Hilfe
meine Oracle – Datenbank spricht nicht mit meiner JSP

Diese Dokumentation, soll die grundlegenden Schritte vermitteln wie man mithilfe einer JSP – Seite auf eine Oracle – Datenbank zugreift und Daten abfragt und manipuliert.

Inhalt

Einleitung 3
Wahl der Projektart und notwendige Einstellungen 3
Testen der Datenbankverbindung 8
JSP –Programmierung 16
Datenabfragen und Manipulation über die JSP 17
Erstellen von Tabellen 17
Abfragen von Tabellen 17
Einfügen von Zeilen 17
Updaten von Zeilen 17
Löschen von Zeilen 17
Löschen von Tabellen 18
Ausgabe der Daten in der JSP 18

Einleitung

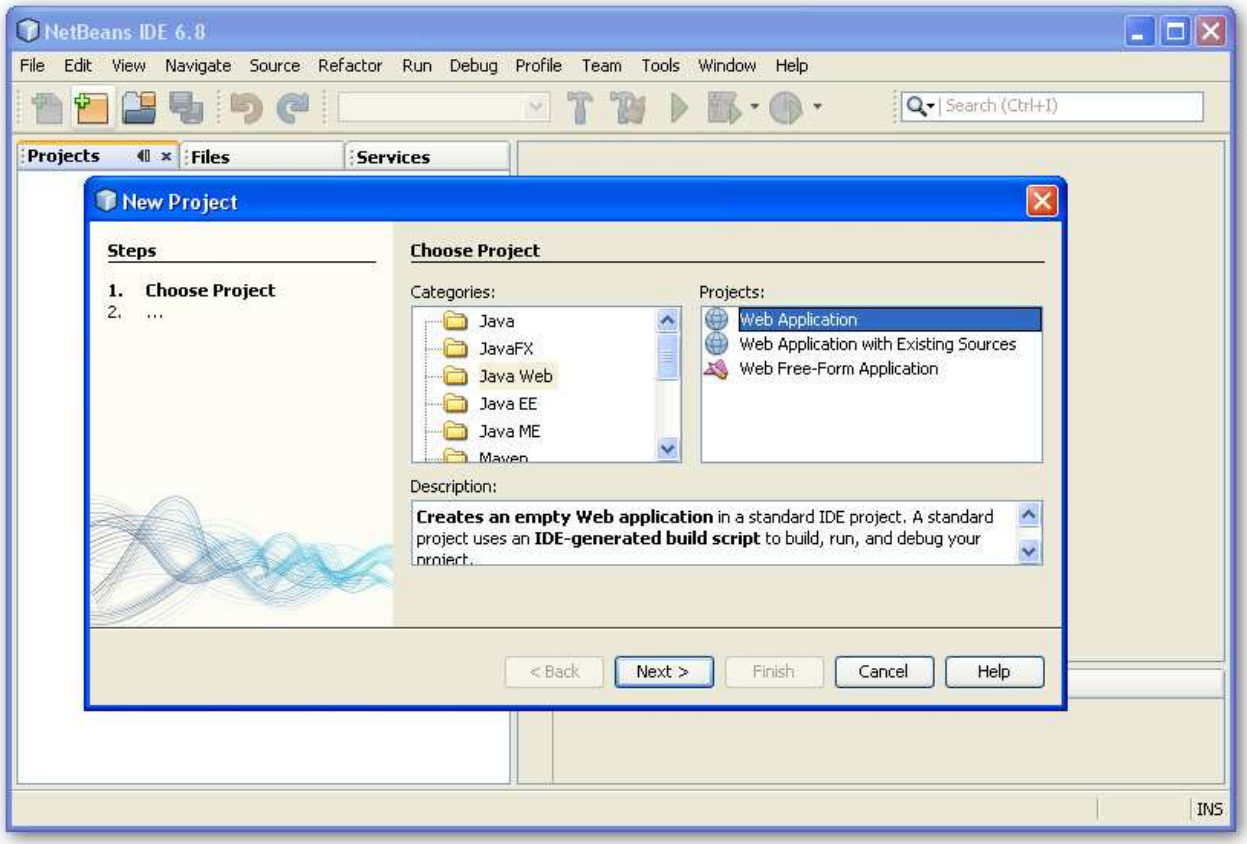
Um in einem NetBeans – Project auf eine Oracle - Datenbank zugreifen zu können, benötigt es einige Vorbereitungen. Anhand einer Webanwendung soll veranschaulicht werden, wie man eine Verbindung zu einer eine Oracle – Datenbank herstellt und dann dort Daten abfragt und später auch manipuliert.

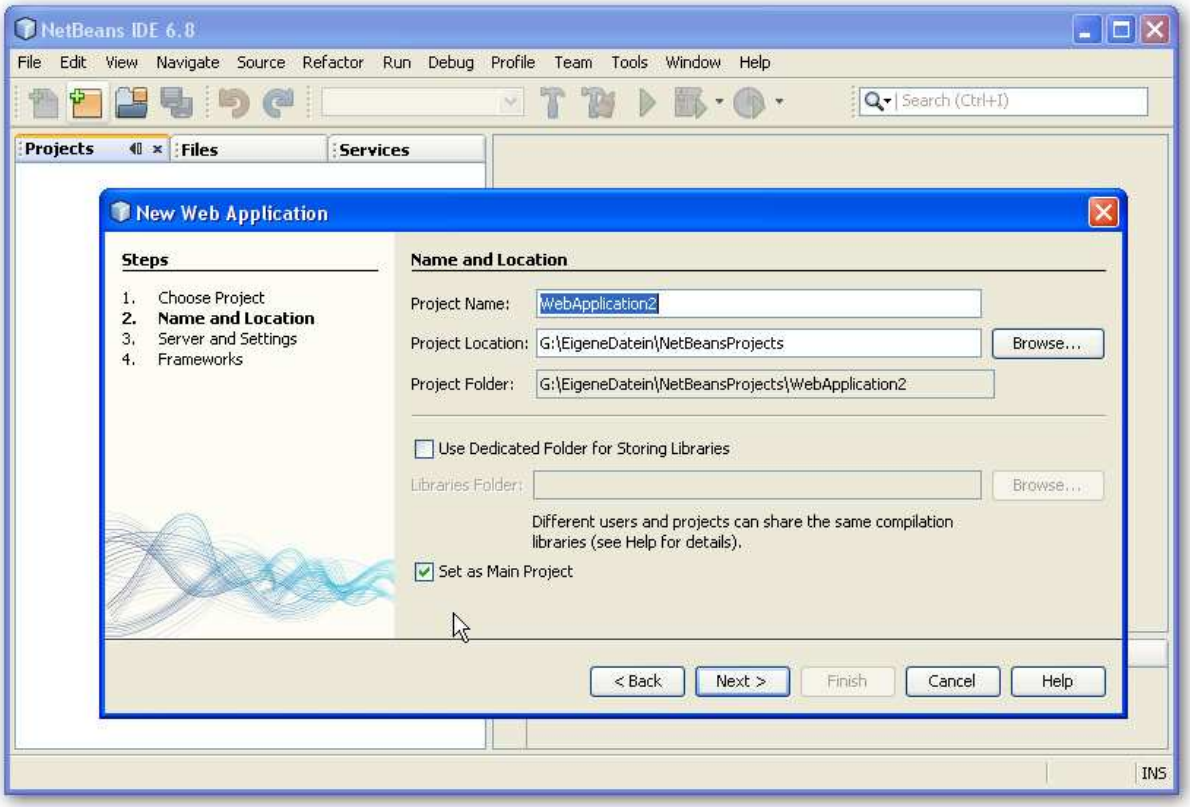
Wahl der Projektart und notwendige Einstellungen

Bei einer JSP – Seite handelt es sich in NetBeans um eine einfache Webanwendung.

