Beans und Enterprise Beans

1.1. Einleitende Bemerkungen

Die Aufgaben liefern Ihnen einen Einblick in die EJB Technologie. Sie zeigen Ihnen, wie Sie Enterprise Java Beans (EJBs) Applikationsserver einsetzen, Enterprise Beans basierte Lösungen entwerfen, implementieren und nutzen können. Die Übungen umfassen auch Clients, damit Sie Ihre Lösungen testen können.

1.1.1. Lernziele

Nach dem Bearbeiten der Übungen sind Sie in der Lage

- Enterprise JavaBeans Technologie basierte Server und Clients zu entwickeln
- Sie kennen die grundlegenden Einsatzmöglichkeiten der EJBs und
- kennen die grundlegenden Schritte zur Entwicklung einer EJB basierten Lösung

1.1.2. Bemerkungen zu den Übungen

Übungen müssen flexibel sein und unterschiedliche Hilfeebenen anbieten, je nach Voraussetzungen der Studenten.

- **Aufgabenstellung**:
  Einige Studenten werden in der Lage sein, diese Übungen ohne Zusatzinformationen zu lösen. Sie verwenden dazu lediglich die Aufgabenstellung und die Aufgabenliste.
- **Lösungshinweise:**
  Andere werden die Lösungshinweise und die Beschreibung der Lösungsschritte benutzen müssen. Diese folgen jeweils am Ende der Übungseinheit.
- **Lösung:**
  einige möchten sich vielleicht einfach mit dem Stoff vertraut machen und Schntitt für Schritt durch die Lösung geführt werden möchten

Die einzelnen Aufgaben bauen aufeinander auf! Am Schluss haben Sie die Lösung für ein echtes, reales, konkretes Problem.

Für jede Aufgabe stehen Musterlösungen, Lösungshinweise und natürlich die Aufgabenbeschreibung zur Verfügung. Daher könnten Sie auch einzelne Aufgaben überpringen und einfach die Musterlösung der vorherigen Aufgabe als Basis für das weitere Vorgehen verwenden.

Zu jeder Aufgabe werden die Voraussetzungen aufgelistet, so dass Sie sehen können, welche Aufgaben Sie bereits gelöst haben sollten, oder welche Musterlösungen Sie einbinden müssen. Zudem finden Sie Web-Links auf die API Beschreibungen und jeweils eine kurze Beschreibung der Lernziele dieser Aufgabe.

1.1.3. Übungs Design Ziele

Es gibt drei Arten von Übungen für Sie:

- **"Blank Screen"**
  Sie sehen einen Bildschirm und müssen die Logik dazu entwicklen.
- **Erweiterung**
  Sie müssen ein bereits funktionierendes Beispiel erweitern
- **Korrektur**
  Sie korrigieren fehlerhaftes oder unerwünschtes Verhalten existierender Programme.

Um die Übungen nicht unnötig komplex zu machen, werden Aspekte wie Security in der Regel nicht bearbeitet, ganz im Gegensatz zu realen Programmen.

Daher werden wir auch immer, wenn Applets involviert sind, mit dem Appletviewer arbeiten, da eventuell jemand das Plugin nicht installiert hat.

1.1.4. Enterprise JavaBeans Technologie - Übungsübersicht

Hier zuerst eine Gesamtübersicht über die Übungen.

1. **Installieren und Konfigurieren von Sun's J2EE Reference Implementation**
   Diese Übung führt Sie schrittweise durch die Installation der J2EE™ Reference Implementation, einem Applikationsserver (wie Webspere, Weblogic, Oracle,...) auf Ihrem PC. Diesen Applikationsserver werden wir in den anschliessenden Übungen verwenden.

   Lernziele:
   - Sie installieren Sun's J2EE Ref.Impl. Application Server
   - Ihr PC ist korrekt konfiguriert, so dass EJB übersetzt und eingesetzt werden können

2. **Kreieren einer Entity Bean**
   In dieser Übung implementieren Sie ein MusikCD Bean.

   Lernziele:
   - Sie lernen, wie man eine EJB implementiert, in diesem Falle eine Entity Bean mit Container gemanageten Persistenz
   - Sie machen sich mit der Entwicklung der Enterprise JavaBean Komponenten und deren Nutzungsumgebung vertraut.

3. **Kreieren einer Stateless Session Bean**
   In dieser Übung implementieren Sie eine Session Bean für die Verwaltung des Lagers.

   Lernziele:
   - Sie lernen zustandslose Session Beans zu definieren und zu implementieren
   - Sie machen sich vetraut mit der Entwicklung der Enterprise JavaBeans und mit deren Laufzeitumgebungen
   - Sie lernen aus EJBs andere EJBs einzusetzen, in diesem Fall aus dem selben middle-tier EJB Server

4. Einsatz von Enterprise Beans in Sun's J2EE Reference Implementation
Diese Übung führt Sie durch den konkreten Einsatz der MusicCD Entity Bean und der Lager Session Bean im Container der Referenzimplementation.

Lernziele:

- Sie wissen, wie EJBs im Container der J2EE Referenzimplementation betrieben werden.

5. Aufsetzen der Datenbank
In dieser Übung erfassen Sie Daten in einer Tabelle einer Datenbank (in diesem Falle einer Objekt-Relationalen Datenbank, die zu 100% in Java geschrieben ist). Zudem lernen Sie Programme kennen, mit denen Sie die daten auch in eine der üblichen Datenbanken (Access, Excel, ...Oracle,...) einfügen können.

Lernziele:

- Sie schaffen die Grundlage für die nachfolgenden Aufgaben, indem Sie
  - eine Datenbankumgebung korrekt aufsetzen
  - Daten in diese Datenbank einfügen

6. Entwicklen eines EJB Clients
Nun schreiben Sie eine Client Applikation, mit der Sie den Einsatz und die Funktionalität der Entity und der Session Beans aus den vorherigen Übungen prüfen können.

Lernziele:

- Sie schreiben eine Applikation, welche eine Session Bean einsetzen
- Sie entwickeln eine Applikation, welche eine Entity Bean einsetzen

1.2. Installation und Konfiguration der J2EE Reference Implementation

Falls Sie Hinweise aus der J2EE JavaDoc benötigen: API Docs

Diese einführende Übung führt Sie schrittweise durch die Instalation von Sun's Java™ 2 Enterprise Edition Reference Implementation (J2EERI) Applikationsserver auf Ihrem PC. Diese Übung ist spezifisch auf die J2EERI ausgerichtet. Falls Sie sich entschliessen sollten einen andern Applikationsserver (WebLogic, WebSphere, Oracle,...) einzusetzen, müssen Sie entsprechend anders vorgehen. Der AS sollte aber mindestens EJB 1.1 kompatibel sein!

1.2.1. Voraussetzungen

keine

1.2.2. Arbeitsschritte

Die Schritte im Einzelnen:

1. Überprüfen Sie als erstes die Systemvoraussetzungen! Für J2EERI finden Sie diese beispielsweise auf dem Web.

   Hyperlink: J2EE.

   Sie sollten eine Java 2 Installation Version mindestens 1.2.2 haben.

2. laden Sie die aktuelle Version von Sun's J2EE (Server und Dokumentation) herunter

   Hyperlink: Sun's J2EE Site sowie Dokumentation
   Die Installationsanleitung befindet sich im Dokumentationsset.

3. Installieren Sie J2EE RI und die Dokumentation.

4. Definieren Sie die Umgebungsvariable `J2EE_HOME`. Diese Variable muss auf das Basisverzeichnis der installierten J2EE RI zeigen.

   Beispiel: auf meinem PC gilt `J2EE_HOME=c:\j2sdkee1.2.1`

5. unter Umständen müssen `c:\j2sdkee1.2.1\bin` in den Pfad aufnehmen (ich habe dies weggelassen, weil ich die Jobs mit Bat-Dateien starte,die Pfade jeweils korrekt setzen).

6. Falls Sie auch die Cloudscape Datenbank installiert haben, und das ist empfehlenswert, testen Sie Ihre Installation mit dem Bat-Skript im Verzeichnis `C:\j2sdkee1.2.1\bin` :

   ```
   cloudscape -start
   ```

   Falls Sie die Software korrekt installiert haben,sollte jetzt ein Datenbankserver gestartet werden.

7. Starten Sie jetzt den Applikationsserver von Sun (J2EE RI) mit dem Skript J2EE:
   ```
   J2EE -verbose
   ```

8. Der Applikationsserver sollte sich melden!

   Wenn Sie alles korrekt gemmacht haben, laufen nun die Objekt-relationale Datenbank Cloudscape und der Applikationsserver von Sun.

9. Stoppen Sie beide Services:

   Zuerst müssen Sie den Datenbankserver herunterfahren:
   ```
   Cloudscape -stop
   ```

   Dann kann auch der Applikationsserver herunter gefahren werden:
   ```
   J2EE -stop
   ```

10. Machen Sie sich etwas mit der Dokumentation vertraut. Sie sollten allerdings kaum Probleme haben!

1.2.3. Musterlösung

Diese Übung besitzt keine Musterlösung. Falls Sie alle Schritte erfolgreich abgeschlossen haben, sollte J2EE RI installiert sein und lauffähig und damit für die Lösung der weiteren Aufgaben zur Verfügung stehen.

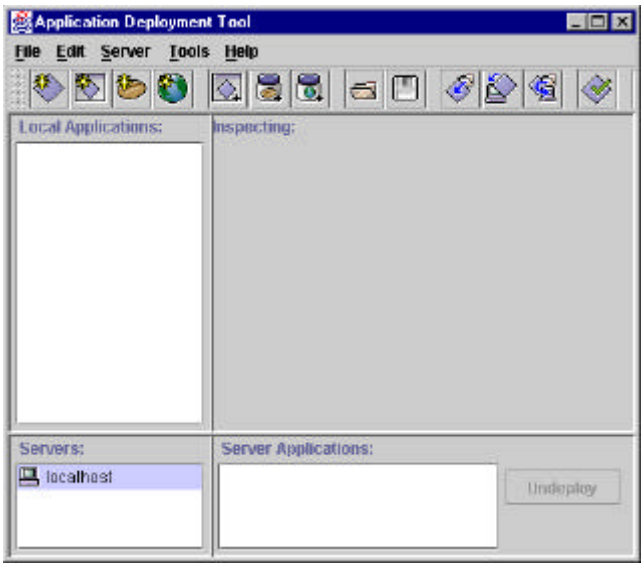
1.2.4. Demonstration

Nach Abschluss der Übung sollte die J2EE RI installiert sein und laufen.

Typischerweise sollten Sie auf dem Bildschirm folgende Ausgabe oder eine ähnliche sehen:

```
j2ee -verbose

Naming service started: :1050
Published the configuration object ...
Binding DataSource, name =
              jdbc/Cloudscape,
    url = jdbc:cloudscape:rmi:
          CloudscapeDB;create=true
Configuring web service using "default"
Web service started: :9191
Web service started: :7000
Configuring web service using "default"
Configuring web service using
  "file:/C:/j2sdkee1.2.1/public_html
                    /WEB-INF/web.xml"
Web service started: :8000
Endpoint [SSL:
  ServerSocket[addr=0.0.0.0/0.0.0.0,port=0,
                            localport=7000]]
    shutdown due to exception:
    javax.net.ssl.SSLException:
    No available certificate corresponds
    to the SSL cipher suites which are enabled.
endpoint down: :7000
J2EE server startup complete.
```

Wenn Sie nun auch noch das `deploytool` starten, sollten Sie folgenden Bildschirm sehen:



Wir werden dieses Hilfsprogramm in einer späteren Übung eimsetzen.

