

Studienarbeit IN-B5  
Thema: Webbasierter Datenbankzugriff  
Zu untersuchende Systeme:  
MySQL und PHP  
Oracle und JSP

Autor: Marco Behnke<sup>1</sup>  
Betreuender Professor: Dr. Wolfgang Gerken<sup>2</sup>

Derzeitiger Seitenumfang: 44

23. November 2004

<sup>1</sup>Email: marco.behnke@web.de, Matrikelnummer: 1562844

<sup>2</sup>Email: gerken@informatik.haw-hamburg.de

---

# Vorwort

In der folgenden Studienarbeit setze ich mich mit der Installation, Konfiguration und Anwendung von Web-basierten Datenbankanwendungen auseinander. Zu untersuchen sind die Unterschiede, die beide Systeme mit sich bringen, sowohl in der Bereichen der Installation/Konfiguration, als auch in der Implementierung der auszuführenden Anwendung.

Für diese Untersuchung habe ich mich für die folgenden beiden System entschieden:

- Zugriff auf eine MySQL-Datenbank über PHP<sup>1</sup>
- Zugriff auf eine Oracle-Datenbank über JSP<sup>2</sup>

MySQL in Verbindung mit PHP stellt mit Sicherheit die am meisten benutzte Internetplattform im Bereich Datenbanken dar, während Oracle sehr großen Zuspruch im Business-Bereich findet.

Einleitend zu dieser Studienarbeit werde ich mich erst einmal für jeweils eine Zielplattform entscheiden (ab Seite 1) und anschließend die ausgewählten Systeme aufsetzen und für den Einsatz konfigurieren (ab Seite 7). Dabei werde ich schon auf einige Unterschiede in der Handhabung der System eingehen und einige Punkte ansprechen, die bei der Installation und Konfiguration zu bedenken sind.

Anschließend werde ich eine kleine Anwendung entwickeln, die auf beiden Systemen zum Einsatz kommen wird (ab Seite 9). Auch hier werde ich auf signifikante Unterschiede in der Implementierung zu sprechen kommen. In diesem Zusammenhang werde ich auch eine Untersuchung durchführen, die sich mit der Performance der Datenbank gestützten Abfragen beschäftigen wird.

Schlußendlich werde ich dann eine abschließende Zusammenfassung präsentieren, die in eine übersichtliche Tabelle eingefasst wird.

Diese Studienarbeit wurde mit  $\LaTeX$  erstellt.

## 0.1 Webbasierter Datenbankzugriff - Wozu ist das gut?

Die einleitende Frage vor dieser Arbeit lautet: **Wozu benötigt man überhaupt einen webbasierten Datenbankzugriff?**

Damit der Kunde Zugriff auf dynamische Daten hat. Nehmen wir als Beispiel einen Online-Shop. Die Firma hat einen sich ständig ändernden Artikelstamm und Warenbestand und verwaltet diesen in einer Datenbank. Die Daten sollen dem Kunden zugänglich gemacht werden.

Dies kann auf der einen Seite über eine Client-Applikation erfolgen. Der Kunde muss ein Programm installieren, um auf die Daten zugreifen zu können. Doch diese Lösung ist nicht ohne Probleme. Zum Einen muss der Kunde nicht zwangsweise Rechte haben, um das erforderliche Programm installieren zu können, die Firma muss für verschiedene Betriebssysteme Anwendungen zur Verfügung stellen und werden in der Anwendung Veränderungen durchgeführt, so muss sichergestellt werden, dass der Kunde auch immer die aktuelle Version hat.

---

<sup>1</sup>Personal Home Page Tools[1]

<sup>2</sup>Java Server Pages[2]

Auf der Anderen Seite steht nun die Webanwendung. Diese ist nur einmal vorhanden und für alle erreichbar. Die Anwendung wird serverseitig ausgeführt und läuft somit unabhängig vom Client. Müssen Veränderungen oder Wartungen am Programm vorgenommen werden, so geschieht dies zentral und ohne Einbeziehung des Kunden.

Die Lösung eines webbasierten Datenbankzugriffs bietet viele Vorteile auf dem Sektor Internet und Onlineshops oder allgemein in Bereichen, in denen dem Kunden dynamische Daten zur Verfügung gestellt werden.

# Inhaltsverzeichnis

<b>Vorwort</b>	<b>ii</b>
0.1 Webbasierter Datenbankzugriff - Wozu ist das gut? . . . . .	ii
<b>1 Die Plattformen</b>	<b>1</b>
1.1 Generelle Fragen . . . . .	1
1.2 MySQL und PHP . . . . .	1
1.2.1 MySQL . . . . .	1
1.2.2 PHP . . . . .	2
1.3 Oracle und JSP . . . . .	4
1.3.1 Oracle . . . . .	4
1.3.2 JSP . . . . .	4
1.4 Das Betriebssystem . . . . .	6
<b>2 Installation und Konfiguration</b>	<b>7</b>
2.1 MySQL und PHP . . . . .	7
2.1.1 Die Komponenten . . . . .	7
2.1.2 Die Installation . . . . .	8
2.2 Oracle und JSP . . . . .	8
2.2.1 Die Komponenten . . . . .	8
2.2.2 Die Installation . . . . .	8
2.3 Fazit . . . . .	8
<b>3 Praktischer Einsatz</b>	<b>9</b>
3.1 Die Anwendung . . . . .	9
3.2 Implementation . . . . .	9
3.2.1 MySQL und PHP . . . . .	9
3.2.2 Oracle und JSP im Vergleich . . . . .	10
<b>4 Fazit</b>	<b>12</b>
4.1 Tabellarische Übersicht . . . . .	12
4.2 Installationsaufwand . . . . .	12
4.3 Kosten . . . . .	13
4.4 Verwaltungstools / Tools allgemein . . . . .	13
<b>I Anhang</b>	<b>14</b>
<b>Diagramme</b>	<b>15</b>
4.1 Anwendungsfall . . . . .	15
4.2 Aktivitätsdiagramm . . . . .	16

---

<b>Quelltexte</b>	<b>17</b>
4.3 PHP	17
4.3.1 connectdb.php - Datenbankverbindung herstellen	17
4.3.2 disconnectdb.php - Datenbankverbindung beenden	17
4.3.3 functions.php - Häufig benutzte Funktionen	18
4.3.4 artikel.php - Der Artikelkatalog	19
4.3.5 buy.inc.php - Kaufverarbeitung	22
4.3.6 warenkorb.php - Warenkorb anzeigen	23
4.3.7 kasse.php - Zur Kasse gehen, Einkauf abschließen	24
4.3.8 index.html - Startseite	25
4.4 JSP	26
4.4.1 artikel.jsp - Der Artikelkatalog	26
4.4.2 warenkorb.jsp - Warenkorb anzeigen	29
4.4.3 create_tables.jsp - Datenbanktabellen erstellen	31
4.4.4 index.html - Startseite	33
4.5 SQL	34
4.5.1 studienarbeit_artikel.sql - Artikelkatalog	34
4.5.2 studienarbeit_warenkorb.sql - Warenkorb	35
<b>Literatur- und Quellenverzeichnis</b>	<b>36</b>
<b>Abbildungsverzeichnis</b>	<b>37</b>
<b>Index</b>	<b>37</b>



# Kapitel 1

## Die Plattformen

### 1.1 Generelle Fragen

Bevor ich mit weiteren Themen beschäftigen werde, ist es erst einmal notwendig, sich für eine Zielplattform zu entscheiden. Dabei gilt es erst einmal folgende Fragen zu klären:

1. Welches Betriebssystem verwende ich?
2. Welche Programmversionen von MySQL, PHP und Oracle soll ich verwenden?
3. Welchen Webserver benutze ich, um JSP und PHP über einen Browser anzusprechen?

Dabei werde ich ein besonderes Augenmerk auf einfache Handhabung in erster Linie und Aktualität und Fehlerfreiheit in zweiter Linie werfen.

Auf den folgenden Seite stelle ich die Plattformen vor und werde an Hand von einigen Beispielen die Handhabung erklären. Desweiteren werde ich Beispiele für die Anwendung der Programmiersprachen geben.

### 1.2 MySQL und PHP

#### 1.2.1 MySQL

MySQL zählt zur Zeit ohne Frage mit zu den populärsten Datenbanksystemen. Zum einen ist es für jeden frei verfügbar, da es unter der GPL<sup>1</sup> angeboten wird. Zum Anderen wird MySQL von der Mehrzahl aller Webhoster als Datenbanksystem für den User mitangeboten, meist in Verbindung mit PHP (zum Beispiel bei 1&1<sup>2</sup> oder Strato<sup>3</sup>).