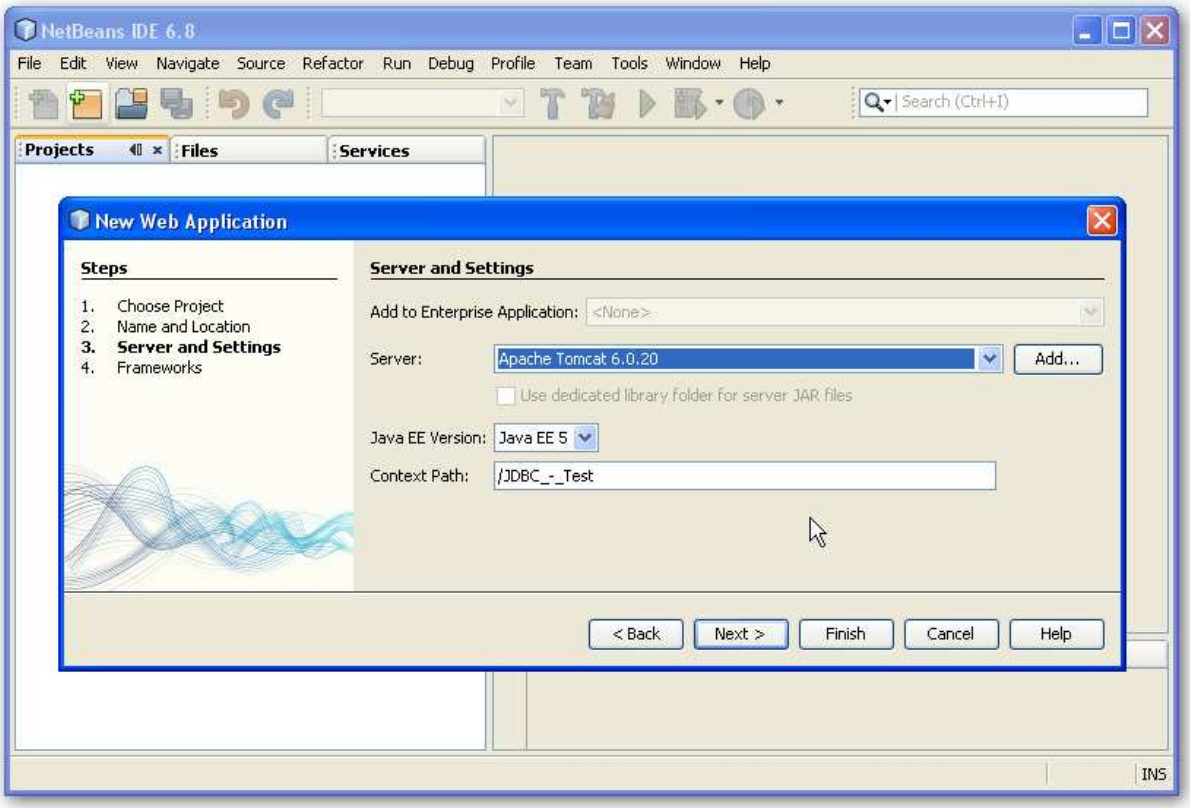
Um eine einfach Webanwendung zu erstellen, geht man wie folgt vor:

File -> New Project oder Strg+Umschalt+N drücken um ein neues Projekt zu erstellen
Categories : Java Web und bei Project Web Application wählen und mit Next weiter in den Einstellungen folgen

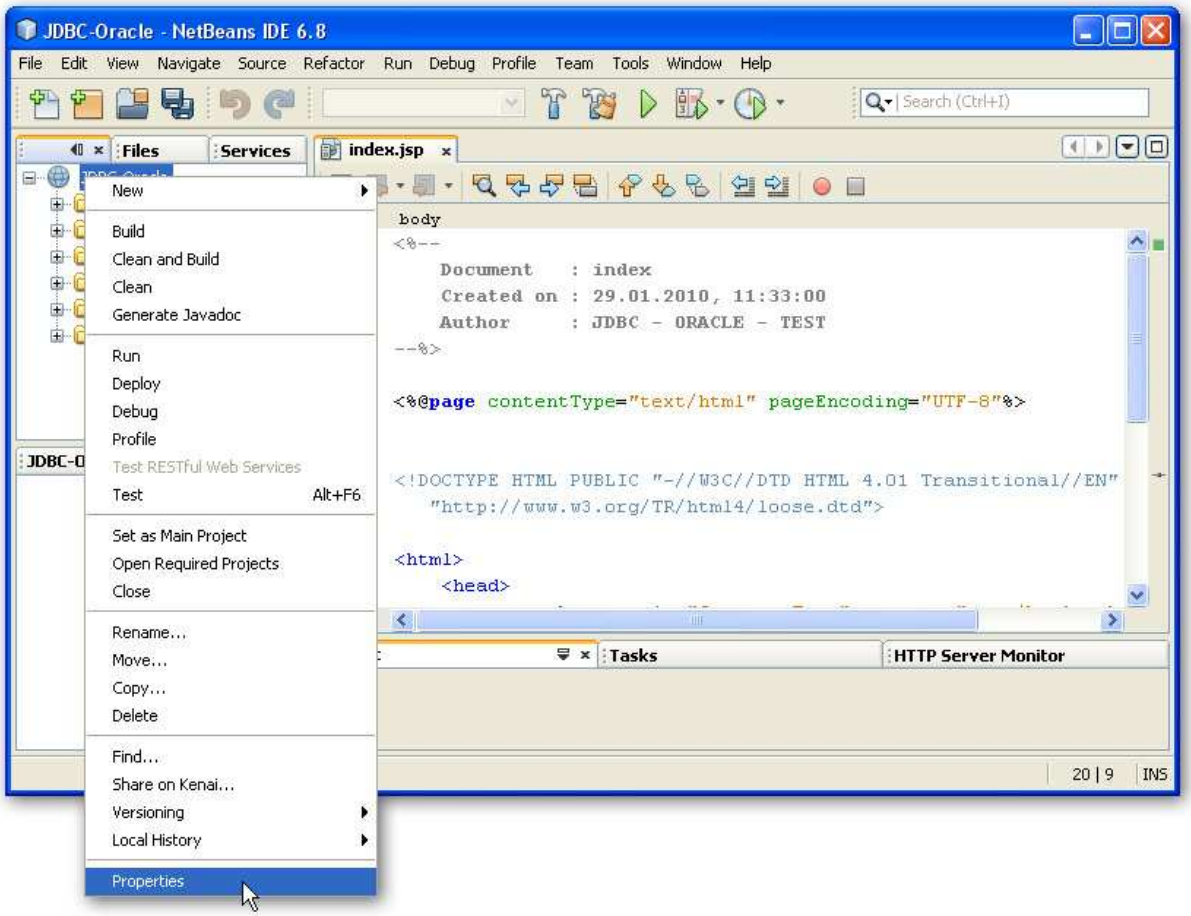




Vergeben sie einen Namen in unserem Beispiel werden wir das Project JDBC – Oracle nennen. Mit Next Auswahl bestätigen und weiter zu den Server und Settings gehen. Ganz wichtig ist die Serverauswahl. Drauf achten, dass der Tomcat ausgewählt ist. Mit Next bestätigen und im folgenden Fenster ohne Auswahl eines Frameworks der Project abschließen



Um mit einer Datenbank sprechen zu können ist eine Schnittstelle nötig. In diesem Beispiel realisieren wir die Kommunikation zur Datenbank über den JDBC Treiber. Der benötigte Treiber der zur Kommunikation mit der Oracle – Datenbank notwendig ist kommt mir der Installation der Datenbank mit. Jedoch ist es notwendig den Treiber dem Projekt zuzuweisen, damit NetBeans die genaue Information hat, wie das Projekt auf die Datenbank zugreifen kann.



Um dem Projekt den JDBC von Oracle zuzuweisen wechselt man in die Eigenschaften des Projekts indem man mit einem Rechtsklick das Kontextmenü öffnet und dann den Punkt Properties anklickt.

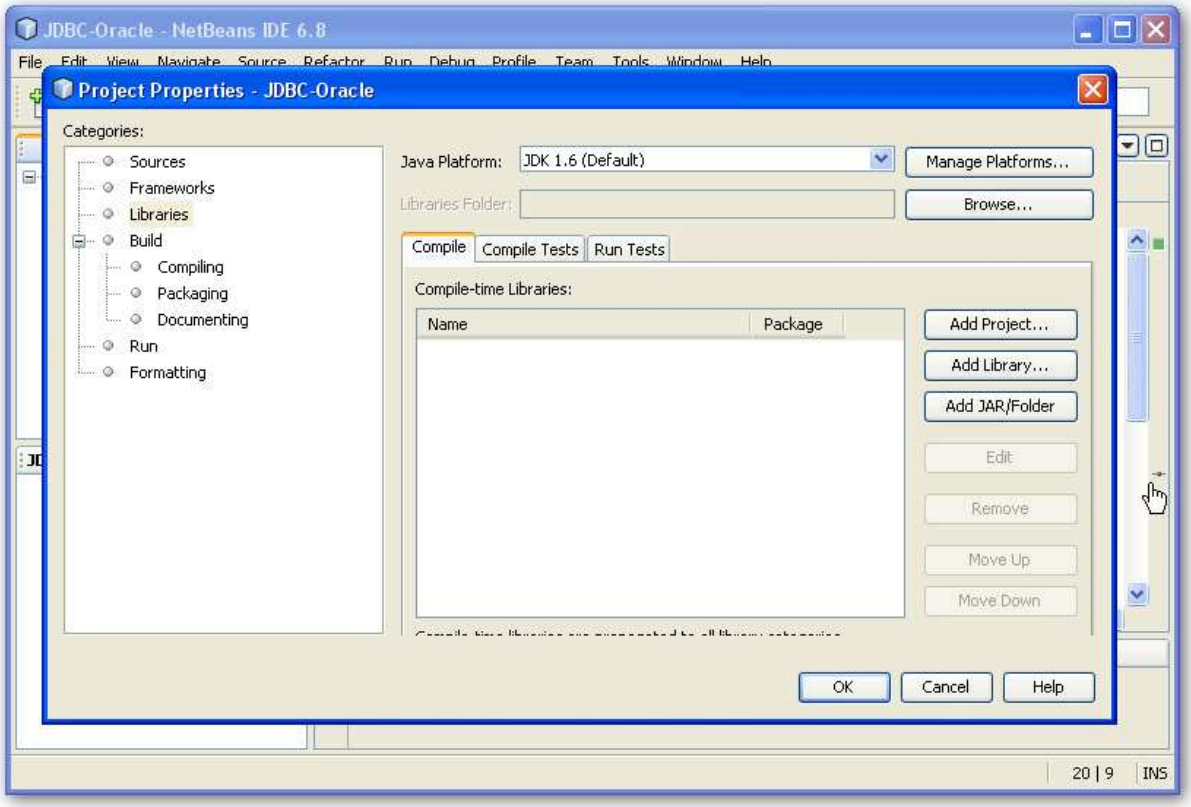
In dem nun neu geöffneten Properties – Fenster kann man dann unter dem Punkt Libraries dann zwei Bibliotheken einbinden.
Zum einem die JSTL - Bibliothek und zum anderen den Treiber um mit der Datenbank in Verbindung treten zu können.