1.2.5. Installation und Konfiguration von Sun's J2EE Referenz Implementation - praktische Hinweise

1.2.5.1. Task 1

Prüfen Sie, ob Ihr System die Voraussetzungen erfüllt, um die J2EE RI zu betreiben.

Typischerweise reicht ein Windows NT PC, aber auch Windows 2000, Windows 98 und Solaris werden unterstützt.

Offiziell werden mindestens 128 MB RAM verlangt, und das als Minimum. Die aktuellen Anforderungen können Sie auf folgender Webseite nachschauen:

J2EE Reference Implementation Tested Configurations.

1.2.5.2. Task 2

Versichern Sie sich, dass Sie die aktuelle Java 2 SDK Version installiert haben, also mindestens Version 1.2.2 oder höher.

J2EE RI benutzt die volle Java 2 SDK Installation. Übrigens läuft die J2EE RI offiziell nicht mit dem Hotspot Compiler.

Auf meinem PC sieht dies so aus:

```
C:\jdk1.3\bin>java -version
java version "1.3.0"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.3.0-C)
Java HotSpot(TM) Client VM (build 1.3.0-C, mixed mode)
```

Offiziell müssen Sie, falls Sie den HotSpot Compiler installiert haben, diesen deinstallieren und anschliessend JDK neu installieren.

1.2.5.3. Task 3

Laden Sie den J2EE RI Server und die Dokumentation herunter.

Hyperlink: für die Software Sun's J2EE download Site
für die Dokumentation Dokumentationssite.
Die Dokumentation umfasst auch eine Installation Anleitung.

Windows oder Solaris Distributionen umfassen je etwa 11Mb. Die Dokumentation umfasst nochmals etwa 7Mb.

1.2.5.4. Task 4

Nach dem Herunterladen brauchen Sie nur den Installer der beiden Pakete zu starten. Als Verzeichnis habe ich die vorgeschlagenen Verzeichnisse gewählt.

Sollten Sie keine Datenbank installiert haben, dann sollten Sie die Objekt-relationale Datenbank Cloudscape installieren. Die Software ist ab Cloudscape.com herunterladbar und ist auch auf dem Server. Der Installer von Cloudscape installiert auch eine JDK/JRE-Version.

1.2.5.5. Task 5

Setzen jetzt die `J2EE_HOME` Umgebungsvariable. Sie muss auf das Basisverzeichnis der installierten J2EE RI zeigen.

Auf Windows setzen Sie die Umgebungsvariablem in der Systemsteuerung - System - Umgebung. Aber Sie können auch mit Bat-Dateien arbeiten und einfach folgende Zeile eintragen:

```
set J2EE_HOME=C:\j2sdkee
```

Beispielsweise auf meinem PC:

```
C:\j2sdkee1.2.1\bin>set j2ee_home
J2EE_HOME=c:\j2sdkee1.2.1
```

1.2.5.6. Task 6

Offiziell sollten Sie jetzt das J2EE Home Verzeichnis auch noch in den Pfad aufnehmen.

Auf Windows Systemen geschieht dies bekanntlich wieder mit Hilfe der Systemsteuerung - System - Umgebung (Windows NT).

Aber Sie können auch mit Bat-Dateien arbeiten und eine Anweisung wie folgende einbauen:

```
set PATH=%PATH%;C:\j2sdkee\bin
```

Auf meinem PC fehlt diese Anweisung auf Systemebene, da sie in den Batch Dateien vorhanden ist.

```
C:\>set path
Path=C:\Cloudscape_3.0_Eval\jre\bin;C:\Perl\bin;C:\WINNT\system32;C:\WINNT;
c:\jdk1.3\bin;
```

Wie Sie sehen, fehlt der Eintrag!

1.2.5.7. Task 7

Sofern Sie Cloudscape installiert ist, und dies ist eine Empfehlung, da im Folgenden immer mit dieser Datenbank gearbeitet wird, können Sie die Datenbank mit folgendem Befehl starten

```
cloudscape -start.
```

Sie sollten dann eine Ausgabe wie unten erhalten (die Datenbak benutzt RMI, JDBC):

```
C:\j2sdkee1.2.1\bin>cloudscape -start
Fri Jul 14 16:15:58 GMT+02:00 2000: [RmiJdbc]
COM.cloudscape.core.JDBCDriver registered in
 DriverManager
Fri Jul 14 16:15:58 GMT+02:00 2000: [RmiJdbc] Binding RmiJdbcServer...
Fri Jul 14 16:15:58 GMT+02:00 2000: [RmiJdbc] No installation of RMI
Security Manager...
Fri Jul 14 16:16:01 GMT+02:00 2000: [RmiJdbc] RmiJdbcServer bound in rmi
registry
```

1.2.5.8. Task 8

Starten sie J2EE RI mit folgendem Befehl:

```
j2ee -verbose.
```

J2EE.Bat ist ein Skript im Verzeichnis J2EE_HOME. Die `-verbose` Option garantiert, dass Sie in etwa folgende Ausgabe erhalten.

```
C:\j2sdkee1.2.1\bin>j2ee -verbose

J2EE server Listen Port: = 1049
Naming service started: :1050
Published the configuration object ...
Binding DataSource, name = jdbc/Cloudscape, url =
jdbc:cloudscape:rmi:CloudscapeDB;create=
true
Web service started: 9191
Web service started: 8000
Web service started: 7000
J2EE server startup complete.
```

1.2.5.9. Task 9

J2EE ist nun installiert und läuft. Schauen Sie sich die Dokumentation der J2EE Reference Implementation etwas genauer an, um sich mit der Software vertraut zu machen.

Hinweis:
Sie können natürlich die beiden Dienste, die Sie eben gestartet haben, mit CTRL-C abbrechen. Aber das ist nicht die feine Art.

Die Befehle für das Stoppen können Sie auf der Kommandozeile erfragen :

```
Cloudscape -stop
J2EE -stop
```

1.3. Kreieren einer Entity Bean

Hyperlink zur API Dokumentation: API Docs

In dieser Übung implementieren Sie eine `MusicCD` Entity Bean, mit der Sie Audio CDs ein einer Datenbank verwalten können.

Sie werden diese Bohne in den folgenden Übungen wieder verwenden.

1.3.1. Voraussetzungen

- Installation und Konfiguration der J2EE Referenz Implementation

1.3.2. Skeleton Code

- `MusicCD.java`
- `MusicCDHome.java`
- `MusicCDBean.java`

1.3.3. Arbeitsschritte

Entwerfen Sie für das `MusicCD` System ein remote Interface sowie ein `MusicCDHome` Interface. Die `MusicCD` Bean muss die folgenden persistenten Datenfelder speichern können:

- `upc`
- `title`
- `artist`
- `type`
- `price`

Das `MusicCD` remote Interface sollte also Zugriffs- und Mutationsmethoden für diese persistenten Datenfelder zur Verfügung stellen. Das `MusicCDHome` Interface sollte eine `findByPrimaryKey`() Methode definieren, welche als Argument eine Zeichenkette enthält, als Platzhalter für `upc` und eine `create`() Methode, welche ein `upc` akzeptiert.

Schreiben Sie die `MusicCDBean` Entity Bean Implementationsklasse. Sie müssen alle Methoden implementieren, die im EntityBean Interface definiert sind.

Sie müssen auch eine Implementation für die Methode `create`() des home Interface implementieren.

Beachten Sie, dass Methoden im remote Interface `remoteExceptions` werfen können.

Fassen Sie Ihre Class Dateien in einem jar zusammen. Die XML Einsatzbeschreibung können Sie aus der Musterlösung übernehmen.

1.3.4. Musterlösung

Die folgenden Dateien enthalten eine vollständige Lösung der Aufgabe:

- `MusicCD.java`
- `MusicCDHome.java`
- `MusicCDBean.java`
- `ejb-jar.xml`

1.3.5. Demonstration

Da es für die Lösung wenig zu zeigen gibt, kann man noch nichts vorführen. Wir werden die Implementation dieser Entity Bean in den späteren Übungsaufgaben testen.

1.3.6. Kreieren einer Entity Bean - Lösungshinweise

Hyperlink zur Dokumentation: API Docs

1.3.6.1. Task 1

Entwerfen Sie für das `MusicCD` System ein remote Interface sowie ein `MusicCDHome` Interface. Die `MusicCD` Bean muss die folgenden persistenten Datenfelder speichern können:

- `upc`
- `title`
- `artist`
- `type`
- `price`

Das `MusicCD` remote Interface sollte also Zugriffs- und Mutationsmethoden für diese persistenten Datenfelder zur Verfügung stellen. Das `MusicCDHome` Interface sollte eine `findByPrimaryKey`() Methode definieren, welche als Argument eine Zeichenkette enthält, als Platzhalter für `upc` und eine `create`() Methode, welche ein `upc` akzeptiert.

Schreiben Sie die `MusicCDBean` Entity Bean Implementationsklasse. Sie müssen alle Methoden implementieren, die im EntityBean Interface definiert sind.

Sie müssen auch eine Implementation für die Methode `create`() des home Interface implementieren.

Beachten Sie, dass Methoden im remote Interface `remoteExceptions` werfen können.

1.3.6.2. Task 2

Schreiben Sie die `MusicCDBean` Entity Bean Implementationsklasse. Sie müssen alle Methoden implementieren, die im EntityBean Interface definiert sind.

Sie müssen auch eine Implementation für die Methode `create`() des home Interface implementieren. Dazu müssen Sie auch noch die Methode `ejbCreate`() implementieren.

`MusicCDHome` muss eine Bean create Method mit der Signature:

```
public MusicCD create(String upc)
               throws CreateException, RemoteException;
```

Aber diese Implementation muss ein `null` Objekt zurück liefern, falls die Persistenz Container-seitig gemanaged wird.

Da die `MusicCDBean` eine Implementation einer Entity Bean ist, besitzt sie einen Primärschlusses und müssen diesen als Parameter an die `create`() und `ejbCreate`() Methode übergeben.

1.3.6.3. Task 3

Übersetzen Sie alle Klassen, aus denen Ihre Entity Bohne zusammengesetzt wird.

Die Klassen aus dem Package javax.ejb befinden sich in `J2EE_HOME\j2ee.jar`.
Hier der Befehl zum Übersetzen der Klassen:

```
javac -classpath %CLASSPATH%;%J2EE_HOME%\lib\j2ee.jar;.
musicstore\*.java
```

1.3.6.4. Task 4

Der Einfachheit halber packen wir alle Klassendateien in ein Jar-Archiv, inklusive der XML Beschreibung ('deployment description').

Das Jar Programm wird mit Java ausgeliefert. Und so wird das Archiv hergestellt:

```
jar cvf MusicCD.jar musicstore\*.class ejb-jar.xml
```

Sie müssen eventuell den Pfad (musicstore\) anpassen, je nachdem wie das Musterlösungsarchiv entpackt oder Ihre Dateien angeordnet haben.

1.4. Kreieren einer zustandslosen Session Bean

Hyperlink zur API Dokumentation: API Docs

In dieser Übung implementieren Sie die Lager Session Bean, mit der die üblichen Lagerverwaltungsoperationen durchgeführt werden.

