

Die Datenquelle mit unserer Anwendung verbinden

Wenn die Datenbank an Ort und Stelle ist, müssen wir unsere Anwendung dazu bringen, diese auch zu benutzen. Wie viel Aufwand dies bedeutet, hängt davon ab, wie eng unsere Anwendung an die Datenquelle gekoppelt ist. Bei einer Anwendung, in der die Datenquelle direkt angesprochen wird, müssten wir den Code zum Suchen in der Datenbank oder zumindest die Umgebungsreferenzen aktualisieren, und wir müssten unsere SQL-Anweisungen so verändern, dass sie mit MySQL kompatibel sind. Glücklicherweise beruht unsere Anwendung jedoch auf container-verwalteter Persistenz, und der Datenbankzugriff ist vollkommen transparent. Daher ist die Migration nicht besonders schwierig.

Wie mache ich das?

Es gibt Schwierigeres als dies. JBoss sucht die Konfiguration der container-verwalteten Persistenz in einem Deployment-Deskriptor namens `jbosscmp-jdbc.xml`. Die vorgegebene Datenquelle für die Anwendung wird im Abschnitt `defaults` der Datei festgelegt, wie es hier zu sehen ist:

```
<!DOCTYPE jbosscmp-jdbc PUBLIC
"-//JBoss//DTD JBOSSCMP-JDBC 4.0//EN"
"http://www.jboss.org/j2ee/dtd/jbosscmp-jdbc_4_0.dtd">

<jbosscmp-jdbc>
  <defaults>
    <datasource>java:MySqlDS</datasource>
    <datasource-mapping>mySQL</datasource-mapping>
  </defaults>
</jbosscmp-jdbc>
```

Das Element `datasource` enthält den JNDI-Namen der Datenquelle, den wir zuvor in der Datei `mysql-ds.xml` festgelegt haben. Der Name in `datasource-mapping` deklariert den Typ der Datenbank. Der Datenbanktyp legt fest, wie die SQL-Anweisungen zu generieren sind und wie die Typabbildung für die jeweilige Datenbank vorzunehmen ist.

Ein Blick in die Datei `conf/standardjbosscmp-jdbc.xml` zeigt Ihnen den vollen Umfang der in JBoss verfügbaren Datenquellen-Abbildungen. Innerhalb des Abschnitts `type-mappings` finden Sie die Typabbildung `mySQL`, auf die sich die Datei `jbosscmp-jdbc.xml` bezieht:

```
<type-mapping>
  <name>mySQL</name>
  ...
</type-mapping>
```

Die Typabbildung definiert SQL-Muster für die Schema-Generierung, die Abbildung von EJB-QL- und JBOSS-QL-Funktionen sowie die vorgegebenen Abbildungen für die Java-Standardtypen. Hier ist die Abbildung für `java.lang.String`:

```
<mapping>
  <java-type>java.lang.String</java-type>
  <jdbc-type>VARCHAR</jdbc-type>
  <sql-type>VARCHAR(250) BINARY</sql-type>
</mapping>
```

Wenn Sie die Abbildung von Strings für alle MySQL-Datenbanken im System ändern müssen, können Sie die Datei `standardjbosscmp-jdbc.xml` ändern. In der Regel werden Sie eher die Abbildung einzelner Felder bearbeiten wollen, aber es ist vielleicht gut zu wissen, dass Sie bei Bedarf auch die Abbildungsregeln systemweit modifizieren können.

Die Datei `jbosscmp-jdbc.xml` befindet sich im Verzeichnis `META-INF` der EJB-JAR-Datei, gleich bei den Deployment-Deskriptoren `ejb-jar.xml` und `jboss.xml`. Um die MySQL-spezifische Datei `jbosscmp-jdbc.xml` hinzuzufügen, ergänzen Sie die `ant`-Befehle zur Erstellung der `ToDo`-Anwendung um `-Doptional.dd=mysql`, wie hier gezeigt:

```
$ ant -Doptional.dd=mysql main deploy
```

Dadurch wird die MySQL-Version der Anwendung generiert.

Was ist gerade geschehen?

Die `ToDo`-Anwendung redet jetzt nicht mehr mit der internen Hypersonic-Datenbank, sondern mit der von Ihnen aufgesetzten MySQL-Instanz. Genau wie zuvor hat JBoss auch jetzt automatisch ein zu unseren Entity-Beans passendes Datenbankschema generiert. Mit dem Befehl `show tables` können Sie sehen, dass unsere Tabelle korrekt angelegt worden ist:

```
mysql> use jbossdb;
Database changed
mysql> show tables;
+-----+
| Tables_in_jbossdb |
+-----+
| Comment            |
| Task               |
+-----+
2 rows in set (0.19 sec)
```

Und mit dem Befehl `describe` können Sie das für die Tabelle generierte Schema prüfen:

Wenn Sie MySQL auf einem anderen Host laufen lassen möchten, müssen Sie dem Datenbankbenutzer den Zugriff von dem Host aus gewähren, auf dem JBoss läuft.