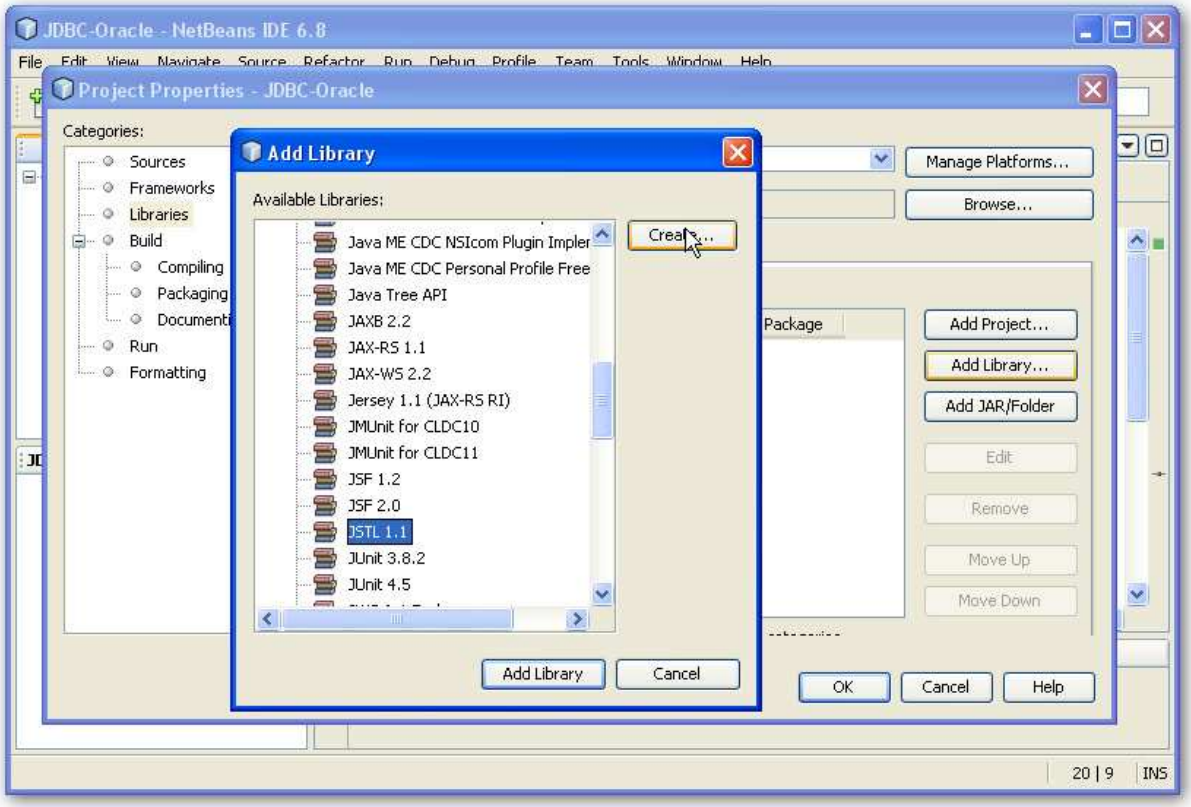
Um die JSTL – Bibliothek einzubinden auf den Add Library Button klicken in dem neuen Fenster durch die Liste scrollen und die JSTL 1.1 auswählen und die Auswahl mit Add Library bestätigen.

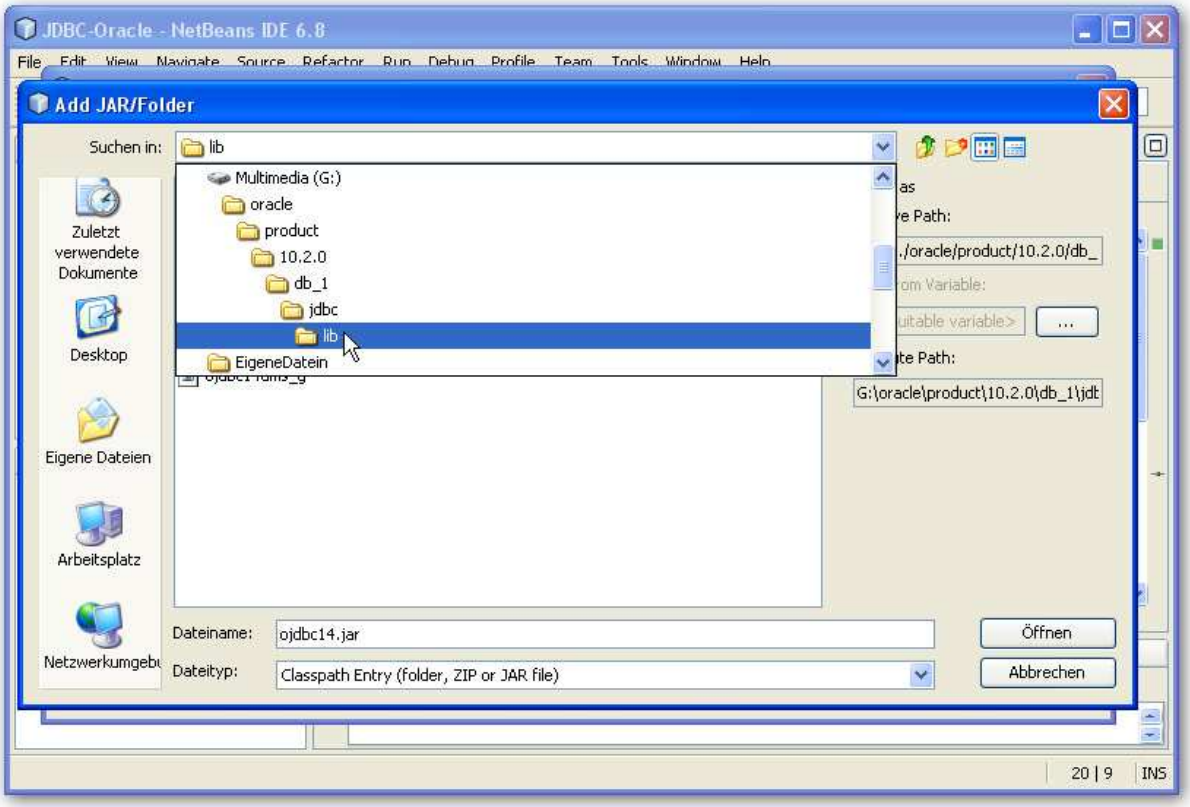
Die Bibliothek zum Oracle – Treiber bindet man ein, indem man mit dem Klick auf den Button Add JAR/Folder ein Öffnen – Dialog aufruft. Dort geht man dann in das Installationsverzeichnis der Oracle – Datenbank. Die JAR – Datei zum Treiber findet man im folgenden Pfad:

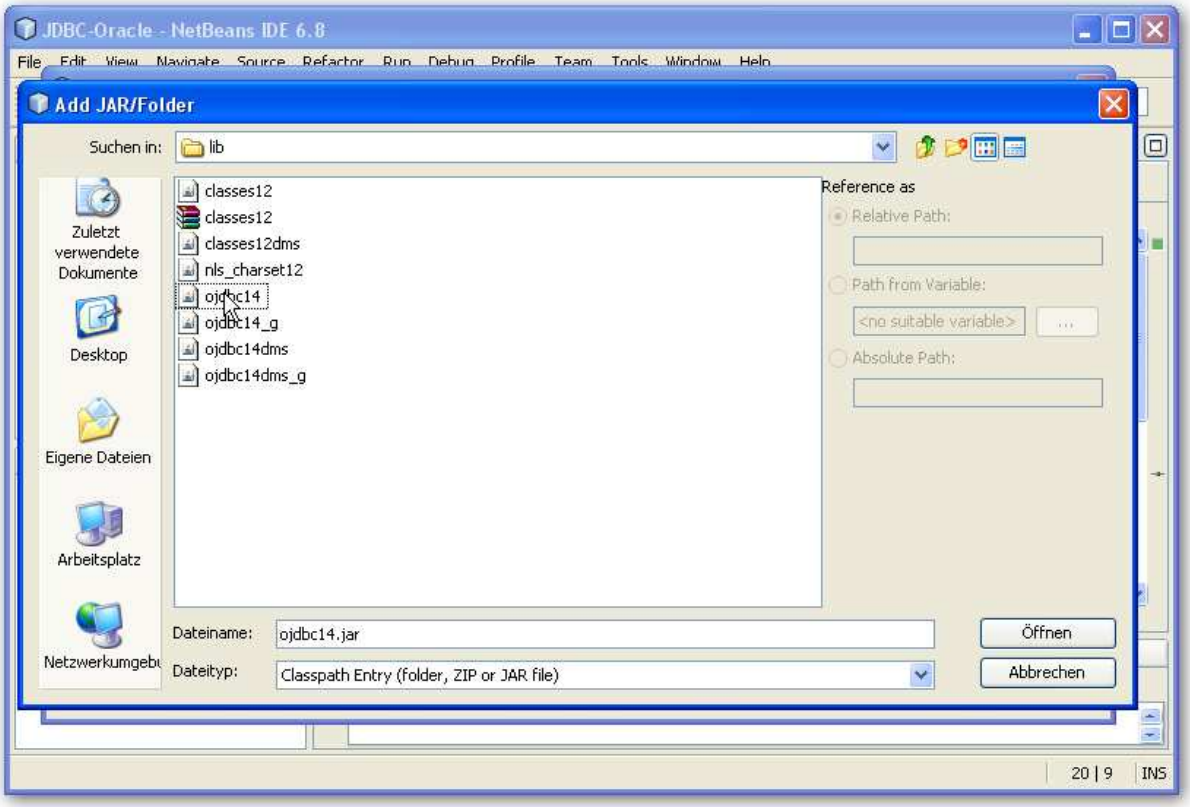
Festplattenpartition:\oracle\product\10.2.0\db_1\jdbc\lib