Diese Methoden nutzen die `MusicCD` Bean und werden vom `MusicInventoryClient` initialisiert.

1.4.1. Voraussetzungen

Installation der Datenbank und des EJB Servers
Kreieren einer Entity Bean

1.4.2. Skeleton Code

Inventory.java
InventoryHome.java
InventoryBean.java
MusicData.java

1.4.3. Tasks

Entwerfen Sie eine Wrapperklasse `MusicData`, welche die persistenten Datenfelder in der Bohne `MusicCD` speichert (`upc, title, artist, type, price`).

Entwerfen Sie eine Inventory Session Bean, welche einen `addInventory`() Dienst zur Verfügung stellt. Als Argument wird ein Array von `MusicDataObject`() verwendet.

Die `addInventory`() Methode stellt eine Referenz auf `MusicCDHome` her. Anschliessend werden in einer Schleife alle Daten (Objekte) als Argumente der `addInventory`() Methode zum Kreieren von `MusicCD` Beans mit entsprechenden Attributen verwendet.

Übersetzen Sie alle Klassen Ihrer Bean.

Packen Sie alle Klassen in ein Jar Archiv, wobei Sie auch die XML Einsatzbeschreibung dazulegen müssen. Die XML Beschreibung finden Sie bei der Musterlösung.

1.4.4. Musterlösung

Sie finden die Musterlösung für die Aufgabe in folgenden Dateien. Die Musterlösung enthält eine vollständig implementierte Inventory Session Bean:

- Inventory.java
- InventoryHome.java
- InventoryBean.java
- MusicData.java
- ejb-jar.xml

1.4.5. Demonstration

Auch für diese Teilaufgabe steht kein lauffähiges Demoprogramm zur Verfügung. Wir müssen erst noch einige weitere Aufgaben lösen.

1.4.6. Kreieren eine zustandslosen Session Bean - Lösungshinweise

Hyperlink auf die API Dokumentation: API Docs

1.4.6.1. Task 1

Entwerfen Sie eine Wrapperklasse `MusicData`, welche die persistenten Datenfelder in der Bohne `MusicCD` speichert (`upc, title, artist, type, price`).

Vergessen Sie nicht *`java.io.Serializable`* zu implementieren.

`MusicData` sollte einen Konstruktor zur Verfügung stellen, mit dem die Datenfelder der Instanzen gesetzt werden können, die `MusicCD` Attribute:

```
...
public MusicData(String upc, String title,
                 String artist, String type,
                                float price) {
  this.upc    = upc;
  this.title  = title;
  this.artist = artist;
  this.type   = type;
  this.price  = price;
}
...
```

1.4.6.2. Task 2

Entwerfen Sie eine Inventory Session Bean, welche einen `addInventory`() Dienst zur Verfügung stellt. Als Argument wird ein Array von `MusicDataObject`() verwendet.

Die `addInventory`() Methode stellt eine Referenz auf `MusicCDHome` her. Anschliessend werden in einer Schleife alle Daten (Objekte) als Argumente der `addInventory`() Methode zum Kreieren von `MusicCD` Beans mit entsprechenden Attributen verwendet.

`InventoryHome` sollte eine einfache Methode zum Kreieren der Bohne zur Verfügung stellen, welche folgende Signatur aufweist:

```
public Inventory create() throws CreateException, RemoteException;
```

`Inventory` sollte nur eine Methode beschreiben, `addInventory()`, mit einer Signatur, die in etwa wiefolgt aussieht (die Details hängen von Ihrem Design ab):

```
public boolean addInventory(MusicData[] data)
               throws RemoteException;
```

Weil `Inventory` eine Session Bean ist, besitzt sie keinen Primärschlüssel und die Methoden zum Kreieren besitzen keinerlei Argumente: `create()` und `ejbCreate()`.

`InventoryBean.addInventory()` ist ähnlich wie beim `MusicClient`. Auch in diesem Fall muss man eine Verbindung zu einem JNDI Context herstellen, die `MusicCD` Bean nachschlagen, und so weiter.

1.4.6.3. Task 3

Übersetzen Sie alle Klassendateien.

Auch in diesem Fall müssen Sie darauf achten, dass die javax.ejb Klassen, welche sich in `J2EE_HOME/j2ee.jar`. Und so sieht der Aufruf aus (passen Sie das musicstore\ Ihrer Umgebung an):

```
javac -classpath %CLASSPATH%;%J2EE_HOME%
         \lib\j2ee.jar;. musicstore\*.java
```

1.4.6.4. Task 4

Packen Sie Ihre Session Bean in ein jar Archiv, wobei Sie die XML Beschreibung dazupacken müssen.

Die folgende Kommandozeile zeigt, wie dies zu geschehen hat;

```
jar -cvf Inventory.jar musicstore\*.class META-INF\ejb-jar.xml
```

1.5. Nutzung / Deployment der Enterprise Beans in Sun's J2EE Reference Implementation

Hyperlink auf die API Dokumentation: API Docs

Nun wollen wir die `MusicCD` Entity Bean zusammen mit der Inventar (`Inventory`) Session Bean konkret einsetzen. Dazu benutzen wir die J2EE RI als Applikationsserver, als Container. Die Vorbereitung der Nutzung einer Bohne ist in der Regel ein Projekt in sich. Sie ist *immer* Applikationsserver spezifisch. Falls Sie einen der andern üblichen Applikationsserver (WebLogic, WebSphere, Oracle) einsetzen möchten, müssen Sie unbedingt die Applikationsdokumention dieser Server lesen und verstehen. Die Bean selbst ist portabel, aber wie sie in den Container eingebaut werden muss, ist Server spezifisch.

Damit Sie die Übung durchführen können, benötigen Sie die zwei Beans:

- die Entity Bean `MusicCD`
- die Session Bean `Inventory`

1.5.1. Voraussetzungen

Installation und Konfiguration der J2EE Reference Implementation
Kreieren einer Entity Bean
Kreieren eine Stateless Session Bean
falls Sie keine Standarddatenbank (mit JDBC) verwenden: Aufsetzen der Datenbank

1.5.2. Skeleton Code

Inventory.jar
MusicCD.jar

1.5.3. Arbeitschritte

Kopieren Sie die oben erwähnten zwei jar Dateien in Ihrem Arbeitsverzeichnis. Die Nächsten Schritte geschehen in Ihrem Arbeitsverzeichnis.

Starten Sie den Datenbankserver, falls nötig (siehe Zusatzabschnitte weiter unten, falls Sie Cloudscape einsetzen möchten, oder falls Sie Fragen zum Aufsetzen der Datenbank haben); starten Sie die J2EE RI oder eine Alternative (WebSphere, WebLogic, Gemstone, Oracle), den Applikationsserver.

Im Folgenden gehe ich davon aus, dass Sie mit Cloudscape und der J2EE RI als Applikationsserver arbeiten.

Starten Sie das `deploytool` Hilfsprogramm. Wahrscheinlich wird Ihr PC etwas Mühe haben, all dies zum Laufen zu bringen, da die Empfehlung ist, mindestens 128 MB Speicher zu verwenden.

Im deploytool müssen Sie "File/New Application" anwählen, um eine neue Applikation "MusicStore" zu kreieren. (Diese Applikation ist als "MusicStore.ear" abgespeichert und der Musterlösung beigefügt, im Archiv.)

Selektieren Sie die 'MusicStore' Applikation aus der Liste der einsetzbaren Anwendungen (linke Seite des `deploytool` GUI).

Selektieren Sie "File/Add EJB JAR for Application..." und verwenden Sie die Dialogboxe um das Archiv `'MusicCD.jar'` zu öffnen. Sie erhalten dieses Archiv als Skeleton oder natürlih vollständig in der Musterlösung.

Jetzt müssen Sie auch noch das Inventar Archiv einbinden. Gehen Sie analog zu oben vor, wobei Sie im Wesentlichen `'MusicCD.jar'` durch `'Inventory.jar'` ersetzen.

Wählen Sie die 'MusicStore' Applikation an und wechseln Sie in den "JNDI" Bildschirmteil des `deploytool` Fensters. Erfassen Sie die JNDI Namen für alle Komponenten, gemäss folgender Tabelle:

```
MusicCDBean       MusicCDBean       ejb/MusicCD
InventoryBean     InventoryBean     ejb/Inventory
InventoryBean      ejb/MusicCD      ejb/MusicCD
```

Nun müssen Sie die 'MusicCD' Entity Bean mit der Datenbank verknüpfen. Dazu wählen Sie die 'MusicCDBean' Komponente im linken Fenster.

Dann selektieren Sie 'Entity' im Hauptfenster und drücken auf den 'Deployment Settings' Knopf. Damit gelangen Sie zum Input Dialog. Der Datenbank JNDI Name ist 'jdbc/MusicStore' .

Deselektieren Sie die Boxen 'Create table on deploy' und 'Delete table on deploy' und drücken Sie dann auf den Knopf "Generate SQL now".

Die Datenbank für die CDs, die Sie entweder bereits aufgesetzt haben oder jetzt dringenst aufsetzen müssten (siehe die Übung und die Anleitung dazu), besitzt unter anderem eine Tabelle "MusicCD" in der die MusicCDBeans abgespeichert werden, also in unserem Fall die Datensätze, welche die CDs beschreiben.

Da das vom Enterprise Applikationsserver generierte SQL noch nicht die korrekten Angaben enthält, müssen Sie Ihre Lifecycle Methoden auf die DB und die Tabelle anstimmen.