MySQL bieten für den Programmierer Schnittstellen und Treiber für eine Reihe von Programmiersprachen

- C, C++, Java, Perl, PHP
- Eiffel, Python und Tcl

und kann auf einer ganzen Reihe von Betriebssystemen eingesetzt werden

- Windows 95 und aufwärts
- Linux, BSD, MacOS
- AIX, DEC, HP-UX und weitere UNIX Derivate

---

<sup>1</sup>GNU Public License

<sup>2</sup>[www.1und1.de](http://www.1und1.de)

<sup>3</sup>[www.strato.de](http://www.strato.de)

- Amiga OS
- und viele andere

MySQL ist zur Zeit in der Version 4.00 als Production Release erhältlich. Version 5 ist jedoch schon seit einiger Zeit in der Entwicklung.

### Funktionsumfang

In der aktuellen Version unterstützt MySQL eine ganze Reihe von Funktionen, die wenig außen vor lassen. Doch leider gibt es auch einiges, das noch nicht so ganz funktioniert:

- Locking von Tabellen unter einigen Betriebssystem (z.B. SunOS)
- Replikation befindet sich noch im BETA Status
- FULLTEXT ist ebenfalls BETA

### Features

MySQL 3.22 kann Tabellen mit einer Größe von bis zu 4G verwalten, seit Version 3.23 wurde diese Grenze auf 8 Millionen Terabytes hochgesetzt. Diese Grenzen werden allerdings noch Betriebssystemabhängig beschränkt.

## 1.2.2 PHP

Was ist PHP? Wozu ist es da? PHP steht als Abkürzung seit ca. 1997 für „PHP: Hypertext Preprocessor“, davor hieß es noch „Personal Home Page Tools“.

PHP ist eine Skriptsprache, d.h. in PHP geschriebene Programme werden zur Laufzeit kompiliert. Die Syntax von PHP ist stark an C, Perl und Java angelehnt. PHP kann in HTML eingebettet werden und somit für die dynamische Generierung von Webseiten genutzt werden. Das wohl wichtigste Feature und damit auch der größte Unterschied gegenüber JavaScript, welches auch als Skriptsprache in HTML eingebunden werden kann, ist der Ort der Kompilierung. Während JavaScript auf Clientseite kompiliert wird, wird der PHP-Code auf dem Server ausgeführt.

Was bedeutet das im Klartext? Auf der einen Seite wird der Quellcode vor den Usern versteckt. Wird über PHP eine Ausgabe erzeugt, wird diese mit in den HTML-Code der Seite gepackt. Auf der anderen Seite kann mit PHP ein viel größerer Funktionsumfang und die damit verbundenen Möglichkeiten angesprochen werden, z.B. Dateihandling auf Serverseite, Datenbankabfragen usw.

### Wie werden PHP Dateien aufgerufen

PHP Dateien liegen grundsätzlich in der selben Ordnerstruktur, wie alle anderen Webdokumente, im Apache als unterhalb von */htdocs*. In der *PHP.ini*-Datei kann angegeben werden, in welchen Verzeichnisse der PHP Prozessor Dateien interpretieren darf.

Die Einbindung in den Apache erfolgt entweder als Modul oder als externe Programmdatei.

Ein einfaches Beispiel für die Verwendung von PHP sieht so aus:

```
<html>
<body>
<?
    echo("<h1>Hello World!!</h1>");
?>
</body>
</html>
```

Als nächstes schauen wir uns mal an, wie man mit PHP eine Verbindung mit einer Datenbank herstellt, eine Anfrage an die Datenbank stellt und die Verbindung beendet.

Abbildung 1.1: PHP Beispiel - Hello World!

```
<?
/* Datenbankbenutzer */
$dbserver= "datenbank.server.tld";
$dbuser= "benutzername";
$dbpassword= "passwort";
$dbdatenbank= "datenbankname";

/* Datenbank verbinden */
MYSQL_CONNECT($dbserver, $dbuser, $dbpassword) or die ( "<H3>Datenbankserver nicht erreichbar</H3>"
MYSQL_SELECT_DB($dbdatenbank) or die ( "<H3>Datenbank nicht vorhanden</H3>");
?>
```

Abbildung 1.2: PHP Beispiel - Datenbankverbindung herstellen

```
<?
echo("Datenbankabfrage wird gestartet");
$query = "SELECT name, vorname FROM users WHERE userid=17;";
$result = MYSQL_QUERY("$query");
while($line=MYSQL_FETCH_ROW($result)) {
    echo("Benutzer $line[0], $line[1] gefunden.");
}
?>
```

Abbildung 1.3: PHP Beispiel - Datenbankabfrage

```
<?
/* Datenbankverbindung trennen */
MYSQL_CLOSE();
?>
```

Hier noch ein paar kurze Erläuterungen zu einigen verwendeten Funktionen:

**MYSQL\_CONNECT(dbserver, dbuser, dbpassword)** Baut die Verbindung zum Datenbankserver auf

**MYSQL\_SELECT\_DB(dbdatenbank)** Wählt eine Datenbank aus, an die die Anfragen gestellt werden

**MYSQL\_QUERY(query) resultset** Stellt die Anfrage an die Datenbank; query ist ein String, der eine korrekt formulierte SQL Anfrage enthalten muss; diese Funktion liefert ein Resultset zurück

**MYSQL\_FETCH\_ROW(result) row** iteriert über das Resultset und liefert mit jedem Aufruf die nächste Ergebnisreihe zurück

Abbildung 1.4: PHP Beispiel - Datenbankverbindung schließen

## 1.3 Oracle und JSP

### 1.3.1 Oracle

Bei Oracle ist mir die Wahl nicht sehr schwer gefallen. Über einen Besuch auf der Oracle-Homepage<sup>4</sup> habe ich mir erst einmal einen Überblick über die vorhandenen Versionen verschafft und so ermittelt, welche Version für meinen Anwendungsbereich in Frage kommt. Für die aktuelle Version 9 des Datenbanksystems gibt es zwei Möglichkeiten: Lite und Personal Edition, letztere ist auch noch für Version 8 erhältlich. Da mir diese Version bereits im Umgang bekannt ist, entscheide ich mich für Oracle 8i Personal Edition Version 8.1.7. In dieser Version sind alle Kinderkrankheiten der Version 8 eliminiert.

Nach einem kurzen Studium des Paketumfanges, stelle ich fest, dass Oracle bereits einen Webserver mitliefert, der JSP unterstützt.

Um sowohl Oracle, als auch den dazugehörigen JSP Server erfolgreich starten zu können, muss noch ein aktuelles J2SDK<sup>5</sup> installiert werden. Dieses ist über die Homepage von Sun[3][4] zu beziehen.

Mit diesen Komponenten habe ich nun alles zusammen, was ich für den Betrieb von Oracle und JSP brauche.

### 1.3.2 JSP

JavaServer Pages, kurz JSP, stellen das Gegenstück zu ASP dar.

JSP sind text-basierte Dokumente, die HTML-Code mit Elementen der Java-Programmiersprache verbinden. Der HTML-Code auf der einen Seite dient dem Design der Seite, er ist statisch. Auf der anderen Seite ist der dynamische Teil in Java programmiert.

*JavaServer Pages sind text-basierte Dokumente, die beschreiben, wie Anfragen von einem Browser an einen Server und Antworten des Servers an den Browser durchzuführen sind.*<sup>6</sup>

Ein einfaches Beispiel für die Verwendung von JSP sieht so aus:

```
<html>
<body>
<%
    out.println("<h1>Hello World!</h1>");
%>
</body>
</html>
```

Abbildung 1.5: JSP Beispiel - Hello World!

Der Aufruf dieses Beispiels zeigt eine Seite an, auf der die Überschrift „Hier eine Überschrift!“ angezeigt wird. Die Funktion **out.println(String str)** erzeugt direkt HTML-Code, der über den HTTP-Stream ausgegeben wird.

Jetzt schauen wir uns noch einmal an, wie man mit JSP eine Datenbankverbindung aufbaut und eine Anfrage an die Datenbank über diese stellt.