Als Treiberdatei wählt man die ojdbc14.jar und bestätigt seine Auswahl wieder.


JDBC-Oracle - NetBeans IDE 6.8
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Project Properties - JDBC-Oracle
Categories:
Sources
Frameworks
Libraries
Build
Compiling
Packaging
Documenting
Run
Formatting
Java Platform:
JDK 1.6 (Default)
Manage Platforms...
Libraries Folder:
Browse...
Compile
Compile Tests
Run Tests
Compile-time Libraries:
Name
Package
Add Project...
Add Library...
Add JAR/Folder
Edit
Remove
Move Up
Move Down
OK
Cancel
Help
20 | 9
INS


JDBC-Oracle - NetBeans IDE 6.8
Project Properties - JDBC-Oracle
Add Library
Available Libraries:
Java ME CDC NSIcom Plugin Impler
Java ME CDC Personal Profile Free
Java Tree API
JAXB 2.2
JAX-RS 1.1
JAX-WS 2.2
Jersey 1.1 (JAX-RS RI)
JMUnit for CLDC10
JMUnit for CLDC11
JSF 1.2
JSF 2.0
JSTL 1.1
JUnit 3.8.2
JUnit 4.5
Create...
Add Library
Cancel
Manage Platforms...
Browse...
Add Project...
Add Library...
Add JAR/Folder
Edit
Remove
Move Up
Move Down
OK
Cancel
Help
20 | 9
INS

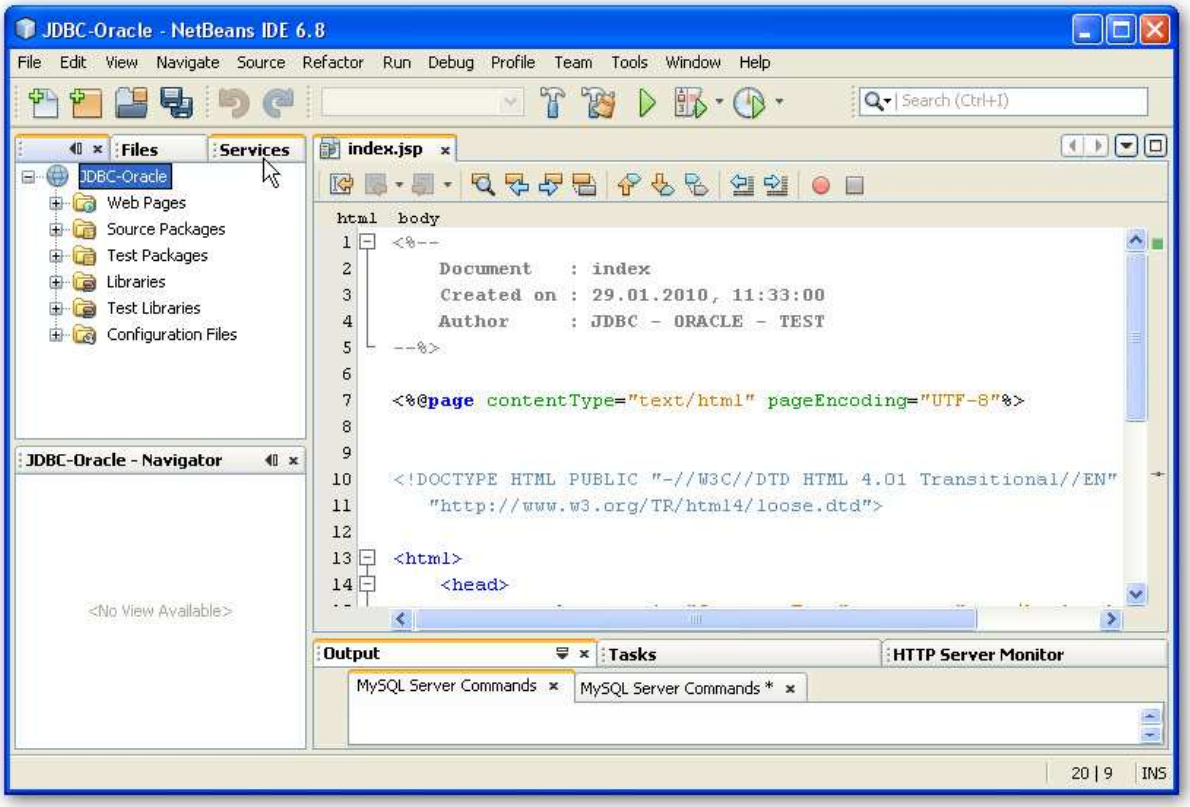




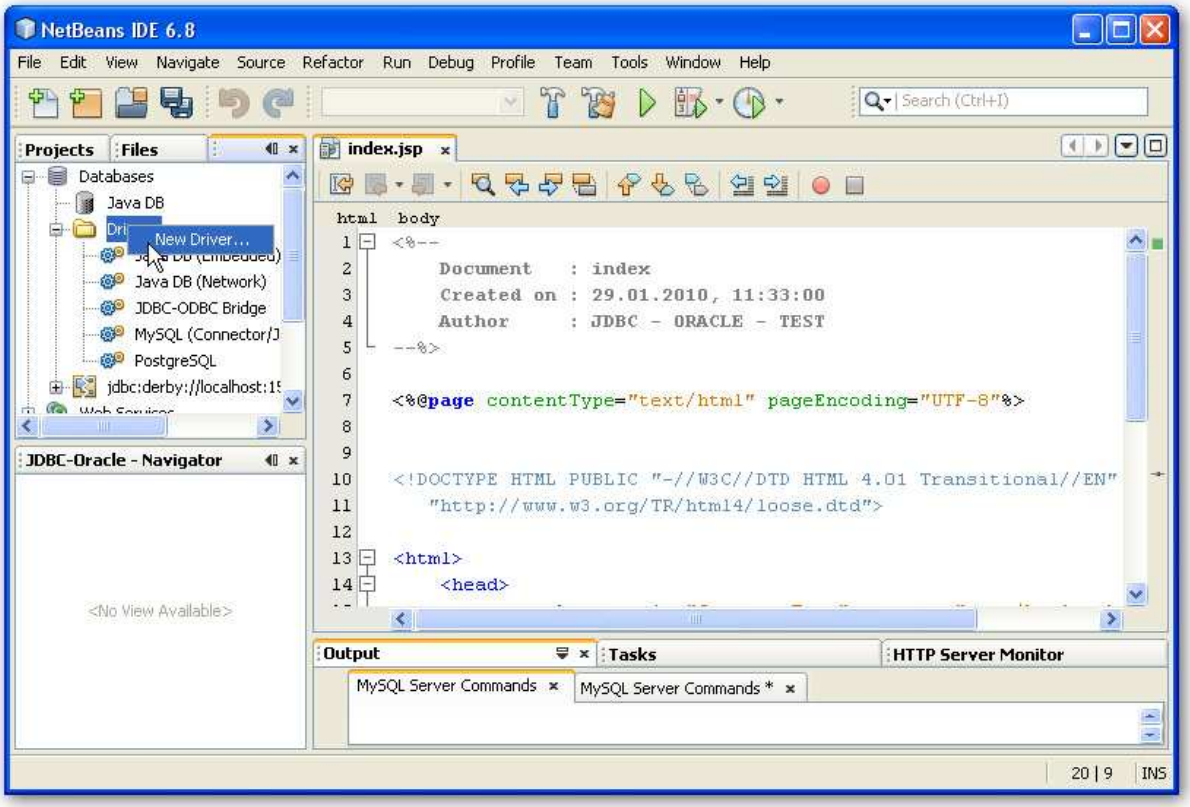
Damit sind nun die notwendigen Einstellungen getan, damit die JSP eine Verbindung mit der Datenbank aufbauen kann. Wie das genau geht, wird in den nächsten Seiten beschrieben.

