Hans Kern

Arbeitsanleitung
Praktikum Datenbanken und Informationssysteme

SS 2012

Inhaltsverzeichnis

Praktikum Datenbanken und Informationssysteme Arbeitsanleitung für das SS12

Das Wichtigste zuerst - zum Ausschneiden und Dabeihaben

Die Arbeitsanleitung will kein Lehrbuch sein. Ich möchte nur auf Schwierigkeiten hinweisen, die bei der gleichzeitigen Verwendung unterschiedlicher Sprachen (HTML, PHP, MySQL, CSS, JS) in einem Projekt entstehen.

1. **Projektkonzeption** zuerst!

2. In Word nur **selbstdefinierte** Formatierungen, die Sie auch für Ihre BA nutzen können!

3. Nach jeder Arbeit **Tagesberichte** fortführen!

4. **Ordnerstruktur** des Projektes einhalten!

5. **Programmierstile** für MySQL, XHTML, PHP, CSS und JS beachten!

6. In allen Codierungen Kommentare und Bezeichner in **Englisch**!

7. In allen Codierungen für Einrückungen nur **Tabs**, nicht Leerzeichen verwenden; ansonsten keine Tabs!

8. Eine einzige **MySQL-Ladedatei** für alle **5(!)** Tabellen, zu jeder Tabelle nur ein Insert!

9. MySQL-Abfragen zunächst **ohne XHTML-Oberfläche** testen und dokumentieren!

10. **Fehlermeldungen** in C:\xampp\apache\logs\error.log müssen behoben werden; desgleichen Fehlermeldungen in der Browser-Fehlerkonsole. Error-Reporting ändern!

11. Ein- und Ausgaben in fehlerfreiem Deutsch mit **ß** und **Umlauten** und korrekte Buchstaben in anderen Sprachen, z.B. **ç**!

12. Wo erforderlich **transparente** Logos, Wappen der Stadt besorgen!

13. Sobald erste XHTML-Seite fertig ist, **sofort** XHTML und CSS validieren! Ebenso **von PHP erzeugte** XHTML-Seiten validieren! Am bequemsten mit **Addons**!

14. Kompatibel zu **UNIX**, daher gleiche Dinge immer gleich nennen, nicht einmal "Linie", ein andermal "linie"! In Pfadangaben / benutzen, nicht \!

15. **Universelle** Einsetzbarkeit anstreben! Projektspezifische Teile auslagern in Dateien, die eingelesen werden!

16. **Formular-Auswertung und Formular-Eingabe** einer Abfrage in einer einzigen php-Datei! Reihenfolge: Auswertung, dann Eingabe**!!!**

17. **Notepad++** nutzen! Dieser Editor unterstützt Ihre Arbeit dadurch, dass Sie erstens alle Projekt-Dateien gleichzeitig laden und durchsuchen können und Sie zweitens ein sehr gutes Syntax-Highlighting für MySQL, XHTML, PHP, CSS und JS haben!

18. In Notepad++ muss unter Format **UTF-8 ohne BOM** ausgewählt sein!

Tagesberichte

Bitte führen Sie ein Word-Dokument "Tagesberichte", in dem Sie die Arbeit am Projekt unter Datumsangabe stichwortartig festhalten. Die zeitlich neueren Angaben stehen dichter am Anfang. Wenn Sie diese Tagesberichte systematisch erstellen, haben Sie es am Ende des Semesters leicht, einen umfassenden Projektbericht zu schreiben.

Projektkonzeption

Am Anfang Ihres Projekts steht eine Projektkonzeption. Dabei formulieren Sie den Zweck des Projekts und benennen die Adressaten. Dann führen Sie die erforderlichen Daten auf und weisen nach, wie die Daten erhoben werden können. Sie stellen den Datenkatalog und das Entity-Relationship-Modell auf und formulieren die Abfragen. Es sollen Entfernungs- und Datumsabfragen enthalten sein. Ergebnisse sollen kartographisch dargestellt werden. Sie sollen möglichst vielfältige SQL- und HTML-Techniken einsetzen.

Datenerfassung

Es müssen auch Daten mit Lage- und Datumsangaben erfasst werden; z.B. bei Sehenswürdigkeiten geographische Koordinate und Jahr der Erstellung. Die Güte der Koordinaten muss geprüft werden.

Dokumentation der Projektarbeit

Ihre Dokumentation der Projektarbeit soll eine Zusammenfassung und ein Fazit, sowie eine Bewertung der Arbeitsanleitung enthalten. Abgabe als Ausdruck und als Word-Datei mit allen Projektdateien auf CD.

Konventionen

<something> steht für das, was konkret zu ersetzen ist.

Ordnerstruktur

Der Platz im Dateibaum, an dem alle Ihre Dateien zu finden sind, ist C:/xampp/htdocs. Alle Dateien liegen dort in dem Ordner www. Der Ordner www hat folgende Unterordner:

Ordner docu: Er enthält Ihre Projektdokumentation als doc- oder docx-Datei.

Ordner sql: Er enthält die sql-Ladedatei und jede MySQL-Abfrage sowohl als eigenständige sql-Datei als auch das Abfrageergebnis als lst-Datei.

Ordner xls: Falls Sie Ihre Daten mit Excel aufbereitet haben, liegen hier die xls- und csv-Dateien.

Ordner grafics: Er enthält die Graphiken, die in der Internetapplikation gebraucht werden.

Im Ordner www selbst befinden sich die css-Datei und alle html-, js- und php-Dateien.

CD-Brennen

Bevor Sie zum Schluss Ihres Projektes eine CD brennen, entfernen Sie bitte alle nicht mehr gebrauchten Dateien.

Namen für Dateien und Variablen

Es ist ausgesprochen unklug, wenn Sie in Dateinamen und Variablen Sonderzeichen wie "ü" und "ß" verwenden. Dateien und Variablen haben immer englische, möglichst selbsterklärende, aber auch nicht zu lange Bezeichnungen.

Excel-Datei in Text-Datei umwandeln

Die reinen Text-Dateien entstehen aus der xls-Datei durch Datei/Speichern unter…/Text(Tapstopp-getrennt)(*.txt) oder durch Datei/Speichern unter…/ CSV(Trennzeichen-getrennt)(*.csv). Dezimalpunkt, nicht Dezimalkomma verwenden. Durch geschickt in Excel eingefügte Spalten können Sie sich die Umwandlung erleichtern.

Eigenschaften der Word-Dokumente

Ihre Word-Dokumente dürfen nur selbst definierte Formatierungen enthalten. Legen Sie die Formatierungen so an, dass Sie sie später auch für Ihre Abschlussarbeit nutzen können. Verwenden Sie Flattersatz mit automatischer Silbentrennung. Bei Blocksatz müssen Sie den Text auf optische Lücken durchsehen. Es dürfen keine leeren Absätze und keine zwei Leerzeichen hintereinander vorkommen (Danach können Sie übrigens suchen!). Die Dokumente sollen automatisch erstellte Inhalts- und Abbildungsverzeichnisse enthalten. Wenn Sie Graphiken und andere Dateien einfügen, ist es klug, diese **automatisch** mit INCLUDE einzubinden. Das ist hilfreich, da diese Dateien üblicherweise häufiger geändert werden.

Bitte weisen Sie nach, welche Programme Sie genutzt haben.

Die Lesbarkeit und der Informationsgehalt Ihres Textes sind wichtig, insbesondere auch der Informationsgehalt der Zusammenfassung, des Fazits und der Bewertung der Arbeitsanleitung.

Mit Ihrer Internet-Anwendung treten Sie an die Öffentlichkeit. Daher die Rechtschreibung so sorgfältig wie bei Ihrer Abschlussarbeit prüfen. Orthographie- und Interpunktionsfehler sind im Internet besonders ärgerlich und (dis)qualifizieren den Verfasser.

Notepad++

Als Editor für html-, php-, css-, sql- und js-Dateien verwenden Sie bitte nur Notepad++. Seine Vorteile sind das bequeme Suchen und Ersetzen gleichzeitig in allen Dateien eines Projekts, die farbliche Hervorhebung der syntaktischen Strukturen für viele Programmier-Sprachen und das einfache Vergleichen von Dateien.

Inhalte aus Notepad++ in Word dokumentieren

In Notepad++ haben Sie eine html-, php-, css-, sql- oder js-Datei geladen. Die Syntax ist jeweils farblich hervorgehoben.

Nun möchten Sie die ganze Datei oder bestimmte Zeilen in eine PowerPoint-Präsentation oder eine Word-Datei einfügen; insbesondere mit den farblichen Hervorhebungen, möglichst auch mit den Zeilennummern, aber nicht als Screenshot, da er immer nur schlecht zu lesen ist und Sie ihn nachträglich nicht mehr punktuell ändern können.

Das können Sie so schaffen:

In Notepad++ ergänzen Sie in den Zeilen zusätzliche(!) Zeilennummern. Dann exportieren Sie den gewünschten Zeilenbereich in den Zwischenspeicher im rtf-

Format. Aus dem Zwischenspeicher können Sie dann ganz normal nach Word oder Powerpoint importieren.

Die zusätzlichen Zeilennummern erhalten Sie mit TextFX/TextFX Tools/Insert Line Numbers. In den Zwischenspeicher exportieren Sie mit Erweiterungen/NppExport/Copy RTF to clipboard.

Die Funktion TextFX ist in neueren Notepad++-Versionen nicht mehr standardmäßig verfügbar. Sie kann über den Plugin-Manager leicht nachgeladen werden.

Der Export **mit** den farblichen Syntax-Hervorhebungen, aber **ohne** Zeilennummern geht einfacher: durch das Markieren des zu kopierenden Abschnittes, anschließendem Rechtsklick auf den Text und Auswahl von Plugin commands/Copy Text with Syntax Highlighting wird der Text in die Zwischenablage kopiert. Von dort kann er durch die Einfügen-Funktion in Microsoft Word eingebunden werden.

Environment am CADCAM-Labor

Die Rechner im CADCAM-Labor sind so eingerichtet, dass Sie auf einer virtuellen Maschine Ihren eigenen Server einrichten können. Damit können Sie das Zusammenspiel von Server und Browser durch serverseitige und browserseitige Programmierung kennen lernen. Voraussetzung ist jedoch, dass Sie sich im Semester jeweils am gleichen Rechner anmelden.

Die virtuelle Maschine starten Sie mit Start/Programme/VMware/VMware Player. Im Fenster "Öffnen" geben Sie ein: d\vmware\winxppro.vmx. Damit laufen auf dem Rechner jetzt die reale und die virtuelle Maschine. Auf der virtuellen Maschine arbeiten Sie immer als Administrator. Beim ersten Mal sollten Sie das Passwort des Administrators setzen, damit Ihre Daten vor fremdem Zugriff sicher sind.

Ordner und Dateien können Sie durch "drag and drop" leicht zwischen den Maschinen hin- und herbewegen. Auf das CD-Laufwerk und den USB-Stick können Sie von der virtuellen Maschine direkt zugreifen.

Dem <ctrl><alt><entf> auf der realen Maschine entspricht <ctrl><alt><einfg> auf der virtuellen Maschine.

Um vom VMwarePlayer ins Internet zu kommen, muss unter "Ethernet" (am oberen Rand des Fensters der Virtuellen Maschine) Connected und NAT gewählt werden (wird deutlich durch Häkchen bzw. Punkt). Anschließend muss der VMwarePlayer **neu gestartet** werden. Beim Aufruf des Internets muss die normale Hochschulkennung, wie bei normaler Benutzung des Internets an der Hochschule, eingegeben werden. Die virtuelle Maschine soll immer geordnet beendet werden.

Fehlerfreiheit

Der Beweis, dass ein Programm das leistet, was es soll, ist nur sehr schwer zu erbringen. Sie behelfen sich, indem Sie an die eigene Programmierung möglichst strenge Anforderungen stellen. Das fängt mit einem durchgehend eingehaltenem übersichtlichen Programmierstil an, setzt sich mit der Validierung der html- und css-Dateien fort und endet damit, dass die Fehlerkonsole des Browsers und die error.log-Datei von Apache keine Einträge enthalten. Es reicht also **bei weitem nicht**, dass Sie den Eindruck haben, dass alles so funktioniert, wie es soll.