---

<sup>4</sup><http://www.oracle.com>

<sup>5</sup>Java 2 Software Development Kit

<sup>6</sup>Das Einsteigerseminar: JavaServer Pages

```
<html>
<head>
<title>DSN Less Connection With Oracle</title>
</head>

<%@ page import="java.sql.*"%>
<%@ page import="oracle.jdbc.OracleConnection"%>

<body>
<%
    Connection conn = null; //Connection String
    Statement stmt = null; //Statement
    ResultSet rset = null; // Resultset

    try {
        DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());

        conn = DriverManager.getConnection(
            "jdbc:oracle:thin:@141.22.10.44:1521:swt02",
            "username",
            "password");
        stmt = conn.createStatement();
        rset = stmt.executeQuery ("select * from cat");

        while (rset.next()) {
            out.println(rset.getString(1)+"<br>");
        }
        rset.close();
        stmt.close();
        conn.close();
    } catch(Exception e) {
        out.println("Verbindung zum Oracle Datenbankserver konnte nicht hergestellt werden!<br>");
        out.println(e);
    } /*finally {
        rset.close();
        stmt.close();
        conn.close();
    }*/
%>
</body>
</html>
```

Abbildung 1.6: JSP Beispiel - Datenbankverbindung aufbauen und eine Anfrage stellen

Hier noch ein paar kurze Erläuterungen zu einigen verwendeten Funktionen:

`<%@ page import="java.sql.*"%>` Dieses Kostrukt wird analog zu den import-Befehlen in Java verwendet

`DriverManager.registerDriver(DatabaseDriver)` hier wird der Datenbanktreiber für die Anwendung registriert

```
DriverManager.getConnection("jdbc:oracle:thin:@141.22.10.44:1521:swt02";"username";"password")
```

Hier wird die Verbindung zur Datenbank aufgebaut

## 1.4 Das Betriebssystem

Als erstes stehe ich hier vor der Frage, ob ich für die beiden Plattformen das jeweils optimale Betriebssystem verwende oder mich auf ein Betriebssystem festlege, welches ich für beide verwende. Da sich hier augenscheinlich die Frage in den Vordergrund wirft, „Was ist optimal“, versuche ich erst einmal herauszufinden, welches Betriebssystem bei den Anwendern den meisten Anklang findet.

Betrachte ich eine Umfrage, die jsp-develop.de Mitte letzten Jahres durchgeführt hat<sup>7</sup>, so ist Windows mit 70% die meistgenutzte Entwicklerplattform. Einer Umfrage von phpwelt.de<sup>8</sup>, stellt sich Windows hier ebenfalls mit fast 70% als meistgenutzte Plattform im Bereich PHP heraus.

Mit über  $\frac{2}{3}$  Anteil ist Windows also Umfragen folgend die meistgenutzte Plattform. Ich wähle also ebenfalls der breiten Masse folgend Windows als Betriebssystem für meine Anwendungen.

---

<sup>7</sup><http://www.jsp-develop.de/umfrage/11/result/>

<sup>8</sup><http://www.phpwelt.de/voting/votes.php?vid=8>

## Kapitel 2

# Installation und Konfiguration

In den folgenden Abschnitten werde ich mich mit der Installation der Plattformen beschäftigen, die Installation von Windows 2000 setze ich dabei als gegeben voraus. Der Weg, den ich hier gehen werde stellt natürlich nur eine Möglichkeit dar und es besteht kein Anspruch darauf, dass er der Einfachste ist.

Die Installationsanleitungen beschreiben kurz alle benötigten Komponenten, eventuelle Probleme bei der Installation und deren Einsatz, sowie einige Punkte, auf die gesondert geachtet werden muss. Im Folgenden erläutere ich noch die verwendeten Abkürzungen.

<**windir**> Windows-Systemverzeichnis, meisten C:\WINNT

<**sysdir**> Windows-Systemverzeichnis, meisten C:\WINNT\SYSTEM32

### 2.1 MySQL und PHP

Um das Rad nicht neu zu erfinden, habe ich mich für das Installationspaket von XAMPP<sup>1</sup> entschieden. XAMPP ist eine Zusammenstellung von Programmen rund um den Apache-Webserver. Der Vorteil für den User ist, dass er sich nicht mit der Installation der einzelnen Komponenten und deren Zusammenspiel und eventueller unliebsamer Fernwirkungen Gedanken machen muss, sondern alles aus einer Hand bekommt. Ich entscheide mich hier für miniXAMPP für Windows, da das große Paket zu viele Programme enthält, die ich gar nicht benötige.

#### 2.1.1 Die Komponenten

miniXAMPP enthält die folgende Programme und Komponenten, die wichtig für uns sind:

**Apache 2.0.49** der eigentlich Webserver

**MySQL 4.0.18** die Datenbank

**PHP 4.3.4** die Programmiersprache

**mod\_php 4.3.4** das Modul, um PHP direkt in den Webserver einbinden zu können

**PHPMyAdmin 2.5.6** ein Konfigurationstool und Administrationstool für MySQL als Webfrontend

Der Download als ausführbare Datei umfasst 26,3 MB.

---

<sup>1</sup><http://www.apachefriends.de/>

### 2.1.2 Die Installation

Nach dem Entpacken der Datei erfolgt eine übersichtliche Konfiguration der einzelnen Komponenten, ein beiliegendes Readme erläutert die einzelnen Schritte und enthält alle voreingestellten Passwörter.

**VORSICHT!** Hier entsteht eine große Sicherheitslücke. Sollte diese Installation in einem zugänglichen Netz verwendet werden, müssen natürlich alle Passwörter geändert werden, da sie sonst weitläufig bekannt sind.

Nach ungefähr 10 Minuten entpacken und 2 Minuten automatischer Konfiguration ist alles erledigt. 3 Stapelverarbeitungsdateien dienen dem Starten und Stoppen des Webservers und der Datenbank.

Zusammenfassend kann man sagen, dass dies eine benutzerfreundliche Installation war. Nach einem kurzen Test der Komponenten kann ich feststellen, dass alles reibungslos funktioniert hat.

## 2.2 Oracle und JSP

### 2.2.1 Die Komponenten

Ähnlich wie die XAMPP Distribution finden wir im Installationspaket alles, was wir für den Betrieb von Oracle und JSP benötigen. Das Datenbankmanagementsystem inkl. einem JSP-Webserver.

Möchte man statt des Oracle Servers einen anderen verwenden (z.B. Apache Tomcat) so ist es erforderlich, die benötigten Datenbanktreiber in dem Java-Klassenverzeichnis des Servers abzulegen.

### 2.2.2 Die Installation

Das Installationsprogramm ist von Grund auf in Java programmiert, was sich schnell im Bildaufbau und den Ladezeiten widerspiegelt. In den folgenden Auswahlmenüs fällt es einem Laien schwer, die richtigen Komponenten für seine Anwendung zu wählen. In einem sehr schwer zu entdeckendem Menüpunkt findet sich jedoch eine gebündelte Paketauswahl, die man seinen Bedürfnissen anpassen kann. Alternativ kann jedoch auch alles installiert werden.

Die Installationszeit selber dauert ca. eine Stunde, bis alle Pakete kopiert sind. Nach Abschluss der Installation muss nur noch der Datenbankdienst gestartet werden und es kann losgelegt werden.

## 2.3 Fazit

Ein Vorteil beider Systeme ist, dass nach der Installation alles funktioniert. Es sind keine manuellen Änderungen nötig, um mit dem Betrieb der Plattformen beginnen zu können. Der Unterschied liegt einzig im Aufwand der Installation.

## Kapitel 3

# Praktischer Einsatz

Für den Praxistest ist der Einsatz von ein und der gleichen Anwendung auf beiden Systemen vorgesehen. Um einen aussagekräftigen Vergleich herzustellen, muss diese auf beiden Systemen den gleichen Funktionsumfang und selbstverständlich die gleiche Datenbankstruktur aufweisen.

### 3.1 Die Anwendung

Um die beiden System nun in der Praxis zu testen, werde ich eine Applikation aufsetzen, die auf beiden Systemen implementiert wird. Da ein Augenmerk bei der Bewertung auf der Kompatibilität der beiden Datenbanksysteme liegt, ist es ein Anliegen, den Aufbau der SQL-Tabellen nur einmal zu machen und auf beiden Systemen zu nutzen, die Datenbank soll austauschbar bzw. portierbar sein, ohne dass große Anpassungen in den Anwendungen vorgenommen werden müssen.