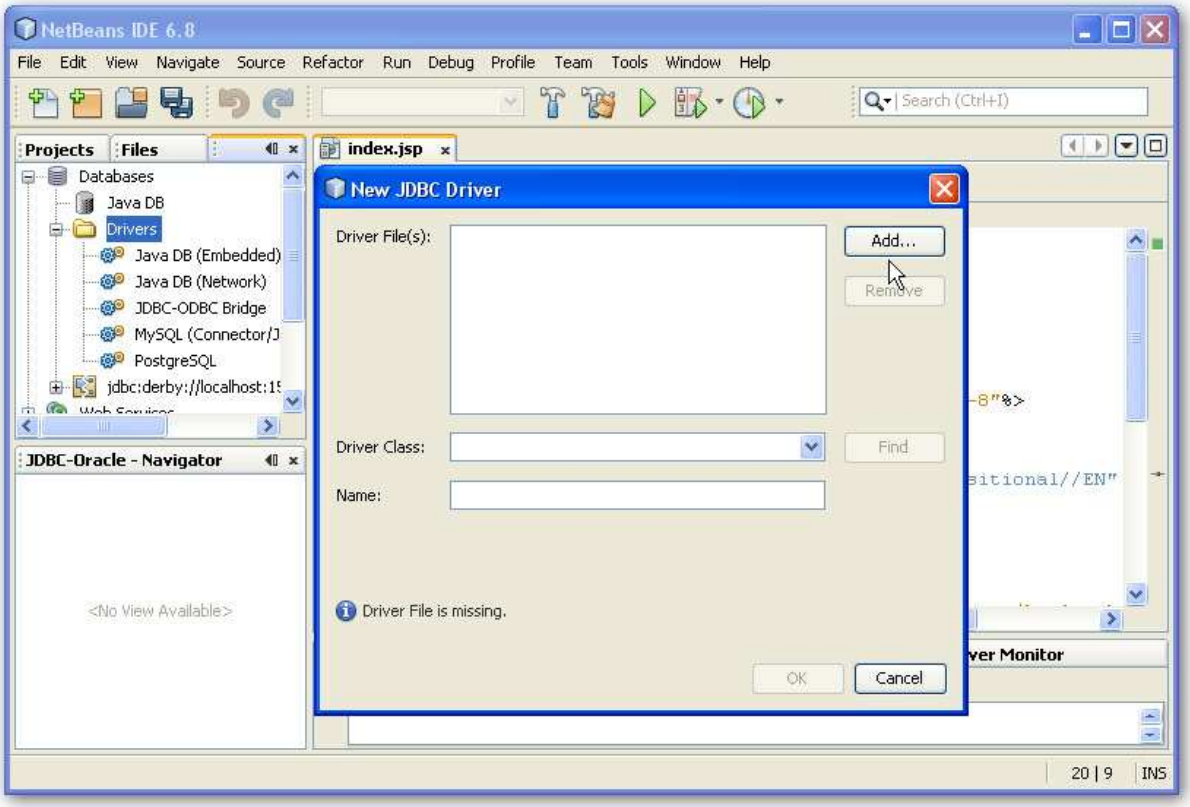
Testen der Datenbankverbindung

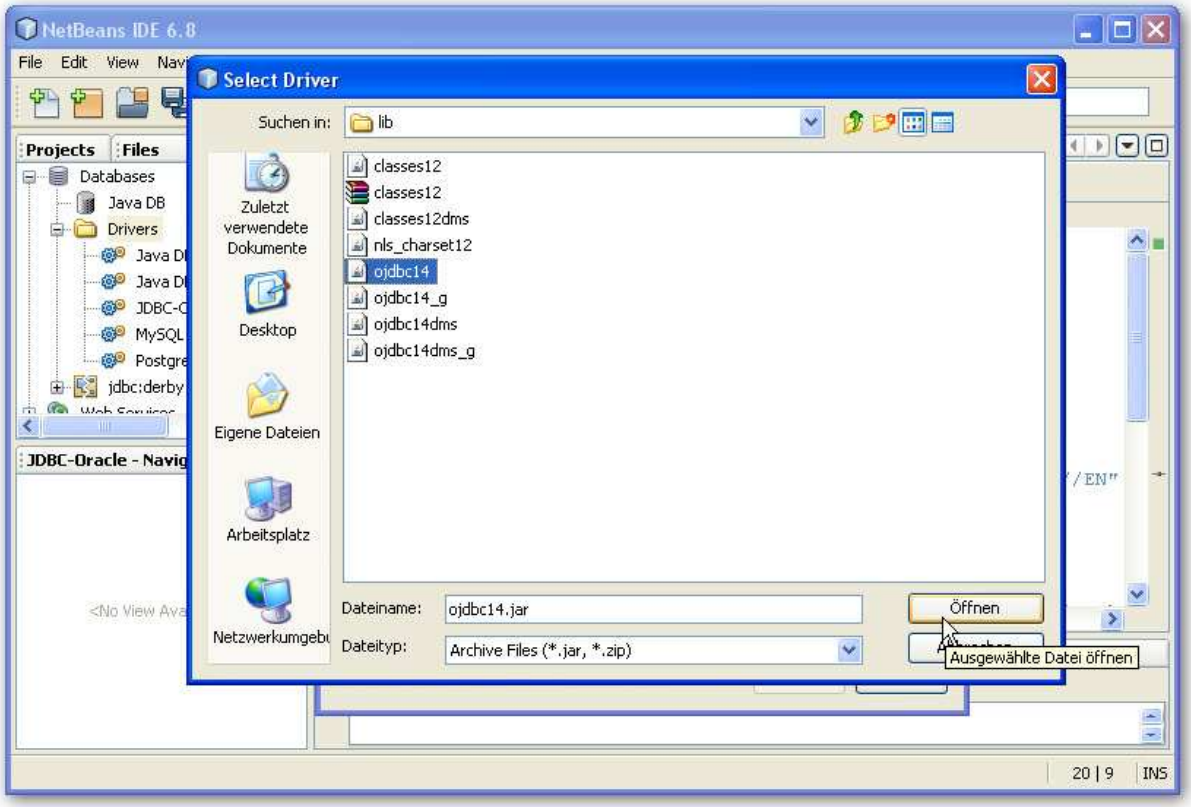
Das Testen der Datenbankverbindung ist nicht unbedingt wichtig, fortgeschrittene Benutzer können diesen Schritt auch auslassen. Anfänger sollten diesen Schritt ruhig ein –zweimal durchmachen um sich den Linkaufbau zur Datenbank zu verinnerlichen und um auch mögliche Fehler auszuschließen.

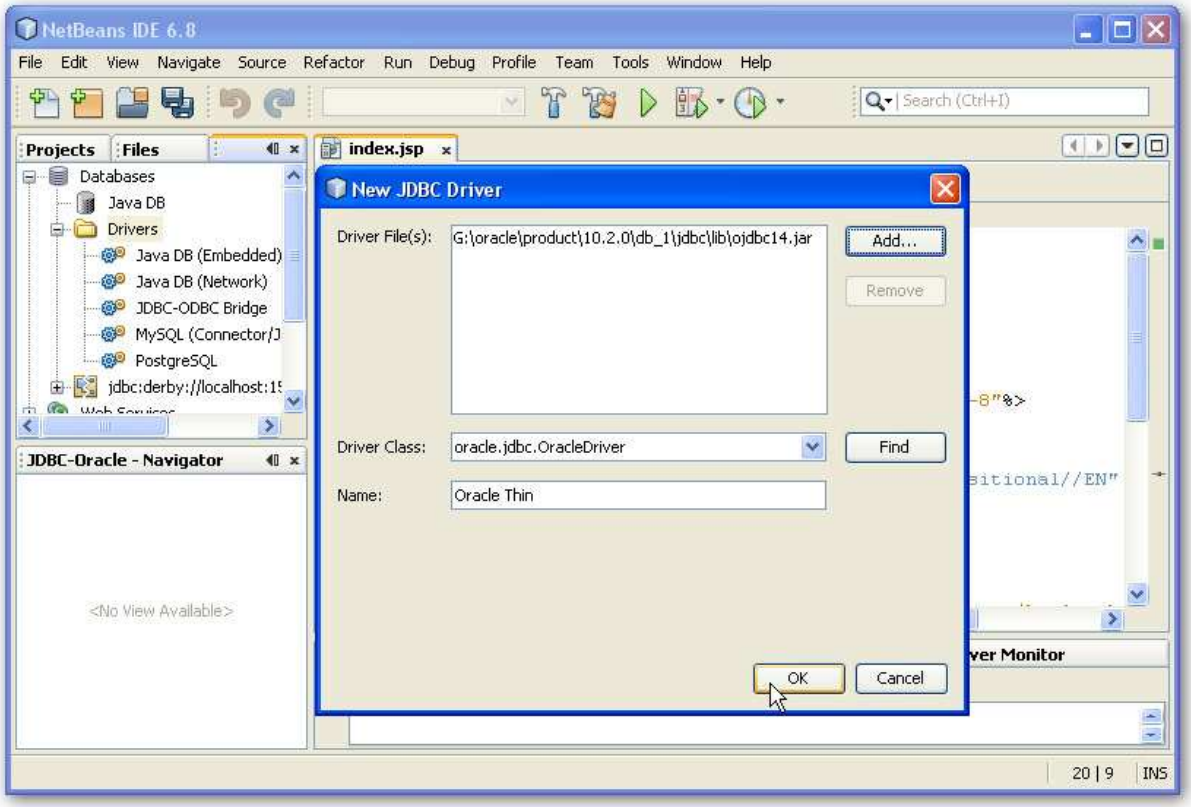


Um die Datenbankverbindung zu testen klickt man auf die Registerkarte Services
Dort findet man dann eine Liste wo unter anderen die zwei Punkte Databases und Driver enthalten sind.
Als erstes fügt man dem Test unter dem Punkt Driver wieder den Oracle Driver in der Weise zu, wie es aus den Voreinstellungen bekannt ist. Mit Rechtsklick auf Driver wählt man New Driver. Ein neuer Dialog öffnet sich und durch ein Klick auf Add… geht das bekannte Auswahl - Fenster auf und wählt denn wieder dem entsprechenden Treiber aus dem lib – Ordner aus.
Nach der Bestätigung sieht man dann die im JAR enthaltenden Treiberinformationen unter anderem die Treiberklassen –Informationen, die später noch einmal benötigt werden.

