

Inhaltsverzeichnis:

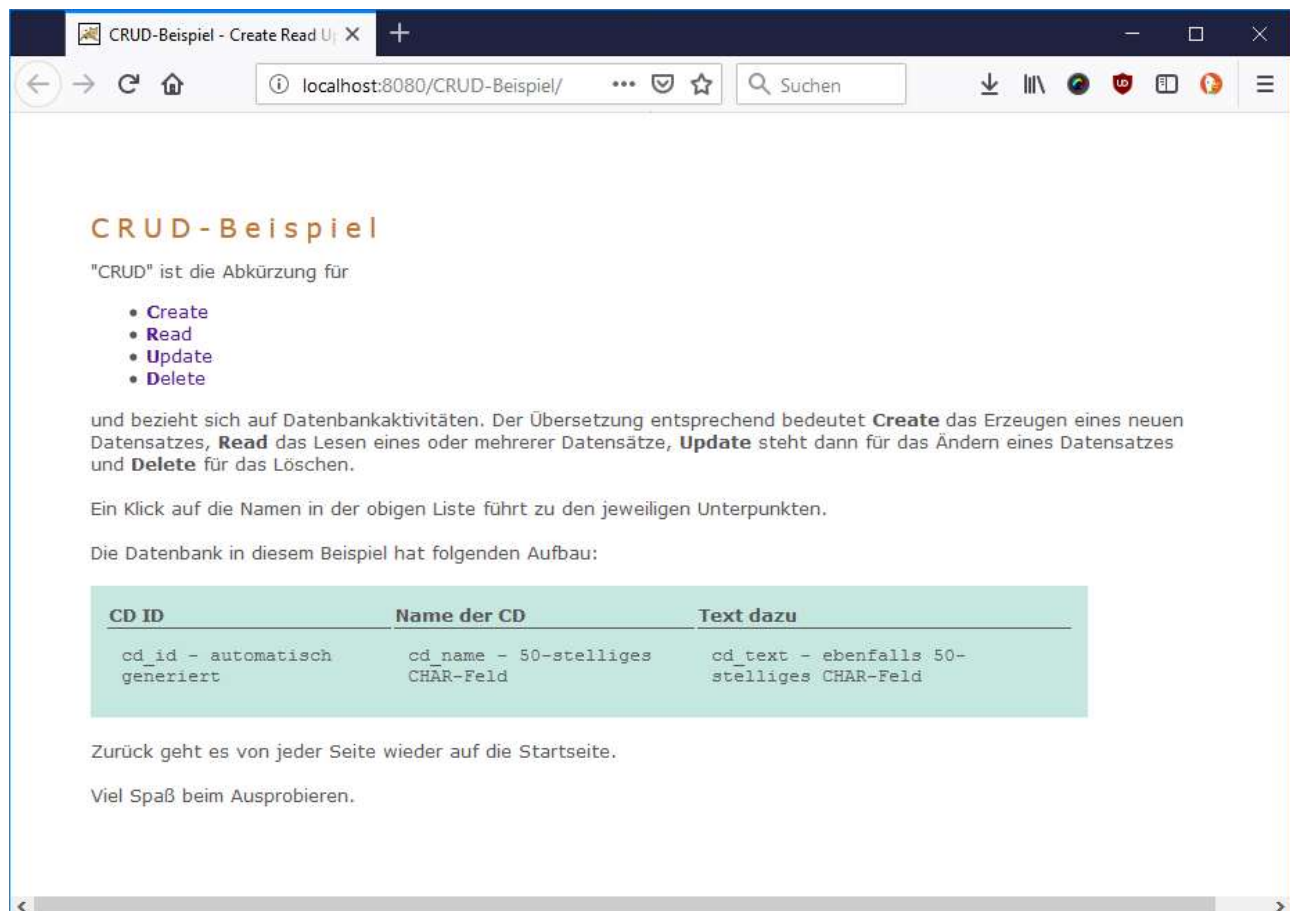
1	Vorwort	2
2	Vorbereitung	3
3	Datenbank einrichten	4
4	Neues Projekt anlegen	10
5	Einstieg, index.html und style.css	12
5.1	index.html - Startseite der Anwendung	12
5.2	style.css – Stylesheet	15
6	Create – Dateien	18
6.1	C1.html – Einstieg in Create	18
6.2	C2.jsp – Speichern des neuen Datensatzes	21
6.3	C3.jsp – Kontrolle, Anzeige aller bisher gespeicherten Datensätze	23
7	Read – Dateien	26
7.1	R1.html – Einstieg in Read	26
7.2	R2.jsp – Lesen aller gespeicherten Datensätze	27
7.3	R3.jsp – Lesen aller gespeicherten Datensätze – Sortierung geändert	28
8	Update – Dateien	31
8.1	U1.html – Einstieg in Update	31
8.2	U2.jsp – Auswahl eines Satzes zur Änderung	33
8.3	U3.jsp – Anzeige aller Attribute eines Satzes	35
8.4	U4.jsp – Aufnahme der Änderungen zu einem Satz	37
8.5	U5.jsp – Übernahme der Änderungen in die Datenbank	41
8.6	U6.jsp – Kontrolle der Übernahme	43
9	Delete – Dateien	45
9.1	D1.html – Einstieg in Update	45
9.2	D2.jsp – Auswahl eines Satzes zur Löschung	46
9.3	D3.jsp – Löschung des Satzes durchführen	48
9.4	D4.jsp – Kontrolle der Löschung	50
10	Abschluss	52

1 Vorwort

Auf der Suche nach einer funktionierenden CRUD-Anwendung (Create, Read, Update, Delete) bin ich über viele Beispiele und Code-Schnipsel gestolpert, die entweder nicht oder nur unvollständig funktioniert haben, die nicht so erklärt waren dass ich es verstehen konnte, oder nicht meinem Bedürfnis entsprochen haben.

Aus diesem Grund habe ich mich daran gemacht, eine vollständige CRUD-Anwendung mit einer 2-spaltigen MySQL-Tabelle zu erstellen, um die Grundfunktionalitäten zunächst erst einmal selbst zu verstehen und dann einen Prototypen zu haben, an dem ich die weiteren Schritte ausprobieren kann. Später soll daraus ein Archiv für Klassik-CDs werden.

In diesem Dokument werde ich mit Prosa und Screenshots hinterlegen, wie ich vorgegangen bin. Sollte dies jemand lesen und Anregungen für eine Überarbeitung haben, bitte gerne per Mail an mich. Auch wenn ich helfen konnte würde ich mich über eine Rückmeldung freuen.



2 Vorbereitung

Grundsätzliche Überlegungen vor dem Start der Entwicklung waren

- es soll nur Freeware oder Open Source genutzt werden
- es muss eine Datenbank angebunden werden
- die Entwicklung findet in *einer* IDE statt
- Die GUI-Programmierung ist zunächst nebensächlich, eine Oberfläche zur Erfassung wird aber benötigt
- für mich als Nicht-Java-Programmierer muss die Sprache leicht erlernbar sein, sollte aber auch allgemein anerkannt sein

So bin ich nach langem Probieren bei folgender Konstellation gelandet:

- IDE ist NetBeans in der Version 7.4, integriert ist der GlassFish-Server 4.0, ich habe mir das Bundle Java EE heruntergeladen
- Die Datenbank ist eine MySQL DB. Da die Erstellung und Anbindung in der IDE sehr einfach ist, habe ich auf das Admin-Tool verzichtet
- Als Browser habe ich Firefox im Einsatz
- Für die Zwischenspeicherung von Code-Beispielen oder sonstiger Textfragmente nehme ich Notepad++
- Programmiert wird in HTML zur Darstellung, JSP für die Kommunikation mit dem Server und innerhalb der JSP-Files mit Java/JavaScript. Keine Angst, ist nicht schlimm...

Die Installation und Inbetriebnahme der NetBeans IDE ist hier nicht erläutert, dazu ist genug Info im Netz vorhanden. Auch zur Anbindung der MySQL-Datenbank ist im Netz prima erklärt.

3 Datenbank einrichten

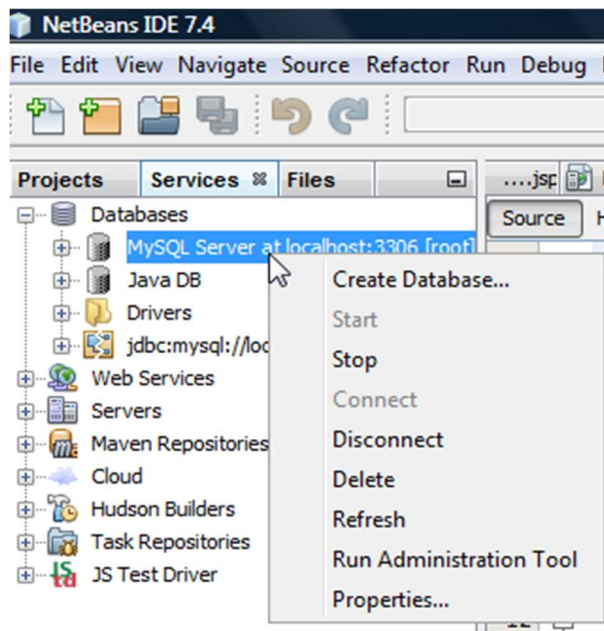
Zunächst kümmern wir uns um die Einrichtung der Datenbank. Ich gehe davon aus, dass dem geneigten Leser die Arbeitsweise einer Datenbank und der Zugriff darauf im Prinzip bekannt ist. Falls nicht, sollte sich darüber im Netz schlau gemacht werden, dort gibt es viele Tutorials und Beispiele, wie und warum Datenbanken und so weiter. An dieser Stelle nur soviel, es gibt ganz unterschiedliche Datenbankformen, die in diesem Beispiel gewählt ist eine relationale Datenbank. Sie ist im Prinzip mit einer Tabelle wie der folgenden zu vergleichen:

CD ID	Name der CD	Text dazu
1	Mozart Symphonien 38 + 39	Schöne Aufnahme
2	Beethoven Eroika	2. Satz zu schnell

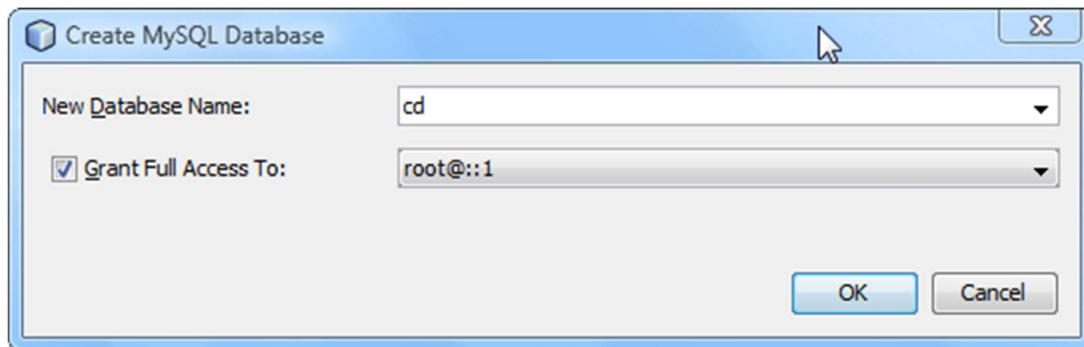
Die Datenbank heißt bei uns cd, die konkrete Tabelle darin cdmeta.

Um sich in der Tabelle einer Datenbank zu bewegen, ist ein Schlüssel zu vergeben, das ist ein Ordnungskriterium, das eindeutig auf eine Zeile in der Tabelle zeigt. In meinem Beispiel ist das die CD ID in der Tabelle. Daneben sind nur 2 Felder enthalten, der Name der CD und ein Freitext dazu.

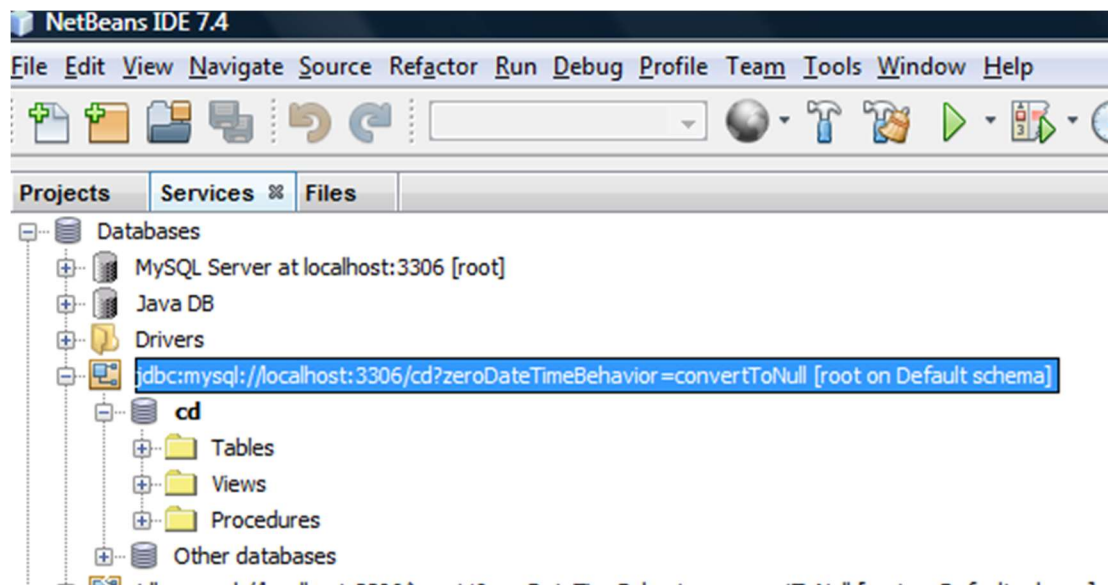
In NetBeans müssen wir vor der Anlage einer Tabelle zunächst eine Datenbank anlegen. Im Fenster Services (Strg+5) ist der erste Topic „Databases“, dann Rechtsklick auf „MySQL Server...“, „Create Database“:



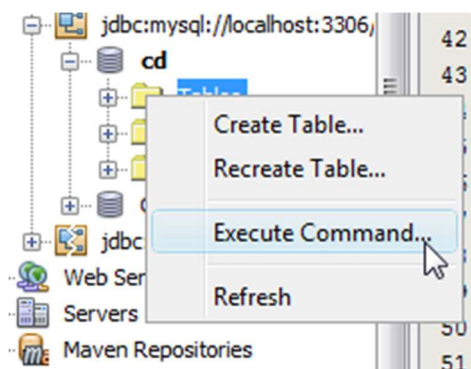
Wir nennen die Datenbank schlicht „cd“:



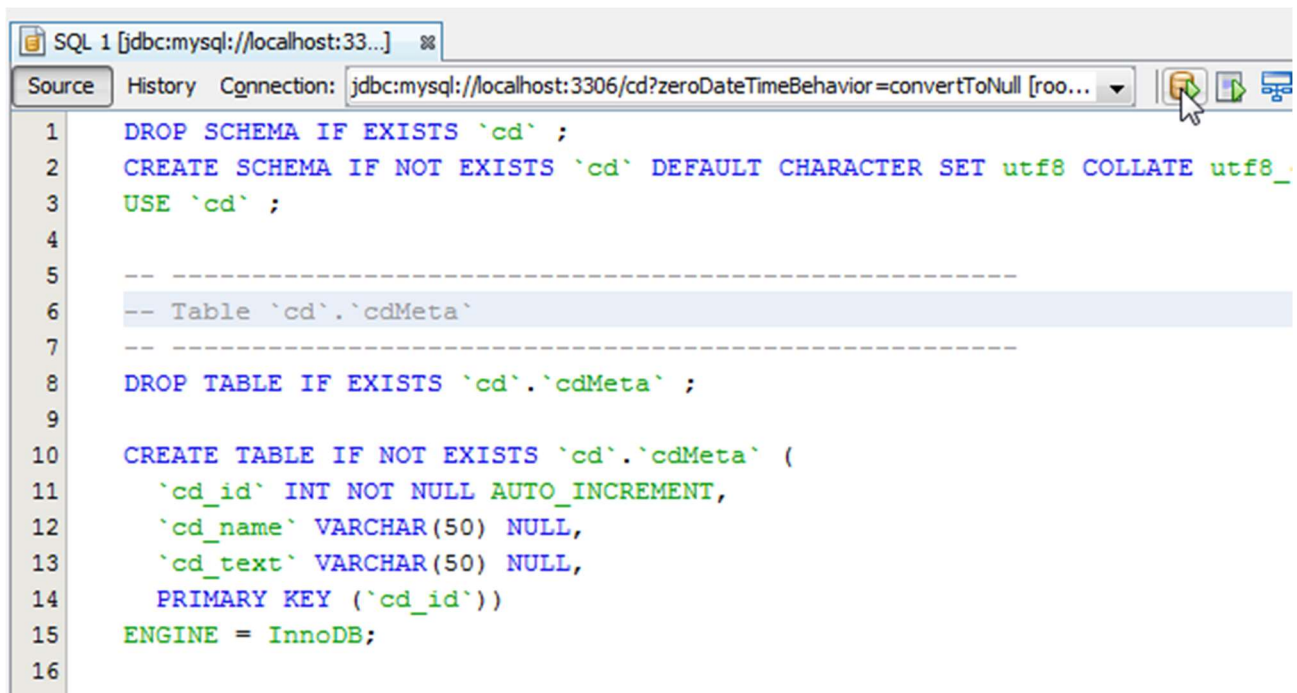
Nach Klick auf „OK“ wird unterhalb von Database automatisch eine Verbindung aufgebaut:



Der Reiter Tables ist noch leer, das wollen wir jetzt ändern, indem wir mit Rechtsklick und dann weiter mit „Execute Command“ den Erstellungs-Befehl (Create) absetzen können:



Im rechten Fenster geht ein Dokument SQL 1 auf:



The screenshot shows a SQL IDE window titled "SQL 1 [jdbc:mysql://localhost:33...]" with a toolbar on the right. The "Source" tab is active, displaying a SQL script. The script includes comments for table creation and the actual CREATE TABLE statement for a table named 'cdMeta' in the 'cd' schema. The table has three columns: 'cd_id' (INT, NOT NULL, AUTO_INCREMENT), 'cd_name' (VARCHAR(50), NULL), and 'cd_text' (VARCHAR(50), NULL). The primary key is set to 'cd_id' and the engine is InnoDB.

```

1  DROP SCHEMA IF EXISTS `cd` ;
2  CREATE SCHEMA IF NOT EXISTS `cd` DEFAULT CHARACTER SET utf8 COLLATE utf8_
3  USE `cd` ;
4
5  -----
6  -- Table `cd`.`cdMeta`
7  -----
8  DROP TABLE IF EXISTS `cd`.`cdMeta` ;
9
10 CREATE TABLE IF NOT EXISTS `cd`.`cdMeta` (
11     `cd_id` INT NOT NULL AUTO_INCREMENT,
12     `cd_name` VARCHAR(50) NULL,
13     `cd_text` VARCHAR(50) NULL,
14     PRIMARY KEY (`cd_id`))
15 ENGINE = InnoDB;
16

```

Der Create-Befehl der Tabelle lautet:

```

DROP SCHEMA IF EXISTS `cd` ;
CREATE SCHEMA IF NOT EXISTS `cd` DEFAULT CHARACTER SET utf8 COLLATE
utf8_general_ci ;
USE `cd` ;

-----
-- Table `cd`.`cdmeta`
-----

DROP TABLE IF EXISTS `cd`.`cdmeta` ;

CREATE TABLE IF NOT EXISTS `cd`.`cdmeta` (
    `cd_id` INT NOT NULL AUTO_INCREMENT,
    `cd_name` VARCHAR(50) NULL,
    `cd_text` VARCHAR(50) NULL,
    PRIMARY KEY (`cd_id`))
ENGINE = InnoDB;

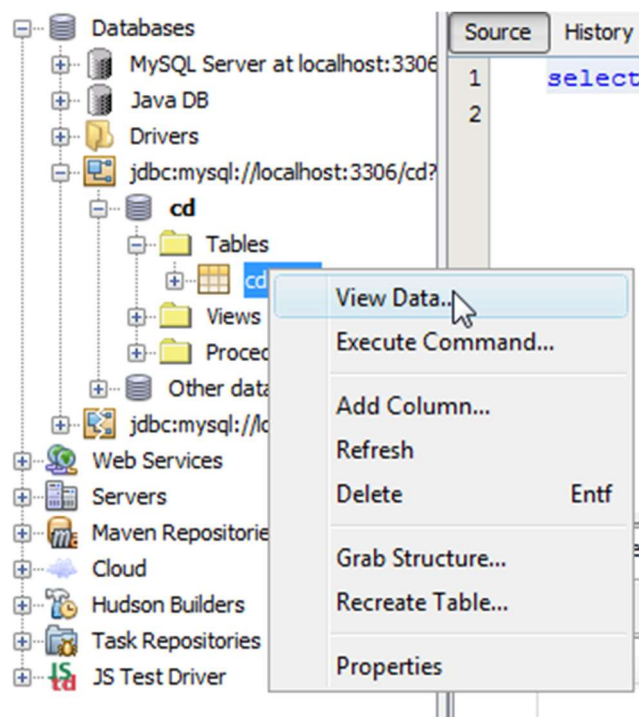
```

Mit dem Klick auf das Datenbanksymbol mit dem grünen Pfeil (oben rechts) wird der SQL-Befehl ausgeführt. Mit dem Zusatz AUTO_INCREMENT im Key „cd_id“ lasse ich den Primär-Schlüssel automatisch generieren. Das hat den Vorteil, dass ich mich um doppelte Schlüssel – das würde die Speicherung in der Tabelle verhindern – nicht kümmern muss.

Nochmal zum Verständnis die Datenbank ist das oberste Kriterium und heißt in unserem Fall „cd“, die in der Datenbank befindliche Tabelle heißt „cdmeta“. Das ist wichtig zu verstehen, die Referenzierung auf die Tabelle erfolgt mit „cd.cdmeta“. Sollte es in der Datenbank später eine weitere Tabelle geben, sagen wir „komponist“, heißt die Referenzierung der neuen Tabelle dementsprechend „cd.komponist“.

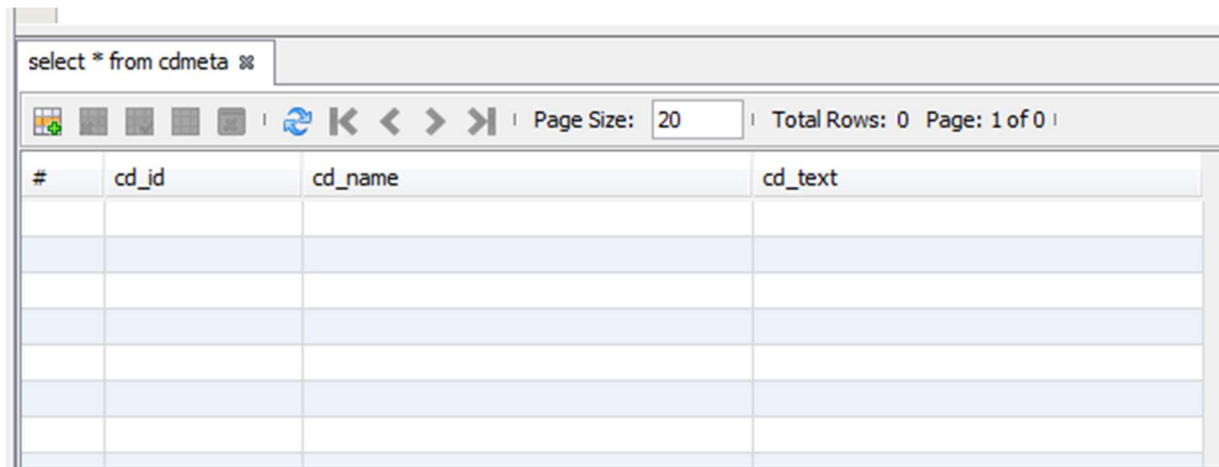
Die Bestätigung erhält man im Output-Fenster (Strg+4):

Für die Anzeige der Tabelle kann man einen Rechtsklick auf die Tabelle machen und dann die Auswahl „View Data...“ wählen:



In den beiden Fenstern auf der rechten Seite erscheint oben der SQL-Befehl:

Unterhalb des Fensters ist die Ergebnistabelle abgebildet, die momentan noch keine Daten enthält:



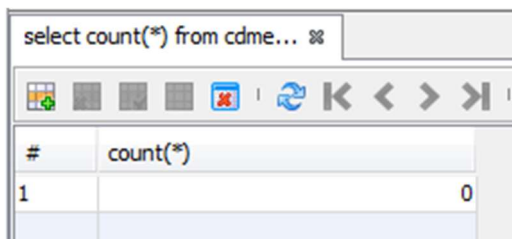
The screenshot shows a SQL editor window with the query `select * from cdmeta` in the top text area. Below the query is a toolbar with icons for saving, undo, redo, and navigation. To the right of the toolbar, it displays `Page Size: 20`, `Total Rows: 0`, and `Page: 1 of 0`. Below the toolbar is a table with four columns: `#`, `cd_id`, `cd_name`, and `cd_text`. The table is currently empty, showing only the header row.

#	cd_id	cd_name	cd_text
---	-------	---------	---------

Durch Überschreiben des SQL-Statements z.B. mit

```
select count(*) from cdmeta;
```

erhalten wir nach Ausführung des SQL die Ergebnistabelle:



The screenshot shows a SQL editor window with the query `select count(*) from cdmeta` in the top text area. Below the query is a toolbar with icons for saving, undo, redo, and navigation. Below the toolbar is a table with two columns: `#` and `count(*)`. The table contains one row with the value `0` in the `count(*)` column.

#	count(*)
1	0

Soll bedeuten, es gibt 0 Einträge in der Tabelle – klar, wir haben ja auch noch nichts eingefügt.

Um nun unsere Beispielsätze einzugeben, können wir auf das CRUD-Beispiel warten und die Neuanlage testen, wir können aber auch den gleichen SQL-Editor nutzen.

Gibt man dort

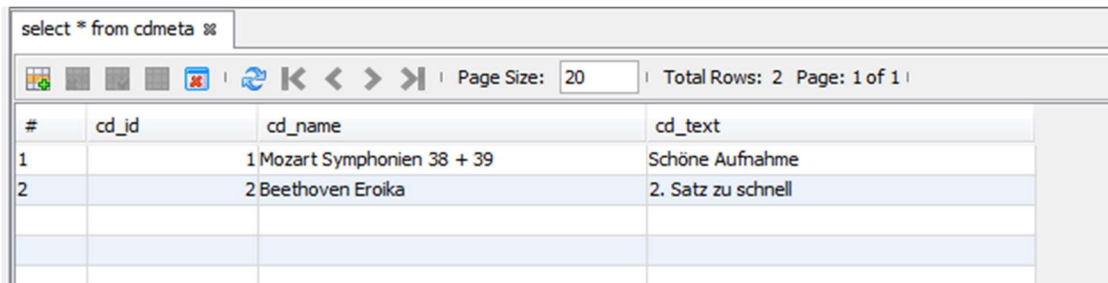
```
insert into cdmeta (cd_name, cd_text) values ('Mozart Symphonien 38 +  
39', 'Schöne Aufnahme');  
insert into cdmeta (cd_name, cd_text) values ('Beethoven Eroika', '2. Satz zu  
schnell');
```

ein, und lässt den SQL laufen, werden die beiden Sätze in die Datenbanktabelle `cdmeta` geschrieben. Dank des `AUTO_INCREMENTS` müssen wir uns um die Bestückung der „`cd_id`“ nicht kümmern!

Zur Kontrolle kann anschließend erneut das select-SQL

```
select * from cdmeta;
```

abgesetzt werden. Diesmal haben wir Daten und die werden auch wie folgt angezeigt:



#	cd_id	cd_name	cd_text
1		1 Mozart Symphonien 38 + 39	Schöne Aufnahme
2		2 Beethoven Eroika	2. Satz zu schnell

Prima – und nun auf zur HTML/JSP/JAVA-Programmierung.

Kleiner Exkurs:

Die Suche in der Datenbank mittels `select *` ist eigentlich keine gute Idee. Man sollte immer die Feldnamen der Tabelle angeben, die man gelesen haben möchte. In unserem Fall

```
select * from cdmeta;
```

sollte es also besser

```
select cd_id, cd_name, cd_text from cdmeta;
```

heißen. Ich habe mich absichtlich für den Stern („*“) entschieden, da diese Schreibweise übersichtlicher ist.

Hintergrund – sollte sich im Laufe der Weiterentwicklung eine neue Spalte ergeben, z.B. `cd_kaufdatum` würde sofort nach Neuerstellung der Tabelle der erste SQL das Feld schon mit ausgeben, obwohl vielleicht die Darstellung im Browser noch nicht geändert wurden. Das würde dann zum Fehler in der Anwendung führen. Ohne Änderung am zweiten SQL ist das Feld noch nicht enthalten, die Verarbeitung kann also theoretisch richtig laufen, ohne Fehler zu melden oder abzustürzen.

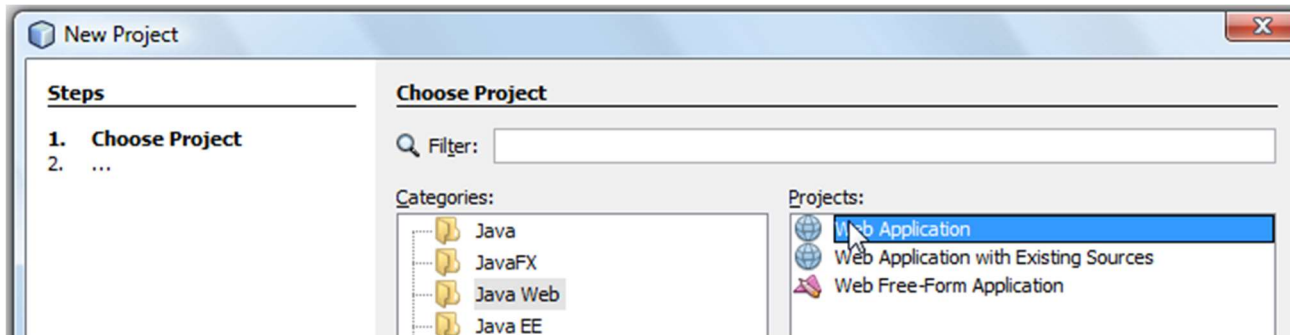
Das gilt natürlich nicht, wenn z.B. die Spalte `cd_text` gelöscht oder umbenannt wird, dann ist auch die Verarbeitung mit expliziten Feldnamen nicht mehr möglich.

Jeder sollte das für sich selbst entscheiden, beide Wege haben Vor- und Nachteile.