Ziel ist eine kleine beispielhafte Anwendung, die die Grundfunktionen von SQL ausnutzt. Ich habe mich für einen Minishop entschieden, der die folgenden Funktionalitäten unterstützen wird: (siehe auch Anwendungsfall und Aktivitätsdiagramm im Anhang S. 15)

**Artikelkatalog anzeigen** Der Kunde kann sich alle Artikel in einer Liste anzeigen lassen

**Artikel anzeigen** Der Kunde kann einen Artikel auswählen und sich die Details dazu anschauen

**Artikel dem Warenkorb hinzufügen** Der Kunde kann einen Artikel zum Warenkorb hinzufügen, dabei wird überprüft, ob der Artikel noch in der gewünschten Menge vorhanden ist und ggf. eine Fehlermeldung ausgegeben

**Warenkorb anzeigen** Der Kunde kann sich seinen Warenkorb anzeigen lassen und sieht dort auf einen Blick alle ausgewählten Artikel und deren Preise

**Zur Kasse gehen und Warenkorb leeren** Der Kunde geht zur Kasse und der Warenkorb wird geleert.

Für einen voll funktionsfähigen Minishop müssten noch weitere Funktionen wie Warenkorb verändern oder Versanddaten eingeben ergänzt werden, da diese für den Rahmen dieser Arbeit nicht nötig sind, fallen sie weg.

### 3.2 Implementation

#### 3.2.1 MySQL und PHP

Welche Komponenten werden benötigt? Ich stelle kurz vor, welche Komponenten bei der Anwendung zum Einsatz kommen. Ein komplettes Programmlisting befindet sich im Anhang.

Die Anwendung kann unter [www.firegate.de/studienarbeit](http://www.firegate.de/studienarbeit) im Einsatz gesehen werden. Der PHP-Quelltext befindet sich im Anhang (siehe S. 17).

## Der Aufbau der Datenbanktabellen

Für diese Anwendung wird es einen Artikelkatalog und einen Warenkorb geben. Der SQL-Quellcode ist im Anhang zu finden (siehe S. 34).

## Benutzerinterface

Über verschiedene HTML Dateien mit eingebundenen PHP Skripten werden dem User die unterschiedlichen Stufen der Anwendung präsentiert. Daten werden über Formulare übermittelt. In PHP werden Prüfungen vorgenommen, die die Richtigkeit der Aktionen überprüfen (Artikelverfügbarkeit etc...). Dies könnte auch direkt bei der Eingabe über JavaScript passieren, hat jedoch den Nachteil, dass die Prüfungen versagen, wenn JavaScript im Browser deaktiviert ist.

## Datenbankzugriffskripte

Hier verwende ich die bereits auf Seite 1 vorgestellten Skripte, um die Datenbankverbindung aufzubauen. Für den Aufbau und den Erhalt der Datenbankverbindungen habe ich mich entschieden, diese jeweils kurz vor der Anfrage erst herzustellen und nach Abschluss und Verarbeitung der Anfrage gleich wieder zu schließen. Das hat den Vorteil, dass einzelne Verbindungen nicht so lange offen gehalten und blockiert werden. Der Nachteil zu einer Verbindung, die zu Beginn des Skriptes aufgebaut und am Ende geschlossen wird ist, dass bei vielen Abfragen innerhalb eines Skriptes, viele Verbindungen in kurzer Zeit geöffnet und geschlossen werden. Je nach Masse der Anfragen von verschiedenen Benutzer könnte es hier zu Ressourcen Engpässen kommen. Man müsste in einem Belastungstest überprüfen, ob eher viele kleine kurze Anfragen oder eine länger andauernde den Server spürbar blockieren. Für diesen Test fehlen mir jedoch die Kapazitäten.

## Skript zum Verarbeiten der Anfragen

In den einzelnen PHP Skripten werden die übermittelten Daten verarbeitet. Anfragen werden an die Datenbank gestellt, Daten werden eingetragen.

### 3.2.2 Oracle und JSP im Vergleich

Mit JSP und Oracle werde ich eine Objektorientierte Implementierung erstellen. Welche Komponenten werden benötigt?

Die Anwendung ist im Internet unter [firegate.hn.org:8080/jsp-examples/studienarbeit/](http://firegate.hn.org:8080/jsp-examples/studienarbeit/) anzuschauen. Der komplette Quelltext befindet sich im Anhang (siehe S. 26).

Um rein programmiertechnisch einen guten Vergleich zu der PHP Anwendung ziehen zu können und um auf die Weise die Portabilität des Quellcodes zu untersuchen, habe ich die Programmierung ähnlich angesetzt, d.h. auf Klassenbildung verzichtet.

In den folgenden Kapitel beschreibe ich nun also nicht nur die Programmierung der JSP/Oracle Variante der Anwendung, sondern werde gleich die Unterschiede im Vergleich aufzeigen.

## Datenbankzugriffsklasse

Der erste große Unterschied ist bereits die Handhabung der Requestvariablen, d.h. die Formulare Daten, die mittels POST bzw. GET übermittel werden. Hat man in PHP sofort Zugriff darauf, müssen diese für JSP erst einmal in Variablen eingelesen werden. Ab dann steht sie Kontextweit zur Verfügung.

```
<%  
    String name = request.getParameter("name");  
>%
```

Abbildung 3.1: JSP Beispiel - Übernehmen von POST/GET-Daten

Wenn man mit einem Formular sehr viele Daten verwendet, dann ist es sehr aufwendig auf die Art und Weise die Formulardaten zuzuordnen. Statt dessen würde man mit einem Feld arbeiten (*Enumeration*  $e = request.getParameterNames()$ ).

Ein weiterer Nachteil der *getParameter()*-Methode gegenüber PHP ist, dass die Daten nur als String vorliegen und für eine Integer-Verwendung erst umgewandelt werden müssen.

```
<%  
    int ianzahl = (new Integer(request.getParameter("anzahl"))).intValue();  
%>
```

Abbildung 3.2: JSP Beispiel - Typenkonvertierung von POST/GET-daten

Ein sehr angenehmer Vorteil von JSP, den ich bei PHP nicht kenne, ist die Möglichkeit mit einem bestimmten Tag eine HTML Ausgabe eines Ausdruckes zu erzeugen. Der komplizierte Umweg über die *out.println()*-Funktion bleibt einem erspart.

```
Guten Tag Herr <%= stringValue%>, wie geht es ihnen?  
Heute ist der <%= mydate.toString()%>
```

Abbildung 3.3: JSP Beispiel - Ausdruck-Tag

## Benutzerinterface

DasBenutzerinterface, sprich das HTML Gerüst kann 1:1 übernommen werden, keine Unterschiede.

# Kapitel 4

## Fazit

Kommen wir nun zum letzten Kapitel der Studienarbeit. Hier werde ich nun die Ergebnisse der Untersuchung zusammenfassen und bewerten.

### 4.1 Tabellarische Übersicht

Zu den einzelnen Bewertungskriterien: + steht für viel, - dementsprechend für wenig. Ich habe mich bewusst gegen die Definition von + als positiv entschieden, da es z.B. sehr subjektiv ist, ob eine große Menge von zusätzlichen Tools positiv oder negativ zu bewerten ist, das liegt schließlich im Auge des Benutzers. Wichtig zu wissen ist, dass ich die beiden Systeme nicht unabhängig und allgemein bewertet habe, sondern im Vergleich zueinander. Z.B. bedeutet 0 beim Funktionsumfang für PHP & MySQL nicht, dass sie nur einen durchschnittlichen Funktionsumfang haben, sondern dass der Funktionsumfang im Vergleich zu JSP & Oracle durchschnittlich ist.

Kriterium	Plattform	
	PHP & MySQL	JSP & Oracle
Installationsaufwand	-	+
Kosten	-	+
Funktionsumfang	0	+
Verwaltungstools	0	+
Referentielle Integrität	-	+

Als nächstes beleuchte ich noch einige Kriterien genauer.

### 4.2 Installationsaufwand

Wie ich bereits im Kapitel Installation und Konfiguration (siehe S. 7) erwähnt habe, geht die Konfiguration von PHP & MySQL mehr als einfach von statten. Innerhalb weniger Minuten ist das System installiert und einsatzbereit.

Der eigentlich Installationsvorgang von Oracle geht ähnlich einfach von statten. Bis man jedoch dort angekommen ist, war es ein langr Weg. Schon im ersten Setup-Konfigurationsmenü wird man von der Fülle der auswählbaren Funktionen erschlagen. Für einen Anfänger ist hier schwer zu erkennen, was er davon genau und minimal zum Betrieb braucht. Negativ aufgefallen ist ebenfalls die ziemlich lange Installationsdauer. Doch auch Oracle inkl. JSP-Server sind nach der Installation sofort einsatzbereit.