Der Umfang der Fehlermeldungen von PHP-Programmen wird durch eine Systemvariable gesteuert. Unter C:\xampp\php steht anfangs bei Ihnen in der Datei php.ini

```
error_reporting = E_ALL & ~E_NOTICE
```

Auf meinem UNIX-Server steht in /etc/php5/apache2/php.ini

```
error_reporting = E_STRICT
```

Ändern Sie in Ihrer php.ini die Fehlerdokumentation auf:

```
error_reporting = E_ALL | E_STRICT
```

Die Validität der html-Dateien, auch der von PHP erzeugten, überprüfen Sie am besten unter validator.w3.org. Sie können die Validierung auch mit dem entsprechenden Addon (s. u.) jeweils "on the fly" vornehmen.

Ob Ihre css-Datei fehlerfrei ist, zeigt Ihnen jigsaw.w3.org/css-validator.

Nützliche Addons für Firefox

https://addons.mozilla.org/de/firefox/addon/html-validator stattet Firefox mit einer mitlaufenden Validierung Ihrer Seiten aus. Die Leistung entspricht der von validator.w3.org, ist aber „on-the-fly“.

https://addons.mozilla.org/en-US/firefox/addon/firebug integriert Firebug in Ihren Firefox-Browser. Das ist nützlich bei der Entwicklung von XHTML, CSS und JavaScript.

https://addons.mozilla.org/de/firefox/addon/dom-inspector-6622 installiert einen DOM-Inspektor in Ihren Firefox-Browser. Sie können sich damit die genaue DOM-Struktur Ihrer Seite ansehen. Das ist bei mancher Fehlersuche hilfreich.

XAMPP

Apache, MySQL, PHP und Perl sind bei XAMPP vorinstalliert. Die Vorinstallation steht auf der realen Maschine auf o:/tmp. Sie ziehen die Vorinstallation auf die virtuelle Maschine und führen sie aus.

Geben Sie jetzt in einem Browser der virtuellen Maschine als URL http://localhost/<projekt> ein, wird eine Datei c:/xampp/htdocs/<projekt>/index.html (bzw. index.php, index.htm oder index.shtml) gesucht und dargestellt.

MySQL-Datenbanken sind auf c:/xampp/mysql/data abgelegt.

Umlaute, ß und Sonderzeichen (Character set)

Wichtig ist, dass Text so dargeboten wird, wie es in der jeweiligen Sprache korrekt ist: Händelstraße, Hôpital, Moët & Chandon, De l'Église, Plaça. Sie müssen also dafür Sorge tragen, dass die Sonderzeichen wie Ä, ä, ß, ë, ô, ç, ', & als Bestandteile von Text auch auf englischen oder ungarischen Rechnern unter UNIX oder Windows korrekt erscheinen. Die Behandlung von Ä, ä, ß, ë, ô, ç ist relativ einfach und im Folgenden für UNIX und Windows beschrieben. Alle Zeichen, die in den Programmiersprachen syntaktische Funktionen haben, können in Text Schwierigkeiten machen. Dazu gehören ', ", /, \, &, <, >.

Es ist zu unterscheiden, ob die Zeichen in html-, php-, css-, sql- oder js-Dateien stehen.

Sonderzeichen in XHTML- und PHP-Dateien unter UNIX

Wenn Sie in Notepad++ das Menü „Kodierung“ öffnen, soll „UTF-8 ohne BOM“ ausgewählt sein. **Achtung:** Ist das nicht der Fall, nutzen Sie „Konvertiere zu UTF-8 ohne BOM“ und prüfen Sie vor dem Speichern, ob die Sonderzeichen erhalten bleiben.

Im html-Header steht die Zeile:

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
```

Dann können Sie im Body innerhalb von Text die Sonderzeichen wahlweise nach Spalte „Darstellung“ oder nach Spalte „XHTML-Entities“ (s. u.) eintragen.

Sonderzeichen in XHTML- und PHP-Dateien unter Windows

Wenn Sie in Notepad++ das Menü „Kodierung“ öffnen, soll „UTF-8 ohne BOM“ ausgewählt sein. **Achtung:** Ist das nicht der Fall, nutzen Sie „Konvertiere zu UTF-8 ohne BOM“ und prüfen Sie vor dem Speichern, ob die Sonderzeichen erhalten blieben.

Im html-Header steht die Zeile:

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
```

Dann können Sie im Body innerhalb von Text die Sonderzeichen wahlweise nach Spalte „Darstellung“ oder nach Spalte „XHTML-Entities“ eintragen.

Darstellung	XHTML-Entities	URL-Kodierung
Ä	Ä	%C4
Ö	Ö	%D6
Ü	Ü	%DC
ä	ä	%E4
ö	ü	%F6
ü	ö	%FC
ß	ß	%DF
â	â	%E2
é	é	%E9
è	è	%E8
ê	ê	%EA
É	É	%C9
Î	Î	%CE
ı (türk. i)	ı	%C4%B1
ô	ô	%F4
õ	õ	%F5
œ	œ	%9C
ç	ç	%E7
©	©	%A9
€	€	%80
&	&	%26
„	„	%84
“	“	%93
‚	‚	%82
‘	‘	%91
Leerzeichen		%20

Diese Liste ist nicht vollständig.

Sonderzeichen in der MySQL-Ladedatei unter UNIX

Wenn Sie in Notepad++ das Menü „Kodierung“ öffnen, soll „UTF-8 ohne BOM“ ausgewählt sein.

Sie bearbeiten die sql-Ladedatei mit Notepad++. Tabellennamen und Spaltennamen sind in Englisch. Dabei entstehen keine Probleme. Bei den Inhalten der Spalten wie "Bülowstraße" oder "De l'Église" setzen Sie die Sonderzeichen so, dass sie richtig zu lesen sind.

Der Anfang der Ladedatei sieht dann so aus:

```
/* Data are taken from Dortmund but reduced and
   changed to suit different aspects. */
DROP DATABASE IF EXISTS sample;
SHOW WARNINGS;
CREATE DATABASE IF NOT EXISTS sample
   DEFAULT CHARACTER SET utf8 COLLATE utf8_unicode_ci;
SHOW WARNINGS;
USE sample;
SHOW WARNINGS;
DROP TABLE IF EXISTS routes;
SHOW WARNINGS;
CREATE TABLE IF NOT EXISTS routes (
  id INT2,
  name VARCHAR(10) BINARY,
  color VARCHAR(10) BINARY,
  colorhex VARCHAR(10) BINARY,
  PRIMARY KEY (id)
);
SHOW WARNINGS;
```

Wenn Sie die Ladedatei jetzt abspeichern, in einer Konsole mit mysql.exe einlesen (s. u.) und die Tabellen ausgeben, werden alle Sonderzeichen korrekt dargestellt.

Sonderzeichen in der MySQL-Ladedatei unter Windows

Wenn Sie in Notepad++ das Menü „Kodierung“ öffnen, soll „UTF-8 ohne BOM“ ausgewählt sein.

Sie bearbeiten die sql-Ladedatei mit Notepad++. Tabellennamen und Spaltennamen sind in Englisch. Dabei entstehen keine Probleme. Bei den Inhalten der Spalten wie "Bülowstraße" oder "De l'Église" setzen Sie die Sonderzeichen so, dass sie richtig zu lesen sind.

Die Ladedatei sieht dann genau so aus wie unter UNIX (s. o).

Wenn Sie die Ladedatei abspeichern, in einer Eingabeaufforderung mit mysql.exe einlesen (s. u.) und die Tabellen ausgeben, werden manche der Sonderzeichen **nicht** korrekt dargestellt. Das hat aber keinen Einfluss auf die späteren Queries.

Die Arbeit mit **phpMyAdmin** hat bisher bei den Sonderzeichen unlösbare Schwierigkeiten gemacht! Daher wurde Mysql bisher immer in der Eingabeaufforderung ausgeführt.

Die Sonderzeichen ', ", \, &, <, > in Auswahllisten

Betrachten Sie die folgende PHP-Zeile:

```
echo "<option value='$station'>$station</option>\n";
```

Wenn `$station` ein Sonderzeichen aus ', ", \, &, <, > enthalten kann, müssen diese in XHTML-Entities umgewandelt werden. Das macht:

```
$station = htmlentities($station,ENT_QUOTES,'UTF-8');
```

Zurückwandeln geht mit:

```
015 $station = html_entity_decode($station,ENT_QUOTES,'UTF-8');
```

Die Sonderzeichen ', ", \ in Queries

Betrachten Sie die folgende PHP-Zeile:

```
014 $query = "SELECT * FROM stations WHERE name = '$station'";
015 $result = mysql_query($query);
```

Wenn `$station` ein Sonderzeichen aus ', ", \ enthalten kann, müssen diese durch ein \ „geschützt“ werden. Das macht:

```
013 $station = addslashes($station);
```

Die hinzugefügten \ können Sie so wieder entfernen:

```
016 $station = stripslashes($station);
```

Prototypen und ihre Verallgemeinerung, benannte Konstanten

Betrachten Sie das folgende Programm (Erde/dieErde1.html) als einen Prototyp.

```
001 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
002   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
003 <html xmlns="http://www.w3.org/1999/xhtml">
004 <head>
005 <meta http-equiv='Content-Type' content='text/html; charset=utf-8' />
006 <title>Die Erde</title>
007 </head>
008 <body>
009   <p>Hier gibt es wichtige Informationen zur Erde:</p>
010   <p>Der Radius der Erde ist 6370 km.</p>
011   <p>Der Umfang der Erde ist 40024 km.</p>
012 </body>
013 </html>
```

In welche Richtungen könnten Sie es ausbauen?

1. Sie könnten das Programm für andere Himmelskörper aufstellen; zum Beispiel für den Mars.

2. Sie könnten das Programm in andere Sprachen übertragen; zum Beispiel ins Englische.

3. Sie könnten weitere Informationen hinzufügen; zum Beispiel die mittlere Dichte von 5,52 g/cm³.

Den Punkt 3 wollen wir nicht weiter behandeln, da er im Wesentlichen nur aus der Einfügung weiterer Zeilen besteht.

Um den Punkt 1 zu erledigen, könnten Sie überall „Erde“ durch „Mars“ ersetzen, dann müssen Sie noch auf den Artikel achten, sowie „6370“ durch „3397“ und „40024“ durch „21344“ ersetzen. Als Ergebnis haben Sie zwei Programme, deren strukturelle Ähnlichkeit nur durch sehr genaues Hinsehen zu entdecken ist. Eine bessere Lösung erhalten Sie, wenn Sie sozusagen „Erde“, „6370“ und „40024“ herausziehen, diese Konstanten benennen und sie als benannte Konstanten wieder einsetzen. Dabei können Sie dann auch die Beziehung zwischen Radius und Umfang explizit machen: u = 2 * π * r.

Das neue Programm (Erde/dieErde3.php) könnte dann so aussehen:

```
001 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
002         "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
003 <html xmlns="http://www.w3.org/1999/xhtml">
004 <head>
005   <meta http-equiv='Content-Type' content='text/html; charset=utf-8' />
006 <?php
007 define('PI',3.14159);
008 define('PLANET','Erde');
```

```
009 $radius = 6370;
010 echo "\t<title>Der Planet ".PLANET."</title>\n";
011 echo "</head>\n";
012 echo "<body>\n";
013 echo "\t<p>Hier gibt es wichtige Informationen zum Planeten
".PLANET.":</p>\n";
014 echo "\t<p>Der Radius des Planeten ".PLANET." ist $radius km.</p>\n";
015 echo "\t<p>Der Umfang des Planeten ".PLANET." ist "
016         . round(2 * PI * $radius) ." km.</p>\n";
017 ?>
018 </body>
019 </html>
```

Was hat sich geändert? Das Programm ist komplexer, dafür aber auch leistungsfähiger geworden. Da es Berechnungen enthält, reicht HTML nicht mehr aus und es wird die Skriptsprache PHP verwendet. Die für die Planeten kennzeichnenden Informationen wurden als benannte Konstanten an den Anfang des Programms gesetzt und die Beziehungen zwischen den Konstanten wurden aufgedeckt. Formal sind nur PI und PLANET Konstanten. $radius dagegen ist eine Variable, die wie eine Konstante verwendet wird, da ihr Wert im Programm nicht geändert wird. Beachten Sie die unterschiedliche Definition und Verwendung! Die Informationen der Zeilen 7, 8 und 9 könnten jetzt aus einer Datenbank abgerufen werden, während der übrige Teil des Programms nicht geändert werden muss.