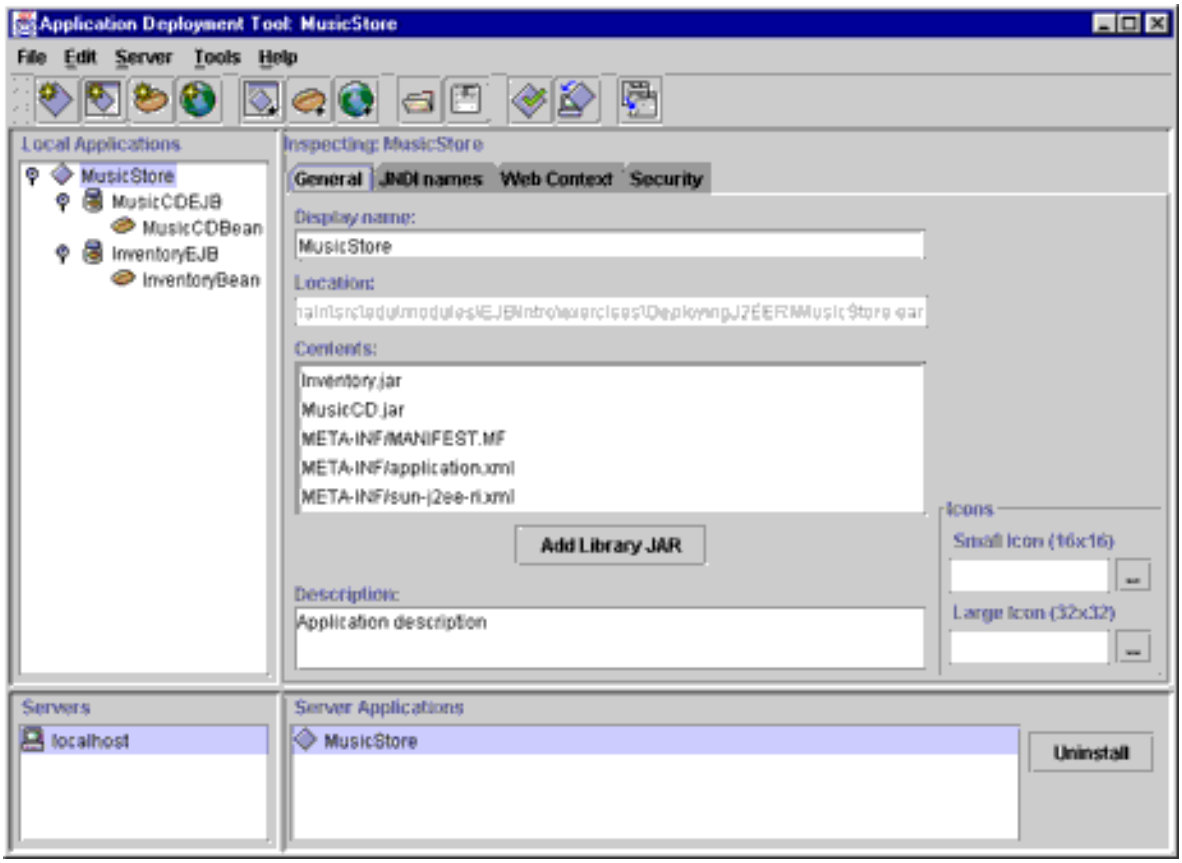
Im `deploytool` selektieren Sie nun "Tools/Deploy/Application..." aus dem Menü. Wählen Sie die Box "Return Client Jar"; den Rest können Sie einfach akzeptieren. Steppen Sie durch den Wizzard, wobei Sie für den Rest jeweils immer die Standardwerte und Standardeinstellungen verwenden. Schliesslich sollten Sie zum "Finish" Button gelangen

Nun sollten Sie in der unteren Hälfte des `deploytool` die MusicStore Applikation sehen. Diese sollte im Panel "Server Applications" erscheinen.

Falls Sie soweit gelangen, haben Sie die Bohnen erfolgreich installiert.

1.5.4. Musterlösung

Falls Sie die MusicStore Application erfolgreich im Server aktiviert haben, sollte der Bildschirm des deploytool etwa folgendermassen aussehen:



1.5.5. Demonstration

Sie können im Begleittext Musterbildschirme sehen. Damit können Sie allfällige offene Fragen vermutlich klären.

1.5.6. Einsatz der Enterprise Beans in Sun's J2EE Reference Implementation - Lösungshinweise

Hyperlink auf die API Dokumentation: API Docs

Im folgenden finden Sie zu jedem Aufgaben-Arbeitsschritt Hinweise und Bildschirm Abbildungen.

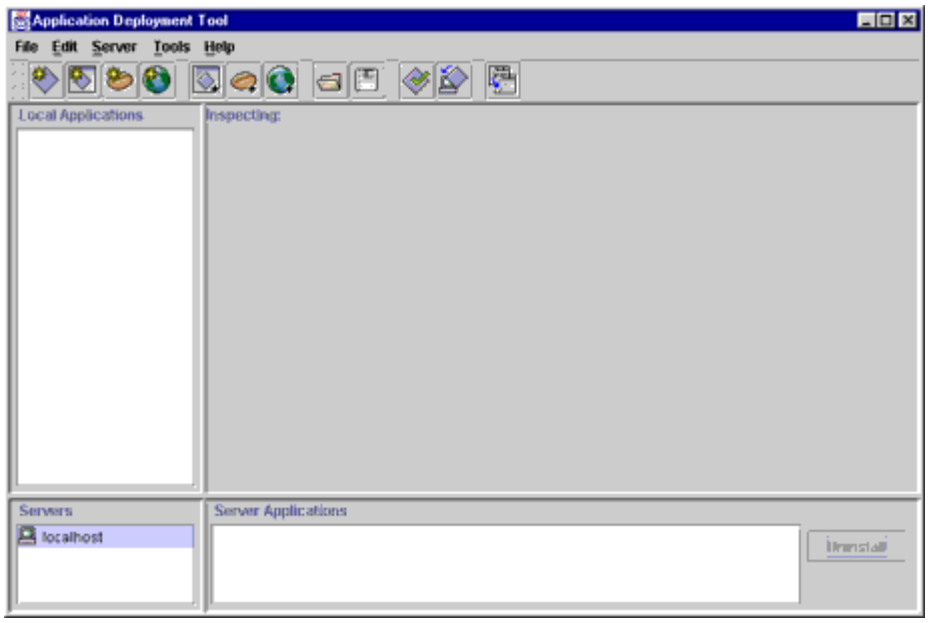
1.5.6.1. Task 1

Am Besten legen Sie sich ein Arbeitsverzeichnis an und kopieren die Skeletons, die Lösungsframworks für die Aufgabe in dieses Verzeichnis.

1.5.6.2. Task 2

Starten Sie das `deploytool` Hilfsprogramm. Dieses greift auf den J2EE RI Applikationsserver zu und erlaubt es Ihnen unter anderem Enterprise Beans zu verwalten.

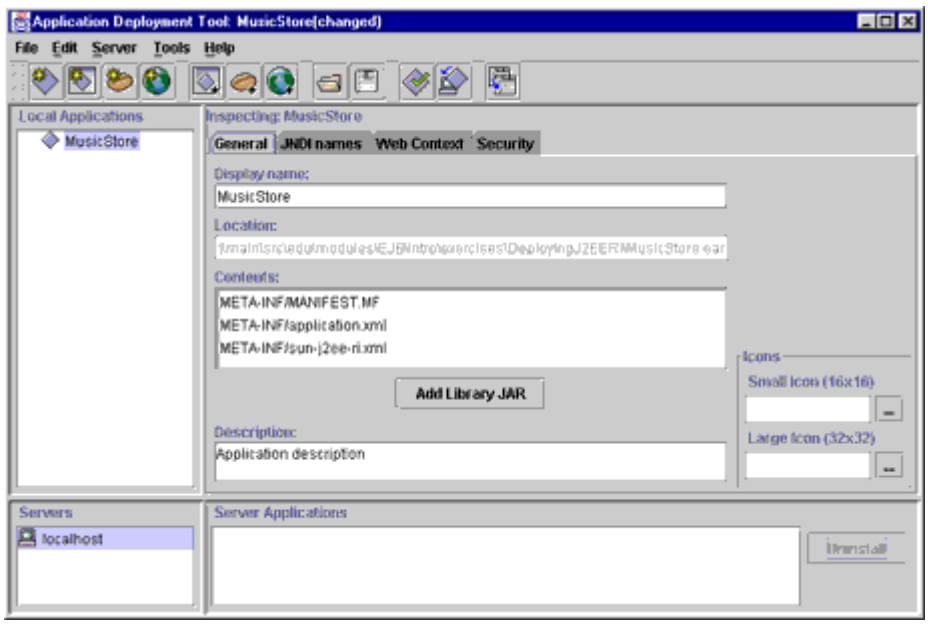
Bevor Sie das Hilfsprogramm starten, sollte Ihr J2EE RI Server und der Datenbankserver laufen: `cloudscape -start` und `j2ee -verbose`.



1.5.6.3. Task 3

Im `deploytool` müssen Sie "File/New Application" anwählen, um eine Applikation "MusicStore" zu kreieren. Diese wird als "MusicStore.ear" Datei abgespeichert.

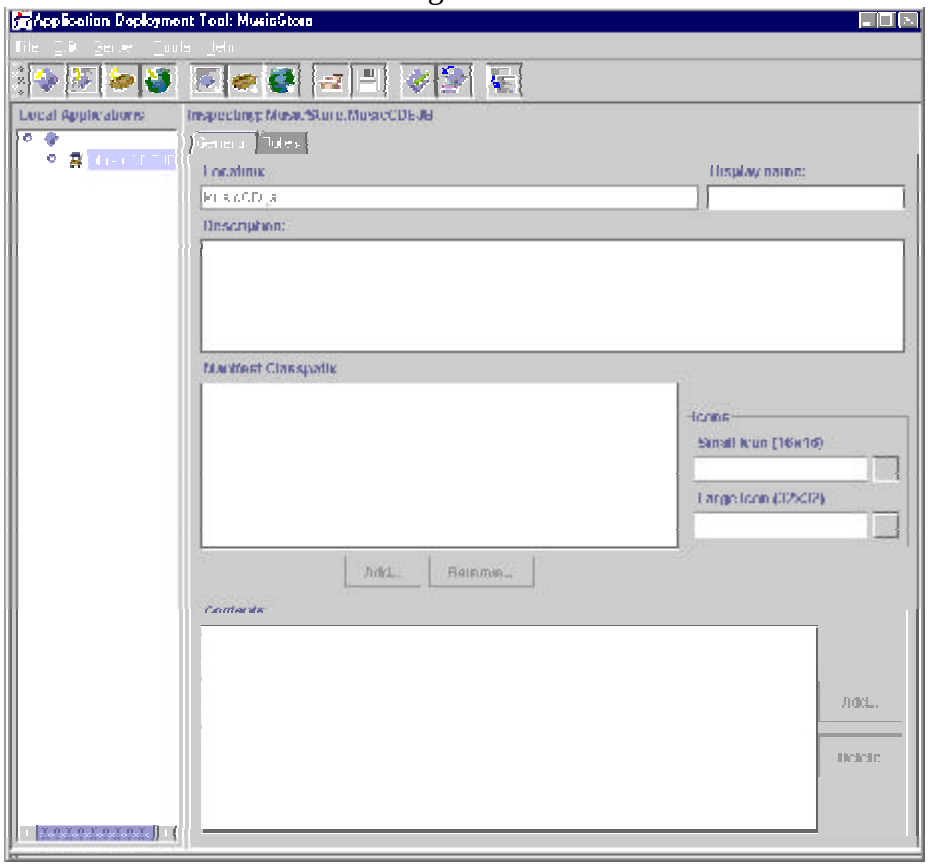
Auf der linken Seite des Bildschirmssollten Sie nun in etwa folgenden Bildschirm sehen.



1.5.6.4. Task 4

Wählen Sie die MusicStore Applikation aus der Liste auf der linken Seite des Bildschirms. Wählen Sie nun "File/add EJB JARto Application ..." und öffnen Sie mit Hilfe der Dialogboxe das `MusicCD.jar`, in dem sich die Skeletons der Applikation befinden.