Wie bereits im Kapitel

### 4.3 Kosten

Betrachtet man die reinen Anschaffungskosten, so hat MySQL die Nase vorne. MySQL steht kostenlos zur freien Verfügung. Die Oracle Datenbankserver können zwar auch kostenlos heruntergeladen und installiert, dürfen jedoch nicht öffentlich zugänglich genutzt werden. Oracle schlägt als Minimum mit ca. EUR 600 zu Buche.

### 4.4 Verwaltungstools / Tools allgemein

Betrachtet man MySQL alleine, so gibt es nur eine kleine Anzahl Tools, die dem Datenbankmanagementsystem beigelegt sind. In der XAMPP Distribution sieht das schon etwas anders aus. Dort verfügt man z.B. über PHPMysqlAdmin, ein PHP-Programm mit Webfrontend zum Management der Datenbank. Hier kann so ziemlich alles gemacht werden (SQL Abfragen, Table Optimierung, Alter Table, Datenbanken erstellen / löschen).

Die Oracle Distribution hingegen kommt mit einem großen Umfang an Tools, um die Arbeit rund um das Oracle Datenbankmanagementsystem zu vereinfachen. Es gibt Tools zum Management der Datenbank und der User, zur Programmierung der Oracle Funktionen in PL/SQL (Trigger etc...), einfache Tools zur Eingabe von SQL Statements ... In meinen Augen fast schon zu viele Tools, so dass man öfters mal nicht genau weiß, welche der vielen Möglichkeiten man nun für die nächste Aufgabe nutzen möchte. Obwohl es mehrere Tools gibt die für die gleiche Aufgabe benutzt werden können, hat jedes Tool seine Eigenarten.

---

# Teil I

# Anhang

# Diagramme

## 4.1 Anwendungsfall

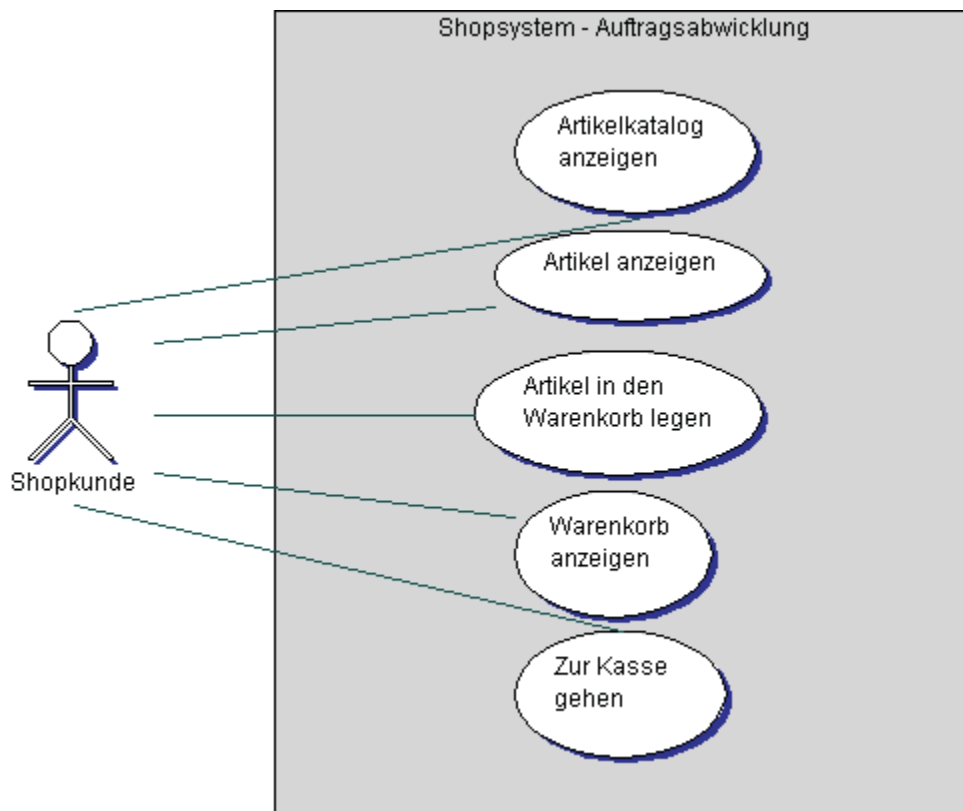


Abbildung 4.1: Anwendungsfall - UML Diagramm

## 4.2 Aktivitätsdiagramm

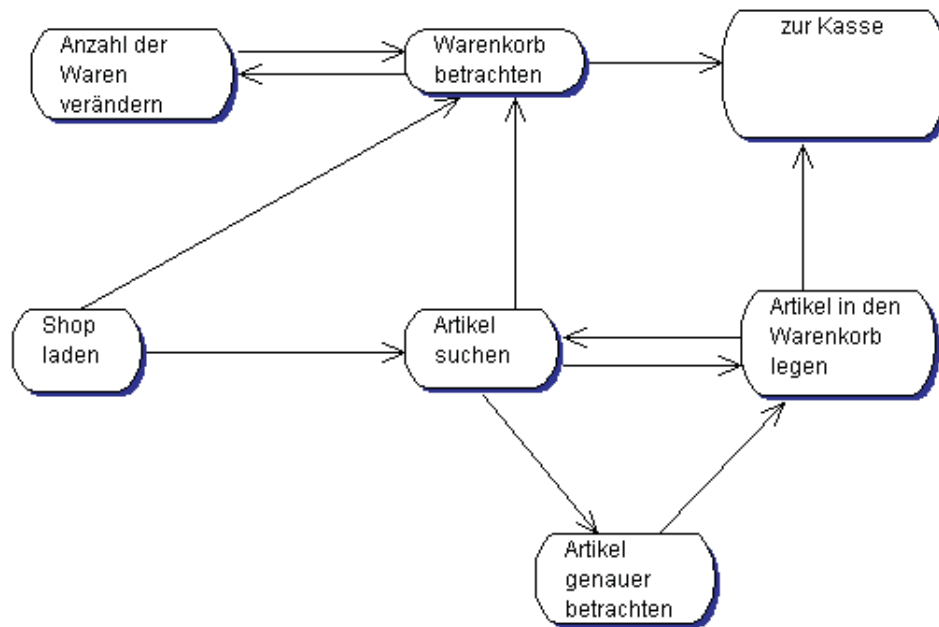


Abbildung 4.2: Aktivitätsdiagramm - UML Diagramm

# Quelltexte

## 4.3 PHP

### 4.3.1 connectdb.php - Datenbankverbindung herstellen

```
<?
$dbserver= "xx";
$dbuser= "xx";
$dbpassword= "xx";
$dbdatenbank= "xx";
MYSQL_CONNECT($dbserver, $dbuser, $dbpassword)
    or die ( "<H3>Datenbankserver nicht erreichbar</H3>");
MYSQL_SELECT_DB($dbdatenbank)
    or die ( "<H3>Datenbank nicht vorhanden</H3>");
?>
```

### 4.3.2 disconnectdb.php - Datenbankverbindung beenden

```
<?
    MYSQL_CLOSE();
?>
```

### 4.3.3 functions.php - Häufig benutzte Funktionen

```
<?
function connectdb() {
    include("connectdb.php");
}

function disconnectdb() {
    include("disconnectdb.php");
}

function getArtikelzahl() {
    connectdb();
    $q_anzahlartikel = "SELECT count(*) FROM studienarbeit_warenkorb;";
    $r_anzahlartikel = MYSQL_QUERY("$q_anzahlartikel")
        or die("<b>Fehler bei der Ausführung:</b> ".MYSQL_ERROR());
    $line = MYSQL_FETCH_ROW($r_anzahlartikel);
    $anzahl = $line[0];
    disconnectdb();
    return $anzahl;
}

function warenkorb_leeren() {
    connectdb();
    $query = "DELETE FROM studienarbeit_warenkorb;";
    $result = MYSQL_QUERY("$query") or die("<b>Fehler bei der Ausführung:</b> ".MYSQL_ERROR());
    disconnectdb();
}
?>
```