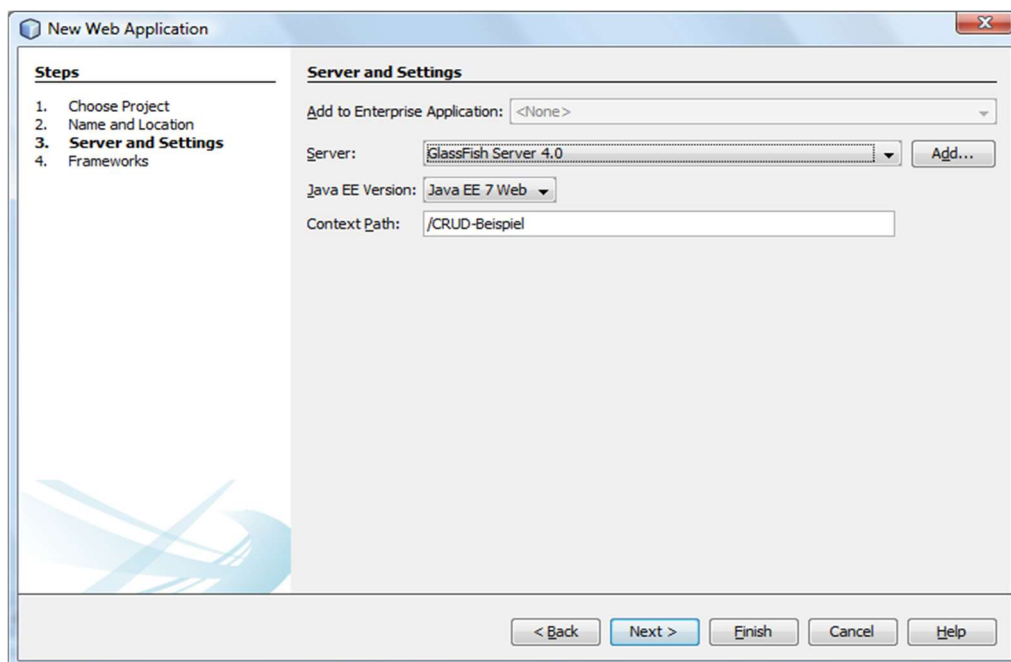
4 Neues Projekt anlegen

Wir wechseln in die NetBeans-IDE.

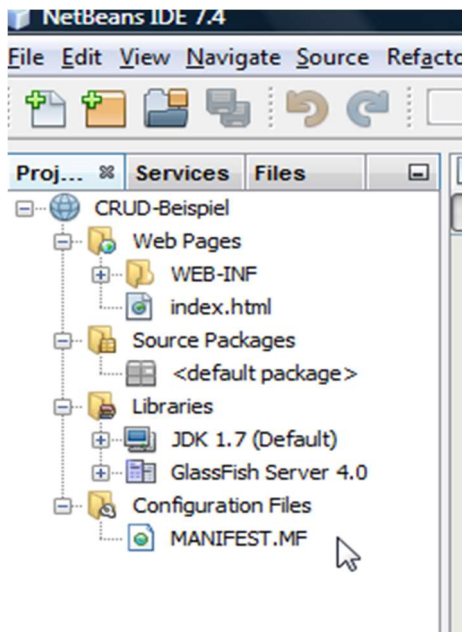
Mit Strg+Umschalt+N öffnet sich der Assistent zur Anlage eines neuen Projektes. Ich habe mich für Java Web und dann Java Application entschieden:



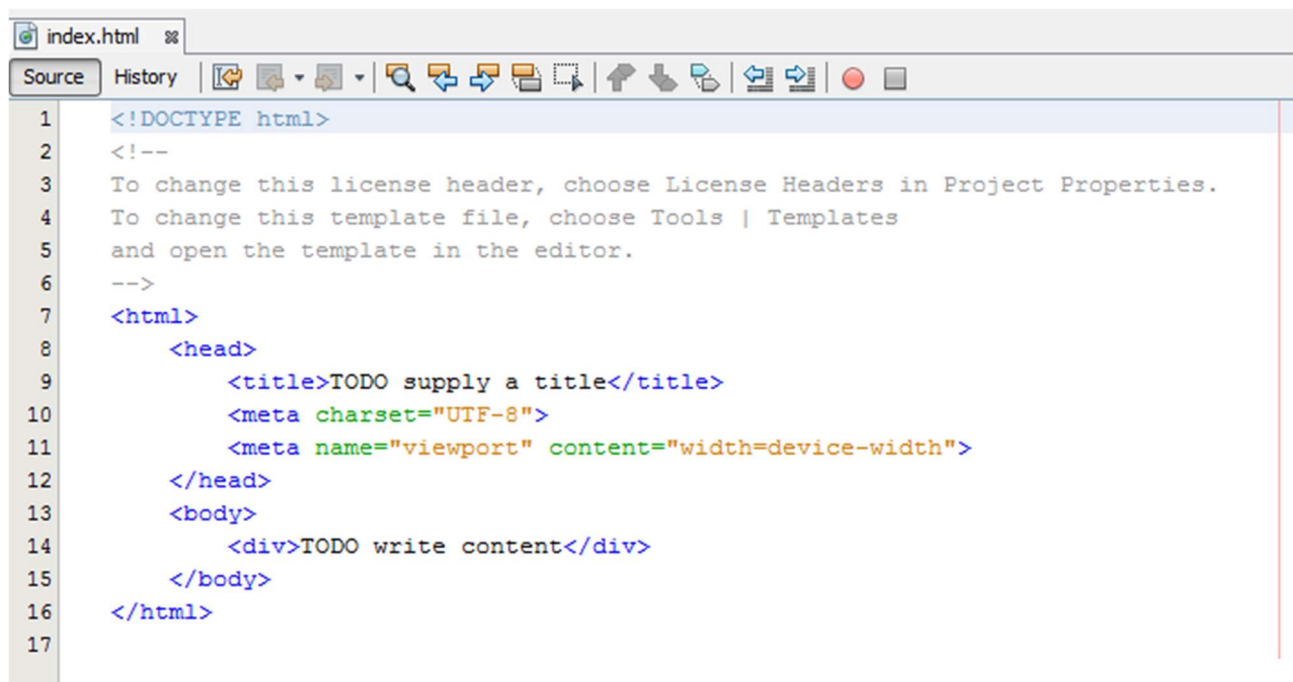
Mit „Next“ weiter, dann Namen auswählen, bei mir heißt es „CRUD-Beispiel“. In der Folgemaske den GlassFish oder einen anderen installierten Server anwählen:



Danach auf „Finish“ klicken. Die IDE legt den Pfad wie nachfolgend selbständig an:



Im Arbeitsfenster rechts daneben, sollte bereits die Datei index.html geöffnet sein:



5 Einstieg, index.html und style.css

Nun gibt es mehrere Wege. In dem von mir erstellten Beispiel brauchen wir 17 Dateien, die alle angelegt werden wollen. Der einfachste Weg ist, alle Dateien im Downloadbereich herunterzuladen und in die IDE zu kopieren.

Da die Einrichtung in der IDE aber 'per Hand' gemacht werden muss, bringt das nicht wirklich viel. Alternativ gehe ich in den nächsten Kapiteln auf jede Datei kurz ein, der Quelltext kann dann gleich kopiert werden.

CRUD steht – wie auf der Homepage schon erwähnt – für **C**reate **R**ead **U**ppdate und **D**eleate, die verschiedenen Datenbankoperationen.

Dementsprechend beginnen die Dateien auch mit „C“ oder „R“, die Nummerierung entspricht der Sequenz in der sie hintereinander aufgerufen werden, Also beispielsweise kommt „C1“ vor „C2“.

5.1 index.html - Startseite der Anwendung

Quelltext:

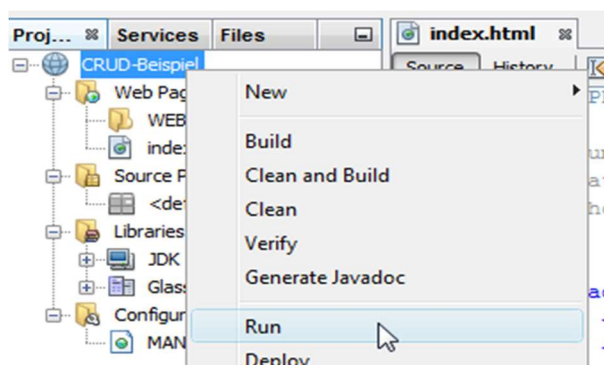
```
<!DOCTYPE html>
<!--
    Document      : index.html
    Created on    : 11.01.2014, 07:32:24
    Author       : papa
-->
<html>
  <head>
    <title>CRUD-Beispiel - Create Read Update Delete eine einfache
Datenbank</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width">
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>
    <h1>CRUD-Beispiel</h1>
    <div>&quot;CRUD&quot; ist die Abk&uuml;rzung f&uuml;r</div>
    <ul>
      <li><a href="C1.html"><b>C</b>reate</a></li>
      <li><a href="R1.html"><b>R</b>ead</a></li>
      <li><a href="U1.html"><b>U</b>pdate</a></li>
      <li><a href="D1.html"><b>D</b>elete</a></li>
    </ul>
    <div>und bezieht sich auf Datenbankaktivit&uuml;ten. Der
&Uuml;bersetzung entsprechend bedeutet
      <b>Create </b>das Erzeugen eines neuen Satzes, <b>Read</b>
das Lesen eines oder
      mehrerer Datens&uuml;tze, <b>Update</b> steht dann f&uuml;r
das &Auml;ndern eines
      Satzes und <b>Delete</b> f&uuml;r das L&ouml;schen.</div>
    <br>
    <div>Ein Klick auf die Namen in der obigen Liste f&uuml;hrt zu
```

```
den jeweiligen Unterpunkten.</div>
<br>
<div>Die Datenbank in diesem Beispiel hat folgenden Aufbau:</div>
<br>
<table>
  <tr>
    <th>CD ID</th>
    <th>Name der CD</th>
    <th>Text dazu</th>
  </tr>
  <tr>
    <td>cd_id - automatisch generiert</td>
    <td>cd_name - 50-stelliges CHAR-Feld</td>
    <td>cd_text - ebenfalls 50-stelliges CHAR-Feld</td>
  </tr>
</table>
<br>
<div>Zurück geht es von jeder Seite wieder auf die
Startseite.</div>
<br>
<div>Viel Spaß; beim Ausprobieren.</div>
</body>
</html>
```

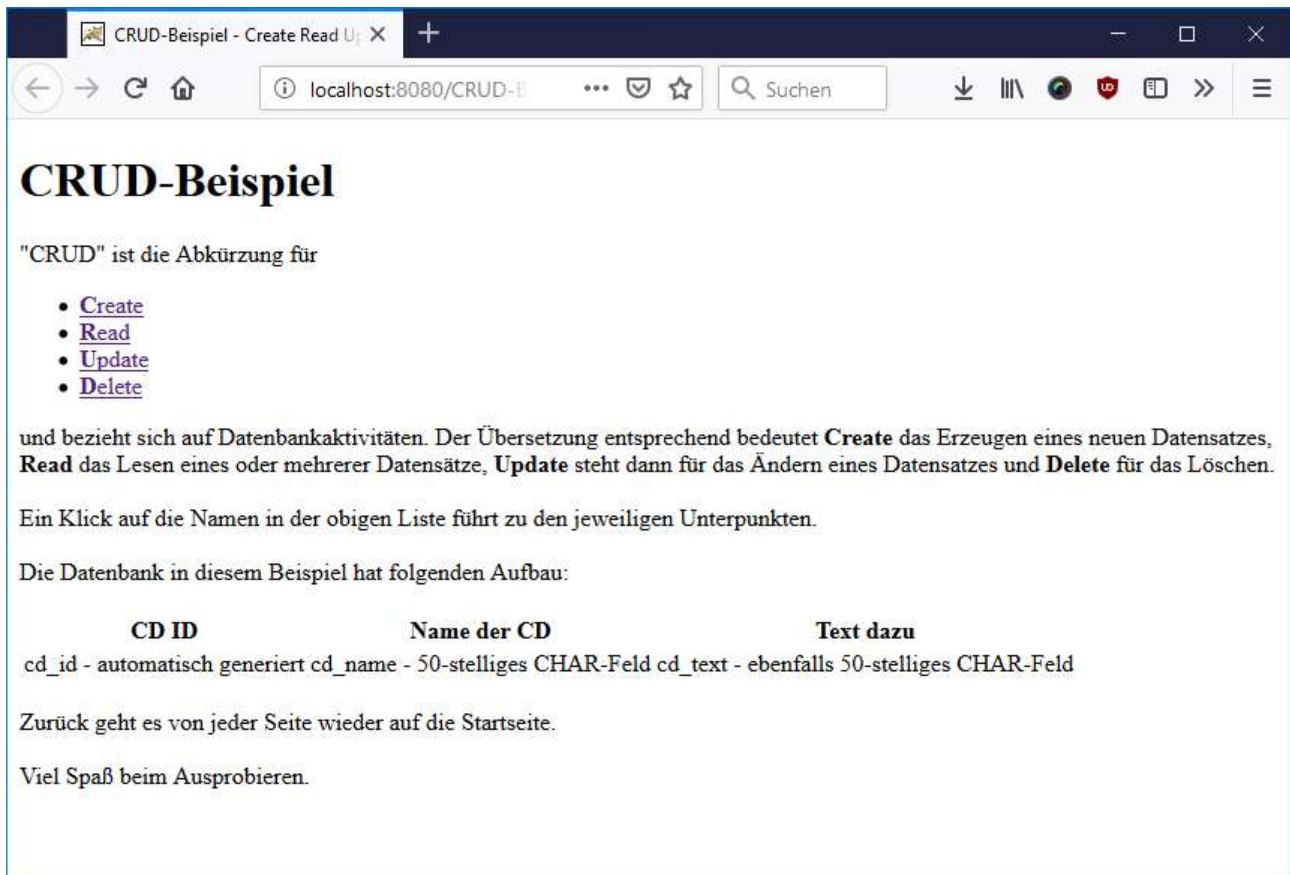
Sobald der Text in das Dokument kopiert wurde, kann auch schon getestet werden. Dazu zuerst die neue Datei durch Klick auf das Diskettensymbol oben links in der IDE (alternativ Speichern aller Dateien mit Strg + Umschalt + S) speichern – das ist in den Folgedateien nicht mehr explizit beschrieben und wird vorausgesetzt.

In der Datei werden 4 Dokumente verlinkt, C1.html, R1.html, U1.html und D1.html. Das sind jeweils die ersten Einstiegsseiten für Create, Read, Update und Delete wie oben beschrieben.

Rechtsklick auf das Projekt und Auswahl Run startet das Projekt:



Wenn alles richtig gemacht wurde sollte im bevorzugten Browser das nachfolgende Bild erscheinen:

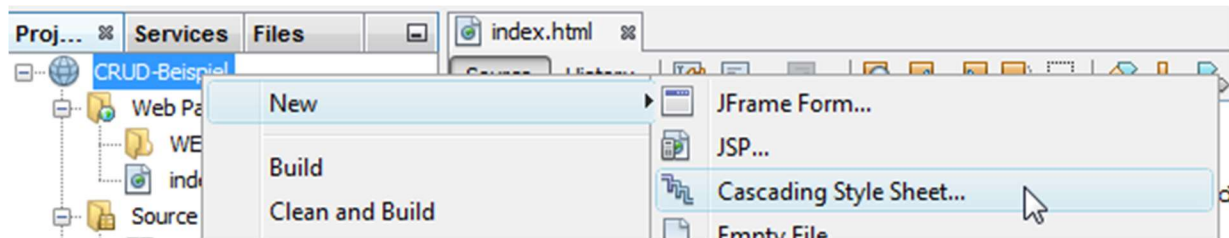


Die Darstellung gefällt mir noch nicht gut, wir werden daher ein stylesheet anlegen, in dem die projektweit geltenden Einstellungen für die html-Darstellung beschrieben sind.

Die Seite nicht schließen, einfach wieder in die IDE wechseln.