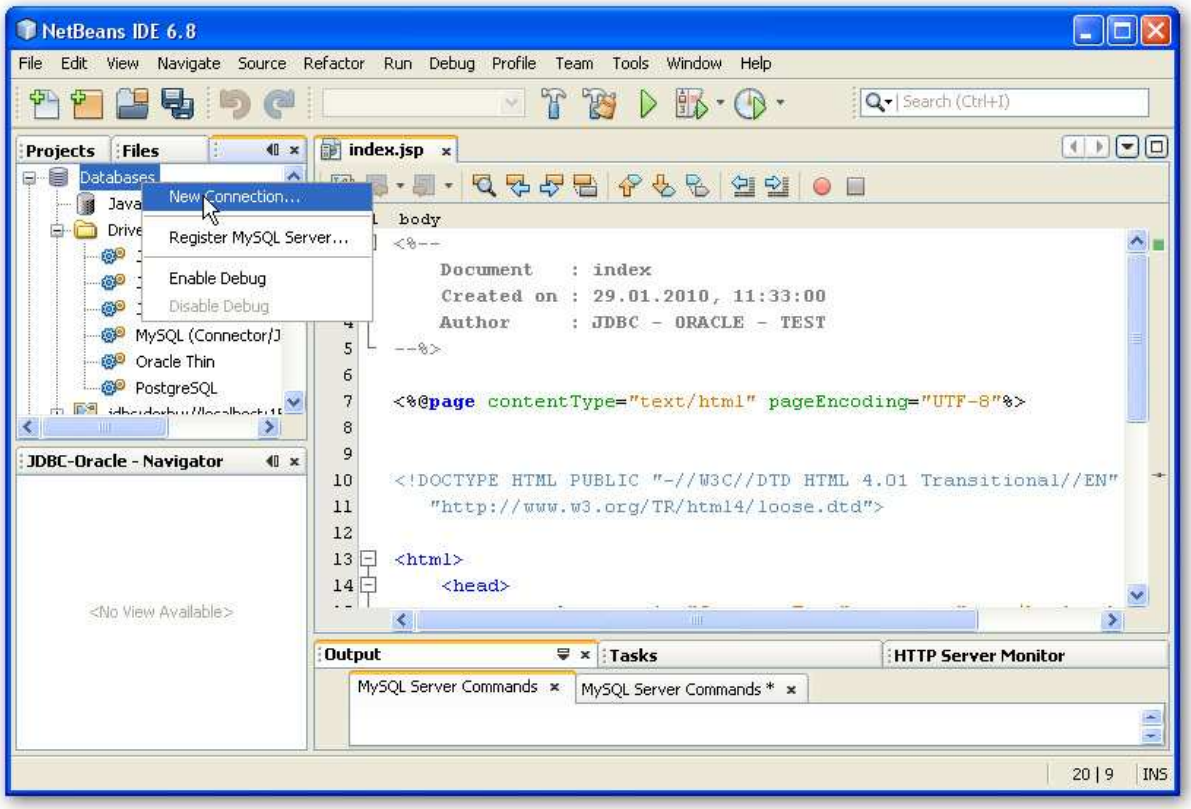

NetBeans IDE 6.8
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
Projects
Files
index.jsp
Databases
Java DB
New Driver...
Java DB (Network)
JDBC-ODBC Bridge
MySQL (Connector/J
PostgreSQL
jdbc:derby://localhost:1
JDBC-Oracle - Navigator
<No View Available>
html body
<%--
Document : index
Created on : 29.01.2010, 11:33:00
Author : JDBC - ORACLE - TEST
--%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
Output
Tasks
HTTP Server Monitor
MySQL Server Commands
MySQL Server Commands *
20 | 9
INS


NetBeans IDE 6.8
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
Projects
Files
index.jsp
Databases
Java DB
Drivers
Java DB (Embedded)
Java DB (Network)
JDBC-ODBC Bridge
MySQL (Connector/J
PostgreSQL
jdbc:derby://localhost:1
JDBC-Oracle - Navigator
<No View Available>
New JDBC Driver
Driver File(s):
Add...
Remove
Driver Class:
Find
Name:
Driver File is missing.
OK
Cancel
20 | 9
INS


NetBeans IDE 6.8
File Edit View Navi
Select Driver
Suchen in: lib
classes12
classes12
classes12dms
nls_charset12
ojdbc14
ojdbc14_g
ojdbc14dms
ojdbc14dms_g
Zuletzt verwendete Dokumente
Desktop
Eigene Dateien
Arbeitsplatz
Netzwerkumgebu
Dateiname: ojdbc14.jar
Dateityp: Archive Files (*.jar, *.zip)
Öffnen
Ausgewählte Datei öffnen
Projects
Files
Databases
Java DB
Drivers
JDBC-Oracle - Navig
20 | 9
INS


NetBeans IDE 6.8
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
Projects
Files
index.jsp
Databases
Java DB
Drivers
Java DB (Embedded)
Java DB (Network)
JDBC-ODBC Bridge
MySQL (Connector/J
PostgreSQL
jdbc:derby://localhost:15
JDBC-Oracle - Navigator
<No View Available>
New JDBC Driver
Driver File(s): G:\oracle\product\10.2.0\db_1\jdbc\lib\ojdbc14.jar
Add...
Remove
Driver Class: oracle.jdbc.OracleDriver
Find
Name: Oracle Thin
OK
Cancel
20 | 9
INS

Nachdem man den Treiber hinzugefügt hat, legt man eine neue Verbindung zu der Datenbank an. Dazu klickt man mit rechts auf Databases klickt und New Connection… auswählt.