### 4.3.4 artikel.php - Der Artikelkatalog

```
<?
  include("functions.php");
  if(isset($buy)) include("buy.inc.php");
?>
<h1>Artikelkatalog</h1>
<hr>

<?
  echo("<font color=\"#FF0000\">$error</font>");

  if(isset($afterbuy)) { ?>
    Artikel wurde dem Warenkorb hinzugefügt.
  <? }
  if(isset($show)) include("show.inc.php");
  if(isset($show_details)) include("show_details.inc.php");
?>

<hr>
Sie haben <? echo(getArtikelzahl()); ?> verschiedene Artikel im Warenkorb
- <a href="artikel.php?show">Zum Artikelkatalog</a>
- <a href="warenkorb.php?show">Zum Warenkorb</a>
- <a href="kasse.php">Zur Kasse</a>
- <a href="index.html">Zurück zum Anfang</a>
```

**show.inc.php - alle Artikel anzeigen**

```

<h3>Klicken sie auf einen Artikel f&uuml;r eine Detailbeschreibung</h3>
<table width="95%">
  <tr>
    <th align="left" valign="top">Artikelnummer</th>
    <th align="left" valign="top">Artikelbezeichnung</th>
    <th align="left" valign="top">Kurzbeschreibung</th>
    <th align="right" valign="top">Preis in cent</th>
  </tr>

<?
function artikelzeile($artikelnummer, $bezeichnung, $kurzbeschreibung, $preis) {
  echo("<form method=\"get\">\n");
  echo("<input type=\"hidden\" name=\"id\" value=\"$artikelnummer\">\n");
  echo("<td align=\"left\" valign=\"top\">$artikelnummer</td>\n");
  echo("<td align=\"left\" valign=\"top\">
    <a href=\"artikel.php?show_details&id=$artikelnummer\">$bezeichnung</a></td>\n");
  echo("<td align=\"left\" valign=\"top\">$kurzbeschreibung</td>\n");
  echo("<td align=\"right\" valign=\"top\">$preis</td>\n");
  echo("<td>&nbsp;</td>");
  echo("<td align=\"left\" valign=\"top\">
    <input type=\"text\" name=\"anzahl\" value=\"1\" size=\"3\">");
  echo("<input type=\"hidden\" name=\"quelle\" value=\"show\">");
  echo("<input type=\"submit\" name=\"buy\" value=\"St&uuml;ck kaufen\"></td>\n");
  echo("</form>\n");
}
connectdb();
$q_articles = "SELECT id, bezeichnung , kurzbeschreibung, preis_in_cent
              FROM studienarbeit_artikel ORDER BY bezeichnung;";
$r_articles = MYSQL_QUERY("$q_articles")
  or die("<b>Fehler bei der Ausf&uuml;hrung:</b> ".MYSQL_ERROR());
while($line = MYSQL_FETCH_ROW($r_articles)) {
  echo("<tr>\n");
  artikelzeile($line[0],$line[1],$line[2],$line[3]);
  echo("</tr>\n");
}
disconnectdb();
?>

</table>

```

**show\_details.php - Artikel detailliert anzeigen**

```
<?
connectdb();
$q_articles = "SELECT id, bezeichnung , kurzbeschreibung, beschreibung, preis_in_cent
              FROM studienarbeit_artikel
              WHERE id=$id
              ORDER BY bezeichnung;";
$r_articles = MYSQL_QUERY("$q_articles")
  or die("<b>Fehler bei der Ausführung:</b> ".MYSQL_ERROR());
while($line = MYSQL_FETCH_ROW($r_articles)) { ?>
  <h3><? echo("$line[1]"); ?></h3>
  <p><strong>Artikelnummer</strong>: <? echo("$line[0]"); ?></p>
  <p><strong>Kurzbeschreibung</strong>: <? echo("$line[2]"); ?></p>
  <p><strong>Beschreibung</strong>: <? echo("$line[3]"); ?></p>
  <p><strong>Preis</strong>: <? echo("$line[4] cent"); ?></p>
  <p><form method="\get\"><input type="text" name="anzahl" value="1" size="3">
    <input type="hidden" name="quelle" value="show_details&id=<? echo("$line[0]"); ?>">
    <input type="hidden" name="id" value="<? echo("$line[0]"); ?>">
    <input type="submit" name="buy" value="Stück kaufen"></form>
  </p>

<? }
disconnectdb();
?>
```

### 4.3.5 buy.inc.php - Kaufverarbeitung

```
<?
// Anzahl Syntax überprüfen
$anzahl = intval($anzahl);
if($anzahl <= 0) {
    $error = "Es wurde ein falscher Wert für die Anzahl verwendet, bitte korrigieren!";
    header("Location: artikel.php?error=$error&$quelle");
} else {
    connectdb();
    $q_anzahl = "SELECT anzahl FROM studienarbeit_artikel WHERE id=$id;";
    $r_anzahl = MYSQL_QUERY("$q_anzahl")
    or die("<b>Fehler bei der Ausführung:</b> ".MYSQL_ERROR());
    $line = MYSQL_FETCH_ROW($r_anzahl);
    $verfuegbar = $line[0];
    if($verfuegbar < $anzahl) {
        $error = "Es wurden $anzahl ausgewählt, jedoch sind nur
            noch $verfuegbar verfuegbar! Bitte Anzahl korrigieren.";
        header("Location: artikel.php?error=$error&$quelle");
        include("disconnectdb.php");
    } else {
        $q_vorhanden = "SELECT anzahl
            FROM studienarbeit_warenkorb
            WHERE artikel = $id;";
        $r_vorhanden = MYSQL_QUERY($q_vorhanden)
        or die("<b>Fehler bei der Ausführung:</b> ".MYSQL_ERROR());
        $line = MYSQL_FETCH_ROW($r_vorhanden);
        if($line[0] > 0) {
            $anzahl += $line[0];
            $q_einpacken = "UPDATE studienarbeit_warenkorb
                SET anzahl=$anzahl
                WHERE artikel=$id;";
        } else {
            $q_einpacken = "INSERT INTO studienarbeit_warenkorb(artikel, anzahl)
                VALUES($id,$anzahl);";
        }
        $r_einpacken = MYSQL_QUERY($q_einpacken)
        or die("<b>Fehler bei der Ausführung:</b> ".MYSQL_ERROR());
        header("Location: artikel.php?afterbuy");
    }
    disconnectdb();
}
?>
```

### 4.3.6 warenkorb.php - Warenkorb anzeigen

```

<?
    include("functions.php");
    if(isset($clear)) warenkorb_leeren();
?>

<h1>Warenkorb</h1>
<hr>
<?
    echo("<font color=\`#\`FF0000\`>$error</font>");
    if(isset($clear)) echo("Warenkorb geleert.");
    if(isset($show)) {
        connectdb();
        $q_warenkorb = "SELECT a.id, a.bezeichnung, a.preis_in_cent, w.anzahl
                        FROM studienarbeit_warenkorb w, studienarbeit_artikel a
                        WHERE w.artikel = a.id
                        ORDER BY a.id;";
        $r_warenkorb = MYSQL_QUERY($q_warenkorb)
            or die("Error: ".MYSQL_ERROR);
        $anzahl = MYSQL_NUM_ROWS($r_warenkorb);
        if($anzahl <= 0) echo("Keine Artikel im Warenkorb");
        else {
            $summe = 0;
            echo("<table>");
            echo("<th align=right>Anzahl</th>");
            echo("<th align=right>Artikelnummer</th>");
            echo("<th align=left>Bezeichnung</th>");
            echo("<th align=right>Einzelpreis</th>");
            echo("<th align=right>Gesamt</th>");
            while($line = MYSQL_FETCH_ROW($r_warenkorb)) {
                echo("<tr>");
                echo("<td><input type=text size=3 value=$line[3]></td>");
                echo("<td align=right>$line[0]</td>");
                echo("<td align=left>$line[1]</td>");
                echo("<td align=right>$line[2] cent</td>");
                echo("<td align=right>".($line[2]*$line[3])." cent</td>");
                echo("</tr>");
            }
            echo("</table>");
        }
    }
?>

<hr>
<a href="warenkorb.php?clear">Warenkorb leeren</a>
- <a href="artikel.php?show">Zum Artikelkatalog</a>
- <a href="kasse.php">Zur Kasse</a>
- <a href="index.html">Zurück zum Anfang</a>

```

### 4.3.7 kasse.php - Zur Kasse gehen, Einkauf abschließen

```
<?
  include("functions.php");
  if(isset($buy)) include("buy.inc.php");
?>

<h1>Kasse</h1>
<hr>

<? echo("<font color=\"\#FF0000\">$error</font>"); ?>

Herzlichen Glückwunsch, sie haben die Artikel gekauft.

<? warenkorb_leeren(); ?>

<hr>
<a href="artikel.php?show">Zum Artikelkatalog</a>
- <a href="index.html">Zurück zum Anfang</a>
```

