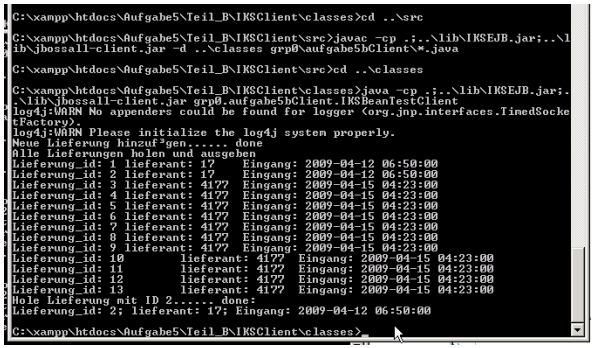


```
sistence.unit:unitName=#DummyPersistenceUnit
6:59:13,937 WARN  [SessionFactoryObjectFactory] InitialContext did not implemen
t EventContext
16:59:13,937 INFO  [NamingHelper] JNDI InitialContext properties:{java.naming.fa
ctory.initial=org.jnp.interfaces.NamingContextFactory, java.naming.factory.url.p
kgs=org.jboss.naming:org.jnp.interfaces}
16:59:14,031 INFO  [SessionSpecContainer] Starting jboss.j2ee:jar=IKSEJB.jar,nam
e=ModelBean,service=EJB3
16:59:14,031 INFO  [EJBContainer] STARTED EJB: grp0.aufgabe5b.ModelBean ejbName:
 ModelBean
16:59:14,046 INFO  [JndiSessionRegistrarBase] Binding the following Entries in G
lobal JNDI:

        ModelBean/remote - EJB3.x Default Remote Business Interface
        ModelBean/remote-grp0.aufgabe5b.Model - EJB3.x Remote Business Interface
```

Wenn das (neue) IKSEJB.jar-Paket auf dem JBoss deployt ist, können Sie wiederum den Client kompilieren und ausführen.

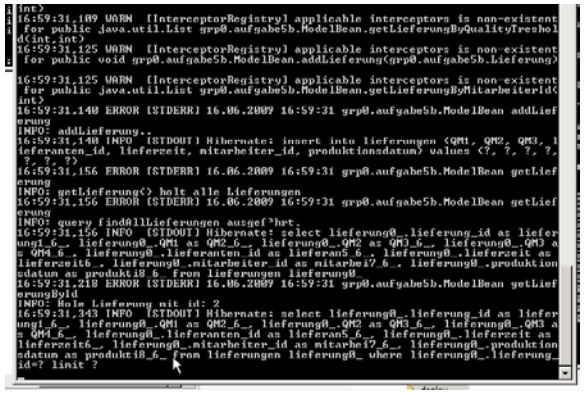
Tipp: Denken Sie daran, dass die Client jeweils die aktuelle Version der IKSEJB.jar im lib-Verzeichnis benötigt!

Ein erfolgreicher Durchlauf des TestClient fügt eine neue Lieferung der Datenbank hinzu, gibt danach alle Lieferungen und schließlich eine bestimmte einzelne Lieferung aus.



```
C:\xampp\htdocs\Aufgabe5\Teil_B\IKSClient\classes>cd ..\src

C:\xampp\htdocs\Aufgabe5\Teil_B\IKSClient\src>javac -cp .;..\lib\IKSEJB.jar;..\l
ib\jbossall-client.jar -d ..\classes grp0\aufgabe5bClient\*.java

C:\xampp\htdocs\Aufgabe5\Teil_B\IKSClient\src>cd ..\classes

C:\xampp\htdocs\Aufgabe5\Teil_B\IKSClient\classes>java -cp .;..\lib\IKSEJB.jar;.
.\lib\jbossall-client.jar grp0.aufgabe5bClient.IKSBeanTestClient
log4j:WARN No appenders could be found for logger (org.jnp.interfaces.TimedSocke
tFactory).
log4j:WARN Please initialize the log4j system properly.
Neue Lieferung hinzuf³gen..... done
Alle Lieferungen holen und ausgeben
Lieferung_id: 1 lieferant: 17    Eingang: 2009-04-12 06:50:00
Lieferung_id: 2 lieferant: 17    Eingang: 2009-04-12 06:50:00
Lieferung_id: 3 lieferant: 4177  Eingang: 2009-04-15 04:23:00
Lieferung_id: 4 lieferant: 4177  Eingang: 2009-04-15 04:23:00
Lieferung_id: 5 lieferant: 4177  Eingang: 2009-04-15 04:23:00
Lieferung_id: 6 lieferant: 4177  Eingang: 2009-04-15 04:23:00
Lieferung_id: 7 lieferant: 4177  Eingang: 2009-04-15 04:23:00
Lieferung_id: 8 lieferant: 4177  Eingang: 2009-04-15 04:23:00
Lieferung_id: 9 lieferant: 4177  Eingang: 2009-04-15 04:23:00
Lieferung_id: 10        lieferant: 4177  Eingang: 2009-04-15 04:23:00
Lieferung_id: 11        lieferant: 4177  Eingang: 2009-04-15 04:23:00
Lieferung_id: 12        lieferant: 4177  Eingang: 2009-04-15 04:23:00
Lieferung_id: 13        lieferant: 4177  Eingang: 2009-04-15 04:23:00
Hole Lieferung mit ID 2..... done:
Lieferung_id: 2; lieferant: 17; Eingang: 2009-04-12 06:50:00

C:\xampp\htdocs\Aufgabe5\Teil_B\IKSClient\classes>
```

Beobachten Sie dabei auch die entsprechende Ausgabe des Hibernate-Persistenzmechanismus in der JBoss-Konsole.