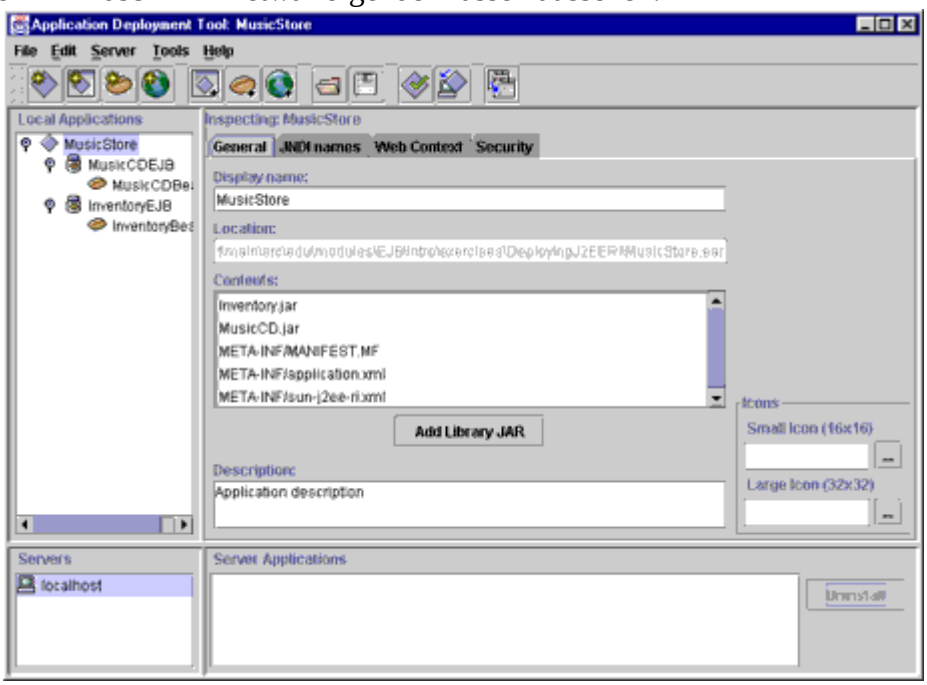
Nun sollte Ihr Bildschirm in etwa folgendermassen aussehen:



1.5.6.5. Task 5

Nun verfahren Sie mit dem Archiv `Inventory.jar` genau wie mit dem Musik CD Archiv.

Danach sollte Ihr Bildschirm in etwa folgendermassen aussehen:

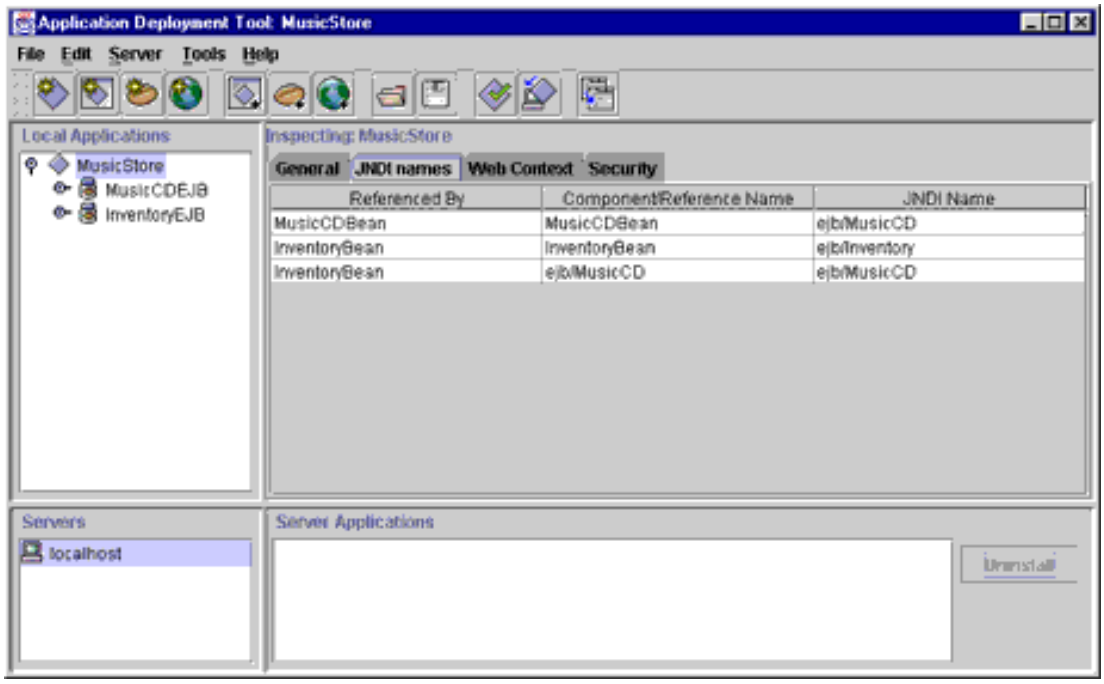


1.5.6.6. Task 6

Innerhalb der MusicStore Anwendung selektieren Sie nun das "JNDI" Panel und ergänzen Sie die Einträge gemäss der folgenden Liste:

MusicCDBean	MusicCDBean	ejb/MusicCD
InventoryBean	InventoryBean	ejb/Inventory
InventoryBean	ejb/MusicCD	ejb/MusicCD

Jetzt sollte Ihr Panel folgendermassen aussehen:



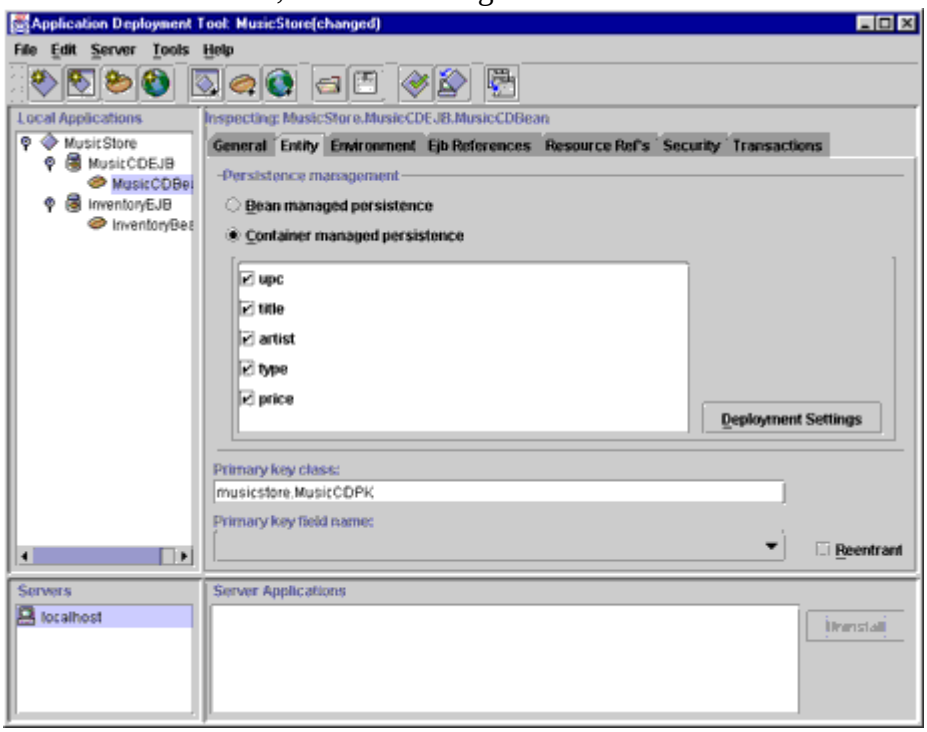
1.5.6.7. Task 7

Nun müssen Sie eine Verbindung herstellen zwischen der Datenbank und der MusicCD Entity Bean. Dazu wählen Sie die "MusicCDBean" Komponente im linken Fenster aus.

Als nächstes wählen Sie die "Entity" Tab im Hauptfenster und drücken Sie auf den "Deployment Settings" Knopf. Damit gelangen Sie zu einem Eingabefenster.

Setzen Sie die Datenbank auf "jdbc/MusicStore" in diesem Dialog. Deselektieren Sie zudem die Boxen : "Create table on deploy" und "Delete table on deploy".
Jetzt können Sie SQL generieren lassen.

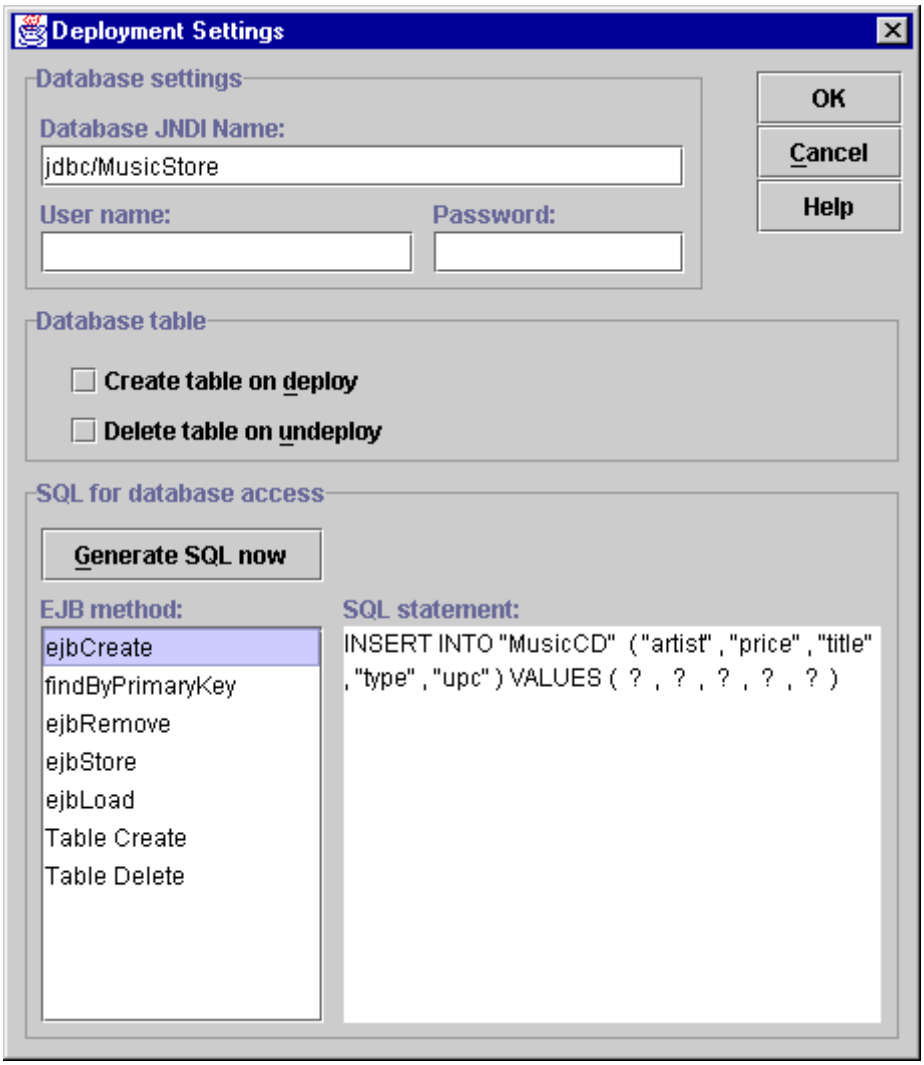
Sie sollten einen Bildschirm sehen, der in etwa folgendermassen aussieht:



Der JNDI Name der Datenbank ist `jdbc/MusicStore` . Sie können in der Übung zum Aufsetzen der Datenbank nachsehen, wie die Datenbank konkret aufgesetzt werden muss.

Falls das Hilfsprogramm sich beschwert, es kenne den Namen der Datenbank nicht, haben Sie offensichtlich die Datenbank falsch aufgesetzt, oder der Server läuft nicht korrekt.