5.2 style.css – Stylesheet

Rechtstlick auf Projekt und Anlage CSS, Name ist „style“:



Quelltext:

```
/*
    Document    : style
    Created on   : 11.01.2014, 07:02:04
    Author      : papa
    Description  : Stylesheet
*/

root {
    display: block;
}
body {
    font-family: Verdana, Arial, sans-serif;
    font-size: smaller;
    padding: 50px;
    color: #555;
}

h1 {
    text-align: left;
    letter-spacing: 6px;
    font-size: 1.4em;
    color: #be7429;
    font-weight: normal;
    width: 900px;
}

table {
    width: 720px;
    padding: 10px;
    background-color: #c5e7e0;
    font-family: Courier, true-type;
}

th {
    text-align: left;
    border-bottom: 1px solid;
    font-family: Verdana, true-type;
}
```

```
td {
    padding: 10px;
}

a:link {
    color: #be7429;
    font-weight: normal;
    text-decoration: none;
}

a:link:hover {
    color: #be7429;
    font-weight: normal;
    text-decoration: underline;
}
```

Achtung:

In der Datei index.html – und in allen weiteren Dateien - müssen wir das Stylesheet erst bekannt machen. In der index.html müssen wir dazu die folgende

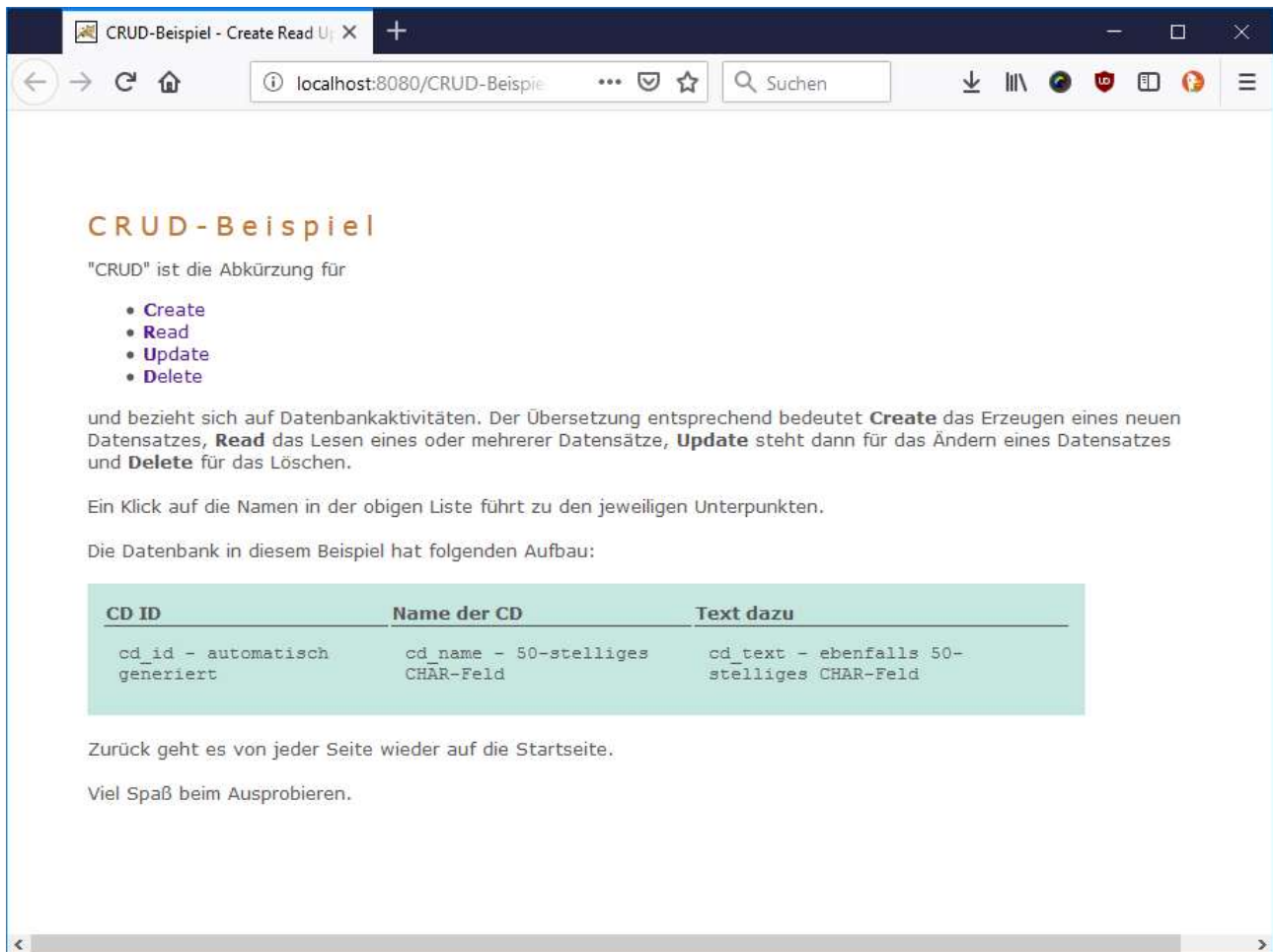
fettgedruckte, grüne Zeile hinzufügen:

```
<!DOCTYPE html>

<head>
    <title>CRUD-Beispiel - Create Read Update Delete eine einfache
Datenbank</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width">
    <link rel="stylesheet" type="text/css" href="style.css">
</head>
```

Danach beide Dateien Speichern (Strg + Umschalt + S). Jetzt wieder auf den Browser mit der geöffneten Startseite wechseln und das Fenster aktualisieren. Dazu reicht es, einfach die Funktionstaste F5 oberhalb der Tastatur zu drücken.

Das Fenster sieht jetzt so aus, oder?



Schon besser! Weitere Änderungen im Stylesheet habe ich nicht vorgenommen, hier kann sich jeder selbst verwirklichen.

Weiter mit der ersten Datei, C1.html.

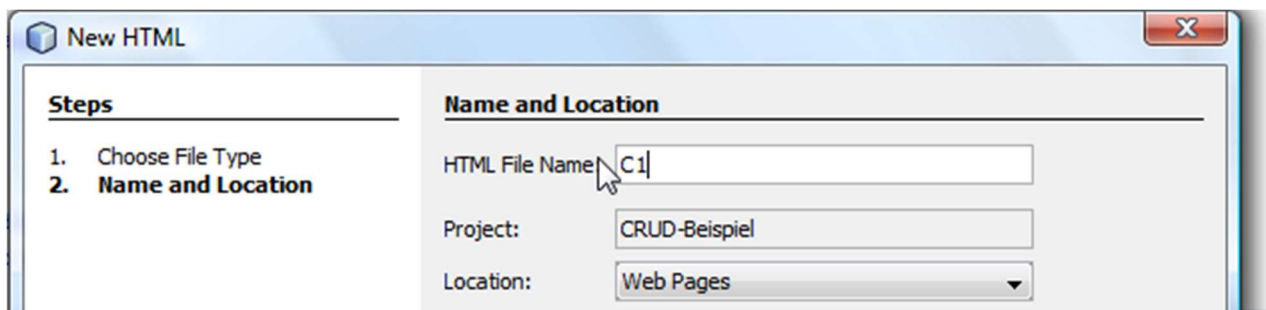
6 Create – Dateien

6.1 C1.html – Einstieg in Create

Da in den Einstiegsseiten noch keine Datenbankaktivitäten stattfinden, werden hier nur html-Seiten angelegt. Erst ab den Folgeseiten ist es notwendig, die Dateien unter JSP laufen zu lassen. Also, Rechtsklick auf Projekt und Anlage HTML, Name ist „C1“:



Das nennen wir jetzt C1 und bestätigen mit Finish:



In den nächsten Dateien habe ich die Kopien des Anlageprozesses weggelassen.

Vorschlag:

Damit frühzeitig getestet werden kann, können auf einen Schlag alle benötigten Dateien angelegt werden. Die IDE spendiert einen Standard-Text, damit sind die Links nicht funktionsunfähig.

Als Alternative kann man auch zunächst nur die Dateien C1, C2 und C3 anlegen.

Quelltext zu C1:

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-
8">
    <link rel="stylesheet" type="text/css" href="style.css">

    <title>CRUD-Beispiel - Create - Formular</title>
  </head>
  <body>
    <h1>Create - Einf&uuml;gen eines neuen Datensatzes</h1>
    <div>F&uuml;r das Einf&uuml;gen eines Datensatzes in eine
bestehende Datenbank
      ist zu beachten, dass zuvor eigentlich eine Pr&uuml;fung auf
die Existenz eines
      Datensatzes erfolgen sollte. Falls das unterlassen wird,
```

```
besteht die Gefahr, dass
    es zu Fehlermeldungen kommt, da ein Datensatz mit einem
eindeutigen Schlüssel
    nur einmal in der Datenbank abgelegt werden kann.</div>
<br>
<div>Mit dem Entschluss, den Primärschlüssel
automatisch durch die Verwaltungssysteme
    der Datenbank selbst generieren zu lassen, bin ich diesem
Problem aus dem Weg gegangen.</div>
<br>
<div>Um die Inhalte in die Datenbank zu bekommen, brauchen wir
zuerst ein Formular mit 2 Eingabefeldern
    für den Titel der CD und den zugehörigen
Text.</div>
<br>
<form action="C2.jsp" method="post" accept-charset="ISO-8859-1">
<p>Name der CD:<br>
<input name="cdName" size="40"></p>
<p>Text dazu:<br>
<input name="cdText" size="40"></p>
<p><input type="submit" value="Einfügen..."></p>
</form>
<br>
<a href="index.html">zurück zur Startseite...</a>
</body>
</html>
```

C1.jsp ist ein Formular, in dem die beiden Felder Name und Text eingebbar sind, sie werden dann an C2.jsp weitergereicht.

Prinzipiell kann nach jedem neuen Schritt getestet werden (Speichern der Datei und Aktualisieren des Browser). Teilweise funktionieren die Links dann nur nicht. Ich habe in einem Aufwasch auch C2 und C3 mit anlegen lassen.

Nach Speichern und Aktualisieren des Browsers (!), kann der Klick auf den ersten Hyperlink Create gemacht werden.

CRUD - Beispiel

"CRUD" ist die Abkürzung für

- Create
- Read
- Update
- Delete

und bezieht sich auf Datenbank

C1 sieht dann so aus:

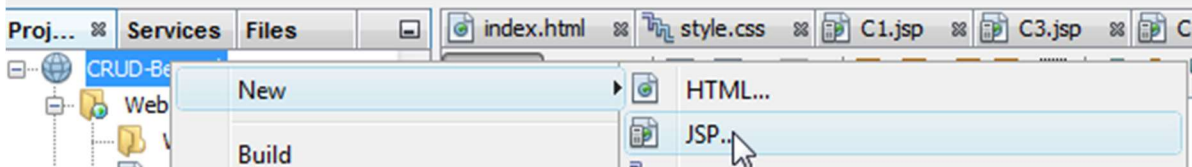


Das Einfügen funktioniert natürlich nicht, ist die Datei C2 mitgeneriert worden, erscheint „Hello World!“ sonst ein Fehlerhinweis 404 – Seite nicht gefunden.

Dementsprechend müssen wir uns zuerst um C2 kümmern.

6.2 C2.jsp – Speichern des neuen Datensatzes

C2.jsp ist ein JSP – das steht für Java Server Pages. Ein JSP wird so angelegt:



Das was aus dem Formular kommt, wird in die Datenbank übernommen.

Achtung:

Es findet keinerlei (!) Prüfung auf logische Inhalte statt, beide Felder können sowohl Müll enthalten als auch leer sein. Die Prüfung muss nachträglich selbst eingebaut werden. Mir war die prinzipielle Funktionsweise wichtiger als die Formalprüfung.

Quelltext zu C2.jsp:

```
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql"%>
<!--
    Document      : C2
    Created on    : 11.01.2014, 07:48:41
    Author       : papa
--%>

<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <link rel="stylesheet" type="text/css" href="style.css">
        <title>CRUD-Beispiel - Create - Best&auml;tigung des
Einf&uuml;gens</title>
    </head>
    <body>

        <sql:setDataSource var="Quelle" driver="com.mysql.jdbc.Driver"
            url="jdbc:mysql://localhost/cd"
            user="root"
            password="root"/>

        <sql:update dataSource="${Quelle}" var="updttable">
            insert into cdmata (cd_name, cd_text)
                values ('${param.cdName}', '${param.cdText}')
        </sql:update>

    <%@page contentType="text/html" pageEncoding="UTF-8"%>
    <h1>Best&auml;tigung des Einf&uuml;gens</h1>
    <div>Prima, die Speicherung des Datensatzes war erfolgreich!
        Das war nicht weiter verwunderlich, da keine Pr&uuml;fungen
auf die Datenfelder
```

```
        erfolgt sind. Hier sind Kreativität und Sorgfalt
gefragt.</div>
    <br>
    <div>&Uuml;ber nachfolgenden Link kann das Ergebnis betrachtet
werden:</div>
    <br>
    <form action="C3.jsp" method="post" accept-charset="ISO-8859-1">
    <input type="hidden" name="cdName" value="{param.cdName}">
    <p><input type="submit" value="Kontrolle..."></p>
    </form>
    <br>
    <a href="index.html">Zur&uuml;ck zur Startseite...</a>
</body>
</html>
```

Damit geprüft werden kann, dass der Insert geklappt hat, ist C3.jsp da. Ich habe C2 und C3 gemeinsam angelegt und getestet, deshalb die Screenshots erst im Nachgang.

6.3 C3.jsp – Kontrolle, Anzeige aller bisher gespeicherten Datensätze

In C3 wird der gerade gespeicherte Datensatz angezeigt.

Der Quellcode dazu sieht folgendermaßen aus:

```
<%@ page import="java.io.*,java.util.*,java.sql.*"%>
<%@ page import="javax.servlet.http.*,javax.servlet.*,javax.ejb.*" %>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql"%>

<%--
    Document      : C3
    Created on    : 11.01.2014, 08:32:40
    Author       : papa
--%>

<sql:setDataSource var="Quelle" driver="com.mysql.jdbc.Driver"
                  url="jdbc:mysql://localhost/cd"
                  user="root"
                  password="root"/>

<sql:query sql="select * from cdmeta where cd_name =
'${param.cdName}'" var="Ergebnis" dataSource="${Quelle}" >
</sql:query>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-
8">
        <link rel="stylesheet" type="text/css" href="style.css">
        <title>CRUD-Beispiel - Create - Kontrolle des
Einf&uuml;gens</title>
    </head>
    <body>
        <h1>Kontrolle des eben eingef&uuml;gten Datensatzes</h1>
        <table border="1" width="80%">
            <tr>
                <th>Nummer</th>
                <th>Name der CD</th>
                <th>Text dazu</th>
            </tr>
            <c:forEach var="row" items="${Ergebnis.rows}">
                <tr>
                    <td>
                        <c:out value="${row.cd_id}"/>
                    </td>
                    <td>
                        <c:out value="${row.cd_name}"/>
                    </td>
                </tr>
            </c:forEach>
        </table>
    </body>
</html>
```

```
<td>
    <c:out value="${row.cd_text}"/>
</td>
</tr>
</c:forEach>
</table>
<p><a href="index.html">Zur&uuml;ck zur Startseite...</a></p>
</body>
</html>
```

Screenshots zum Thema Create. C1:



C2 sieht dann so aus:



Drückt man den Kontrolle...- Button landet man in C3:



Mit dem Verweis auf die Startseite geht es wieder zu index.html. Das „C“ in CRUD haben wir verarztet, kommen wir zum „R“, dem Read.

7 Read – Dateien

7.1 R1.html – Einstieg in Read

R1 ist der Einstieg in die Anzeigen. Da hier auch noch keine Datenbankaktivitäten stattfinden, ist dies eine html-Datei. Anlegen also nicht als JSP... sondern HTML...

R1.html hat folgenden Code:

```
<!DOCTYPE html>
<!--
    Document      : R1.html
    Created on    : 11.01.2014, 07:32:24
    Author       : papa
-->
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-
8">
        <link rel="stylesheet" type="text/css" href="style.css">
        <title>CRUD-Beispiel - Read - Lesen Tabelle</title>
    </head>
    <body>
        <h1>Lesen aller Datensätze</h1>
        <div>Das Lesen von Daten kann beliebig komplex sein. Die
Steuerung was gelesen werden soll erfolgt
            in der Regel über das SQL-Statement, also die
Datenbankabfragesprache.</div>
        <br>
        <div>In diesem Beispiel habe ich mich darauf beschränkt,
eine Seite anzubieten, in der alle
            bisher gespeicherten Datensätze angezeigt werden und
eine, in der mit dem Namen etwas
            gespielt werden kann.</div>
        <p><a href="R2.jsp">Anzeige aller Datensätze...</a></p>
        <p><a href="R3.jsp">gezieltes Lesen zu "Name"</a></p>
        <p><a href="index.html">Zurück zur Startseite...</a></p>
    </body>
</html>
```

Hier habe ich 2 Abfragen hinterlegt, R2.jsp liest alle Datensätze aus R3 ebenfalls, sortiert diese aber nach dem Namen. Die Unterscheidung liegt nur im SQL-Statement.

7.2 R2.jsp – Lesen aller gespeicherten Datensätze

R2 liest alle bisher gespeicherten Daten aus der Datenbank aus, Sortiert nach dem Schlüssel (cd_id) aufsteigend.

Quellcode:

```
<%@ page import="java.io.*,java.util.*,java.sql.*"%>
<%@ page import="javax.servlet.http.*,javax.servlet.*,javax.ejb.*" %>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql"%>

<!--
    Document      : R2
    Created on    : 11.01.2014, 08:33:41
    Author       : papa
-->

<sql:setDataSource var="Quelle" driver="com.mysql.jdbc.Driver"
                    url="jdbc:mysql://localhost/cd"
                    user="root"
                    password="root"/>

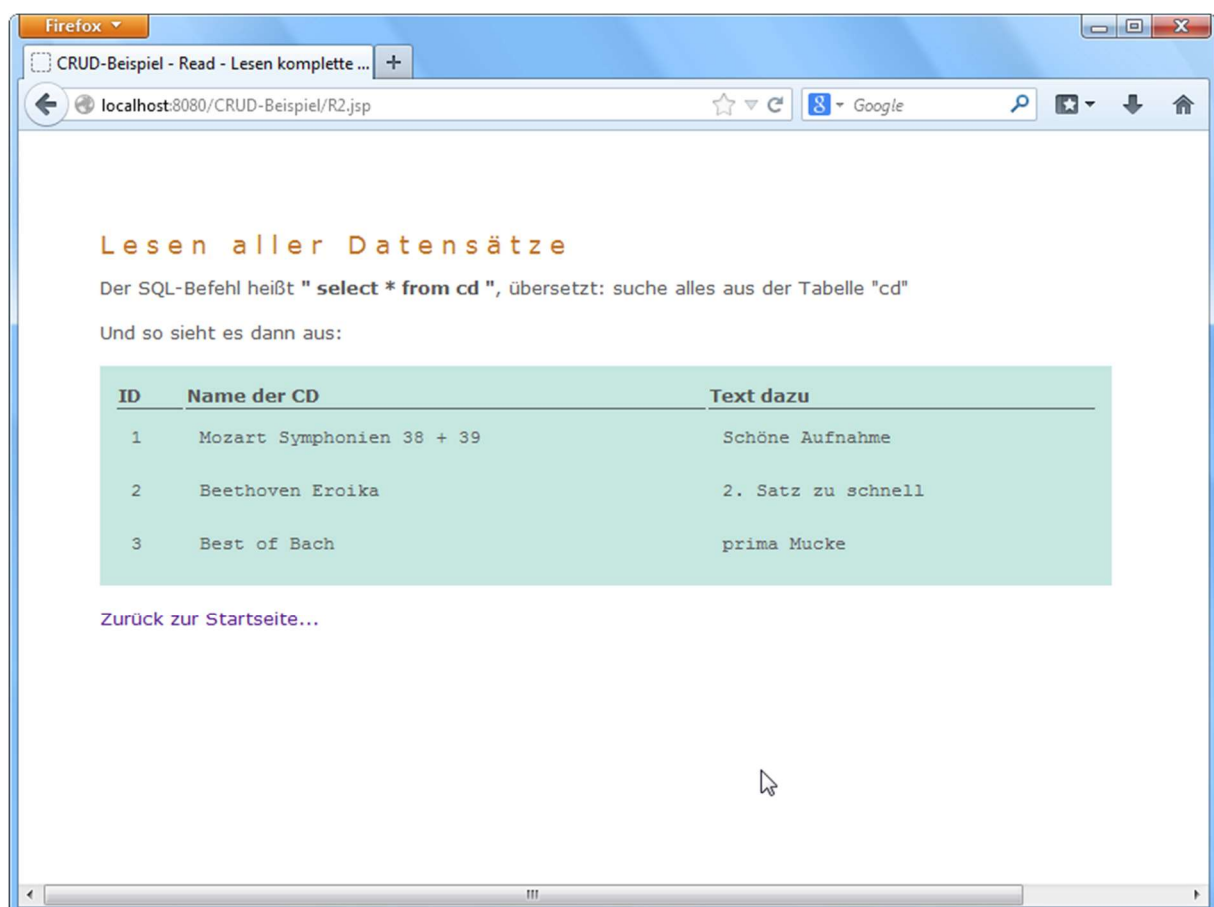
<sql:query sql="select * from cdmeta " var="Ergebnis"
dataSource="${Quelle}" >
</sql:query>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-
8">
        <link rel="stylesheet" type="text/css" href="style.css">
        <title>CRUD-Beispiel - Read - Lesen komplette Tabelle</title>
    </head>
    <body>
        <h1>Lesen aller Datensätze</h1>
        <div>Der SQL-Befehl heißt: <b>"select * from cd
        </b>,
            übersetzt: suche alles aus der Tabelle
        <b>"cd"</b></div>
        <br>
        <div>Und so sieht es dann aus:</div>
        <br>
        <table border="0" width="80%">
            <tr>
                <th>ID</th>
                <th>Name der CD</th>
                <th>Text dazu</th>
            </tr>
            <c:forEach var="row" items="${Ergebnis.rows}">
                <tr>
                    <td>
                        <c:out value="${row.cd_id}"/>

```

```
</td>
<td>
    <c:out value="${row.cd_name}"/>
</td>
<td>
    <c:out value="${row.cd_text}"/>
</td>
</tr>
</c:forEach>
</table>
<br>
<a href="index.html">Zurück zur Startseite...</a>
</body>
</html>
```

So sieht's aus:



7.3 R3.jsp – Lesen aller gespeicherten Datensätze – Sortierung geändert

R3.jsp unterscheidet sich von R2.jsp nur durch den SQL-Befehl, hier findet eine Sortierung nach dem CD-Namen statt. In unserem Beispiel kommt Beethoven vor Mozart.

Der Code sieht so aus:

```
<%@ page import="java.io.*,java.util.*,java.sql.*"%>
<%@ page import="javax.servlet.http.*,javax.servlet.*,javax.ejb.*" %>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql"%>

<!--
    Document      : R3
    Created on    : 11.01.2014, 08:33:41
    Author       : papa
-->

<sql:setDataSource var="Quelle" driver="com.mysql.jdbc.Driver"
                  url="jdbc:mysql://localhost/cd"
                  user="root"
                  password="root"/>

<sql:query sql="select * from cdmeta order by cd_name" var="Ergebnis"
dataSource="${Quelle}" >
</sql:query>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-
8">
        <link rel="stylesheet" type="text/css" href="style.css">
        <title>CRUD-Beispiel - Read - Lesen komplette Tabelle</title>
    </head>
    <body>
        <h1>Lesen aller Datensätze</h1>
        <div>Der SQL-Befehl heißt: <b>"select * from cdmeta
order by cd_name"</b>,
            &uuml;bersetzt: suche alles aus der Tabelle
            &quot;cdmeta&quot;, sortiere die Ausgabe nach dem Feld cd_name</div>
        <br>
        <div>Und so sieht es dann aus:</div>
        <br>
        <table border="1" width="80%">
            <tr>
                <th>ID</th>
                <th>Name der CD</th>
                <th>Text dazu</th>
            </tr>
            <c:forEach var="row" items="${Ergebnis.rows}">
                <tr>
                    <td>
                        <c:out value="${row.cd_id}"/>
                    </td>
                    <td>
                        <c:out value="${row.cd_name}"/>
                    </td>
                    <td>

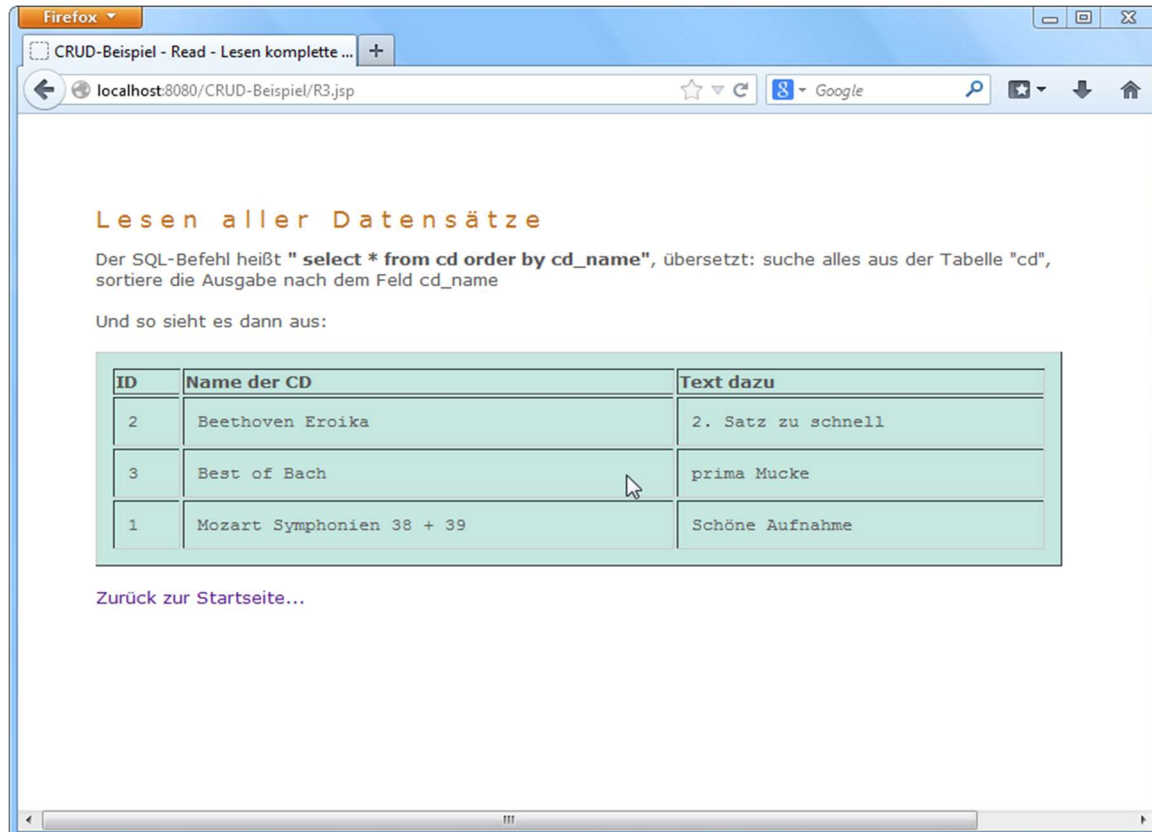
```

```

        <c:out value="\${row.cd_text}"/>
      </td>
    </tr>
  </c:forEach>
</table>
<br>
<a href="index.html">Zurück zur Startseite...</a>
</body>
</html>

```

Der Screenshot dazu:



8 Update – Dateien

Der Update-Teil ist komplexer, da zuerst gezielt ein Satz gelesen werden muss, und beim Update auch nur dieser Datensatz geändert werden darf. Deshalb haben wir hier 6 Dateien, U1.html bis U6.jsp:

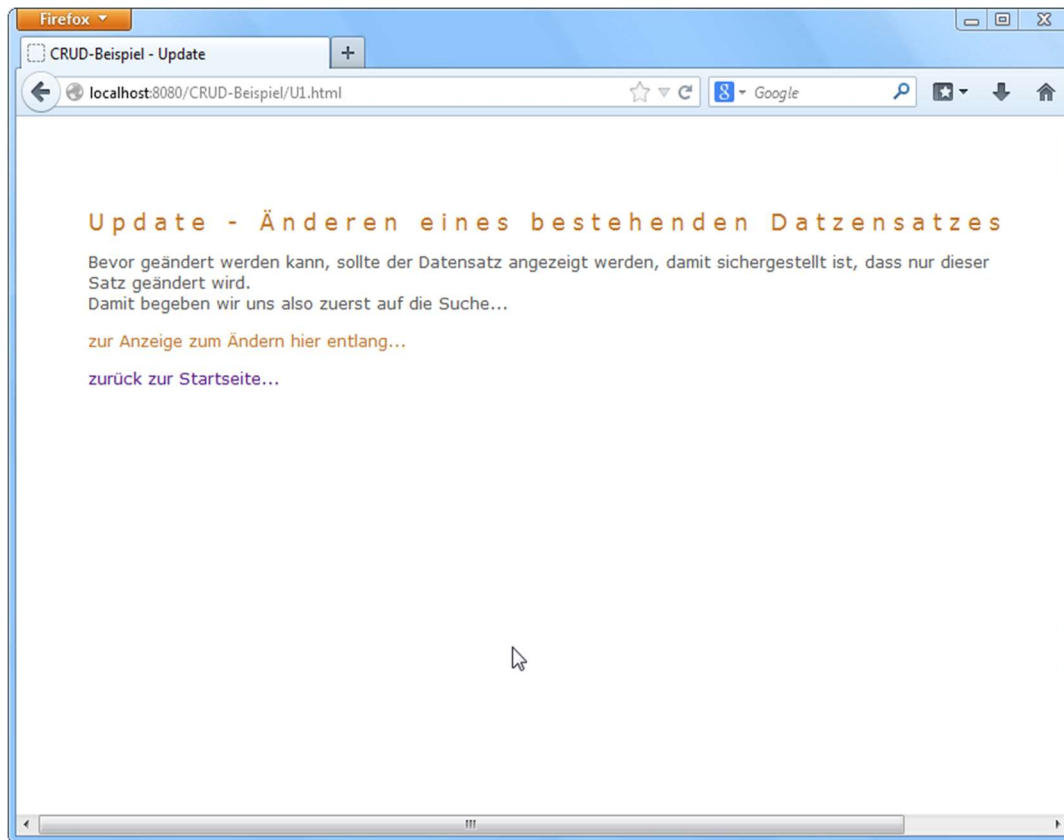
8.1 U1.html – Einstieg in Update

Auch hier ist wieder keine Datenbankaktivität von Nöten, deshalb U1 auch wieder als html-Datei.

Quellcode U1.html:

```
<!DOCTYPE html>
<!--
    Document      : U1.html
    Created on    : 11.01.2014, 07:36:24
    Author       : papa
-->
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-
8">
        <link rel="stylesheet" type="text/css" href="style.css">
        <title>CRUD-Beispiel - Update</title>
    </head>
    <body>
        <h1>Update - &Auml;nderen eines bestehenden Datensatzes</h1>
        <div>Bevor ge&auml;ndert werden kann, sollte der Datensatz
angezeigt werden, damit
            sichergestellt ist, dass nur dieser Satz ge&auml;ndert wird.
        </div>
        <div>Damit begeben wir uns also zuerst auf die Suche...</div>
        <p><a href="U2.jsp">zur Anzeige zum &Auml;ndern hier
entlang...</a></p>
        <p><a href="index.html">zur&uuml;ck zur Startseite...</a></p>
    </body>
</html>
```

Screenshot zur Update-Seite:



Dem Link „zur Anzeige zum Ändern hier entlang...“ folgen.

8.2 U2.jsp – Auswahl eines Satzes zur Änderung

Bei Start U2 wird zuerst die Datenbank einmal komplett gelesen und die Auswahlbox zeigt nur das Feld cd_name von jedem Datensatz. Der Satz, der in Auswahlbox selektiert wird, wird an U3 weitergereicht.

Quellcode U2.jsp:

```
<%@page import="java.sql.*"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>

<%--
    Document      : U2
    Created on    : 11.01.2014, 07:48:40
    Author       : papa
--%>

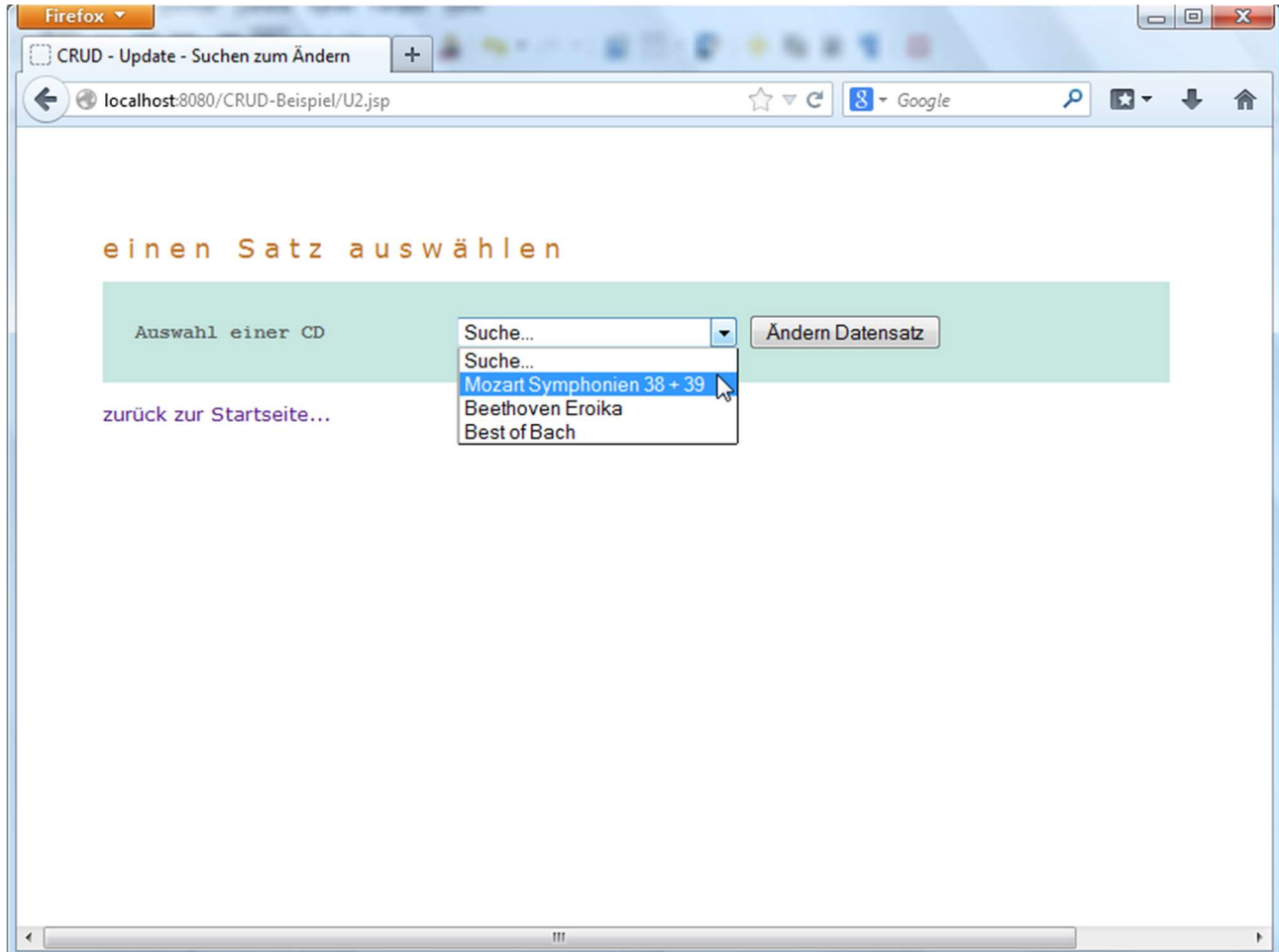
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-
8">
        <link rel="stylesheet" type="text/css" href="style.css">
        <title>CRUD - Update - Suchen zum &Auml;ndern</title>
    </head>
    <body>
        <h1>einen Satz ausw&auml;hlen</h1>
<form name="form" action="U3.jsp">
<table border="0">
    <tr>
        <td><b>Auswahl einer CD</b>&nbsp;<td>
        <select name="cd_ID">
            <option value="">Suche...</option>
        <%
            Class.forName("com.mysql.jdbc.Driver").newInstance();
            String connectionURL = "jdbc:mysql://localhost:3306/cd";

            Connection connection=
DriverManager.getConnection(connectionURL, "root", "root");
            PreparedStatement psmnt = connection.prepareStatement("select
cd_id, cd_name from cdmeta ");
            ResultSet results = psmnt.executeQuery();

            while(results.next()){
                String cd_name = results.getString(2);
                int id = results.getInt(1);
                %>
                <option value="<%= id %"><%=
out.println(cd_name); %></option>
            <%> results.close(); psmnt.close(); %>
        </select>
        <input type="submit" value="&Auml;ndern Datensatz"/><br>
    </tr>
</table>
</form>
```

```
<p><a href="index.html">zurück zur Startseite...</a></p>
</body>
</html>
```

Der Screenshot dazu:



8.3 U3.jsp – Anzeige aller Attribute eines Satzes

Wie beschrieben erhält U3 die Daten von U2. Hier wird jetzt aber der gesamte Datensatz zu der Auswahl aus U2 angezeigt.

U3.jsp:

```
<%@ page import="java.io.*,java.util.*,java.sql.*"%>
<%@ page import="javax.servlet.http.*,javax.servlet.*,javax.ejb.*" %>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql"%>

<!--
    Document      : U3
    Created on    : 14.12.2013, 09:02:26
    Author       : papa
-->

<%
    String scd_id = request.getParameter( "cd_ID" );
    String scd_name = null;
    String scd_text = null;
%>

<sql:setDataSource var="Quelle" driver="com.mysql.jdbc.Driver"
    url="jdbc:mysql://localhost/cd"
    user="root"
    password="root"/>

<sql:query sql="select * from cdmeta where cd_id = ?" var="Ergebnis"
dataSource="${Quelle}" >
    <sql:param value="${param.cd_ID}" />

</sql:query>

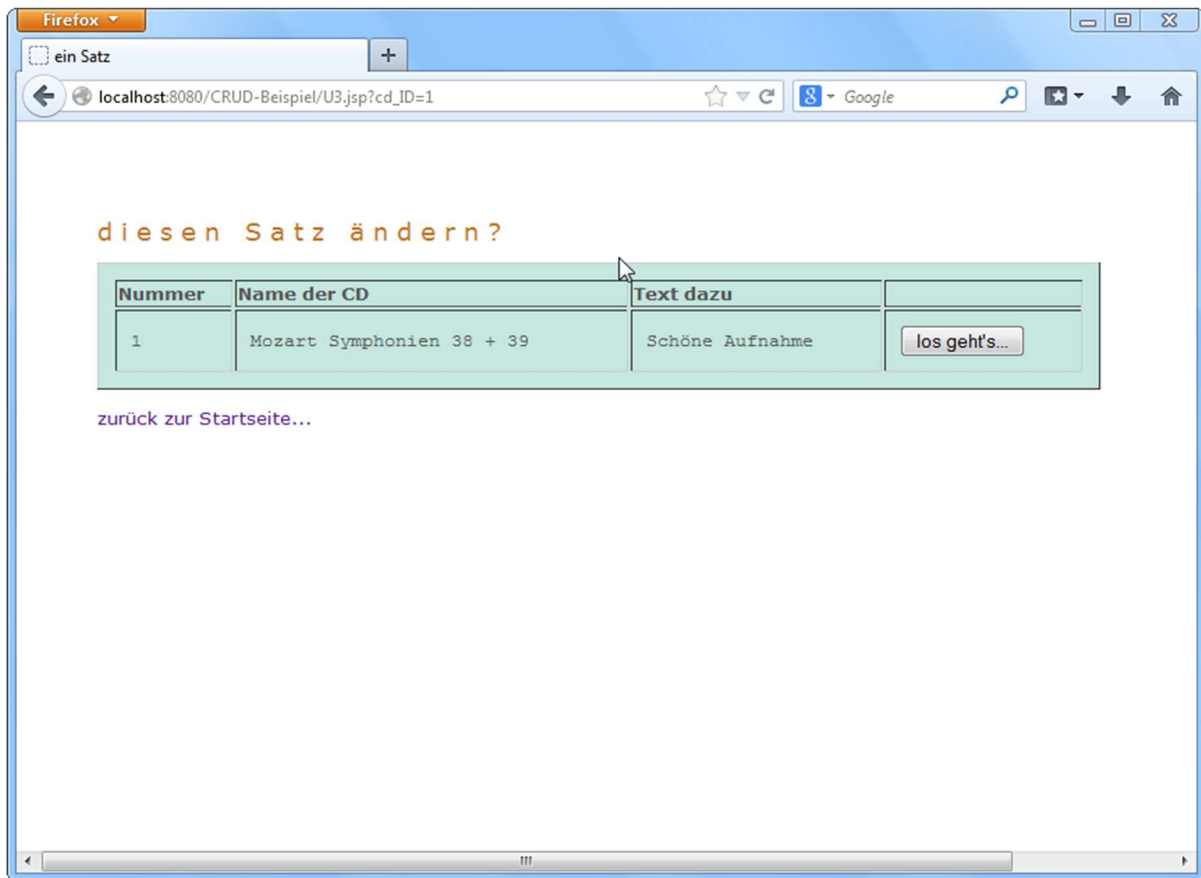
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-
8">
        <link rel="stylesheet" type="text/css" href="style.css">
        <title>ein Satz</title>
    </head>
    <body>
        <h1>diesen Satz &auml;ndern? </h1>
        <form name="frm" method="post" action="U4.jsp">
            <table border="1" width="80%">
                <tr>
                    <th>Nummer</th>
                    <th>Name der CD</th>
                    <th>Text dazu</th>
                    <th>&nbsp;</th>
                </tr>
                <c:forEach var="row" items="${Ergebnis.rows}">
                    <tr>
```

```

        <td>
            <c:out value="\${row.cd_id}"/>
            <c:set var="scd_id" value="\${row.cd_id}"/>
        </td>
        <td>
            <c:out value="\${row.cd_name}"/>
            <c:set var="scd_name" value="\${row.cd_name}"/>
        </td>
        <td>
            <c:out value="\${row.cd_text}"/>
            <c:set var="scd_text" value="\${row.cd_text}"/>
        </td>
        <td>
            <input type="hidden" name="cd_id"
value="\${scd_id}">
            <input type="hidden" name="cd_name"
value="\${scd_name}">
            <input type="hidden" name="cd_text"
value="\${scd_text}">
            <input type="submit" name="submit" value=" los
geht's... ">
        </td>
    </tr>
</c:forEach>
</table>
</form>
<p><a href="index.html">zur&uuml;ck zur Startseite...</a></p>
</body>
</html>

```

Screenshot zu U3:



Durch Klicken auf den „los geht's...“-Button gelangt man zu U4.

8.4 U4.jsp – Aufnahme der Änderungen zu einem Satz

U4 ist das Formular, in dem die neuen Daten eingegeben werden sollen.

Damit man weiß wo man sich befindet, sind die derzeitigen Inhalte als Vorbelegungen mitgegeben. Die Änderungen werden übernommen, sobald der Button „Ändern...“ geklickt wird.

Auch hier wieder der Hinweis, dass keinerlei Prüfungen auf die neuen Felder eingebaut sind, das sollte dann jeder für sich selbst machen.

Quellcode zu U4.jsp:

```
<%@ page import="java.io.*,java.util.*,java.sql.*"%>
<%@ page import="javax.servlet.http.*,javax.servlet.*,javax.ejb.*" %>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql"%>

<%--
    Document      : U4
    Created on    : 11.01.2014, 12:17:47
    Author       : papa
--%>

<%
    String scd_id = request.getParameter( "cd_id" );
    String scd_name = request.getParameter( "cd_name" );
    String scd_text = request.getParameter( "cd_text" );
%>

<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-
8">
    <link rel="stylesheet" type="text/css" href="style.css">
    <title>CRUD - Update - neue Werte aufnehmen</title>
</head>
<body>
    <h1>Erfassen der neuen Werte</h1>
    <div>Der Satz wird als ganzes gespeichert, das bedeutet, dass
beim Klicken auf
        den Button beide Felder gleichzeitig ge&uuml;ndert werden.
Eine gesonderte
        Pr&uuml;fung erfolgt nicht.
    </div>
    <br>

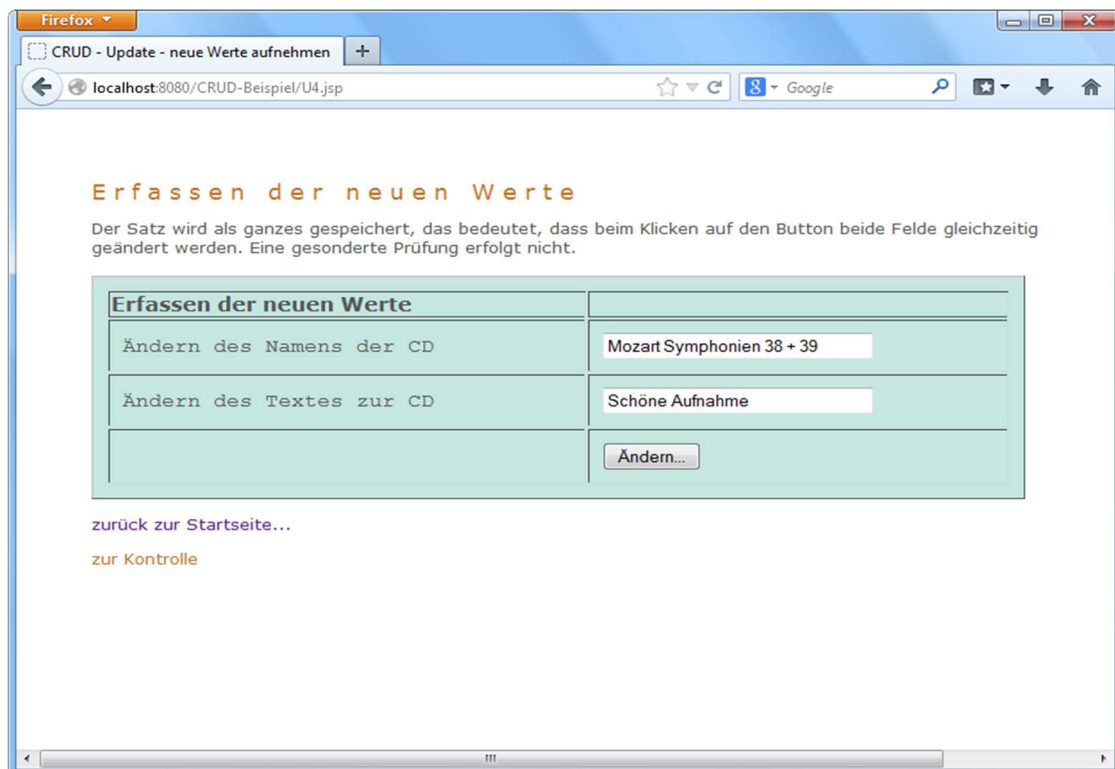
    <form name="frm" action="U5.jsp">
    <input type="hidden" name="cd_id" value="<%= scd_id %>">
    <table width="80%" border="1">
        <tr>
            <th>Erfassen der neuen Werte</th>
            <th>&nbsp;</th>
        </tr>
        <tr>
            <td>&Auml;ndern des Namens der CD</td>
            <td><input type="text" name="cd_name" value="<%=
scd_name %>"></td>
        </tr>
        <tr>
            <td>&Auml;ndern des Textes zur CD</td>
            <td><input type="text" name="cd_text" value="<%=
scd_text %>"></td>
        </tr>
    </table>
    </form>
</body>
</html>
```