Nun folgt die Anpassung der (neuen) IKS-Businesslogik an die (alte) Model-Klasse aus Aufgabe 4. Öffnen Sie dazu die Schnittstellen-Klasse der EJB Model.java. Fügen Sie ganz unten die **Signaturen** der Businessmethoden der Model-Klasse aus Aufgabe 4 hinzu. (Damit kann der Controller aus Aufgabe 4 die neuen EJB genau so nutzen wie die alte Model-Klasse!). Überprüfen Sie auch die vollständige und korrekte Implementierung der Methoden innerhalb der Session Bean!

```
63    */
64   public List<Lieferung> getLieferungByQualityTreshold(int bedingung,
65       int treshold);
66
67   /*
68    * ==============================
69    * ab hier zusätzliche Methoden, für die Einhaltung des "Vertrages" der
70    * IKS-Model-Klasse aus Aufgabe5 Teil A
71    */
72
73   public String halloWelt();
74
75   //***TODO*** weitere Methodensignaturen für die Businessfunktionen
76   // "Holen aller Lieferungen" und "Eine Lieferung in die DB einfügen"
77
78 }
```

Führen Sie zum Abschluss den kompletten Deployment-Prozess noch einmal durch. Testen Sie die korrekte Funktionalität der IKS-Businesslogik mit dem TestClient! Damit stellen Sie sicher, dass die IKS-Businesslogik vollständig und korrekt arbeitet, bevor Sie die Weboberfläche aus Aufgabe 4 editieren.

Änderung der Weboberfläche

Wenn die IKS-Businesslogik korrekt arbeitet, ist mit wenig Aufwand Ihre Weboberfläche von Aufgabe 4 so konfiguriert, dass an Stelle der alten Model-Klasse nun die EJB zum Einsatz kommen.

Löschen Sie die Datei Model.java aus Ihrer Lösung von Aufgabe 4 (nachdem Sie natürlich eine Sicherheitskopie angelegt haben ☺). Diese wird nun **nicht mehr** benötigt, da der Controller so editiert wird, dass er nun das Model-Interface der EJB-basierten IKS-Businesslogik verwendet.

Öffnen Sie die Controller-Klasse Ihrer Weboberfläche. Suchen Sie die Definition des Model-Attributes des Controllers, und die init()-Methode des Controllers. Dort müssen Sie den Controller dahingehend ändern, dass dieser nicht mehr eine Model-Instanz der Model-Klasse

(„**Model myModel = new Model()**“) erzeugt (denn diese Model-Klasse gibt es ja nicht mehr).

Stattdessen müssen Sie in der **init()**-Methode das Model-Objekt per RMI aus der EJB im JBoss-Server holen. Eine Vorlage dazu finden Sie im Quelltext des **IKSBeanTestClient** (denn dieser macht zu Beginn genau das gleiche!)
Tipp: Denken Sie dabei daran, dass dazu evtl. auch weitere Bibliotheken zu importieren sind!

```
public class ControllerServlet extends HttpServlet {
  private static final long serialVersionUID = 1L;

  private HashMap<String, Action> actions;
  Model model;

  public void init() throws ServletException {
    System.out.println("ControllerServlet.init()");
    // Model-Interface holen:
    Hashtable<String, String> props = //.....****TODO***
    try {
    InitialContext initialContext = new InitialContext(props);
//*******TODO*****
    this.model = lvw;
    System.out.println("Model-Interface wurde geholt (oder nicht :-) )");
    } catch (NamingException ne) {
      throw new ServletException();
    }

    //========ab hier das alte...

    actions = new HashMap<String, Action>();
    ShowAllLieferungenAction action1 = new ShowAllLieferungenAction(model);
```

Wenn die Model-Schnittstelle der IKS-Businesslogik in den EJB nach dem Vorbild der alten Model-Klasse aus Teil A die gleichen Businessfunktionen bereitstellt, müssen in der Weboberfläche sonst keine Veränderungen mehr vorgenommen werden!

Zum Deployen der Weboberfläche als Webapplikation auf dem JBoss beachten Sie die Hinweise im Praktikumsguide Aufgabe 4.

Am Ende sollte das IKS für den Benutzer genauso aussehen und funktionieren wie Ihre Lösung von Aufgabe 4!

Verstehen und dokumentieren Sie also den kompletten Deployment-Prozess, damit Sie Ihr IKS zum Testat auch in eine entsprechende Live-JBoss-Umgebung deployen können!

Teil III

Testat:

1. Die Testatabnahme findet ausschließlich in der Liveumgebung statt.
2. Am Testattermin senden Sie mir unter juergen.bader@fh-kempten.de ein Deploymentpaket bis 8:00 Uhr zu.
3. Bemühen Sie sich um einen Testattermin.
4. Bringen Sie Ihr Testatblatt zur Testatabnahme mit.