### 4.3.8 index.html - Startseite

```
<h1>Willkommen beim Teleshopping!</h1>
<h3>Was machen sie tun?</h3>
<ul>
  <li><a href="artikel.php?show">Die Artikel im Katalog anschauen</a></li>
  <li><a href="warenkorb.php?show">Meinen Warenkorb betrachten</a></li>
  <li><a href="kasse.php">Zur Kasse gehen</a></li>
</ul>
```

## 4.4 JSP

### 4.4.1 artikel.jsp - Der Artikelkatalog

```
<%@ page import="java.sql.*"%>
<%@ page import="oracle.jdbc.OracleConnection"%>
<%
    Connection conn = null;
    Statement stmt = null;
    ResultSet rset = null;
    try {
        conn = DriverManager.getConnection(
            "jdbc:oracle:thin:@141.22.10.44:1521:swt02","scott","tiger");
        stmt = conn.createStatement();
    }
    catch(Exception e) {
        out.println("Verbindung zum Oracle Datenbankserver
            konnte nicht hergestellt werden!<br>");
        out.println(e);
    }
    String error = "";
    String buy = request.getParameter("buy");
    String aid = request.getParameter("id");
    try {
        if(buy!=null) {
            String anzahl = request.getParameter("anzahl");
            int ianzahl = (new Integer(anzahl)).intValue();
            if(ianzahl <= 0) {
                error = "Es wurde ein falscher Wert für die Anzahl verwendet, bitte korrigieren!";
            } else {
                String getAnzahl = "SELECT anzahl FROM studienarbeit_artikel WHERE id="+aid;
                rset = stmt.executeQuery (getAnzahl);
                int verfuegbar=0;
                while(rset.next()) verfuegbar = rset.getInt(1);
                if(verfuegbar < ianzahl) {
                    error = "Es wurden "+ianzahl+" ausgewählt, jedoch sind nur noch "
                        +verfuegbar+" verfuegbar! Bitte Anzahl korrigieren.";
                } else {
                    String vorhanden = "SELECT anzahl FROM studienarbeit_warenkorb WHERE artikel = "+aid;
                    rset = stmt.executeQuery (vorhanden);
                    boolean check = rset.next();
                    if(check == false) out.println("null");
                    String einpacken = "";
                    if((check) && (rset.getInt(1) > 0)) {
                        ianzahl += rset.getInt(1);
                        einpacken = "UPDATE studienarbeit_warenkorb SET anzahl="
                            +ianzahl+" WHERE artikel="+aid;
                    } else {
                        einpacken = "INSERT INTO studienarbeit_warenkorb(artikel, anzahl)
                            VALUES("+aid+", "+anzahl+")";
                    }
                    rset = stmt.executeQuery (einpacken);
                }
            }
        }
    }
    catch(Exception e) {
        out.println("Fehler bei der Verarbeitung der Anfrage: "+e);
    }
}
```

```

        }
    }
} catch(Exception e) {out.println("fehler beim kaufen<br>");out.println(e);}
%>

<body>
<h1>Artikelkatalog</h1>
<hr>

<%
    if(!error.equalsIgnoreCase("")) out.println("<font color=\"\#FF0000\">"+error+"</font>");

    String show = request.getParameter("show");
    if(show.equalsIgnoreCase("1")) {
        String afterbuy = request.getParameter("afterbuy");
        if(afterbuy.equalsIgnoreCase("1")) out.println("Artikel wurde dem Warenkorb hinzugefuegt.");
    }
%>

<h3>Klicken sie auf einen Artikel f&uuml;r eine Detailbeschreibung</h3>
<table width="95%">
    <tr>
        <th align="left" valign="top">Artikelnummer</th>
        <th align="left" valign="top">Artikelbezeichnung</th>
        <th align="left" valign="top">Kurzbeschreibung</th>
        <th align="right" valign="top">Preis in cent</th>
    </tr>

<%
    try {
        String getArticles = "SELECT id, bezeichnung , kurzbeschreibung, preis_in_cent
                                FROM studienarbeit_artikel
                                ORDER BY bezeichnung";
        rset = stmt.executeQuery (getArticles);
        while (rset.next()) {
            out.println("<tr>\n");
            out.println("<form method=\"get\">\n");
            out.println("<input type=\"hidden\" name=\"id\" value="+rset.getString(1)+">\n");
            out.println("<td align=\"left\" valign=\"top\">"+rset.getString(1)+"</td>\n");
            out.println("<td align=\"left\" valign=\"top\">");
            out.println("<a href=artikel.jsp?show=2&id="+rset.getString(1)+">");
            out.println(rset.getString(2));
            out.println("</a>");
            out.println("</td>\n");
            out.println("<td align=\"left\" valign=\"top\">"+rset.getString(3)+"</td>\n");
            out.println("<td align=\"right\" valign=\"top\">"+rset.getString(4)+"</td>\n");
            out.println("<td>&nbsp;&nbsp;&nbsp;</td>");
            out.println("<td align=\"left\" valign=\"top\">
                <input type=\"text\" name=\"anzahl\" value=\"1\" size=\"3\">");
            out.println("<input type=\"hidden\" name=\"quelle\" value=\"show\">");
            out.println("<input type=\"hidden\" name=\"show\" value=\"1\">");
            out.println("<input type=\"hidden\" name=\"afterbuy\" value=\"1\">");

```

```

        out.println("<input type=\"submit\" name=\"buy\" value=\"Stück kaufen\"></td>\n");
        out.println("</form>\n");
        out.println("</tr>\n");
    }
} catch(Exception e) {
    out.println("Verbindung zum Oracle Datenbankserver konnte nicht hergestellt werden!<br>");
    out.println(e);
} finally {
    rset.close();
    stmt.close();
    conn.close();
}
}
out.println("</table>");
}

if(show.equalsIgnoreCase("2")) {
    String id = request.getParameter("id");
    String getArticle = "SELECT id, bezeichnung , kurzbeschreibung, beschreibung, preis_in_cent
                        FROM studienarbeit_artikel
                        WHERE id="+id+" ORDER BY bezeichnung";
    rset = stmt.executeQuery (getArticle);
    while (rset.next()) {%>
        <h3><%=rset.getString(2)%></h3>
        <p><strong>Artikelnummer</strong>: <%=rset.getString(1)%></p>
        <p><strong>Kurzbeschreibung</strong>: <%=rset.getString(3)%></p>
        <p><strong>Beschreibung</strong>: <%=rset.getString(4)%></p>
        <p><strong>Preis</strong>: <%=rset.getString(5)%> cent</p>
        <p><form method="get"><input type="text" name="anzahl" value="1" size="3">
        <input type="hidden" name="quelle" value="show_details&id=<%=rset.getString(1)%>">
        <input type="hidden" name="id" value="<%=rset.getString(1)%>">
        <input type="hidden" name="show" value="1">
        <input type="hidden" name="afterbuy" value="1">
        <input type="submit" name="buy" value="Stück kaufen"></form></p>
    <%}
}
%>

<hr>
Sie haben <? echo(getArtikelzahl()); ?> verschiedene Artikel im Warenkorb
- <a href="artikel.jsp?show=1&afterbuy=0">Zum Artikelkatalog</a>
- <a href="warenkorb.jsp?show=1">Zum Warenkorb</a>
- <a href="kasse.jsp">Zur Kasse</a>
- <a href="index.html">Zurück zum Anfang</a>