Nach all diesen Arbeiten sollte Ihr Bildschirm folgender Abbildung ähnlich sein:



1.5.6.8. Task 8

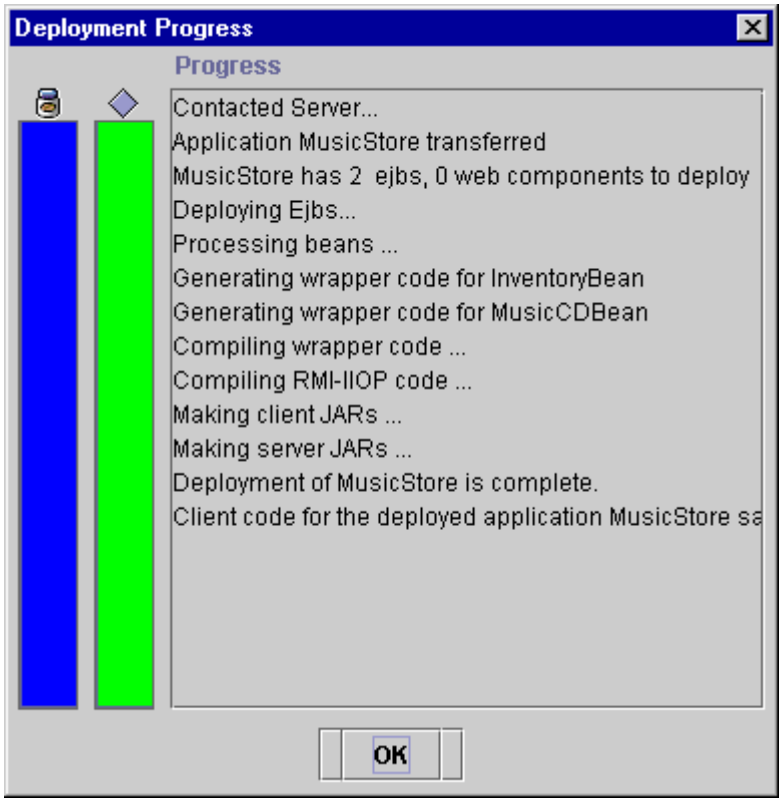
Die MusicStoreDB, welche Sie aufgesetzt haben (siehe separate Übung) verwendet eine Tabelle "MusicCD", in der die MusicCD Beans abgespeichert werden. Die SQL Kommandos, die der J2EE RI Server generiert sind nicht automatisch mit der richtigen Tabelle verbunden.

Es obliegt Ihnen, diese Anpassungen zu machen. Dazu müssen Sie in jede Lifecycle Methode eingreifen und das generierte SQL Skript anpassen.

Auf dem Bildschirm zum vorigen Arbeitsschritt sehen Sie, wie das SQL Skript aussieht.

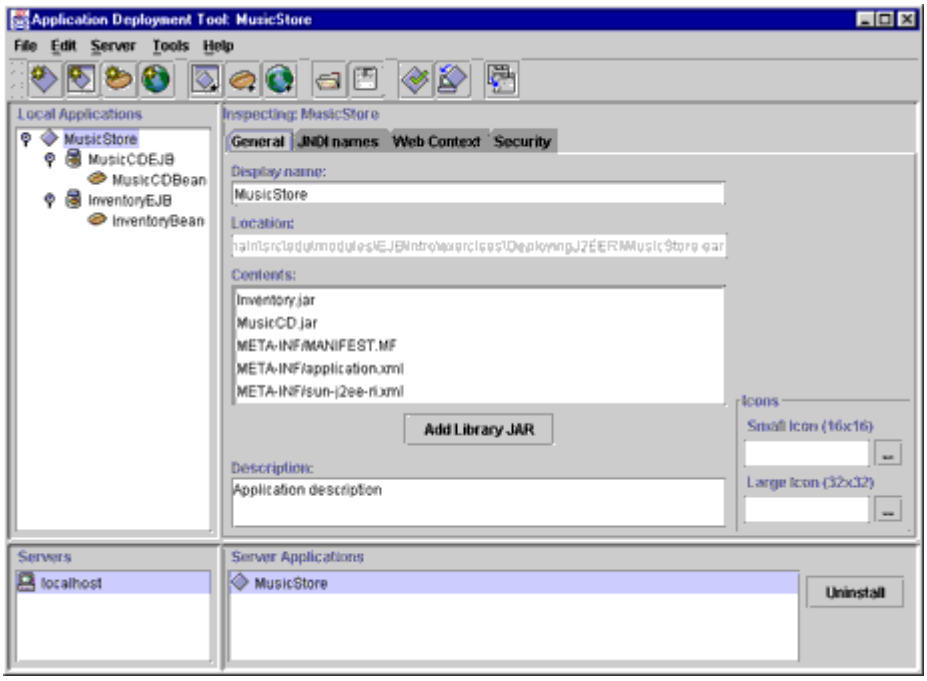
1.5.6.9. Task 9

Nun müssen wir den Client einbinden. Das Kreieren eines Clients wird in einer andern Übungsaufgabe besprochen.



1.5.6.10. Task 10

Sie sollten Sie gerade installierte Applikation im uneren Teil des Bildschirms sehen. Fals dies der Fall ist, haben Sie Ihre Bohnen korrekt installiert.



1.6. Aufsetzen der Datenbank

Hyperlink zur API Dokumentation: API Docs

In dieser Übung setzen Sie eine Tabelle in der Datenbank auf, in der die Informationen über die Audio-CDs abgespeichert werden. Der Einfachheit halber steht Ihnen eine Cloudscape Datenbank als Lösung zur Verfügung, im JAR Format.

Wie oben stehen auch "Lösungsframeworks" (Skeletons) zur Verfügung. Ein Programm kann auch eingesetzt werden, um für andere Datenbanken die Daten zu erfassen, sofern die Datenbank über einen JDBC Treiber verfügt. Das Hilfsprogramm besitztauch eine einfache Hilfe, "-usage" sowie "-help" oder "-h" Optionen.

1.6.1. Voraussetzungen

Installation und Konfiguration der J2EE RI

1.6.2. Resourcen

- DatabaseTool.java
- MusicStoreDB.jar
- MusicCDCreateTables.java
- MusicCDInsertRecords.java

1.6.3. Tasks

WählenSie ein Datenbanksystem aus welches auf Ihrer Arbeitsumgebung funktioniert.

Installieren Sie entweder die Clouscale Datenbank aus dem Archiv oder laden Sie die Daten mit dem Hilfsprogramm, sofern Sie eine JDBC fähige Datenbank gewählt haben.

Als Daten können Sie beliebige Audio-CDs auswählen. Die Demodatenbank besteht aus einigen Titeln aus einem Online Store, einem der grössten Bücher, Video und Audio Supermarktketten in den USA.

Verifizieren Sie die Datenbank beispielsweise mit einem Werkzeug des Datenbankherstellers oder mit Hilfe des Hilfsprogramms `DatabaseTool.java`.

1.6.4. Musterlösung

Zu dieser Teilaufgabe gibt es keine Musterlösung, da zuviele unterschiedliche mögliche Lösungen existieren.

```
c:\> cloudscape -start
Wed Jan 12 11:51:20 PST 2000:
     [RmiJdbc] COM.cloudscape.core.JDBCDriver
                    registered in DriverManager
Wed Jan 12 11:51:20 PST 2000:
     [RmiJdbc] Binding RmiJdbcServer...
Wed Jan 12 11:51:20 PST 2000:
     [RmiJdbc] No installation of
                       RMI Security Manager...
Wed Jan 12 11:51:22 PST 2000:
     [RmiJdbc] RmiJdbcServer bound
                          in rmi registry
```

Und beim Start der J2EE könnte es etwa folgendermassen aussehen:

```
myhost> j2ee -verbose

Naming service started: :1050
Published the configuration object ...
Binding DataSource, name = jdbc/Cloudscape,
    url =
     jdbc:cloudscape:rmi:CloudscapeDB;create=true
Binding DataSource, name = jdbc/MusicStore,
    url =
     jdbc:cloudscape:rmi:MusicStoreDB;create=false
Configuring web service using "default"
Web service started: :9191
Web service started: :7000
Configuring web service using "default"
Configuring web service using
  "file:/c:/j2sdkee1.2.2/public_html/WEB-INF/web.xml"
Web service started: :8000
Endpoint [SSL:
  ServerSocket[addr=0.0.0.0/0.0.0.0,port=0,
                              localport=7000]]
    shutdown due to exception:
    javax.net.ssl.SSLException:
    No available certificate corresponds
        to the SSL cipher suites
        which are enabled.
endpoint down: :7000
J2EE server startup complete.
```

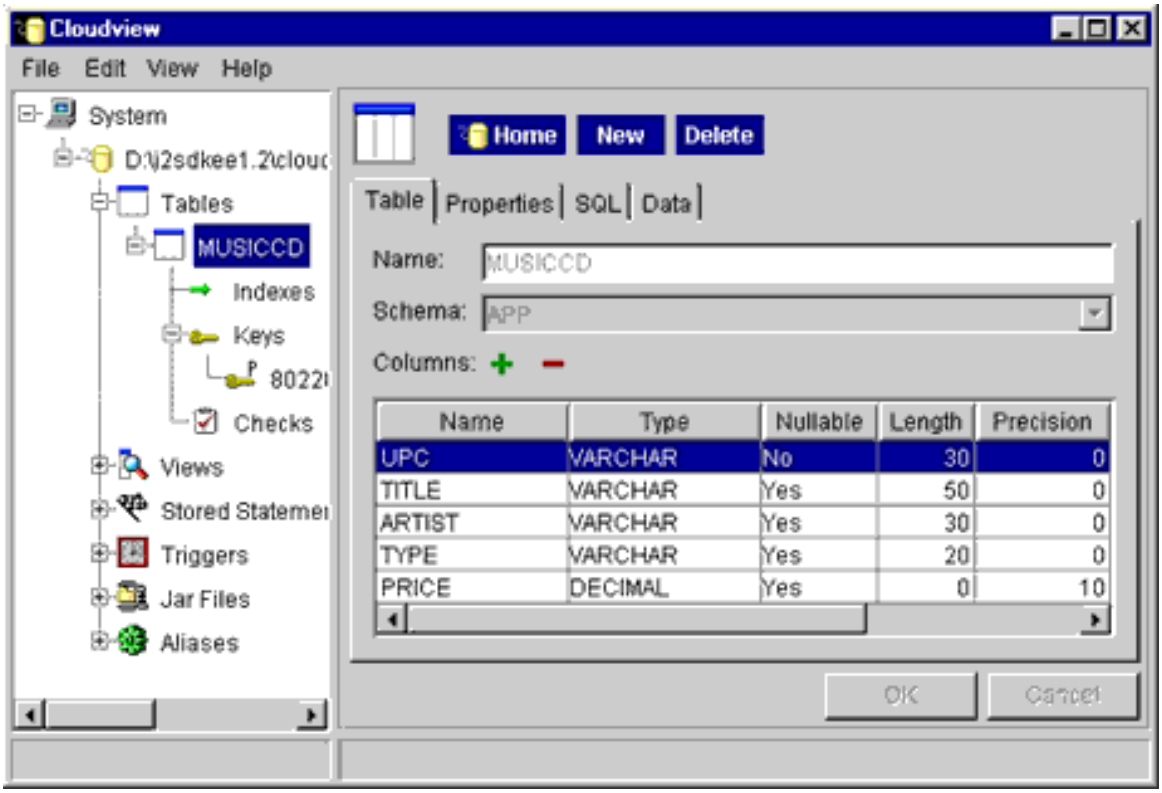
1.6.5. Demonstration

Das Systemverhalten ist Datenbank abhängig. Die meisten Datenbanken verfügen über grafische Hilfsmittel, wie Tabellen Browser, mit denen Sie die Integrität der aufgesetzten Lösung kontrollieren können.