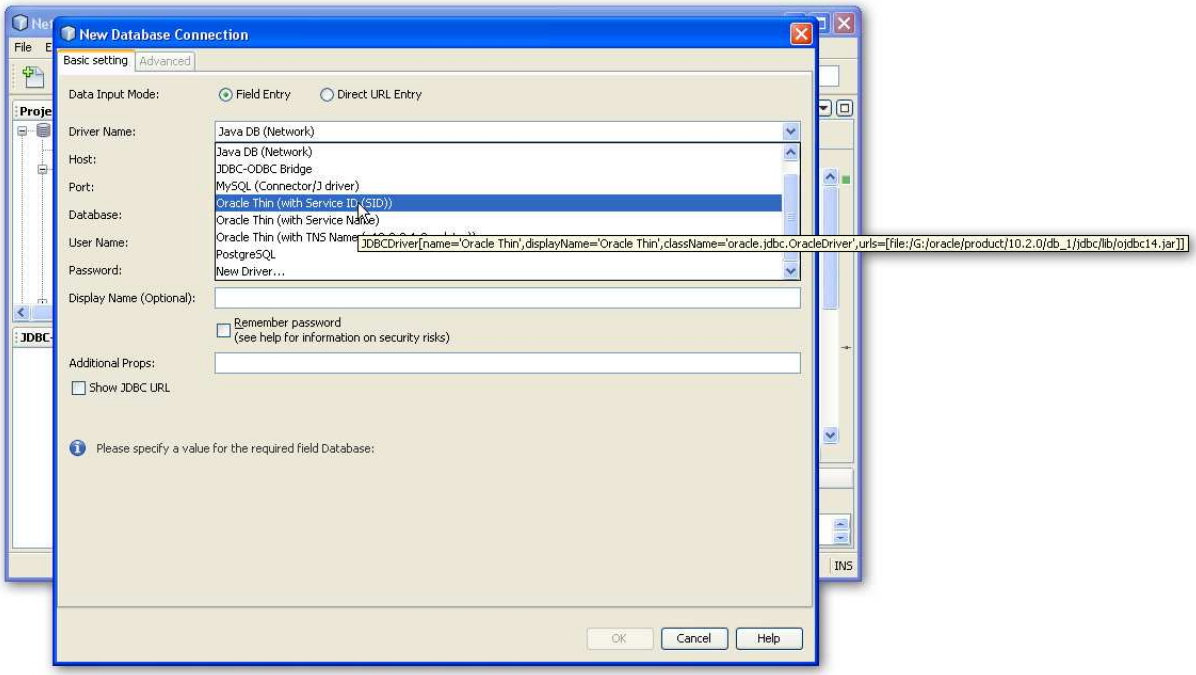


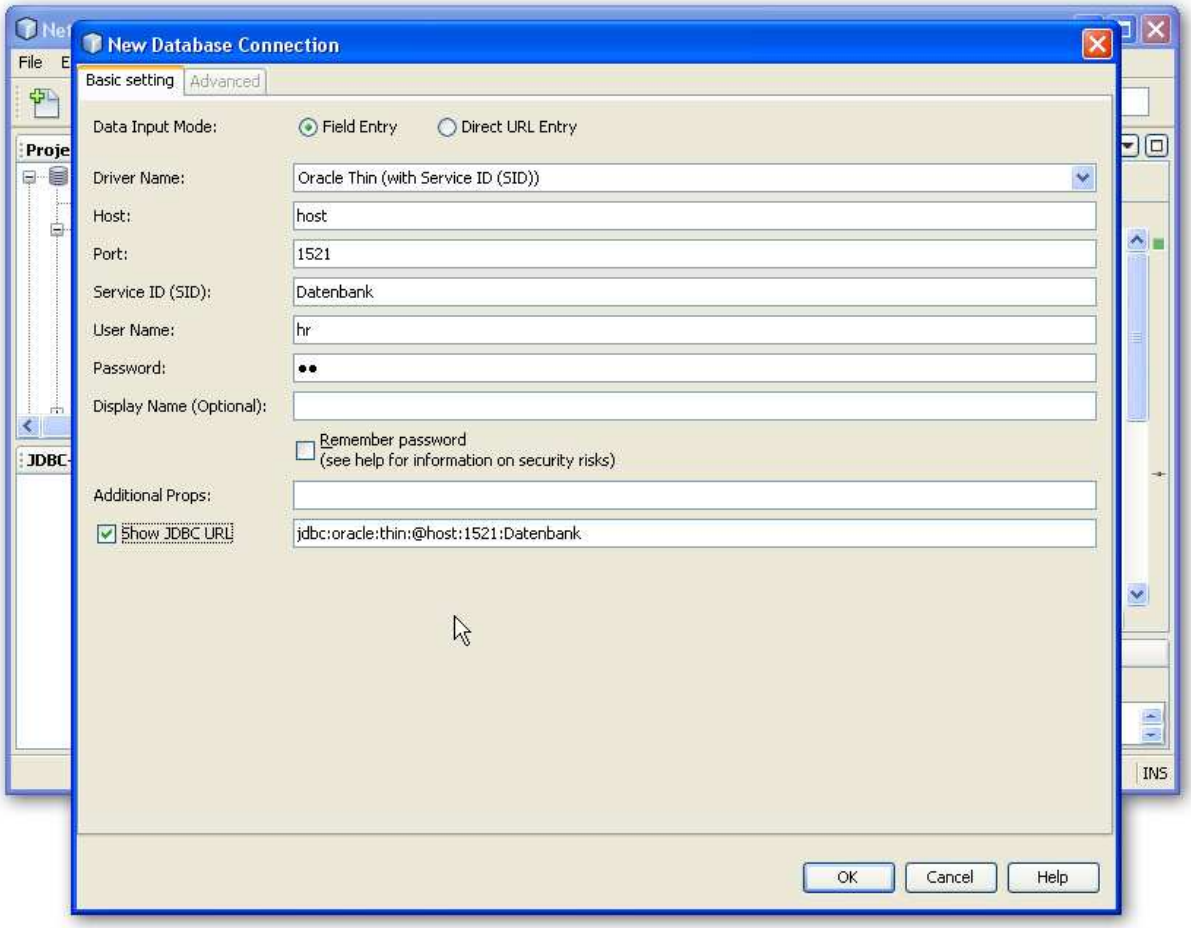
Im neuen Dialogfenster wählt man unter Driver Name den OracleTreiber aus. Dadurch, dass drei Treiber angeboten werden, wählt man einfacherhalber den ersten Treiber aus.
Unter Host wird die Host-ID angegeben auf die auch der Listener lauscht. Um die genaue Bezeichnung zu finden kann man auch in die TNSNAMES.ora der Oracle Installation aus zufinden im Pfad

Festenplattenbezeichner:\oracle\product\10.2.0\db_1\NETWORK\ADMIN

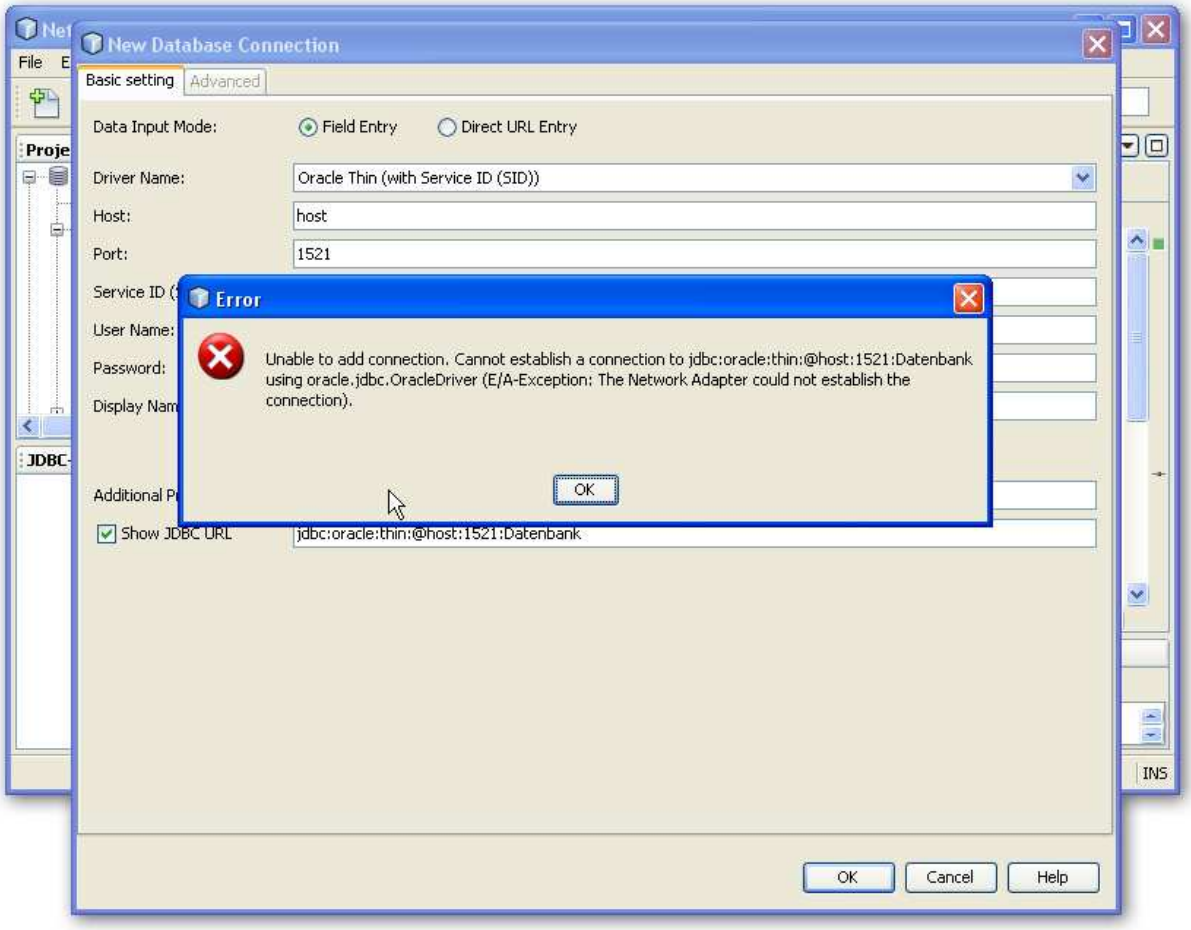
Dort findet man auch den Port und die Datenbankbezeichnung. Dann gibt man noch den User und Passwort an, der später genutzt werden soll an.