Ähnlich ist das Vorgehen, um das Programm in andere Sprachen zu übertragen. Dazu werden die deutschen Textstrings – es sind unbenannte Konstanten - herausgezogen und benannt. Unbenannte Konstanten werden auch als magic numbers bezeichnet und sind in Programmen verpönt!

Für Ihre Arbeit ist also wichtig, dass Sie in Ihren Programmen möglichst alle unbenannten Konstanten an den Anfang stellen und benennen. Insbesondere müssen Werte, die in der Datenbank enthalten sind, auch von dort geholt werden. Das wurde in den folgenden Fällen **nicht** berücksichtigt:

XHTML

```
<option>Adenauerplatz</option>
<option>Bahnhof Dossenheim</option>
<option>Bahnhof Wieblingen</option>
```

MySQL

```
SELECT name FROM pois WHERE kind = 'Kirche';
```

PHP

```
$db = @MYSQL_CONNECT( 'localhost', 'root', 'geheim');
mysql_select_db( 'Trondheim', $db);
```

Javasript

```
var point = new Array();
var text = new Array();
point[0] = new GLatLng( 51.51350, 7.46500);
text[0] = "Hauptstraße";
```

\n (LF), \r (CR), \t (HT) und PHP

Was machen im obigen Programm das „\n“ und das „\t“? Das wird erst sichtbar, wenn Sie sich auf dem Browser die vom Programm ausgegebene Seite im Quelltext ansehen. Im Quelltext werden an den mit „\n“ markierten Stellen die Zeilen umbrochen. Ohne „\n“ würde eine einzige, bisweilen sehr lange Zeile erzeugt, in der Sie dann vom Validator gemeldete Fehler nur schwer finden.

Zeilenende bei Windows, Unix und Mac

In Windows werden Zeilenenden durch CR LF markiert, in Unix durch LF und beim Mac durch LF CR, wie Sie in Notepad++ unter Einstellungen/Optionen…/Dateien leicht prüfen können.

MySQL

ER-Modell

Mit dem Programm DBDesigner können Sie bequem aus einer bestehenden Datenbank ein Entity-Relationship-Model anfertigen; Stichwort Reverse Engineering.

Starten von MySQL in der Eingabeaufforderung

Wenn MySQL korrekt installiert wurde, können Sie es in der Eingabeaufforderung starten mit:

```
cd c:\xampp\mysql\bin
mysql -u user –ppasswort
```

Achtung! Zwischen -u und dem Benutzernamen ist ein Leerzeichen, zwischen -p und dem Passwort dagegen nicht.

Ladedatei

Das Einrichten der Datenbank, das Kreieren und Füllen der Tabellen erfolgt am besten mit der Ladedatei sample.sql.

Mit der Ladedatei können Sie die gesamte Datenbank mit einer Anweisung löschen und erneuern (s. u.). Die Ladedatei `sample.sql` soll daher gleich zu Beginn der Projektarbeiten erstellt werden. Und zwar deshalb, weil in der Regel Umbenennungen der Tabellennamen und Spaltennamen während der Projektarbeit notwendig werden. Damit erstrecken sich diese Änderungen nicht auf mehrere Dateien, sondern erfolgen in einer einzigen Datei. Falls im Laufe des Projekts weitere Änderungen erfolgen müssen, werden sie jeweils in der Ladedatei vorgenommen.

Beispiel (Sql/sample.sql)

```
001 /* Data are taken from Dortmund but reduced and
002    changed to suit different aspects. */
003 DROP DATABASE IF EXISTS sample;
004 SHOW WARNINGS;
005 CREATE DATABASE IF NOT EXISTS sample
006    DEFAULT CHARACTER SET utf8 COLLATE utf8_unicode_ci;
007 SHOW WARNINGS;
008 USE sample;
009 SHOW WARNINGS;
010 DROP TABLE IF EXISTS routes;
011 SHOW WARNINGS;
012 CREATE TABLE IF NOT EXISTS routes (
013   id INT2,
014   name VARCHAR(10) BINARY,
015   color VARCHAR(10) BINARY,
016   colorhex VARCHAR(10) BINARY,
017   PRIMARY KEY (id)
018 );
019 SHOW WARNINGS;
020 INSERT INTO routes (id, name, color, colorhex) VALUES
021 (1, '404', 'cyan', '#00FFFF'),
022 (2, '403', 'grün', '#008000'),
023 (3, 'U47', 'kaki', '#FF69B4'),
024 (4, 'U41', 'gelb', '#FF69B4')
```

```
025 ;
026 SHOW WARNINGS;
```

Beachten Sie bitte den Befehl `show warnings;` Er gibt Hinweise auf Unstimmigkeiten, untersucht aber nur den vorangegangenen Befehl. Daher muss für die Inserts eine spezielle, nur bei MySQL erlaubte Form gewählt werden!

Ausführen von MySQL-Dateien

Die Dateien mit den MySQL-Befehlen können Sie so in MySQL ausführen:

```
source <path mit Laufwerk>/<file>;
```

Drop, Create und Use einer Datenbank (Sql/sample.sql)

Drop, Create und Use einer Datenbank werden so vorgenommen:

```
003 DROP DATABASE IF EXISTS sample;
005 CREATE DATABASE IF NOT EXISTS sample
006    DEFAULT CHARACTER SET utf8 COLLATE utf8_unicode_ci;
008 USE sample;
```

Löschen einer Tabelle und Kreieren mit Schlüssel und Constraints (Sql/sample.sql)

```
056 DROP TABLE IF EXISTS s_r;
058 CREATE TABLE IF NOT EXISTS s_r (
059   s_id INT2,
060   r_id INT2,
061   seq INT2 UNSIGNED DEFAULT NULL,
062   spa INT4 DEFAULT NULL,
063   dur TIME DEFAULT NULL,
064   PRIMARY KEY (s_id, r_id),
065   FOREIGN KEY (s_id) REFERENCES stops(id),
066   FOREIGN KEY (r_id) REFERENCES routes(id)
067 );
```

Dokumentation der Ausführung

Wenn Sie die eigentlichen MySQL-Befehle mit den beiden folgenden Zeilen klammern, erhalten Sie eine für unsere Zwecke geeignete Dokumentationsdatei:

```
\T <path>\<file>.lst
\t
```

Benutzerdefinierte Variablen (Sql/variables.sql)

Mit benutzerdefinierten Variablen können Sie Ausdrücke (expressions) bilden. Sie können aber keine Namen von Tabellen oder Spalten dabei nutzen.

```
001 -- user-defined variable using SET
002 SET @currentColor = 'cyan';
003 SELECT * FROM routes WHERE color = @currentColor;
004 -- two user-defined variables using SELECT
005 SELECT @minX:=MIN(x),@maxX:=MAX(x) FROM pois;
006 SELECT * FROM pois WHERE x=@minX OR x=@maxX;
```

Soweit ich weiß, gibt es keine Möglichkeit, die benutzerdefinierten Variablen aufzulisten. `SHOW VARIABLES` zeigt nur die Environment-Variablen!

MySQL ohne PHP testen

Welche Haltestellen gibt es an einer ausgewählten Linie? Diese Abfrage soll später in Ihrem Projekt dynamisch an die Datenbank gestellt werden. Es ist klug, sich zunächst zu vergewissern, dass MySQL korrekte Ergebnisse liefert. Dort, wo später PHP dynamisch Werte einsetzt, werden jetzt benutzerdefinierte Variablen verwendet, die mit `SET` vereinbart werden.

```
001 SET @currentRouteName="U41";
```

```
002 SELECT stops.name
003     FROM routes, stops, r_s
004     WHERE stops.id=r_s.s_id
005     AND r_s.r_id=routes.id
006     AND routes.name=@currentRouteName
007     ORDER BY r_s.succession;
```

Nützliches

Zum Zeigen der MySQL-Kommandos (nicht Standard-SQL):

```
HELP;
```

Zum Zeigen der Environment-Variablen:

```
SHOW VARIABLES;
```

Zum Zeigen der Datenbanken:

```
SHOW DATABASES;
```

Zum Zeigen der Tabellen:

```
SHOW TABLES;
```

Zum Zeigen der Spalten einer Tabelle:

```
SHOW COLUMNS FROM <table>;
```

Die gleiche Wirkung hat:

```
DESCRIBE <table>;
```

Fortgeschrittenes

An der Stelle einer FROM-Tabelle kann ein SELECT stehen:

```
SELECT AVG(x) FROM (SELECT x FROM stops) AS s;
```

Gruppenfunktion GROUP_CONCAT:

Durch welche Linien werden die Haltestellen bedient?

```
SELECT s_id, COUNT(*),
       GROUP_CONCAT(DISTINCT r_id ORDER BY r_id SEPARATOR ', ') AS 'Linien'
       FROM s_r GROUP BY s_id;
```

Reguläre Ausdrücke:

```
SELECT * FROM stops WHERE y REGEXP '^570[0-9]+';
```

Reguläre Ausdrücke sind ein mächtiges Werkzeug und es lohnt sich, sie nutzen zu lernen. In diesem Beispiel bedeuten: ^ der Ausdruck steht am Anfang, + ein- oder mehrfache Wiederholung, [0-9] die Zeichen 0 bis 9. Insgesamt: Die Variable y fängt mit den Ziffern 570 an, danach folgen mindestens eine der Ziffern 0 bis 9.

Case:

```
SELECT s_id,
   CASE r_id
      WHEN 1 THEN 'Bus 404'
      WHEN 2 THEN 'Bus 405'
      ELSE 'Uxx' END AS 'Linie' FROM s_r;
```

Limit:

```
SELECT * FROM stops LIMIT 5, 4;
```

Zeige ab der 5. insgesamt 4 Haltestellen.

Schwierigere SQL-Abfragen

-- Gib zu einer gewählten Haltestelle auf einer gewählten Linie die Namen der Vorgänger- und Nachfolger-Haltestellen aus!

-- Zeige alle Haltestellen, die von einer gewählten Haltestelle durch Einmal-Umsteigen (Zweimal, …) zu erreichen sind!

-- Zeige alle Pois, die in einem Band einer zu wählenden Linie liegen!

-- Zeige die Namen aller Haltestellen und Sehenswürdigkeiten!

-- Zeige die Namen aller Haltestellen, die mit dem Namen einer Sehenswürdigkeit übereinstimmen.!