Im Falle von Cloudscape nennt sich dieses Tool Cloudview.
Cloudview starten Sie mit folgendem Kommando:

```
c:\> java
    -Dcloudscape.system.home=%J2EE_HOME%
       \cloudscape \-classpath %J2EE_HOME%
            \lib\cloudscape\tools.jar;
  %J2EE_HOME%\lib\cloudscape\cloudscape.jar;
  %J2EE_HOME%\lib\cloudscape
              \license.jar;%CLASSPATH% \
-ms16m -mx32m COM.cloudscape.tools.cview
```

Die MusicCD Tabelle sieht in Cloudview folgendermassen aus:

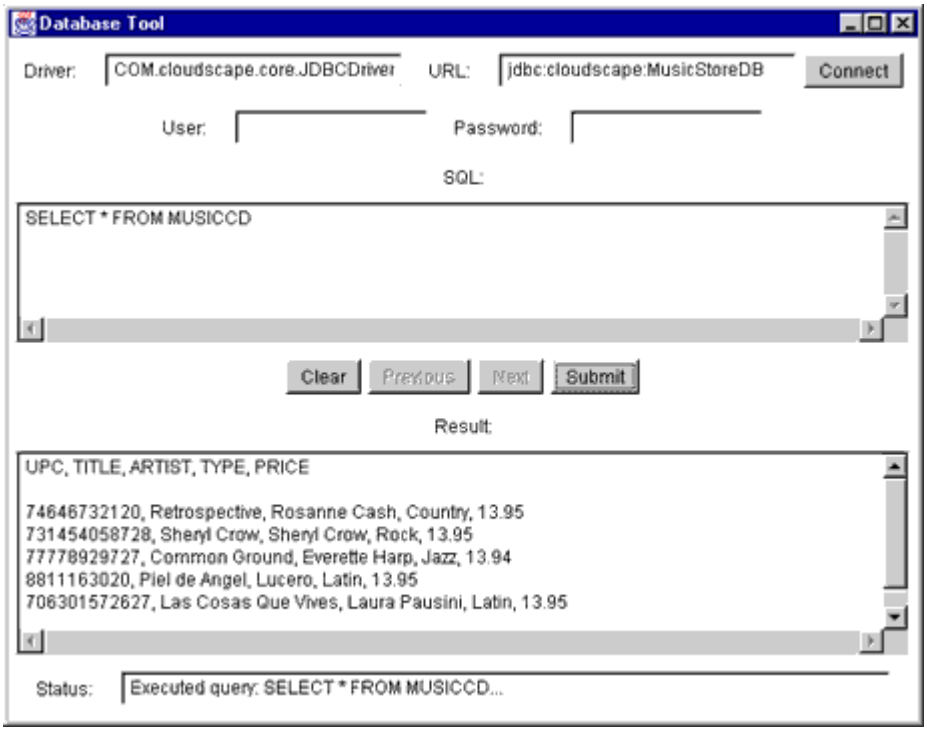


Falls Sie mit einer Datenbank ohne spezielle Grafikwerkzeuge arbeiten, zum Beispiel Textdateien, dann können Sie das Datenbank-Werkzeug `DatabaseTool.java` aus der Musterlösung übersetzen und einsetzen, sofern Sie einen JDBC Treiber für das Datenbanksystem haben.:

```
c:\> java
       -Dcloudscape.system.home=%J2EE_HOME%
                                \cloudscape\
-classpath %J2EE_HOME%\lib
                    \cloudscape\tools.jar;
  %J2EE_HOME%\lib\cloudscape\cloudscape.jar;
  %J2EE_HOME%\lib\cloudscape\license.jar;
                               %CLASSPATH%\
-ms16m -mx32m DatabaseTool
```

Die obige Anweisung steht auf einer einzigen Zeile.

Falls Sie das DatabaseTool einsetzen, sollte Ihre MusicCD Tabelle etwa folgendermassen aussehen:



In vielen Fällen ist das `DatabaseTool` natürlich schneller als die Werkzeuge des Datenbankherstellers, da diese viel universeller sind.

1.6.6. Aufsetzen der Datenbank - Lösungshinweise

Hyperlink zur API Dokumentation: API Docs

1.6.6.1. Task 1

Sie können ein Datenbanksystem Ihrer Wahl verwenden, sofern Sie einen JDBC Treiber dafür finden.

Sun's J2EE RI wird mit einer Laufzeitversion der Cloudscape Datenbank ausgeliefert. Sie können diese Datenbank, die inzwischen zu Informix gehört, auch als Testversion herunterladen. Eine Kopie sollte auch auf dem Server sein.

Der Einsatz von Cloudscape erleichtert Ihnen die Installation. Die Datenbank ist vielleicht nicht sehr schnell; aber Sie sollten kaum Probleme damit haben.

Zur Referenzinstallation von J2EE RI finden Sie bei Sun fast beliebig viel Informationen:

- Configuration Guide
- Release Notes

Microsoft Access verfügt nicht über den nötigen EJB Support!

Falls Sie Cloudscape einsetzen können Sie eine Referenzdatenbank für diese Übung aus dem `MusicStoreDB.jar` in das Verzeichnis `%J2EE_HOME%\cloudscape` kopieren. Speichern Sie das Archiv in diesem Verzeichnis und verwenden Sie jar oder WinZip und das Archiv zu entpacken.

Falls Sie andere Datenbanksysteme einsetzen können Sie mit den Hilfsprogrammen `MusicCreateTable.java` und `MusicCDInsertRecords.java` die nötigen Tabellen.

Sie können auch Hilfsprogramme von Sun einsetzen:

- `DatabaseTool`,
- oder `JDBCTest` (`TestTool`)

von Sun's JDBC Site.

Am Schluss müssen Sie die Property Datei im J2EEHome Verzeichnis anpassen:

```
%J2EE_HOME%\config\default.properties
```

um die Datenquelle anzugeben. Hier die Änderungen, die Sie durchführen müssen:

```
jdbc.datasources=jdbc/Cloudscape|
  jdbc:cloudscape:rmi:CloudscapeDB;create=true

(alles auf einer Zeile, auch die folgende Anweisung)
jdbc.datasources=jdbc/Cloudscape|
  jdbc:cloudscape:rmi:CloudscapeDB;create=true|
  jdbc/MusicStore|jdbc:cloudscape:rmi:MusicStoreDB;create=false
```

Der obige Schritt ist in der Regel nicht nötig, wenn Sie die Cloudscape Datenbank Trialversion installieren, also die Laufzeitversion überschreiben.

Nun sollten Sie Clouscape und J2EE RI neu starten.

Sie haben sicher keine Probleme CDs anzugeben. Falls doch: bei Barnes and Noble und andere Grossanbieter im Internet (CDNow) können Sie alle relevanten Daten sehen (UPC, Titel, Interpret, Musikrichtung usw.).

1.6.6.2. Task 2

Verifizieren Sie die Integrität der Datenbank mit Hilfe eines Hilfsprogramms des Datenbankanbieters, oder verwenden Sie das `DatabaseTool.java` Hilfsprogramm.

1.7. Kreieren eines EJB Clients

Hyperlink zur API Dokumentation: API Docs

In dieser Übung implementieren Sie zwei Clients : standalone Anwendungen, die Zugriff auf die `MusicCD` Entity Bean und die `Inventory` Session Bean, die Sie inden vorhergehenden Übungen entwickelt und 'deployed', also im EJB Server installiert und aktiviert haben.

1.7.1. Voraussetzungen

- Installation und Konfiguration der J2EE Referenzimplementation
- Kreieren einer Entitäten Bohne
- Kreieren einer Session Bohne
- Installieren und Aktivieren / Nutzen der EJBs in der J2EE RI
- Aufsetzen einer Datenbank für EJBs

1.7.2. Skeleton Code

Sie finden auf dem Server Lösungsframeworks für folgende Programme:

- MusicClient.java
- MusicInventoryClient.java
- Inventory.jar
- MusicCD.jar

1.7.3. Aufgaben

Starten Sie den Cloudscape Datenbankserver:

```
cloudscape -start.
```

Starten Sie die J2EE RI :

```
j2ee -verbose.
```

Stellen Sie fest, im Log auf dem Bildschirm, ob die MusicStoreDB geladen wird. Falls nicht, dann sollten Sie die Übung zum Aufsetzen der Datenbank nochmals genau anschauen, speziell die Lösungshinweise.

Passen Sie das `MusicClient.java` Skeleton an, damit sie auf den JNDI Context zugreifen können. Benutzen Sie diesen Context, um eine Referenz auf MusicCDHome zu erhalten.

Sie können eine neue `MusicCD` Bean zum UPC Code im Skeleton und passenden Datenfeldern gemäss Skeleton eingeben. Aber Sie können auch beliebige eigene CDs erfassen.

Ergänzen Sie das Lösungsframework, indem Sie eine Suchmethode implementieren, zum Suchen der CDs.

Übersetzen und starten Sie das Programm.

Editieren Sie jetzt das `MusicInventoryClient.java` Skeleton, um den JNDI Context zu erhalten und mit diesem eine Referenz auf InventoryHome herzustellen.

Kreieren Sie eine neue Inventar Bean.

Kreieren Sie eine Instanz von MusicData und geben Sie die CD Daten ein. Benutzen Sie dann die Inventar Bean, um mit deren Business Methoden die Aufgabe (Speicherung der Daten in der Datenbank) abzuschliessen.

Übersetzen und starten Sie die Applikation.

Falls Sie Hilfe benötigen, lesen Sie die Lösungshinweise nach dieser Aufgabe.

1.7.4. Musterlösung

Die Musterlösung zu dieser Aufgabe finden Sie im Archiv `MusicStoreClient.jar` . Sie finden darin die Container-generierten Stubs und Skeletons und all die andern Dateien, die Sie für den J2EE™ Server benötigen.