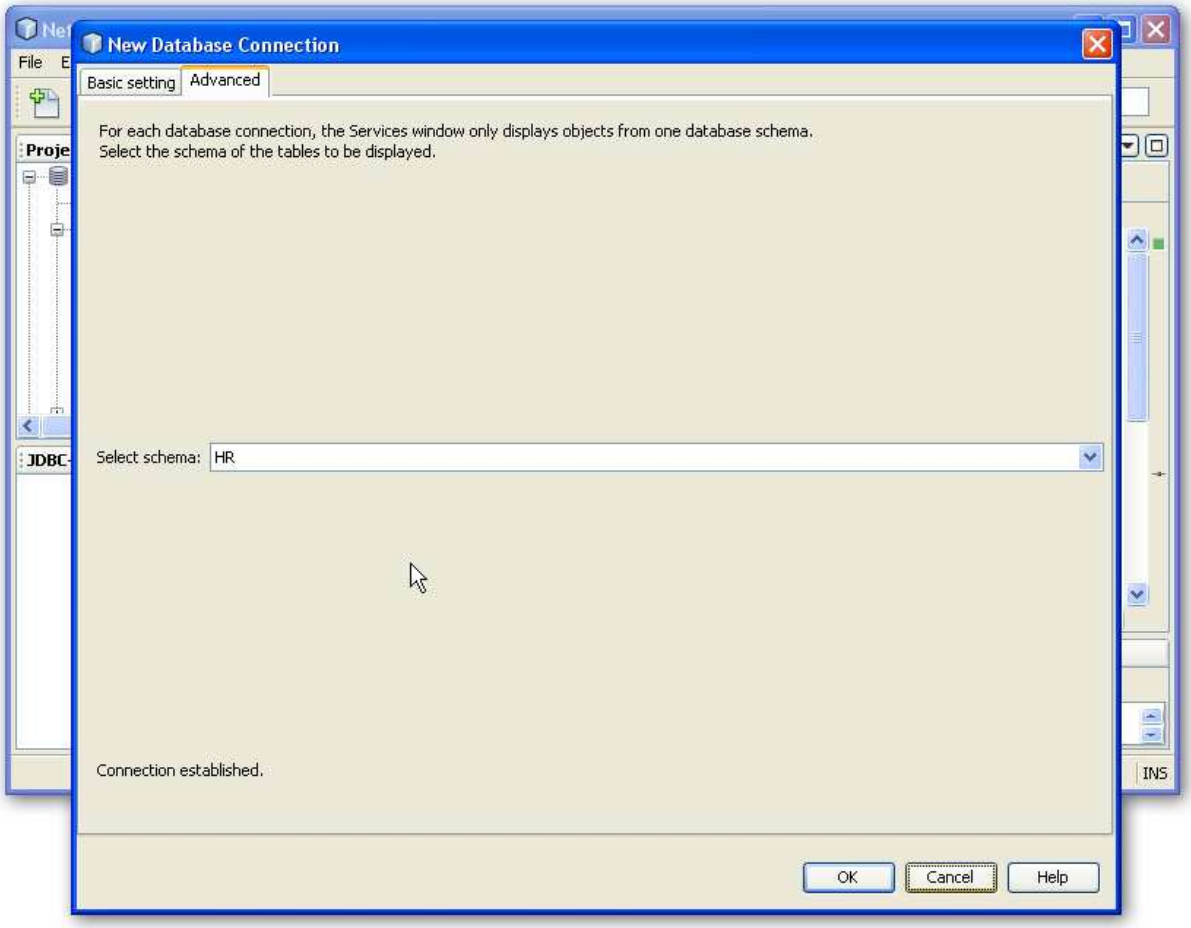
Wenn ein Haken in Show JDBC URL ist, dann wird die Verbindungsadresse angezeigt die später noch einmal gebraucht wird. Anfänger können sich diesen Link auch in eine Textdatei zum Beispiel zwischenspeichern.


New Database Connection
Basic setting
Advanced
Data Input Mode:
Field Entry
Direct URL Entry
Driver Name:
Java DB (Network)
Host:
Port:
Database:
User Name:
Password:
Display Name (Optional):
Additional Props:
Remember password
(see help for information on security risks)
Show JDBC URL
Please specify a value for the required field Database:
Java DB (Network)
JDBC-ODBC Bridge
MySQL (Connector/J driver)
Oracle Thin (with Service ID (SID))
Oracle Thin (with Service Name)
Oracle Thin (with TNS Name
PostgreSQL
New Driver...
JDBCDriver[name='Oracle Thin',displayName='Oracle Thin',className='oracle.jdbc.OracleDriver',urls=[file:/G:/oracle/product/10.2.0/db_1/jdbc/lib/ojdbc14.jar]]
OK
Cancel
Help



Wenn alle Angaben gemacht wurden, diese mit Ok bestätigen. Dann wird versucht eine Verbindung zur Datenbank mit dem angegebenen User zu erstellen. Sollte denn ein Fehler auftreten wie in der folgenden Abbildung, sollten die Dienste überprüft werden.


New Database Connection
Basic setting
Advanced
Data Input Mode:
Field Entry
Direct URL Entry
Driver Name:
Oracle Thin (with Service ID (SID))
Host:
host
Port:
1521
Service ID (
User Name:
Password:
Display Nam
Additional P
Show JDBC URL
jdbc:oracle:thin:@host:1521:Datenbank
OK
Cancel
Help
Error
Unable to add connection. Cannot establish a connection to jdbc:oracle:thin:@host:1521:Datenbank using oracle.jdbc.OracleDriver (E/A-Exception: The Network Adapter could not establish the connection).
OK



Ist der Verbindungsaufbau erfolgreich wird das Schema angezeigt und mit der Aussage Connection established dies auch noch einmal bestätigt. Mit OK das dann in Kenntnis nehmen.

JSP –Programmierung

Nachdem nun die notwendigen Voreinstellungen getätigt wurden und Netbeans auch schon erfolgreich eine Verbindung aufbauen konnte , soll nun die JSP “lernen“ mit der Datenbank zu reden.

Damit man später einfacher wie im normalen HTML oder XML auf einfache Tags zugreifen kann, implementiert man am Anfang erst einmal zwei taglib

<%@ taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>

Damit man dies so einfach tun kann, wird durch die am Anfang eingebundene JSTL 1.1 Library realisiert.

Im späteren Quelltextverlauf können nun mit dem sql – Tag einfach alle für die Kommunikation mit der Datenbank genutzt werden. Während mit dem c-Tag dann die Auswertung genutzt wird.

Danach folgt die Verbindung mit der Datenbank. Diese wird mit dem sql-Tag realisiert und zwar mit der Property setDataSource. Als Attribute werden der verwendete Treiber, der datenbankspezifische Link und die Userdaten.

<sql:setDataSource driver="oracle.jdbc.driver.OracleDriver"
url="jdbc:oracle:thin:@host:port:Datenbankname" user="hr" password="hr" />

Der HR – User ist in dem Fall ein User mit normalen Rechten. Ein Einlogen als Sysdba oder Sysoper kann man realisieren, indem man das Userattribut mit der entsprechenden Rolle erweitert, d.h. zum Beispiel der User Sys ,der als Sysdba auf die Datenbank connecten möchte

user=“ Sys as sysdba“

Damit ist dann die Verbindung hergestellt.

Das wichtigste ist damit also schon einmal erledigt.

Datenabfragen und Manipulation über die JSP

Folgend noch eine kleine Auflistung der Befehle die zur Manipulation der Datenbank genutzt werden. Die Befehle folgen dem SQL Standard zum Erstellen, Ändern und Löschen von Tabellen und Zeilen.

Eine Sicherheit der Daten beim Löschen und Updaten muss noch mit geeigneten Mitteln herbeigeführt werden, d.h. bei den Beispielen wird nicht drauf geachtet das Constraint Verletzungen auftreten können, da auf Constraints verzichtet wurden.

Die einzelnen Werte können auch über Variablen ermittelt werden.

Erstellen von Tabellen

```
<sql:update var="createTable">
        create table mit ( id number, name varchar(50))
 </sql:update>
```

Abfragen von Tabellen

```
<sql:query var="selectTable" >
        SELECT * FROM mit
</sql:query>
```

Einfügen von Zeilen

```
<sql:transaction>
    <sql:update var="updateCount">
         INSERT INTO mit VALUES (1,'John Doe')
    </sql:update>
    <sql:update var="updateCount">
         INSERT INTO mit Values(2,'Jane Doe')
    </sql:update>
</sql:transaction>
```

Updaten von Zeilen

```
<sql:update var="updateCount" >
        UPDATE mit SET name=? <sql:param value="Scott Tiger"/> WHERE id=1
 </sql:update>
```

Löschen von Zeilen

```
<sql:update var="updateCount" >
        DELETE FROM mit WHERE id=2
 </sql:update>
```

Löschen von Tabellen

```
<sql:update var="newTable" >
        drop table mit
</sql:update>
```

Ausgabe der Daten in der JSP

Dieses Beispiel gibt alle Daten aus die in einer „select *“- Abfrage ermittelt wurden.

```
<table border="1">
<%-- Auslesen der Tabellenspalten --%>
    <c:forEach var="columnName" items="${selectTable.columnNames}">
        <th><c:out value="${columnName}"/></th>
    </c:forEach>
 <%-- Inhalt der Zeilen --%>
    <c:forEach var="row" items="${selectTable.rows}">
        <tr>
            <c:forEach var="column" items="${row}">
                <td>
                    <c:out value="${column.value}"/>
                </td>
            </c:forEach>
        </tr>
    </c:forEach>
</table>
```

Dieses Beispiel gibt die Daten ausgewählter Spalten aus die bei einer Select – Anweisung ermittelt wurden.

```
<table border="1">
<%-- Auslesen der Tabellenspalten --%>
    <c:forEach var="columnName" items="${selectTable.columnNames}">
        <th><c:out value="${columnName}"/></th>
    </c:forEach>
 <%-- Inhalt der Zeilen --%>
    <c:forEach var="row" items="${myvar.rows}">
        <tr>
            <td>
                <c:out value="${row.first_name}"/>
            </td>
            <td>
                <c:out value="${row.job_id}"/>
            </td>
    </c:forEach>
</table>
```