Arten von Joins (Sql/jointypes.sql)

Im Folgenden finden Sie eine sql-Datei, die die verschiedenen in MySQL erlaubten Tabellenverknüpfungen zeigt.

```
01 DROP DATABASE IF EXISTS jointypes;
02 CREATE DATABASE IF NOT EXISTS jointypes
03     DEFAULT CHARACTER SET utf8 COLLATE utf8_unicode_ci;
04 USE jointypes;
05 DROP TABLE IF EXISTS eltern;
06 CREATE TABLE IF NOT EXISTS eltern (
07   id INT2,
08   ename VARCHAR(6));
09 INSERT INTO eltern (id, ename) VALUES
10 (1, 'Klaus'),
11 (2, 'Paula'),
12 (3, 'Heinz');
13 DROP TABLE IF EXISTS kinder;
14 CREATE TABLE IF NOT EXISTS kinder (
15   id INT2,
16   eid INT2,
17   kname VARCHAR(8));
18 INSERT INTO kinder (id, eid, kname) VALUES
19 (1,2, 'Mäxchen'),
20 (1,3, 'Mäxchen'),
21 (2,3, 'Lenchen');
22
23 /*1. Inner Join, hier ohne WHERE*/
24 SELECT * FROM eltern, kinder;
25 /*2. Inner Join mit CROSS JOIN*/
26 SELECT * FROM eltern CROSS JOIN kinder;
27 /*3. Inner Join mit STRAIGHT_JOIN*/
28 SELECT * FROM eltern STRAIGHT_JOIN kinder WHERE eltern.id=kinder.eid;
29 /*4. Inner Join mit INNER JOIN*/
30 SELECT * FROM eltern INNER JOIN kinder ON eltern.id=kinder.eid;
31 /*5. Outer Join mit LEFT OUTER JOIN*/
32 SELECT * FROM eltern LEFT OUTER JOIN kinder ON eltern.id=kinder.eid;
33 /*6. Outer Join mit NATURAL LEFT OUTER JOIN*/
34 SELECT * FROM eltern NATURAL LEFT OUTER JOIN kinder;
35 /*7. Outer Join mit RIGHT OUTER JOIN*/
36 SELECT * FROM eltern RIGHT OUTER JOIN kinder ON eltern.id=kinder.eid;
37 /*8. Outer Join mit NATURAL RIGHT OUTER JOIN*/
38 SELECT * FROM eltern NATURAL RIGHT OUTER JOIN kinder;
```

Wenn sich `eltern` und `kinder` als Alias auf die gleiche Tabelle beziehen, sprechen wir von einem Auto- oder Self-Join. Bei 1. kann auch eine WHERE-Bedingung stehen. Bei 6. und 8. muss es in `eltern` und `kinder` übereinstimmende Spaltennamen, hier `id`, geben. Ist der Vergleichsoperator wie hier das Gleichheitszeichen, sprechen wir von einem Equi-Join, sonst von einem Not-Equi-Join.

Mengenoperationen

Einige Datenbanksysteme, zum Beispiel Oracle, kennen Methoden, um Vereinigungsmengen (UNION), Schnittmengen (INTERSECT) und Differenzmengen (MINUS) zu erzeugen. Mysql und Access kennen nur UNION. INTERSECT und MINUS können mit WHERE … NOT IN realisiert werden.

Guter und erwünschter MySQL-Programmierstil

Im Programmcode dürfen für Einrückungen am Anfang einer Zeile nur Tabs verwendet werden. An anderen Stellen sollen Tabs nicht eingesetzt werden.

MySQL-spezifische Bezeichner in upper-case, aber benutzerdefinierte in lower-case; also z.B. `SELECT, ROUND`, aber `routes, name`. Die Bezeichner für Entitäten-Tabellen stehen im Plural; also `stops`. Die Bezeichner für Relationen-Tabellen werden mit einem Verb gebildet; also `stop_ison_route` (in diesem Fall verkürzt zu `s_r`).

Die Primärschlüssel der Tabellen heißen immer `id`, Namen heißen immer `name`; es sei denn Sie brauchen z.B. `familyname` und `firstname`. Die Aliase für Tabellen sollen möglichst nur aus einem Buchstaben bestehen und den Tabellennamen aufnehmen; also `routes AS r, stops AS s`. Spalten sollen voll qualifiziert werden; also `stops.name` bzw. `s.name`.

Wählen Sie einprägsame Bezeichner. Sie sollen nicht zu kurz sein, aber auch nicht zu lang. `x_coordinate` ist zu lang, hier ist `x` sogar ausreichend. Im Sinne der Internationalisierung sollen Sie Bezeichner und Kommentare in Englisch wählen.

In den SELECT-Anweisungen dürfen keine unbenannten Konstanten (magic numbers) enthalten sein: verboten ist also zum Beispiel `'Ettlinger Tor'` in `WHERE name = 'Ettlinger Tor'` oder `500` in `WHERE distance < 500`. Sie sollen an solchen Stellen benutzerdefinierte Variablen (s. o.) verwenden.

Wenn Sie eine Anweisung auf mehrere Zeilen verteilen, dann ist das Ende einer Klausel eine geeignete Trennstelle.

Vor dem Komma kein Leerzeichen, danach ein Leerzeichen.

Vor und nach <, >, = ein Leerzeichen.

Vor und nach dem Punkt kein Leerzeichen.

HTML und XHTML

In c:/xampp/htdocs/www liegen die für Ihre Internet-Anwendung erforderlichen Dateien.

Insbesondere liegt dort Ihre Startdatei mit dem Namen index.html oder index.php; in jedem Fall ist der Dateiname index. Dann können Sie Ihr Projekt mit `http://localhost/www` durch den Server auf Ihrem Browser präsentiert bekommen. Dabei wird Ihre Start-Seite von Apache auf Ihrem Browser ausgeliefert. Wenn Sie Ihre Start-Seite doppelklicken, wird als URL im Browser `file:///c:/xampp/htdocs/www/index.html` angezeigt. Auf diese Art wurde die Seite aber nicht durch Apache interpretiert, was aber notwendig ist.

Bilder als Icons sollen transparent auf dem Untergrund stehen und keinen weißen Rand haben. Wie Sie das erreichen, finden Sie unten.

Seiten, die im Dateibaum unterhalb von index.html liegen, werden in Ihrem Programm mit einem relativen, nicht absoluten Pfad angegeben. Also nicht:
`http://localhost/www/query01.php`
sondern:
`./query01.php`
Das setzt voraus, dass index.html und query01.php im gleichen Ordner liegen. Wenn query01.php im Unterordner "php" liegt, dann geht es so:
`./php/query01.php`

Kennen Sie den Unterschied von `"./"` und `"../"`?

Ihre Lösungen möchte ich später auf meiner Homepage ins Netz stellen, siehe `www.hans-f-kern.de/PDB.shtml`.

HTML in XHTML wandeln

Für die Einbindung von Google-Maps in Internetseiten wird der Einsatz von XHTML empfohlen. XHTML ist strenger, daher besser zu validieren und führt damit zu eher browser-unabhängigen Ergebnissen. Sie sollten daher von Anfang an mit XHTML arbeiten. Hier folgen die wichtigsten Unterschiede von HTML und XHTML:

`<br>` wird zu `<br />`. Jedes Tag hat ein End-Tag oder eine dem `<br />` entsprechende Gestalt.

`<TABLE CLASS=menue>` wird zu `<table class="menue">` Alle Tag-Namen und Attribut-Namen werden klein geschrieben. Die Attributwerte werden in " oder ' eingeschlossen.

Das Attribut `NORESIZE` wird zu `noresize="noresize"`. Jedes Attribut hat also einen Attributwert.

`<img>` braucht das alt-Attribut, `<table>` braucht das summary-Attribut, `<table>` darf nicht innerhalb `<p>` stehen. Sonderzeichen werden ersetzt: & wird &

DOCTYPEs und Validierung (Doctypes/XHTMLstrict.html)

Doctypes sind für die korrekte Anzeige und Validierung von Seiten erforderlich. Unter http://www.w3.org/QA/2002/04/valid-dtd-list.html schlägt das World Wide Web Consortium (W3C) diese Schablone für eine Seite vor:

```
01 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
02     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
03 <html xmlns="http://www.w3.org/1999/xhtml">
04
05 <head>
06   <title>An XHTML 1.0 Strict standard template</title>
07   <meta http-equiv="content-type" content="text/html;charset=utf-8" />
08 </head>
09
10 <body>
11
12   <p>... Your XHTML content here ...</p>
13
14 </body>
15 </html>
```

Für Framesets und Targets werden andere Doctypes (siehe `XHTMLframes.html` und `XHTMLtarget.html` im Ordner `Doctypes`) verwendet. Das können Sie auch als Hinweis interpretieren, dass Sie Framesets und Targets (s. u.) mit moderneren Techniken (`<div>`) ersetzen sollten.

Ob Ihr XHTML-Dokument fehlerfrei ist, können Sie unter validator.w3.org überprüfen oder Sie lassen die Validierung mit dem entsprechenden Addon (s. o.) jeweils "on the fly" vornehmen.

Achtung: `<body border="0">` wird immer als Fehler gemeldet. Fehlerfrei und gleichwertig dagegen ist `<body style="margin:0; padding:0; border:0">`.

<frameset> und <target> durch ersetzen

Oben wurde darauf verwiesen, dass Sie bei Benutzung von Frames und Target nicht den strikten XHTML-Doctype verwenden können. Die Verwendung von Target wird sogar missbilligt. Und das Arbeiten mit `<div>` gilt als der modernere Ansatz. Die Dokumentation einer Internetseite mit Frames ist recht übersichtlich. Daher finden Sie im Ordner `Sample` den Unterordner `SampleFrame+Google`. Die modernere Lösung mit `<div>` enthält der Unterordner `SampleDiv+Google`. Die Dateien darin sind aber wenig elegant, da sie sehr große gleiche Abschnitte enthalten, die nur umständlich gepflegt

werden können. Unter dem Gesichtspunkt der Pflege und Weiterentwicklung steht der Unterordner `SampleDivInc+Google`. Er enthält Abfragen an GoogleMaps, die in der Google Api Version 2 formuliert wurden. Mit Google Api Version 3 wurden die Abfragen in `SampleDivInc+Google3` formuliert.

http://de.selfhtml.org/css/eigenschaften/positionierung.htm bietet einen guten Überblick zur Verwendung von `<div>`. Die Datei `index.html` im Ordner `DIV_example` ist von dort genommen. Das Beispiel Dortmund wurde ursprünglich mit Frames gelöst, siehe den Ordner `SampleFrame+Google`. Die Umstellung auf `<div>` finden Sie in den Ordnern `SampleDiv+Google, SampleDivInc+Google` und `SampleDivInc+Google3`.

Bilder transparent machen

Bei den Logos möchten Sie in der Regel Transparenz erreichen. Das können Sie mit Photoshop erledigen, aber auch so:

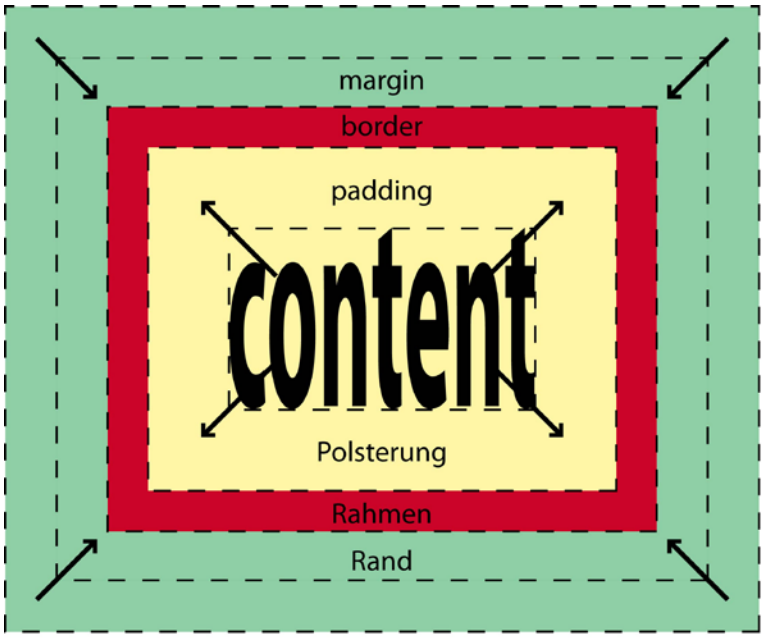
Öffnen Sie http://www.gifworks.com/

Klicken Sie "How do I get started?" Click here to begin.

Sie können das Bild hochladen, transparent machen und wieder herunterladen.

Polsterung (padding), Rahmen (border) und Rand (margin)

Die verschiedenen XHTML-Elemente auf einer Seite sind rechteckige Bereiche, die mit Polsterung, Rahmen und Rand versehen sind. Die Beziehungen zeigt die folgende Graphik.