- MusicClient.java
- MusicInventoryClient.java
- MusicStoreClient.jar

1.7.5. Demonstration

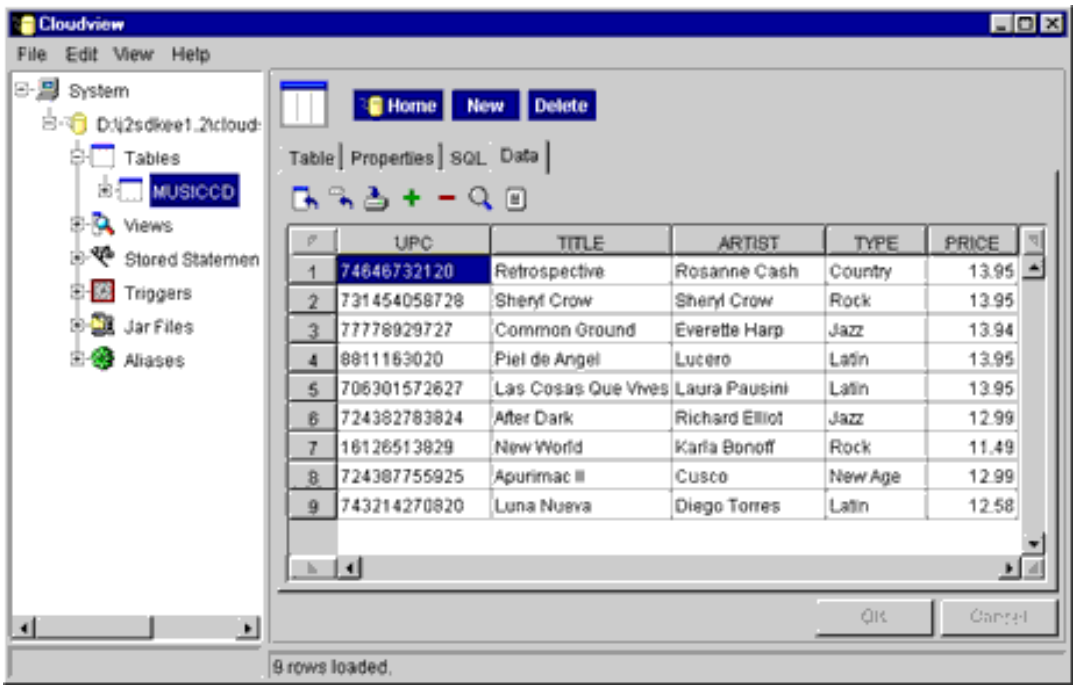
`MusicClient` produziert eine Ausgabe in der Standardausgabe:

```
c:\> java -classpath %CLASSPATH%;
       MusicStoreClient.jar; \%J2EE_HOME%
         \lib\j2ee.jar  musicstore.MusicClient

Titel: Luna Nueva
Artist: Diego Torres
Type: Latin
Preis: 12.58

c:\>
```

`MusicInventoryClient` sollte eine Ausgabe produzieren, die gleich oder ähnlich wie unten aussieht:

```
c:\> java -classpath %CLASSPATH%;MusicStoreClient.jar; \
        %J2EE_HOME%\lib\j2ee.jar  musicstore.MusicInventoryClient

Aufbau einer neuen Inventarliste ...
Eingabe der neuen Inventarliste  ...
Die neue Inventarliste wurde in die Datenbank eingefügt.
c:\>
```

Bei einer Abfrage Ihrer Datenbank sehen Sie natürlich die CDs, die Sie eingefügt haben:



1.7.6. Kreieren eines EJB Clients - Lösungshinweise

Hyperlink zur API Dokumentation: API Docs

1.7.6.1. Task 1

Starten Sie den Cloudscape Datenbank Server mit folgendem Befehl:
`cloudscape -start`.

Falls Cloudscape schon aus einer vorherigen Übung oder auch warum auch immer bereits läuft, brauchen Sie den Server natürlich nicht neu zu starten. Eigentlich sollte es nicht möglich sein, zwei Versionen des DB-Servers gleichzeitig auf der selben Maschine zu starten. Beim zweiten Starten sehen Sie etwas in folgendem Stil

```
C:\j2sdkee1.2.1\bin>cloudscape -start
Sat Jul 15 16:36:37 GMT+02:00 2000: [RmiJdbc] COM.cloudscape.core.JDBCDriver
registered in  DriverManager
Sat Jul 15 16:36:37 GMT+02:00 2000: [RmiJdbc] Binding RmiJdbcServer...
Sat Jul 15 16:36:38 GMT+02:00 2000: [RmiJdbc] No installation of RMI Security Manager...
Got Exception: Port already in use: 1099; nested exception is:
        java.net.BindException: Address in use: JVM_Bind
java.rmi.server.ExportException: Port already in use: 1099; nested exception is:
        java.net.BindException: Address in use: JVM_Bind
java.net.BindException: Address in use: JVM_Bind
        at java.net.PlainSocketImpl.socketBind(Native Method)
        at java.net.PlainSocketImpl.bind(PlainSocketImpl.java:397)
        at java.net.ServerSocket.<init>(ServerSocket.java:170)
        at java.net.ServerSocket.<init>(ServerSocket.java:82)
        at sun.rmi.transport.proxy.RMIDirectSocketFactory.createServerSocket(RMIDirectSock
etFactory.java:30)
        at
sun.rmi.transport.proxy.RMIMasterSocketFactory.createServerSocket(RMIMasterSock
etFactory.java:312)
        at sun.rmi.transport.tcp.TCPEndpoint.newServerSocket(TCPEndpoint.java:559)
        at sun.rmi.transport.tcp.TCPTransport.listen(TCPTransport.java:200)
        at sun.rmi.transport.tcp.TCPTransport.exportObject(TCPTransport.java:172)
        at sun.rmi.transport.tcp.TCPEndpoint.exportObject(TCPEndpoint.java:319)
        at sun.rmi.transport.LiveRef.exportObject(LiveRef.java:119)
        at sun.rmi.server.UnicastServerRef.exportObject(UnicastServerRef.java:125)
        at sun.rmi.registry.RegistryImpl.setup(RegistryImpl.java:95)
        at sun.rmi.registry.RegistryImpl.<init>(RegistryImpl.java:81)
        at java.rmi.registry.LocateRegistry.createRegistry(LocateRegistry.java:167)
        at RmiJdbc.RJJdbcServer.register(RJJdbcServer.java:155)
        at RmiJdbc.RJJdbcServer.main(RJJdbcServer.java:243)
C:\j2sdkee1.2.1\bin>
```

Beachten Sie folgendes:
Sobald Sie Ihre Datenbank eingerichtet haben, sollten Sie es vermeiden den Datenbankserver mit CTRL-C zu stoppen. Dies kann zu inkonsistenten Daten führen. Verwenden Sie statt dessen `cloudscape -stop`.

1.7.6.2. Task 2

Starten Sie die J2EE RI mit dem Befehl
`j2ee -verbose`.

Auch hier sollten Sie überprüfen, ob die MusicStoreDB, die Sie in der vorigen Übung aufgesetzt haben, geladen wird.

Beim Starten der J2EE RI mit der `-verbose` Option sollten Sie aine Ausgabe sehen, die in etwa folgender entspricht:

```
c:\> j2ee -verbose

Naming service started: :1050
Published the configuration object ...
Binding DataSource, name = jdbc/Cloudscape,
    url = jdbc:cloudscape:rmi:CloudscapeDB;create=true
Binding DataSource, name = jdbc/MusicStore,
    url = jdbc:cloudscape:rmi:MusicStoreDB;create=false
Configuring web service using "default"
Web service started: :9191
Web service started: :7000
Configuring web service using "default"
Configuring web service using
    "file:/D:/j2sdkee1.2/public_html/WEB-INF/web.xml"
Web service started: :8000
Endpoint [SSL:
    ServerSocket[addr=0.0.0.0/0.0.0.0,port=0,localport=7000]]
    shutdown due to exception:
    javax.net.ssl.SSLException: No available certificate corresponds
        to the SSL cipher suites which are enabled.
endpoint down: :7000
Loading jar:/C:/j2sdkee1.2/MusicStore/MusicStore947749297345Server.jar
/C:/j2sdkee1.2/MusicStore/MusicStore947749297345Server.jar
Looking up authenticator...
Binding name:java:comp/env/ejb/MusicCD
J2EE server startup complete.
```

Die entscheidende Zeile ist:

```
Binding DataSource, name = jdbc/MusicStore,
    url = jdbc:cloudscape:rmi:MusicStoreDB;create=false
```

die anzeigt, dass die MusicStore Datenbank dem J2EE Server bekannt ist, und

```
Loading jar:/C:/j2sdkee1.2/MusicStore/MusicStore947749297345Server.jar
/C:/j2sdkee1.2/MusicStore/MusicStore947749297345Server.jar
Looking up authenticator...
Binding name:java:comp/env/ejb/MusicCD
```

die anzeigt, dass die MusicStore Anwendung einsatzbereit ist.

1.7.6.3. Task 3

Editieren Sie den `MusicClient.java` Programmrahmen und ergänzen Sie ihn so, dass das Programm einen JNDI Context erhält und diesen benutzt, um eine Referenz auf MusicCDHome zu erhalten.

Sie erhalten den JNDI Context, indem Sie eine neue Instanz von `javax.naming.InitialContext` und speichern Sie diese Referenz in einer Variable vom Datentyp `javax.naming.Context`.

1.7.6.4. Task 4

Kreieren Sie eine neue MusicCD Bean mit dem UPC Code im Programm-Skeleton oder von irgend einer eigenen CD und ergänzen Sie die restlichen Daten wie im Skeleton vorgegeben.

Kreieren Sie mit der `create()` Method und dem UPC Code als Argument, um eine Bean zu kreieren. Mit Hilfe der Zugriffsmethoden und den Mutationsmethoden des remote Interfaces können Sie auf die Daten der Beans zugreifen oder diese mutieren.

1.7.6.5. Task 5

Ergänzen Sie das Programmfragment zum Suchen der MusicCD Bean, die Sie in der Datenbank kreiert haben und die Ihnen die Datenfelder, die Sie im vorigen Arbeitsschritt eingegeben haben, ausgibt.

Kreieren Sie eine Instanz von `MusicCDPK (Primary Key)`, setzen Sie das `upc` Datenfeld auf einen passenden Wert und benutzen Sie die Methode `findByPrimaryKey()` des MusicCD home Interfaces, um die entsprechende MusicCD Bean zu finden.

1.7.6.6. Task 6

Übersetzen und starten Sie die Client Applikation.

Weil der Client beide Beans, die MusicCD Bean und die Inventory Bean, benutzen müssen Sie die beiden Archive `Inventory.jar` und `MusicCD.jar` in ihrem CLASSPATH sein, damit Sie die Applikation korrekt übersetzen können. Zudem muss noch das `j2ee.jar` Archiv im CLASSPATH sein, weil sich darin die `javax.ejb` Klassen befinden.

1.7.6.7. Task 7

Editieren Sie das `MusicInventoryClient.java` Skeleton, um einen JNDI Context zu erhalten und setzen Sie diesen Context ein, um eine Referenz auf InventoryHome zu erhalten.

Sie erhalten einen JNDI Context, indem Sie eine neue Instanz von `javax.naming.InitialContext` bilden und in einer Referenzvariable vom Datentyp `javax.naming.Context` abspeichern.

1.7.6.8. Task 8

Kreieren Sie eine neue Inventory Bean.

Verwenden Sie die `create()` Methode des Inventory home Interfaces.

1.7.6.9. Task 9

Kreieren Sie eine Instanz von `MusicData` und fügen Sie die Daten Ihrer LieblingsCDs in die Datenbank ein. Verwenden Sie anschliessend die Business Methoden der Inventory Bean, um Daten abzufragen oder zu mutieren.

Die `addInventory()` Methode der Inventory Bean verwendet eine Instanz von `MusicData` als Argument und fügt diese Daten in die Datenbank ein.

1.7.6.10. Task 10

Übersetzen und starten Sie Ihre Client Applikation.

Weil der Client beide Beans,die MusicCD Bean und die Inventory Bean verwendet, müssen die beiden Archive `Inventory.jar` und `MusicCD.jar` in Ihrem CLASSPATH sein, damit Sie die Programme übersetzen können. Die J2EE Klassen befinden sich im Archiv `j2ee.jar` . Dieses Archiv muss also auch noch in den CLASSPATH, da sich darin die gesamten `javax.ejb` Klassen befinden.

Viel Spass und ...Bean me up Scotty ...