```

### 4.4.2 warenkorb.jsp - Warenkorb anzeigen

```

<%@ page import="java.sql.*"%>
<%@ page import="oracle.jdbc.OracleConnection"%>
<%
    Connection conn = null; // Connection String
    Statement stmt = null; // Statement
    ResultSet rset = null; // Resultset
    try {
        conn = DriverManager.getConnection(
            "jdbc:oracle:thin:@141.22.10.44:1521:swt02","scott","tiger");
        stmt = conn.createStatement();
    } catch(Exception e) {
        out.println("Verbindung zum Oracle Datenbankserver
            konnte nicht hergestellt werden!<br>");
        out.println(e);
    }
%>

<h1>Warenkorb</h1>
<hr>

<%
String show = request.getParameter("show");
if(show!=null) { /* Show-Anfang */
    String warenkorb = "SELECT a.id, a.bezeichnung, a.preis_in_cent, w.anzahl
        FROM studienarbeit_warenkorb w, studienarbeit_artikel a
        WHERE w.artikel = a.id ORDER BY a.id";

    rset = stmt.executeQuery(warenkorb);
    int summe = 0;
    int gesamt = 0;
    out.println("<table>");
    out.println("<th align=right>Anzahl</th>");
    out.println("<th align=right>Artikelnummer</th>");
    out.println("<th align=left>Bezeichnung</th>");
    out.println("<th align=right>Einzelpreis</th>");
    out.println("<th align=right>Gesamt</th>");
    while(rset.next()) {
        out.println("<tr>");
        out.println("<td><input type=text size=3 value="+rset.getString(4)+"></td>");
        out.println("<td align=right>"+rset.getString(1)+"</td>");
        out.println("<td align=left>"+rset.getString(2)+"</td>");
        out.println("<td align=right>"+rset.getString(3)+" cent</td>");
        gesamt = rset.getInt(3)*rset.getInt(4);
        summe += gesamt;
        out.println("<td align=right>"+gesamt+" cent</td>");
        out.println("</tr>");
    }
    out.println("<tr><td colspan=4 align=right><b>Gesamtpreis:</b></td>");
    out.println("<td align=right>"+summe+" cent</td></tr>");
    out.println("</table>");
    if(summe==0) out.println("Warenkorb leer");
}

```

```
    }  
%>  
  
<hr>  
<a href="warenkorb.php?clear">Warenkorb leeren</a>  
- <a href="artikel.jsp?show=1&afterbuy=0">Zum Artikelkatalog</a>  
- <a href="kasse.jsp">Zur Kasse</a>  
- <a href="index.html">Zurück zum Anfang</a>
```

### 4.4.3 create\_tables.jsp - Datenbanktabellen erstellen

```
<html>
<head>
<title>DSN Less Connection With Oracle</title>
</head>

<%@ page import="java.sql.*"%>
<%@ page import="oracle.jdbc.OracleConnection"%>

<body>
<%
    Connection conn = null; // Connection String
    Statement stmt = null; // Statement
    ResultSet rset = null; // Resultset

    String create_studienarbeit_artikel =
        "CREATE TABLE studienarbeit_artikel(
            id int NOT NULL primary key,
            bezeichnung char(25) NOT NULL,
            kurzbeschreibung char(250) NULL,
            beschreibung char(1000) NULL,
            preis_in_cent int NOT NULL,
            anzahl int NOT NULL)";

    String create_studienarbeit_warenkorb =
        "CREATE TABLE studienarbeit_warenkorb(
            artikel int NOT NULL references studienarbeit_artikel primary key,
            anzahl int NOT NULL)";

    // An dieser Stelle folgen eine Reihe von Insert
    // Statements, um den Katalog zu füllen

    try {
        DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
        conn = DriverManager.getConnection(
            "jdbc:oracle:thin:@141.22.10.44:1521:swt02","scott","tiger");
        stmt = conn.createStatement();
        out.println("Tabellen für die Anwendung werden angelegt.<br>");
        out.println("Erstelle art_id<br>");
        try {
            checker = stmt.execute("create sequence art_id");
        } catch(SQLException e) {out.println(e+"<br>");}
        out.println("Erstelle studienarbeit_artikel<br>");
        try {
            checker = stmt.execute(create_studienarbeit_artikel);
        } catch(SQLException e) {out.println(e+"<br>");}
        out.println("Füge Artikel ein<br>");
        try {
            checker = stmt.execute(insert_artikel1);
            checker = stmt.execute(insert_artikel2);
            checker = stmt.execute(insert_artikel3);
```

```
    } catch(SQLException e) {out.println(e+"<br>");}
    out.println("Erstelle studienarbeit_warenkorb<br>");
    try {
        checker = stmt.execute(create_studienarbeit_warenkorb);
    } catch(SQLException e) {out.println(e+"<br>");}
    rset = stmt.executeQuery ("select * from studienarbeit_warenkorb");
    while (rset.next()) {
        out.println(rset.getString(1)+"<br>");
    }

} catch(Exception e) {
    out.println("Verbindung zum Oracle Datenbankserver konnte nicht hergestellt werden!<br>");
    out.println(e);
} finally {
    if(rset!=null) rset.close();
    if(stmt!=null) stmt.close();
    if(conn!=null) conn.close();
}
%>
</body>
</html>
```

#### 4.4.4 index.html - Startseite

```
<h1>Willkommen beim Teleshopping!</h1>
<h3>Was m&ouml;chten sie tun?</h3>
<ul>
  <li><a href="artikel.php?show">Die Artikel im Katalog anschauen</a></li>
  <li><a href="warenkorb.php?show">Meinen Warenkorb betrachten</a></li>
  <li><a href="kasse.php">Zur Kasse gehen</a></li>
</ul>
```

## 4.5 SQL

Hier folgt nun der SQL-Quellcode für die Datenbanktabellen. Die Unterschiede zu Oracle habe ich ausgezeichnet.

### 4.5.1 studienarbeit\_artikel.sql - Artikelkatalog

#### MySQL:

```
CREATE TABLE studienarbeit_artikel (  
  id int(11) NOT NULL auto_increment,  
  bezeichnung varchar(100) NOT NULL default '',  
  kurzbeschreibung varchar(250) default NULL,  
  beschreibung longtext,  
  preis_in_cent int(11) NOT NULL default '0',  
  anzahl int(11) NOT NULL default '0',  
  PRIMARY KEY (id),  
  UNIQUE KEY bezeichnung (bezeichnung)  
) TYPE=MyISAM;
```

**Oracle:** Das auto\_increment von MySQL wird bei Oracle über eine Sequence gelöst. Beim Einfügen von Datensätzen wird diese über sequence.nextval angesprochen.

```
CREATE SEQUENCE art_id;  
CREATE TABLE studienarbeit_artikel (  
  id int NOT NULL,  
  bezeichnung varchar(100) NOT NULL,  
  kurzbeschreibung varchar(250) NULL,  
  beschreibung varchar(1000),  
  preis_in_cent int NOT NULL,  
  anzahl int NOT NULL,  
  PRIMARY KEY (id),  
  UNIQUE KEY bezeichnung (bezeichnung)  
)
```

## 4.5.2 studienarbeit\_warenkorb.sql - Warenkorb

### MySQL:

```
CREATE TABLE studienarbeit_warenkorb (  
  artikel int(11) NOT NULL default '0',  
  anzahl int(11) NOT NULL default '0',  
  PRIMARY KEY (artikel)  
) TYPE=MyISAM;
```

**Oracle:** Bei Oracle ist es im Gegensatz zu MySQL möglich referentielle Zusammenhänge herzustellen.

```
CREATE TABLE studienarbeit_warenkorb (  
  artikel int NOT NULL REFERENCES studienarbeit_artikel,  
  anzahl int NOT NULL,  
  PRIMARY KEY (artikel)  
)
```

# Literaturverzeichnis

- [1] PHP, Quelle im Internet: <http://www.php.net/>
- [2] JSP, Quelle im Internet: <http://java.sun.com/products/jsp/>
- [3] SUN, Quelle im Internet: <http://www.sun.com>
- [4] Java von SUN, Quelle im Internet: <http://java.sun.com>
- [5] apache-Webserver leicht gemacht, Quelle im Internet <http://www.apachefriends.de/> (letzter Zugriff 31.3.2004)

---

# Abbildungsverzeichnis

1.1	PHP Beispiel - Hello World! . . . . .	3
1.2	PHP Beispiel - Datenbankverbindung herstellen . . . . .	3
1.3	PHP Beispiel - Datenbankanfrage . . . . .	3
1.4	PHP Beispiel - Datenbankverbindung schließen . . . . .	4
1.5	JSP Beispiel - Hello World! . . . . .	4
1.6	JSP Beispiel - Datenbankverbindung aufbauen und eine Anfrage stellen . . . . .	5
3.1	JSP Beispiel - Übernehmen von POST/GET-Daten . . . . .	11
3.2	JSP Beispiel - Typenkonvertierung von POST/GET-daten . . . . .	11
3.3	JSP Beispiel - Ausdruck-Tag . . . . .	11
4.1	Anwendungsfall - UML Diagramm . . . . .	15
4.2	Aktivitätsdiagramm - UML Diagramm . . . . .	16

# Index

## B

Business, ii

## D

Datenbankanwendung, ii

## I

Installation, ii

Implementierung, ii

Installation, ii

Internetplattform, ii

## J

JSP, ii

## K

Konfiguration, ii

Konfiguration, ii

## M

MySQL, ii, 1, 2

## O

Oracle, ii

## P

PHP, ii, 2

## W

webbasiert, ii