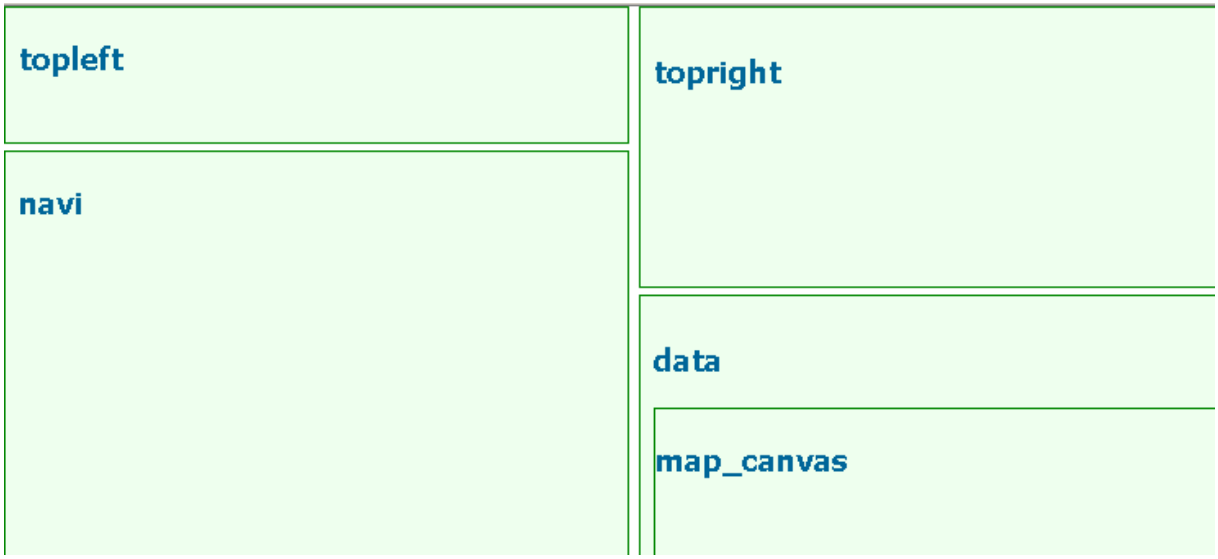
Polsterung (padding) ist der Abstand zwischen Inhalt (content) und dem Rahmen (border). Dieser Bereich wird mit der Hintergrundfarbe gefüllt, die für den Inhalt (content) definiert ist.

Rand (margin) bezeichnet den transparenten Abstand zu einem anderen Objekt. Dieser Bereich wird mit der Hintergrundfarbe gefüllt, die für das umgebende Element definiert ist .

Rand, Rahmen und Polsterung können für alle vier Seiten unterschiedlich definiert werden.

Seitenlayout

Als Seitenlayout stelle ich Ihnen hier das Layout vor, das auf das erste Projekt eines Stadtinformationssystems mit Öffentlichem Personennahverkehr (ÖPNV) und Sehenswürdigkeiten von Jana Nücklich zurück geht. Sie arbeitete noch mit Frames und es gab die Version 2 für das GoogleMap-Api. Was vorher mit Frames geleistet wurde, kann nun eleganter mit Containern ( ) erreicht werden. Und wir verwenden jetzt die Version 3 von GoogleMaps.



Die Seite hat zwei Spalten, die linke Spalte besteht aus den Containern topleft und navi, die rechte aus topright und data. data seinerseits enthält den Container map_canvas. Wenn Sie topleft und topright in der Höhe angleichen, können Sie sie auch zu einem Container header zusammenfassen. Bei einem Mehr-Spalten-Design finden Sie oft neben header auch footer.

Beispiel (SampleDivInc+Google3/layout.php)

Die Datei, die obiges Layout realisiert, sieht so aus:

```
001 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
002      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
003 <html xmlns="http://www.w3.org/1999/xhtml">
004   <head>
005      <title>Das Straßenbahnnetz der Stadt Dortmund</title>
006      <meta http-equiv="Content-Type" content="text/html; charset=utf-8"
/>
007      <link rel='stylesheet' media='all' type='text/css'
href='sample.css' />
008   </head>
009   <body>
010      <div id='topleft'>
011         <p class="menue">topleft</p>
012      </div>
013      <div id='topright'>
014         <p class="menue">topright</p>
015      </div>
016      <div id='navi'>
017         <p class="menue">navi</p>
018      </div>
019      <div id='data'>
020         <p class="menue">data</p>
021         <div id='map_canvas'>
022            <p class="menue">map_canvas</p>
023         </div>
```

```
024       </div>
025    </body>
026 </html>
```

Ein Vorteil von Containern ist, dass wir topleft, topright, navi und data in beliebiger Reihenfolge in obiger Datei aufführen können, wenn wir nur in der CSS-Datei (siehe Zeile 7) ihre Lage auf dem Bildschirm festlegen. Die Reihenfolge der Container wählen Sie am besten so, dass Container, die auf jeder Seite mit gleichem Inhalt gebraucht werden, entweder am Anfang oder am Ende der Datei stehen. Container mit variablem Inhalt stehen dann in der Mitte. Neben dem Container data hat zwar auch map_canvas wechselnden Inhalt, er wird aber an anderer Stelle erzeugt und dann nur auf den Container map_canvas "gemalt" (canvas = Leinwand). Damit steht also nur Zeile 20 (!) für den wechselnden Inhalt auf unseren Seiten. Die Zeilen vor Zeile 20 stellen den header, die Zeilen nach Zeile 20 den footer dar.

Beispiel (SampleDivInc+Google3/query28.php)

Eine einfache Seite ohne Datenbankzugriff und ohne GoogleMap hat diese Gestalt:

```
001 <?php
002 require_once('./inc/header.inc.php');
003 ?>
004         <p class="menue">Verwendete Literatur:</p>
005         <p>Wolfgang D. Misgeld: ORACLE für Profis, Hanser, 1991</p>
006         <p>Eva Kraut u. Theodor Seidl: Oracle-SQL, it-Verlag, o.J.</p>
007         <p>Gregor Kuhlmann u. Friedrich Müllmerstaft: SQL Der Schlüssel
zu rel. Datenbanken, Rowohlt, 2000</p>
008         <p>Alex Morrison u. Alice Rischert: Oracle SQL, Prentice
Hall</p>
009         <p>Rasmus Lerdorf u. Kevin Tatroe: Programmieren mit PHP,
O´Reilly, 2003</p>
010         <p>Rasmus Lerdorf: PHP kurz und gut, O´Reilly, 2003</p>
011         <p>Michael Kofler u. Bernd Öggl: PHP5 & MySQL 5, Addison-
Wesley, 2006</p>
012         <p>Stefan Hinz, Michael Seeboerger-Weichselbaum: MySQL 5 - GE-
PACKT, RedLine, 2006</p>
013         <p>Dieter Staas: PHP4 - Hot Stuff, Franzis, 2001</p>
014         <p>Stefan Mintert (Hg.): XHTML, CSS & Co, Addison-Wesley,
2003</p>
015         <p>Chuck Musciano u. Bill Kennedey: HTML-Das umfassende
Referenzwerk, O´Reilly, 1997</p>
016         <p>Self-HTML <a
href="http://de.selfhtml.org/index.htm">http://de.selfhtml.org/index.htm</a
></p>
017         <p>Self-PHP <a
href="http://www.php.net/manual/de/">http://www.php.net/manual/de/</a></p>
018         <p>Self-PHP <a
href="http://www.selfphp.net/selfphp/">http://www.selfphp.net/selfphp/</a><
/p>
019 <?php
020 require_once('./inc/footer.inc.php');
021 ?>
```

Guter und erwünschter XHTML-Programmierstil

Ihre XHTML-Dokumente sollen möglichst auf allen Betriebssystemen und allen Browsern richtig dargestellt werden. Ob Ihr XHTML-Dokument fehlerfrei ist, können Sie unter validator.w3.org überprüfen oder Sie lassen die Validierung mit dem entsprechenden Addon (s.o.) jeweils "on the fly" vornehmen.

Bei Pfad- und Dateinamen unterscheiden Windows und Internet-Explorer nicht zwischen Groß- und Kleinschreibung, UNIX wohl. Nicht an einer Stelle grau.jpg, an anderer Grau.JPG verwenden!

Im Programmcode dürfen für Einrückungen am Anfang einer Zeile nur Tabs verwendet werden. An anderen Stellen sollen Tabs dagegen nicht eingesetzt werden.

In Notepad++ soll unter Kodierung „**UTF-8 ohne BOM**“ ausgewählt sein.

Wählen Sie einprägsame Bezeichner. Sie sollen nicht zu kurz sein, aber auch nicht zu lang. Im Sinne der Internationalisierung sollen die Bezeichner und Kommentare in Englisch sein.

In reinen html-Dateien sollen Klammerungen durch Einrückungen sichtbar gemacht werden. Bei umfangreichen Dokumenten führt das zu einer zu tiefen Staffelung. Dann können Sie die Einrückungen vereinfachen, wie das oben bereits für `<title>` gezeigt ist.

Verwenden Sie keine absoluten Pfade, sondern immer Pfade relativ zum Ordner, in dem `index.html` liegt.

CSS

Guter und erwünschter CSS-Programmierstil

Die typographische Gestaltung einer Seite soll mit dem Inhalt nicht "fest verdrahtet" sein, sondern mit Hilfe von CSS realisiert werden. Damit können Sie alle Attribute in Marken, die die Typographie beeinflussen, vermeiden.

Statt zum Beispiel auf der XHTML-Seite im Element body das Attribut bgcolor und die missbilligten Attribute text und background zu verwenden,

```
<body bgcolor="#FF9900" text="#FF0000" background="bild.gif">
```

soll in der css-Datei stehen:

```
body {
  background-color : #FF9900;
  background-image  :url(bild.gif);
  color : #FF0000;
}
```

Statt zum Beispiel auf der XHTML-Seite die Element font und b zu verwenden,

```
<p><font face="Arial, sans-serif" size="3" color="#660000">
  <b>Gegenstand</b>
</font></p>
```

soll in der css-Datei stehen:

```
p {
  font-family : Arial, sans-serif;
  font-size : 10pt;
  color : #660000;
  font-weight : bold;
}
```

Darstellungsorientierte Marken wie `<b>`, `<br>`, `<i>`, `<u>` sollen ganz vermieden werden.

Die Bildschirme der Benutzer haben jeweils unterschiedliche Auflösungen und es kann dann alles durcheinander geraten. Es ist gerade ein Konstruktionsprinzip des Web, dass auch bei Verkleinerung des Darstellungsfensters alles lesbar bleibt. Probieren Sie es aus! Was passiert, wenn Sie den Bildschirm verkleinern? Das meiste verrutscht in sinnvoller Weise. Den Hinweis "Optimiert für Browser xyz mit Auflösung nnn" empfinde ich als eine Ausrede.

Im Programmcode dürfen für Einrückungen am Anfang einer Zeile nur Tabs verwendet werden. An anderen Stellen sollen Tabs nicht eingesetzt werden.

Wählen Sie einprägsame Bezeichner. Sie sollen nicht zu kurz sein, aber auch nicht zu lang. Im Sinne der Internationalisierung sollen Sie Bezeichner und Kommentare in Englisch wählen.

Ob Ihr CSS-Dokument fehlerfrei ist, können Sie unter jigsaw.w3.org/css-validator überprüfen.