```

        <td>&nbsp;</td>
        <td><input type="submit" name="submit" value=" &Auml;ndern...
"></td>
    </tr>
</table>
</form>
    <p><a href="index.html">zur&uuml;ck zur Startseite...</a></p>
    <p><a href="Kontrolle.jsp">zur Kontrolle</a></p>
</body>
</html>

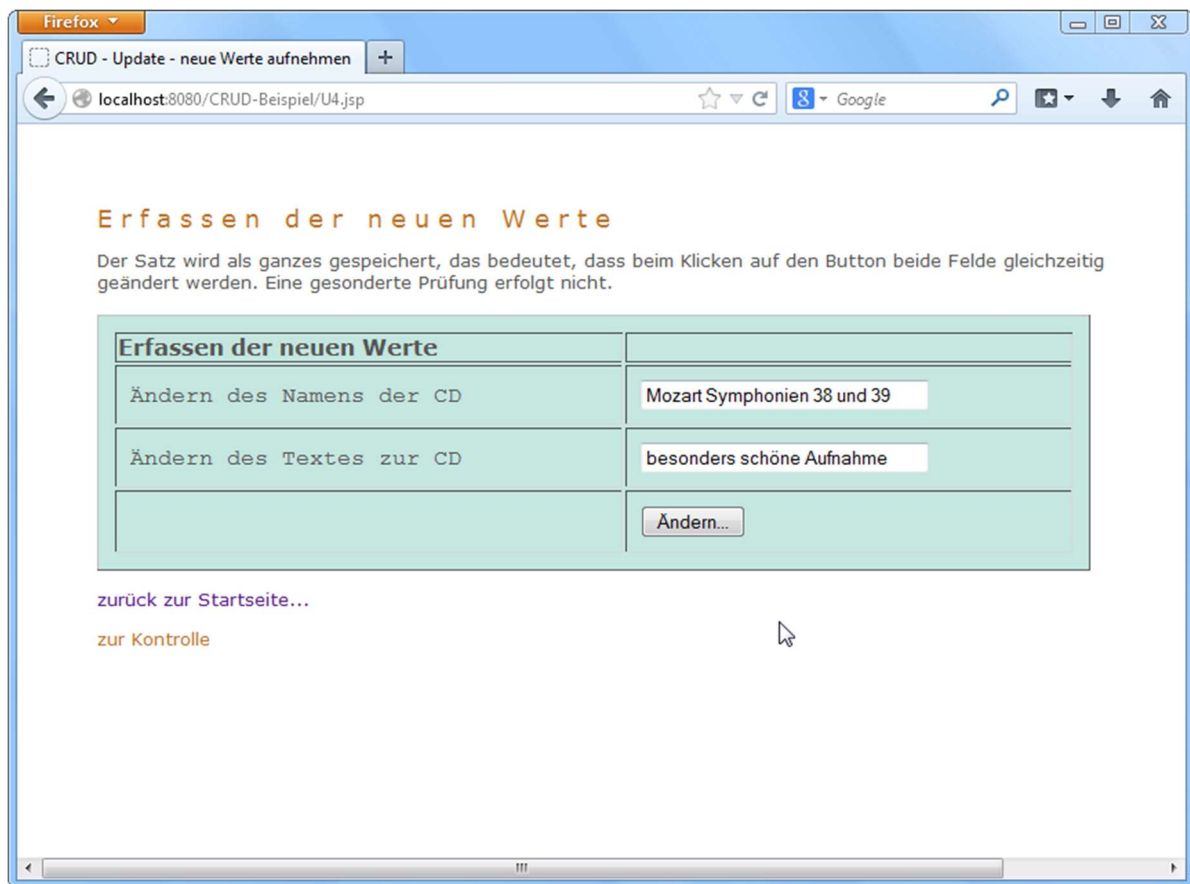
```

Screenshot VOR Änderung:



Aus dem „+“ im Namen soll ein „und“ und aus „Schöne Aufnahme“ „besonders schöne Aufnahme“ werden.

Screenshot dazu:



Mit „Ändern“ in U5.jsp

8.5 U5.jsp – Übernahme der Änderungen in die Datenbank

In U5 werden die geänderten Zeilen in die Datenbank übernommen. Auch hier wieder gibt es keine Prüfung auf die Inhalte!

Quellcode U5.jsp:

```
<%@ page language="java" import="java.sql.*" errorPage="" %>
<!--
    Document      : U5
    Created on    : 11.01.2014, 12:16:41
    Author       : papa
--%>

<%
    Connection conn = null;

    Class.forName("com.mysql.jdbc.Driver").newInstance();
    String jdbcURL="jdbc:mysql://localhost:3306/cd";
    conn = DriverManager.getConnection(jdbcURL,"root", "root");

    PreparedStatement psUpdateRecord=null;
    String sqlUpdateRecord=null;

    int cd_id=Integer.parseInt(request.getParameter("cd_id"));
    String cd_name=request.getParameter("cd_name");
    String cd_text=request.getParameter("cd_text");

    try
    {
        sqlUpdateRecord="update cdmeta set cd_name=?, cd_text=? where
cd_id=?";
        psUpdateRecord=conn.prepareStatement(sqlUpdateRecord);
        psUpdateRecord.setString(1,cd_name);
        psUpdateRecord.setString(2,cd_text);
        psUpdateRecord.setInt(3,cd_id);

        psUpdateRecord.executeUpdate();
    }
    catch(Exception e)
    {
        response.sendRedirect("U3.jsp");
        ///// On error it will send back to addRecord.jsp page
    }

    try{
        if(psUpdateRecord!=null)
        {
            psUpdateRecord.close();
        }

        if(conn!=null)
        {
            conn.close();
        }
    }
    %>
```

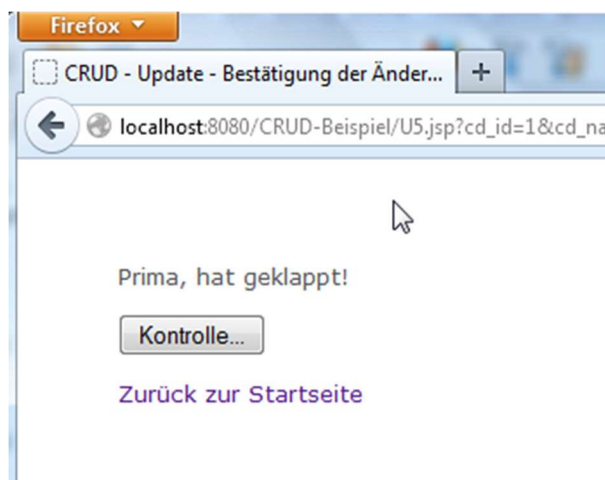
```

    }
  }
  catch(Exception e)
  {
    e.printStackTrace();
  }
%>
<html>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-
8">
  <link rel="stylesheet" type="text/css" href="style.css">
  <title>CRUD - Update - Bestätigung der Änderung</title>
<body>
  <div>Prima, hat geklappt!
  </div>
  <form action="U6.jsp" method="post" accept-charset="ISO-8859-1">
  <input type="hidden" name="cd_ID" value="${param.cd_id}">
  <p><input type="submit" value="Kontrolle..."></p>
  </form>
  <p><a href="index.html">Zurück zur Startseite</a></p>

</body>
</html>

```

U5 sollte sich nur im Fehlerfall bemerkbar machen, ansonsten erscheint nur eine neue Seite mit dem Hinweis, dass alles geklappt hat.



Mit Klick auf „Kontrolle...” landet man in U6.jsp.

8.6 U6.jsp – Kontrolle der Übernahme

In U6 wird der soeben geänderte Satz gelesen und angezeigt.

Quellcode U6.jsp:

```
<%@ page import="java.io.*,java.util.*,java.sql.*"%>
<%@ page import="javax.servlet.http.*,javax.servlet.*,javax.ejb.*" %>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql"%>

<!--
    Document      : U6
    Created on    : 11.01.2014, 12:17:47
    Author       : papa
-->

<sql:setDataSource var="Quelle" driver="com.mysql.jdbc.Driver"
                  url="jdbc:mysql://localhost/cd"
                  user="root"
                  password="root"/>

<sql:query sql="select * from cdmeta where cd_id = ${param.cd_ID}"
var="Ergebnis" dataSource="${Quelle}" >
</sql:query>

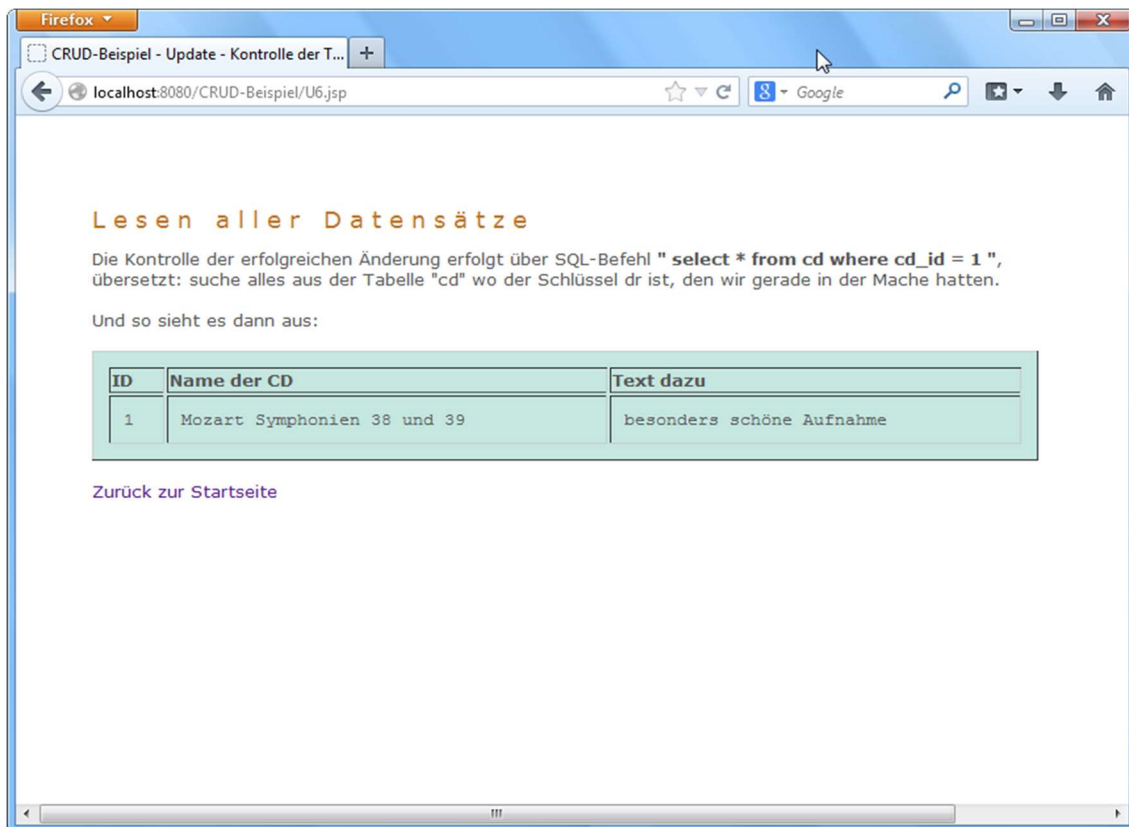
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-
8">
        <link rel="stylesheet" type="text/css" href="style.css">
        <title>CRUD-Beispiel - Update - Kontrolle der Tabelle</title>
    </head>
    <body>
        <h1>Lesen aller Datensätze</h1>
        <div>Die Kontrolle der erfolgreichen Änderung erfolgt über
SQL-Befehl
        <b>"select * from cdmeta where cd_id = ${param.cd_ID}
";</b>,
        übersetzt: suche alles aus der Tabelle
"cdmeta" wo der Schlüssel dr ist,
den wir gerade in der Mache hatten.</div>
<br>
<div>Und so sieht es dann aus:</div>
<br>
<table border="1" width="80%">
    <tr>
        <th>ID</th>
        <th>Name der CD</th>
        <th>Text dazu</th>
    </tr>
    <c:forEach var="row" items="${Ergebnis.rows}">
        <tr>
            <td>
```

```

        <c:out value="\${row.cd_id}"/>
    </td>
    <td>
        <c:out value="\${row.cd_name}"/>
    </td>
    <td>
        <c:out value="\${row.cd_text}"/>
    </td>
</tr>
</c:forEach>
</table>
<br>
<a href="index.html">Zurück zur Startseite</a>
</body>
</html>

```

Screenshot dazu:



9 Delete – Dateien

Der Delete-Teil ist auch recht komplex. Hier muss ebenfalls sichergestellt sein, dass nur der ausgewählte Satz auch gelöscht werden darf. Hier habe ich 4 unterschiedliche Dateien.

9.1 D1.html – Einstieg in Update

Auch hier ist wieder keine Datenbankaktivität von Nöten, deshalb D1 auch wieder als html-Datei.

Quellcode D1.html:

```
<!DOCTYPE html>
<!--
    Document      : D1
    Created on    : 11.01.2014, 07:32:24
    Author       : papa
-->

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-
8">
    <link rel="stylesheet" type="text/css" href="style.css">

    <title>CRUD-Beispiel - Delete</title>
  </head>
  <body>
    <h1>Delete - Löschen eines bestehenden Datensatzes</h1>
    <div>Bevor gelöscht werden kann, sollte der Datensatz
angezeigt werden, damit
        sichergestellt ist, dass nur dieser Satz gelöscht wird.
Andernfalls fehlen einem
        gegebenenfalls die Schlüsselinformationen, um den Satz
zu fassen zu bekommen.
    </div>
    <p>Damit begeben wir uns also zuerst auf die Suche...</p>
    <p><a href="D2.jsp">zur Anzeige zum Löschen hier
entlang...</a></p>
    <p><a href="index.html">zurück zur Startseite...</a></p>
  </body>
</html>
```

9.2 D2.jsp – Auswahl eines Satzes zur Löschung

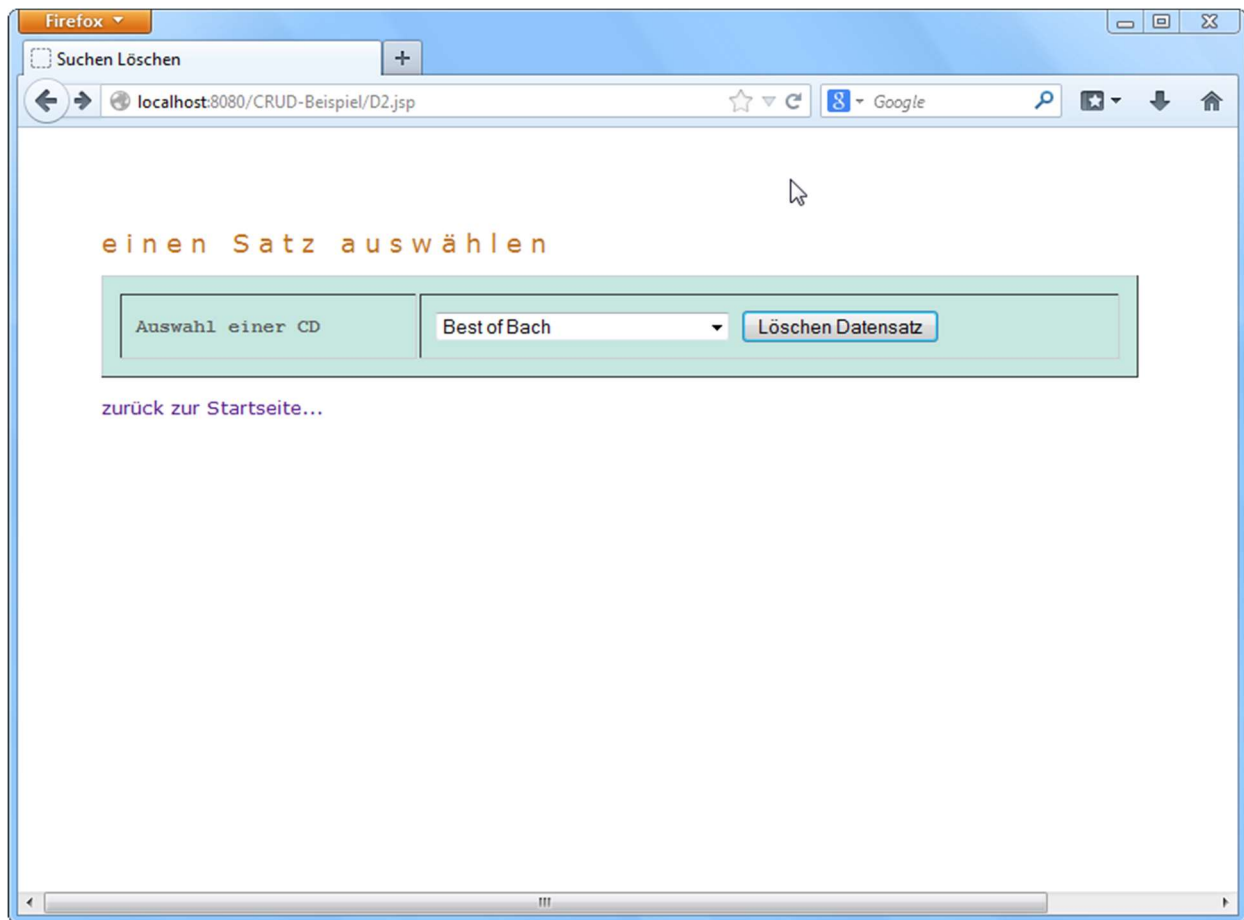
In D2 suchen wir uns analog U2 den Datensatz raus, den wir löschen wollen.

Quellcode D2.jsp:

```
<%@page import="java.sql.*"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%--
    Document      : D2
    Created on    : 11.01.2014, 07:48:40
    Author       : papa
--%>

<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-
8">
        <link rel="stylesheet" type="text/css" href="style.css">
        <title>Suchen L&ouml;schen</title>
    </head>
    <body>
        <h1>einen Satz ausw&auml;hlen</h1>
    <%--
--%>
        <form name="form" method="post" action="D3.jsp">
            <table border="1">
                <tr>
                    <td><b>Auswahl einer CD</b>&nbsp;<td>
                        <select name="cd_ID"><option value="">Suche...</option>
                        <%
                            Class.forName("com.mysql.jdbc.Driver").newInstance();
                            String connectionURL =
"jdbc:mysql://localhost:3306/cd";
                            Connection connection=
DriverManager.getConnection(connectionURL, "root", "root");
                            PreparedStatement psmnt =
connection.prepareStatement("select cd_id, cd_name, cd_text from cdmeta
");
                            ResultSet results = psmnt.executeQuery();
                            while(results.next()){
                                String cd_name = results.getString(2);
                                int id    = results.getInt(1);
                                %>
                                <option value="<%= id %"><%=
out.println(cd_name); %></option>
                                <%> results.close(); psmnt.close(); %>
                            </select>
                            <input type="submit" value="L&ouml;schen Datensatz"/><br>
                        </tr>
                    </table>
                </form>
                <p><a href="index.html">zur&uuml;ck zur Startseite...</a></p>
            </body>
        </html>
```

Screenshot dazu



Mit „Löschen Datensatz“ geht es zu D3.jsp.

9.3 D3.jsp – Löschung des Satzes durchführen

In D3 wird die Löschung durchgeführt

Quellcode D3.jsp:

```
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql"%>
<%--
    Document      : D3
    Created on    : 11.01.2014, 07:48:41
    Author       : papa
--%>

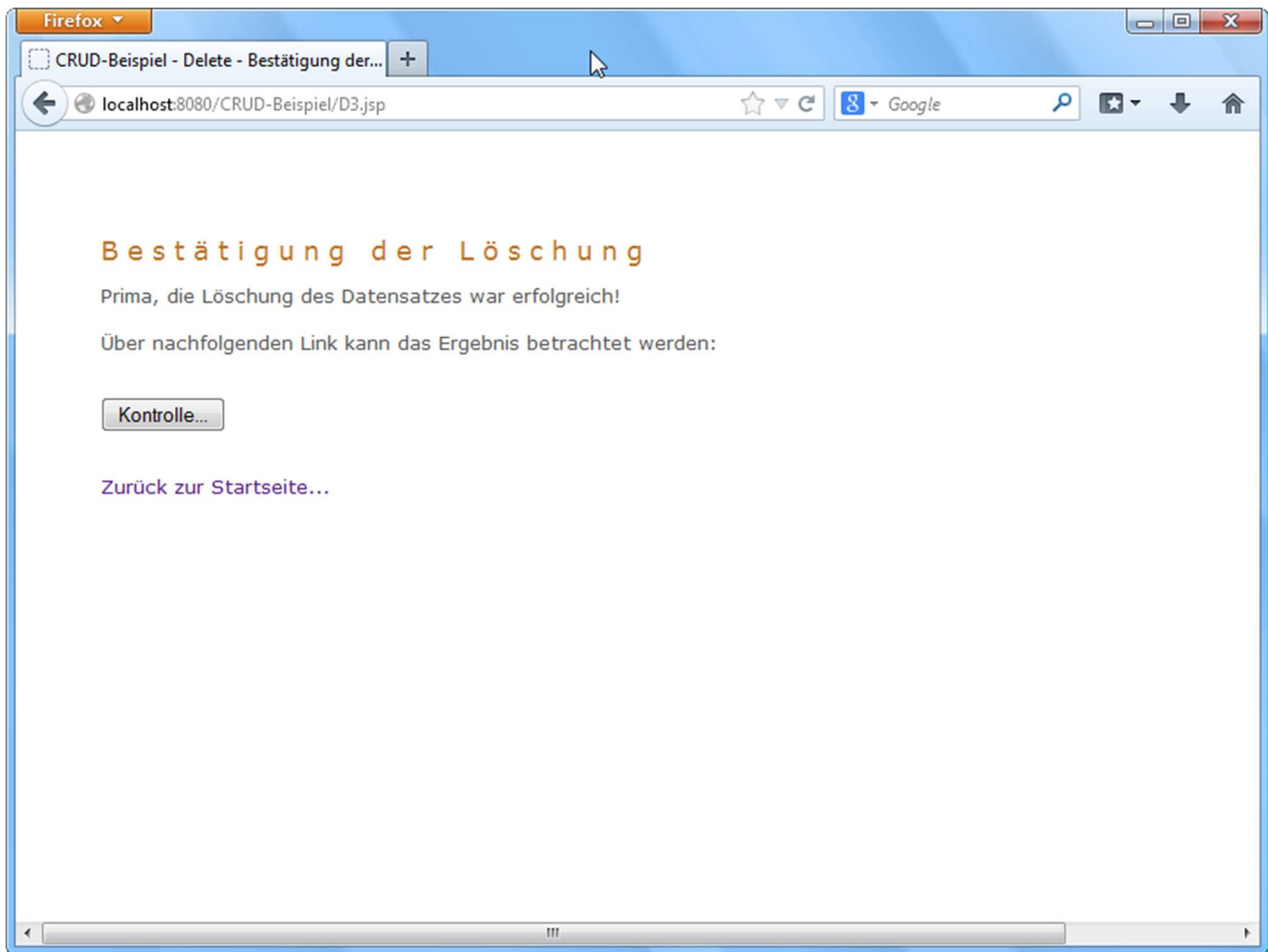
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-
8">
        <link rel="stylesheet" type="text/css" href="style.css">
        <title>CRUD-Beispiel - Create - Best&uuml;tigung des
Einf&uuml;gens</title>
    </head>
    <body>

        <sql:setDataSource var="Quelle" driver="com.mysql.jdbc.Driver"
            url="jdbc:mysql://localhost/cd"
            user="root"
            password="root"/>

        <sql:update dataSource="${Quelle}" var="updttable">
            delete from cdmeta where cd_id = ${param.cd_ID}
        </sql:update>

        <%@page contentType="text/html" pageEncoding="UTF-8"%>
        <h1>Best&uuml;tigung des Einf&uuml;gens</h1>
        <div>Prima, die Speicherung des Datensatzes war erfolgreich!
            Das war nicht weiter verwunderlich, da keine Pr&uuml;fungen
auf die Datenfelder
            erfolgt sind. Hier sind Kreativität und Sorgfalt
gefragt.</div>
        <br>
        <div>&Uuml;ber nachfolgenden Link kann das Ergebnis betrachtet
werden:</div>
        <br>
        <form action="D4.jsp" method="post" accept-charset="ISO-8859-1">
            <input type="hidden" name="cdName" value="${param.cdName}">
            <p><input type="submit" value="Kontrolle..."></p>
        </form>
        <br>
        <a href="index.html">Zur&uuml;ck zur Startseite...</a>
    </body>
</html>
```


Zur Bestätigung wird das Folgende angezeigt:



9.4 D4.jsp – Kontrolle der Löschung

Da der Satz im Idealfall gelöscht wurde, kann man ihn auch nicht mehr anzeigen. Deshalb wird in D4 jeder gespeicherte Datensatz angezeigt.

Quellcode:

```
<%@ page import="java.io.*,java.util.*,java.sql.*"%>
<%@ page import="javax.servlet.http.*,javax.servlet.*,javax.ejb.*" %>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql"%>

<!--
    Document      : R2
    Created on    : 11.01.2014, 08:33:41
    Author       : papa
-->

<sql:setDataSource var="Quelle" driver="com.mysql.jdbc.Driver"
                  url="jdbc:mysql://localhost/cd"
                  user="root"
                  password="root"/>

<sql:query sql="select * from cdmeta " var="Ergebnis"
dataSource="${Quelle}" >
</sql:query>

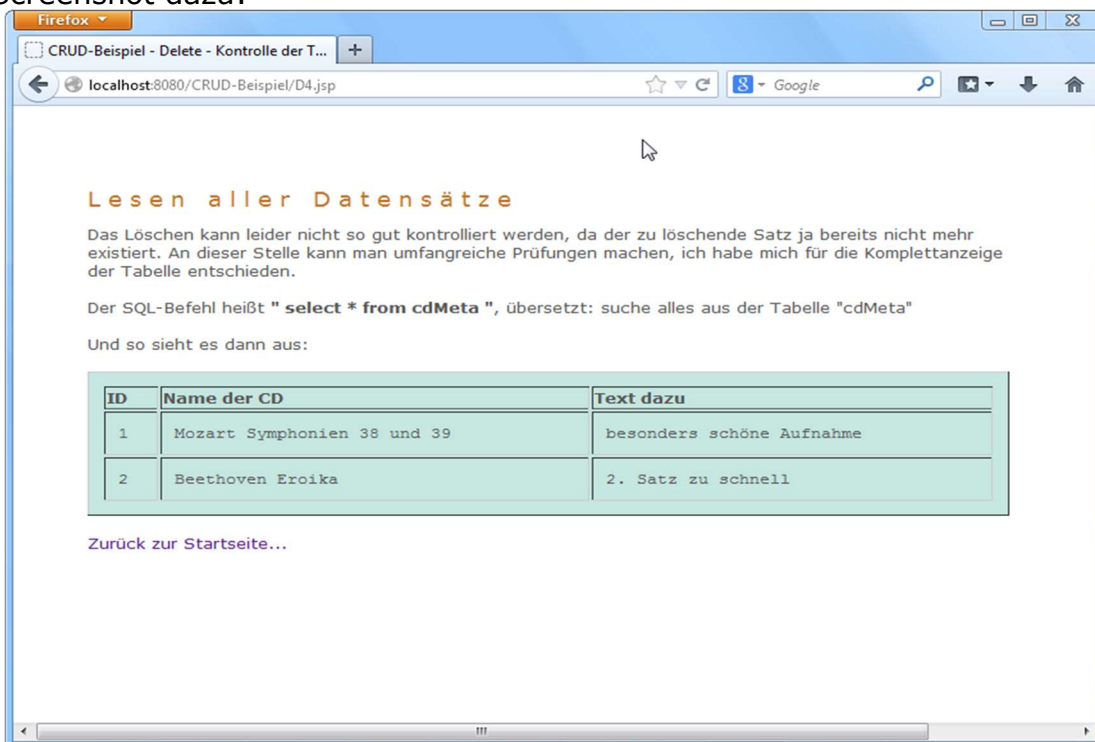
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-
8">
        <link rel="stylesheet" type="text/css" href="style.css">
        <title>CRUD-Beispiel - Delete - Kontrolle der Tabelle</title>
    </head>
    <body>
        <h1>Lesen aller Datensätze</h1>
        <div>Das Löschen kann leider nicht so gut kontrolliert
werden, da der zu
        löschende Satz ja bereits nicht mehr existiert. An
dieser Stelle kann
        man umfangreiche Prüfungen machen, ich habe mich
für die Komplettanzeige
        der Tabelle entschieden. </div>
        <br>
        <div>Der SQL-Befehl heißt <b>"select * from cd
">
        übersetzt: suche alles aus der Tabelle
">cdmeta"</div>
        <br>
        <div>Und so sieht es dann aus:</div>
        <br>
        <table border="1" width="80%">
            <tr>
                <th>ID</th>
```

```

        <th>Name der CD</th>
        <th>Text dazu</th>
    </tr>
    <c:forEach var="row" items="${Ergebnis.rows}">
    <tr>
        <td>
            <c:out value="${row.cd_id}"/>
        </td>
        <td>
            <c:out value="${row.cd_name}"/>
        </td>
        <td>
            <c:out value="${row.cd_text}"/>
        </td>
    </tr>
    </c:forEach>
</table>
<br>
<a href="index.html">Zurück zur Startseite...</a>
</body>
</html>

```

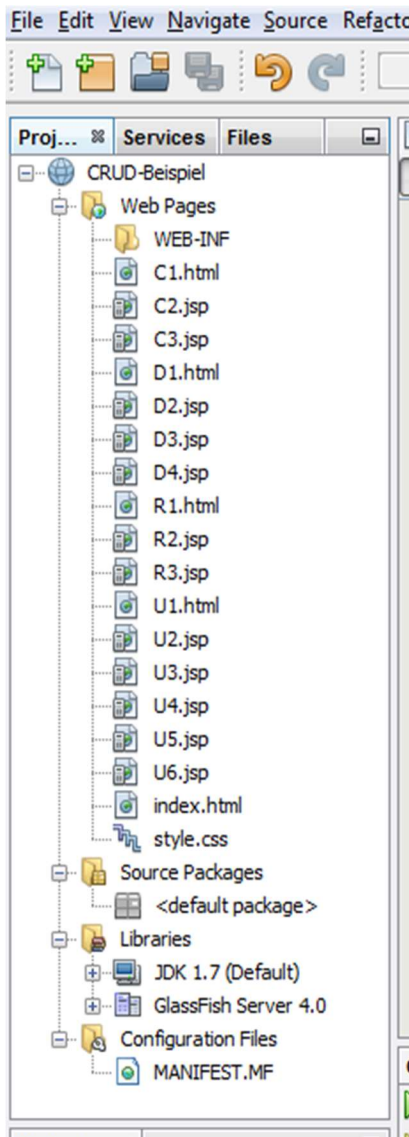
Screenshot dazu:



10 Abschluss

Wir haben es geschafft, alle Datenbankaktivitäten sind erledigt!!

Die Struktur in der IDE sollte in etwa so aussehen:



Das Ausgestalten, Umbauen, Anflanschen kann damit beginnen – viel Erfolg und Spaß dabei!

Liebe Grüße,
papa