Beispiel für CSS (SampleDivInc+Google3/sample.css)

```
/*
This css is intended for an application using DIVs instead of frames.
Should contain all used tags
E.g. a, body, div, em, h1, h2, h3, h4, h5, img,
li, ol, p, table, td, th, tr, ul,
*/
a:link, a:visited, a:hover, a:active {
  color : #006699;
  text-decoration : none;
}
a:hover {
  background-color : #ffbbaa;
}
a:active {
  background-color : #ffbbaa;
}
p, p.menue, p.submenue, span.bold, input, textarea {
  font-size : 10pt;
  color : #0;
  font-family : Tahoma;
}
p.menue {
  font-weight : bold;
  color : #006699;
}
p.submenue {
  font-size : 9pt;
  font-weight : bold;
  color : #ff866a;
}
span.bold {
  font-weight : bold;
}
img, table, tr, td {
  margin : 0 0 0 0;
  border : 0;
  padding : 0 0 0 0;
}
body {
  background-color : #ffffff;
}
#topleft {
  position: absolute;
  top: 0;
  left: 0;
  width:374px;
  height:74px;
  margin:0 5px 5px 0;
  padding:0 0 8px 8px;
  background-color : #eeffee;
}
```

```
052 #navi {
053   position: absolute;
054   top:88px;
055   left:0px;
056   width:374px;
057   height:996px;
058   margin:0 5px 0 0;
059   padding:0 0 8px 8px;
060   background-color : #eeffee;
061 }
062 #topright {
063   position: absolute;
064   top:0px;
065   left:388px;
066   width:860px;
067   height:154px;
068   margin:0 0 5px 0;
069   padding:8px;
070   background-color : #eeffee;
071 }
072 #data {
073   position: absolute;
074   top:176px;
075   left:388px;
076   width:860px;
077   height:900px;
078   margin:0;
079   padding:8px;
080   background-color : #eeffee;
081 }
082 #map_canvas {
083   position: relative;
084   top:0;
085   left:0;
086   margin:0;
087   border:0;
088   padding:0;
089   background-color : #eeffee;
090   width: 860px;
091   height: 515px;
092 }
```

PHP

Allgemeines

Die Abfragen sollen immer als Frage formuliert werden und zwar so, dass aus dem Text eindeutig hervorgeht, was der Benutzer auszuwählen hat? Zum Beispiel:

`Welche Haltestellen hat eine auszuwählende Linie?`

Der Benutzer muss also eine Linie auswählen!

`Welche Haltestellen haben einen tiefen Einstieg?`

Der Benutzer muss hier nichts auswählen!

Bitte keine absoluten Pfade verwenden, sondern immer relativ zum Ordner von `index.html`.

Ihre PHP-Dokumente sollen möglichst auf allen Betriebssystemen und allen Browsern richtig dargestellt werden. Bei Pfad- und Dateinamen unterscheiden Windows und Internet-Explorer nicht zwischen Groß- und Kleinschreibung, UNIX wohl. Nicht an einer Stelle grau.jpg, an anderer Grau.JPG verwenden!

Für die Behandlung einer Abfrage soll nur eine (!) php-Datei eingesetzt werden.

Im Programmcode dürfen für Einrückungen am Anfang einer Zeile nur Tabs verwendet werden. An anderen Stellen sollen Tabs nicht eingesetzt werden.

Der PHP-Quelltext soll mit <?php eingeleitet werden, <? kann zu Fehlern führen, wenn PHP nicht als Default-Scriptsprache eingestellt ist..

Wählen Sie einprägsame Bezeichner. Sie sollen nicht zu kurz sein, aber auch nicht zu lang. `x_coordinate` ist zu lang, hier ist `x` sogar ausreichend. Im Sinne der Internationalisierung sollen Sie Bezeichner und Kommentare in Englisch wählen.

Pro Zeile nur eine Anweisung, nicht mehrere.

Vor dem Komma kein Leerzeichen, danach ein Leerzeichen.

Vor und nach <, >, =, -> kein Leerzeichen.

Guter und erwünschter PHP-Programmierstil

Zur besseren Lesbarkeit sollen immer { und } eingesetzt werden, auch in den Fällen, in denen diese Klammern nicht zwingend sind. Dort, wo im folgenden `anweisung` steht, können auch mehrere Anweisungen stehen. Einrückungen nur dort, wo es in der Programmstruktur eine Etage tiefer geht; also wie unten gezeigt.

Ich bevorzuge den "1TBS"-Einrückungsstil (One True Brace Style), eine Alternative ist der Allman-Stil, der aber zu längeren Texten führt, andererseits bisweilen mit Notepad++ Vorteile bietet.

IF

```
if (ausdruck) {
  anweisung;
} elseif (ausdruck) {
  anweisung;
} else {
  anweisung;
}
```

SWITCH

```
switch (ausdruck) {
  case ausdruck:
    anweisung;
    break;
  case ausdruck:
    anweisung;
    break;
  default:
    anweisung;
  break;
}
```

WHILE

```
while (ausdruck) {
  anweisung;
}
```

DO-WHILE

```
do {
  anweisung;
} while (ausdruck);
```

FOR

```
for (start_ausdruck; cond_ausdruck; iter_ausdruck) {
  anweisung;
}
```

FOREACH

```
foreach (array_ausdruck as $wert) {
  anweisung;
}
```

MySQL unter PHP

MySQL-Befehle

Das Ergebnis einer SELECT-Anweisung müssen Sie sich immer als eine Tabelle mit Zeilen und Spalten vorstellen auch dann, wenn nur ein einziger Wert als Ergebnis erzeugt wird. Dabei können Sie auf die Zeilen und Spalten durch ihre Nummer, auf die Spalten auch durch ihren Namen zugreifen. Mit mysql_query erhalten Sie einen Handle (Pointer, hier `$result`) auf das Ergebnis einer SELECT-Anweisung.

```
$query = "SELECT name, x, y FROM stops";
$result = mysql_query($query) OR die("Mist! ". mysql_error());
```

Die Anzahl der Zeilen erhalten Sie mit:

```
$rows = mysql_num_rows($result);
```

Die Anzahl der Spalten erhalten Sie mit:

```
$cols = mysql_num_fields($result);
```

An den Wert in der **5**. Zeile und **3**. Spalte kommen Sie mit:

```
$y=mysql_result($result, 4, 2);
```

Eine ganze Zeile erhalten Sie mit:

```
$row = mysql_fetch_object($result);
```

Wenn ein Handle nicht mehr gebraucht wird, sollen Sie es freigeben mit:

```
mysql_free_result($result);
```

Wenn Sie alle Zeilen und Spalten eines SELECT verarbeiten möchten, können Sie das so machen:

```
$query = "SELECT name, x, y FROM stops ";
$result = mysql_query($query) OR die("Mist! ". mysql_error());
$rows = mysql_num_rows($result);
$cols = mysql_num_fields($result);
for($r=0; $r< $rows; $r++) {
  for($c=0; $c<$cols; $c++) {
    $name = mysql_result($result, $r, 0);
    $x = mysql_result($result, $r, 1);
    $y = mysql_result($result, $r, 2);
    verarbeite $name, $x, $y
  }
}
```

Das ist nicht sehr elegant, weil die Beziehung zwischen name, x, y und 0, 1, 2 nicht aufgedeckt wird.

Die Verarbeitung aller Zeilen und Spalten können Sie eleganter so erledigen:

```
$query = "SELECT name, x, y FROM stops ";
$result = mysql_query($query) OR die("Mist! ". mysql_error());
while ($row = mysql_fetch_object($result)) {
  $name = $row->name;
  $x = $row->x;
```

```
  $y = $row->y;
  verarbeite $name, $x, $y
}
```

Dabei ist $row ein Objekt und name, x und y sind Eigenschaften des Objekts.

Es geht aber auch elegant so:

```
$query = "SELECT name, x, y FROM stops ";
$result = mysql_query($query) OR die("Mist! ". mysql_error());
while ($row = mysql_fetch_array($result)) {
  $name = $row['name'];
  $x = $row['x'];
  $y = $row['y'];
  verarbeite $name, $x, $y
}
```

Jetzt ist $row ein Array und name, x und y sind assoziative Indizes des Array.

Achtung

Da Ihre Projekt-Applikation von beliebigen Nutzern verwendet wird, können Sie diesen Benutzern natürlich keine Veränderungen in Ihrer Datenbank gestatten. Sie müssen also alle die Datenbank ändernden Befehle vermeiden. Das folgende MySQL-Beispiel zeigt, wie Sie ein CREATE umgehen können.

```
CREATE TABLE dist AS SELECT MAX(distance) maxim FROM h_l GROUP BY l_id;
SELECT SUM(maxim) FROM dist;
```

wird zu

```
SELECT SUM(dist.maxim) FROM (SELECT MAX(distance) maxim FROM h_l GROUP BY
l_id) dist;
```

Hier sehen Sie, dass an der Stelle einer FROM-Tabelle auch ein SELECT stehen kann.

Häufiger Fehler bei der Auswertung von Formulareingaben

```
002   if (array_key_exists('poi',$_GET)) {
003    $poi = $_GET['poi'];
004   }
005   if ($poi != "") {
006     echo "<p>Sie haben die Sehenswürdigkeit ".$poi." ausgewählt.</p>";
007   } else {
008     echo "<p>Bitte wählen Sie eine Sehenswürdigkeit!</p>";
009   }
```

Wenn `'poi'` in Zeile 2 nicht übertragen wird, ist `$poi` in Zeile 5 nicht definiert, was als Fehler gilt. Das Programmstück muss daher lauten:

```
002   if (array_key_exists('poi',$_GET)) {
003     $poi = $_GET['poi'];
004   } else {
005     $poi = "";
006   }
007   if ($poi != "") {
008     echo "<p>Sie haben die Sehenswürdigkeit ".$poi." ausgewählt.</p>";
009   } else {
010     echo "<p>Bitte wählen Sie eine Sehenswürdigkeit!</p>";
011   }
```

Besser fassen Sie die beiden if-Anweisungen zusammen:

```
002   if (array_key_exists('poi',$_GET)) {
003     $poi = $_GET['poi'];
004     echo "<p>Sie haben die Sehenswürdigkeit ".$poi." ausgewählt.</p>";
005   } else {
006     $poi = "";
```

```
007     echo "<p>Bitte wählen Sie eine Sehenswürdigkeit!</p>";
008   }
```

Die Zeilen 4 und 7 stehen dabei jeweils für die gesamte erforderliche Bearbeitung des if- bzw. else-Blockes.

PHP-Fehler finden (debugErrors.php)

Wie können Sie Fehler in PHP besser lokalisieren und dann beheben? Dazu gibt es die PHP-Funktion error_log. Die Funktion heißt zwar error_log, mit ihr lässt sich aber insbesondere der Ablauf des PHP-Programms verfolgen. Die Anwendung zeige ich Ihnen in einer kleinen php-Datei.

```
001 <?php
002 # Es gibt eine vordefinierte Funktion error_log() zum Erzeugen von
Meldungen
003 # Damit können Sie erstens verfolgen, durch welche Codeteile das
Programm gelaufen ist.
004 # Zweitens kann man sich den Inhalt von Variablen ausgeben lassen.
005 # Der erste Parameter enthält die frei gewählte Meldung.
006 # Der zweite Parameter kann diese Werte haben:
007 #     0 Ausgabe der Fehlermeldung auf die Standardfehlerdatei;
008 #       in der Regel xampp/apache/logs/error.log
009 #     1 Fehler wird per E-Mail verschickt; in der Regel abgeschaltet.
010 #     3 Fehler wird auf eine private Datei ausgegeben;
011 #       im Beispiel error.log
012 error_log("Meine erste Fehlermeldung\n",0);
013 error_log("Meine zweite Fehlermeldung\n",1,"hans.kern@hs-
karlsruhe.de");
014 error_log("Meine dritte Fehlermeldung\n",3,"error.log");
015 $a = "hugo";
016 $b = "hugo ";
017 if ($a == $b) {
018   error_log("Die Variable \$a hat den Wert ***$a***.".
019   " Dieser stimmt mit dem Wert der Variablen \$b überein.\n",3,
020   "error.log");
021 } else {
022   error_log("Die Variable \$a hat den Wert ***$a***.".
023   " Dieser stimmt nicht mit dem Wert der Variablen \$b überein.\n",3,
024   "error.log");
025 }
026 # Sie haben natürlich in diesem einfachen Beispiel sofort gesehen, dass
das Programm
027 # durch den else-Block läuft; denn $a enthält 4 Zeichen, $b dagegen 5.
028 # Der Wert einer der kritischen Variablen wird ebenfalls gezeigt.
029 $buchstaben = range('a', 'z');
030 error_log("Die Variable \$buchstaben hat den Wert *$buchstaben*. \n",3,
031           "error.log");
032 error_log("Die Variable \$buchstaben[1] hat den Wert *$buchstaben[1]*.
\n",3,
033           "error.log");
034 ?>
```

Fehler finden Sie auch, wenn Sie sich den Inhalt kritischer Variablen ausgeben lassen. Das geht mit echo häufig nicht, da dann möglicherweise der Variablenwert in ein `<div>` geschrieben wird, das nicht sichtbar ist! Besser ist in solchen Fällen `var_dump($variable)`.

Beispiel für Client-Server-Interaktion (Client-Server-Interaktion/index.php)

```
01 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
02    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
03 <html xmlns="http://www.w3.org/1999/xhtml">
04 <head>
```

```
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
<title>Client-Server-Interaktion</title>
</head>
<body>
<p>Mit &quot;Abschicken&quot; werden Ihre Eintr&auml;ge in den Feldern an den
Server &uuml;bertragen. Der Server veranla&szlig;t dann, da&szlig; die Skriptdatei, die unter &quot;action&quot; aufgef&uuml;hrt ist, Ihre Eingaben
verarbeitet. Dazu m&uuml;ssen Sie diese Skriptdatei
programmiert haben, wozu Sie eine der Skriptsprachen Perl, PHP oder
JSP verwenden k&ouml;nnen. An der Hochschule Karlsruhe sind aus
Sicherheitsgr&uuml;nden nur Perl-Skripte zugelassen.</p>
<?php
    $size = count($_GET);
    if ($size) {
          echo "<h3>Formular-Auswertung</h3><p>_GET hat ".$size." Elemente.<br/>\n";
    }
    foreach ($_GET as $key => $value) {
          if (is_array($_GET[$key])) {
                foreach ($_GET[$key] as $underkey => $value) {
                      echo "Der Schl&uuml;ssel ".$key." mit Unterschl&uuml;ssel ".$underkey." hat den Wert ".$value."<br/>\n";
                }
          } else {
                echo "Der Schl&uuml;ssel ".$key." hat den Wert ".$value."<br/>\n";
          }
    }

    echo "<br/></p>\n";
    echo "<p>\n";

    if (array_key_exists('familienname',$_GET)) {
          $familienname = $_GET['familienname'];
          echo "Ihr Familienname ist: ".$familienname."<br/>\n";
    }
    if (array_key_exists('passwort',$_GET)) {
          $passwort = $_GET['passwort'];
          echo "Ihr Passwort ist: ".$passwort."<br/>\n";
    }
    if (array_key_exists('geschlecht',$_GET)) {
          $geschlecht = $_GET['geschlecht'];
          echo "Ihr Geschlecht ist: ".$geschlecht."<br/>\n";
    }
    if (array_key_exists('eigenschaft',$_GET)) {
          $eigenschaft = $_GET['eigenschaft'];
          foreach ($eigenschaft as $key => $value) {
                echo "Der Array Eigenschaft mit Schl&uuml;ssel ".$key." hat den Wert ".$value."<br/>\n";
          }
    }
    if (array_key_exists('bemerkungen',$_GET)) {
          $bemerkungen = $_GET['bemerkungen'];
          echo "Ihre Bemerkungen sind: ".$bemerkungen."<br/>\n";
    }
    if (array_key_exists('sport',$_GET)) {
          $sport = $_GET['sport'];
          foreach ($sport as $key => $value) {
                echo "Der Array Sport mit Schl&uuml;ssel ".$key." hat den Wert ".$value."<br/>\n";
```

```
60          }
61     }
62     echo "</p>\n";
63 ?>
64
65 <h3>Formular-Eingabe</h3>
66 <form method="get" action="">
67
68     <p>Ihr Name:<input type="text" size="30" maxlength="50"
name="familienname" value="Mayer" /><br/>
69
70     Ihr Passwort:<input type="password" size="30" maxlength="50"
name="passwort" value="geheim" /><br/>
71
72     m&auml;nnlich:<input type="radio" checked="checked" name="geschlecht"
value="m" />
73     weiblich:<input type="radio" name="geschlecht" value="f" /><br/>
74
75     Brillentr&auml;ger:<input type="checkbox" checked="checked"
name="eigenschaft[]" value="brille" />
76     Autofahrer:<input type="checkbox" name="eigenschaft[]" value="auto"
/></p>
77
78     <p>F&uuml;r Ihre Bemerkungen:<br/>
79     <textarea cols="20" rows="6"
name="bemerkungen">Anregungen:</textarea></p>
80
81     <p>Ihre sportlichen Pr&auml;ferenzen:<br/>
82     <select multiple="multiple" name="sport[]" size="3">
83          <option selected="selected" value="ten">Tennis</option>
84          <option value="fus">Fu&szlig;ball</option>
85          <option selected="selected" value="squ">Squash</option>
86          <option value="sno">Snowboard</option>
87     </select><br/>
88
89     <input type="reset" name="RESET" value="Zur&uuml;cksetzen" />
90     <input type="submit" name="SUBMIT" value="Abschicken" /></p>
91 </form>
92
93 </body>
94 </html>
```

Beispiel für eine PHP-Include-Datei (SampleDivInc+Google3/dbConnect.inc.php)

```
001 <?php
002   $db_server = "localhost";
003   $db_name = "sample";
004   $db_user = "hans";
005   $db_passwort = ".kelapA,";
006 /* Connect with database */
007   $dbh = mysql_connect($db_server,$db_user,$db_passwort);
008   mysql_select_db($db_name,$dbh) OR die("Mist! ".mysql_error());
009 ?>
```

Beispiel für die Zugriffsmöglichkeiten auf Datenbanken mit PHP (Mysql/mysql_all.php)

```
001 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
002      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
003 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
004 <head>
005 <title>Alle wichtigen Typen von MySQL-Anwendungen</title>
006 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
007 <style type="text/css">p {font-family : "Courier";}</style>
```

```
008 </head>
009 <body>
010 <!--
011 //    This file is part of a php tutorial
012 //
013 //    (C) Copyright 2009 Hans Kern
014 //
015 //    Created by   : Hans Kern
016 //    Creation date: 09.03.2010
017 -->
018 <?php
019 require_once("dbConnect.inc.php");
020
021 echo "<h3>Welche Datenbanken gibt es?</h3>\n";
022 echo "<p>";
023 $query = "SHOW DATABASES";                                          //
Die Variable $query enthaelt einen SQL-Befehl!
024 $sth = mysql_query($query);                                         //
sth = statement handle
025 while ( $row  =  mysql_fetch_array($sth,MYSQL_ASSOC) ) {            //
Hole Zeile fuer Zeile aus der Tabelle!
026   foreach ($row as $value) {                                        //
Mach was mit allen Spalten!
027     echo "\$row hat \$value $value  \n";                       //
Oops! Die Zeilen haben nur eine Spalte!
028     $db = $value;                                                   //
Merk dir die letzte Datenbank!
029   }
030   echo "<br />\n";
031 }
032 mysql_free_result($sth);
033 echo "</p>";
034
035 $db ='sample';
036
037 echo "<h3>Welche Tabellen hat die Datenbank '".$db."'?</h3>\n";
038 echo "<p>";
039 $sth = mysql_query("SHOW TABLES FROM $db");                         //
Ohne $query fuer den SQL-Befehl
040 while ( $row = mysql_fetch_array($sth,MYSQL_ASSOC) ) {              //
Hole Zeile fuer Zeile aus der Tabelle!
041   foreach ($row as $value) {                                        //
Mach was mit allen Spalten!
042     echo "\$row hat \$value $value  ";                         //
Oops! Die Zeilen haben nur eine Spalte!
043     $table = $value;                                                //
Merk dir die letzte Tabelle!
044   }
045   echo "<br />\n";
046 }
047 mysql_free_result($sth);
048 echo "</p>";
049
050 echo "<h3>Zeige die Namen der Spalten und ihre Attributwerte für die
Tabelle '".$table."' der Datenbank '".$db."'!</h3>\n";
051 echo "<p>";
052 $query = "SHOW COLUMNS FROM $table FROM $db";
053 $sth = mysql_query($query);
054 while ( $row = mysql_fetch_array($sth,MYSQL_NUM) ) {
055   for ($i = 0;  $i < count($row);  $i++) {                          //
Jetzt einmal mit einer for-Loop!
056     echo "'$row[$i]',  ";
```

```
057   }
058   echo "<br />\n";
059 }
060 mysql_free_result($sth);
061 echo "</p>";
062
063 echo "<h3>Zeige alle Zeilen der Tabelle '".$table."' der Datenbank
'".$db."'!</h3>\n";
064 echo "<p>";
065 $query = "SELECT * FROM $db.$table";
066 $sth = mysql_query($query);
067 while ( $row = mysql_fetch_row($sth) ) {
068   for ($i = 0; $i < count($row); $i++) {
069     echo "'$row[$i]',  ";
070   }
071   echo "<br />\n";
072 }
073 mysql_free_result($sth);
074 echo "</p>";
075
076 echo "<h3>Zeige einige Zeilen der Tabelle '".$table."' der Datenbank
'".$db."' mit mysql_fetch_row!</h3>\n";
077 echo "<p>";
078 $query = "SELECT * FROM $db.$table LIMIT 2";
079 $sth = mysql_query($query);
080 while ( $row = mysql_fetch_row($sth) ) {
081   for ($i = 0; $i < count($row); $i++) {
082     echo "'$row[$i]',  ";
083   }
084   echo "<br />\n";
085 }
086 mysql_free_result($sth);
087 echo "</p>";
088
089 echo "<h3>Zeige einige Zeilen der Tabelle '".$table."' der Datenbank
'".$db."' mit mysql_fetch_assoc!</h3>\n";
090 echo "<p>";
091 $query = "SELECT * FROM $db.$table LIMIT 2";
092 $sth = mysql_query($query);
093 while ( $aref = mysql_fetch_assoc($sth) ) {
094   foreach ( @$aref as $key => $value) {
095     echo "'($key=$value)',  ";
096   }
097   echo "<br />\n";
098 }
099 mysql_free_result($sth);
100 echo "</p>";
101
102 echo "<h3>Zeige einige Zeilen der Tabelle '".$table."' der Datenbank
'".$db."' mit mysql_fetch_object!</h3>\n";
103 echo "<p>";
104 $query = "SELECT name, x, y FROM $db.$table LIMIT 2";
105 $sth = mysql_query($query);
106 while ( $object = mysql_fetch_object($sth) ) {
107   echo "$object->name,   $object->x,   $object->y,  ";
108   echo "<br />\n";
109 }
110 mysql_free_result($sth);
111 echo "</p>";
112
113 echo "<h3>Zeige die Namen der Spalten der Tabelle '".$table."' der
Datenbank '".$db."' mit mysql_field_name!</h3>\n";
```

```
114 echo "<p>";
115 $query = "SELECT * FROM $db.$table";
116 $sth = mysql_query($query);
117 for ($i = 0; $i < mysql_num_fields($sth); $i++) {
118   $field_name = mysql_field_name($sth,$i);
119   echo "'$field_name',  ";
120 }
121 echo "<br />\n";
122 mysql_free_result($sth);
123 echo "</p>";
124
125 echo "<h3>Zeige die Namen der Spalten und einige Zeilen der Tabelle
'".$table."' der Datenbank '".$db."'!<br />\nBeachte die gleiche
Reihenfolge!</h3>\n";
126 echo "<p>";
127 $query = "SELECT * FROM $db.$table LIMIT 2";
128 $sth = mysql_query($query);
129 for ($i = 0; $i < mysql_num_fields($sth); $i++) {
130   $field_name = mysql_field_name($sth,$i);
131   echo "'$field_name',  ";
132 }
133 echo "<br />\n";
134
135 while ( $row = mysql_fetch_array($sth,MYSQL_NUM) ) {
136   for ($i = 0; $i < count($row); $i++) {
137     echo "'$row[$i]',  ";
138   }
139   echo "<br />\n";
140 }
141 mysql_free_result($sth);
142 echo "</p>";
143
144 echo "<h3>Hebe die Verbindung zum Datenbankhandle \$dbh auf!</h3>\n";
145 mysql_close($dbh);
146 ?>
147 </body>
148 </html>
```

Beispiel für eine PHP-Seite mit einem Drop-Down-Menü (SampleDivInc +Google3/query01.php)

```
001 <?php
002 require_once('./inc/header.inc.php');
003 ?>
004         <p class='menue'>Es werden die Sehenswürdigkeiten angezeigt,
die
005         man zu einem gewählten maximalen Eintrittspreis besuchen
kann.</p>
006         <p>Der Eintrittspreis für Kinder ist stets &euro; 0.</p>
007 <?php
008 /* first part: process form input */
009 if (array_key_exists('price',$_GET)) {/*Branch if form variable does
exist*/
010   $price = $_GET['price'];
011   echo "<p class='submenue'>Sie haben als maximalen Eintrittspreis
&euro;$price ausgewählt.</p>\n";
012   $query = "SELECT name FROM pois WHERE price_adults <= $price ORDER BY
name";
013   $result = mysql_query($query) OR die("Mist! ". mysql_error());
014   echo "<p><span class='bold'>Die möglichen
Sehenswürdigkeiten:</span><br />\n";
```

```
015   while ($row = mysql_fetch_array($result, MYSQL_ASSOC)) { /* row by
row as associative array*/
016      echo $row['name']."<br />\n";
017   }
018   echo "</p>\n";
019   mysql_free_result($result);
020 } else {/*Branch if form variable does not exist*/
021   echo "<p class='submenue'>Bitte wählen Sie den maximalen
Eintrittspreis pro Erwachsenem!</p>\n";
022 }
023 /* second part: generate form */
024 echo "<form action='' method='get'><p>\n";
025 echo "<select name='price' size='1'>\n";
026 $query = "SELECT DISTINCT price_adults FROM pois ORDER BY
price_adults";
027 $result = mysql_query($query);
028 while ($row = mysql_fetch_array($result, MYSQL_ASSOC)) { /* row by row
as associative array*/
029   echo "<option
value='".$row['price_adults']."'>".$row['price_adults']."</option>\n";
030 }
031 mysql_free_result($result);
032 echo "</select>\n";
033 echo "<input type='submit' value='OK' />\n";
034 echo "</p></form>\n";
035 ?>
036 <?php
037 require_once('./inc/footer.inc.php');
038 ?>
```

Beispiel für eine PHP-Seite mit zwei Drop-Down-Menüs (SampleDivInc+ Google3/query05.php)

```
001 <?php
002 require_once('./inc/header.inc.php');
003 ?>
004         <p class='menue'>Es wird die Luftlinienentfernung zwischen zwei
gewählten Sehenswürdigkeiten angezeigt.</p>
005         <p> </p>
006 <?php
007 /* first part: process form input */
008 /* Is there input from a form? */
009 if (array_key_exists('poi1',$_GET) AND array_key_exists('poi2',$_GET))
{
010   /* Yes, there is input from a form? */
011   $poi1 = $_GET['poi1'];
012   $poi2 = $_GET['poi2'];
013   echo "<p class='submenue'>Sie haben die Sehenswürdigkeiten $poi1 und
$poi2 ausgewählt.</p>\n";
014   $query = "SELECT name, x, y FROM pois WHERE name='$poi1' ORDER BY
name";
015   $result = mysql_query($query) OR die("Mist! ". mysql_error());
016   $objekt1 = mysql_fetch_object($result); /* first row as object */
017   $query = "SELECT name, x, y FROM pois WHERE name='$poi2' ORDER BY
name";
018   $result = mysql_query($query) OR die("Mist! ". mysql_error());
019   $objekt2 = mysql_fetch_object($result); /* first row as object */
020   /* $result is not used anymore */
021   mysql_free_result($result);
022   if ($objekt1->name != $objekt2->name) {
023      echo "<p>Der Abstand zwischen den Sehenswürdigkeiten <br />";
024      echo "<span class='bold'>".$objekt1->name."</span> und ";
```

```
025       echo "<span class='bold'>".$objekt2->name."</span> beträgt: ";
026       $abstand = ROUND(ACOS(
027         SIN($objekt1->x/180.0*PI()) * SIN($objekt2->x/180.0*PI()) +
028         COS($objekt1->x/180.0*PI()) * COS($objekt2->x/180.0*PI()) *
029         COS($objekt1->y/180.0*PI()-$objekt2->y/180.0*PI()))*6372.795,2);
030       echo "<br /><span class='bold'>";
031       echo $abstand;
032       echo " km</span></p>\n";
033    } else {
034       echo "<p>Sie haben die gleichen Sehenswürdigkeiten gewählt.";
035       echo "<br />Bitte versuchen Sie es noch einmal!</p>\n";
036    }
037 } else {
038    /* No, there isn't input from a form? */
039    $poi1 = "";
040    $poi2 = "";
041    echo "<p class='submenue'>Bitte wählen Sie zwei
Sehenswürdigkeiten!</p>\n";
042 }
043 /* second part: generate form */
044 $query = "SELECT DISTINCT name FROM pois ORDER BY name";
045 $result = mysql_query($query);
046 echo "<form action='' method='get'><p>\n";
047 echo "<select name='poi1' size='1'>\n";
048 while ($row = mysql_fetch_array($result, MYSQL_ASSOC)) { /* row by row
as associative array*/
049    echo "<option value='".$row['name']."'>".$row['name']."</option>\n";
050 }
051 echo "</select>\n";
052 echo "    ";
053 /* in order to read $result twice you must reset $result */
054 mysql_data_seek($result, 0);
055 echo "<select name='poi2' size='1'>\n";
056 while ($row = mysql_fetch_array($result, MYSQL_ASSOC)) { /* row by row
as associative array*/
057    echo "<option value='".$row['name']."'>".$row['name']."</option>\n";
058 }
059 echo "</select>\n";
060 echo "<input type='submit' value='OK' />\n";
061 echo "</p></form>\n";
062 mysql_free_result($result);
063 ?>
064 <?php
065 require_once('./inc/footer.inc.php');
066 ?>
```

PHP-Dynamikstufen und -Komplexität

Dynamikstufe 0

Es sind keine Eingaben erforderlich. Es wird eine Seite aufgerufen, die nicht von der Eingabe des Benutzers abhängt. Ein Beispiel dafür: Liste alle Linien!

Dynamikstufe 1

Der Benutzer füllt ein Formular aus und der Server antwortet auf die Formulareingaben. Beispiel: Liste alle Pois, die in einer gewählten Entfernung eines gewählten Pois liegen! Das ist die Dynamikstufe 1, obwohl zwei Eingaben erfolgen!

Dynamikstufe 2

Nach dem Ausfüllen eines Formulars schickt der Server in Abhängigkeit davon ein weiteres Formular. Beispiel: Liste die Fahrzeit zwischen zwei gewählten Haltestellen

einer gewählten Linie! Das ist die Dynamikstufe 2, denn nach der Wahl der Linie, zeigt der Server nur die Haltestellen dieser Linie. Es erfolgen drei Eingaben!

Dynamikstufe 3

Beispiel: Zeige die Haltestellen, die von einer gewählten Linie zu erreichen sind, die eine gewählte Umsteigehaltestelle bedient, die auf der Linie einer gewählten Haltestelle liegt!

Geben Sie für diese Stufen jeweils die generelle Struktur der PHP-Datei an!

Schwierigeres PHP

Wenn eine Seite ein weiteres Mal gezeigt wird, sollten die Eingaben vom vorherigen Mal stehen geblieben sein. Wenn die Seite unmittelbar vorher gezeigt wurde, nutzen wir hidden-Input. Eingaben können wir während einer Sitzung speichern mit der session-Technik.

Ajax

In herkömmlichen Anwendungen, wie auch in Ihrem Projekt, wird im Browser ein Formular ausgefüllt und SUBMIT gegeben. Der Browser überträgt die Seite an den Server und die Antwort des Servers wird wieder auf den Browser zurück übertragen. Mit Ajax (**A**synchronous **J**avaScript **a**nd **X**ML) wird nach Ausfüllen eines Formularfeldes dieses Feld asynchron auf den Server übertragen, ausgewertet und das Ergebnis an den Browser zurück gegeben. Wir können so erreichen, dass sich der Browser wie übliche Desktop-Anwendungen verhält, da nicht mehr volle Seiten neu aufgebaut werden müssen. Ein Beispiel finden Sie im Ordner Ajax.

JavaScript

Die Bemerkungen zu PHP gelten weitgehend auch für JavaScript, insbesondere was zur Verwendung von { und } und zu den Einrückungen dargestellt wurde.

JavaScript-Programmteile dürfen keine php-Dateien aufrufen, da JavaScript auf dem Client läuft, PHP aber nur auf dem Server laufen soll. Leeren Sie einmal "C:\Dokumente und Einstellungen\<name>\Lokale Einstellungen\Temporary Internet Files". Dort steht alles, was Ihr Browser so nach und nach aus dem Netz bezogen hat. Dort dürfen keine php-Dateien zu finden sein.

Google Maps JavaScript API Version 3

Ihr Datenbankprojekt soll auch Karten zeigen. In lange vergangenen Semestern wurde es so gemacht, dass geeignete Karten gescannt und eingebunden wurden, um dann bei der Vorführung des Projektes auf dem lokalen Rechner präsentiert zu werden. Die Projektarbeit insgesamt konnte wegen der Urheberrechte an den Kartenauszügen so nicht ins Netz gestellt werden. Und der Behelf, statt einer Karte nur eine graue Fläche zu zeigen, ist nicht sehr cool. Daher sparen Sie sich die Arbeit, Karten zu scannen, und bereichern stattdessen mit Google-Maps Ihr eigenes Projekt.

Für die Google Maps JavaScript API Version 3 finden Sie auf der folgenden Internetadresse zu allen Features einfache, gut nachvollziehbare Beispiele:

http://code.google.com/intl/de-DE/apis/maps/documentation/javascript/examples/index.html

Eine vollständige Referenz finden Sie unter:

http://code.google.com/intl/de-DE/apis/maps/documentation/javascript/reference.html

Google-Koordinaten

Google verwendet geographische Koordinaten; also Länge (longitude) und Breite (latitude). Bei Google wird es immer in der Reihenfolge latitude, longitude eingesetzt. Wenn Sie geographische Koordinaten erfassen müssen, finden Sie Hilfe unter http://itouchmap.com/latlong.html. Wenn Sie eine andere Adresse verwenden, achten Sie darauf, dass Sie die Koordinaten bequem kopieren können, nämlich beide Werte mit einem Kopiervorgang.

Gauß-Krüger Koordinaten

Im Ordner `Sample` sind Beispiele enthalten, wie Sie Gauß-Krüger-Koordinaten in geographische umrechnen können. Erledigt wird das mit der PHP-Datei `gaussk.inc.php`. Wenn Sie von Anfang an geographische Koordinaten verwenden, brauchen Sie `gaussk.inc.php` nicht.

Beispiel 1 (SampleDivInc+Google3/query21.html)

Dieses Beispiel zeigt die Verwendung von GoogleMaps ohne Zugriff auf die Datenbank mittels PHP. JavaScript-Teile werden mit `//<![CDATA[` und `//]]>` "eingewickelt". Das Beispiel bietet Ihnen die generelle Struktur. Für die direkte Verwendung in Ihrem Projekt ist es nicht geeignet, da Sie die Punktinformationen aus der Datenbank holen müssen. Wenn Sie in Ihrem Projekt eine Google-Anwendung geschrieben haben und Sie sehen sich die Ausführung im Browser als XHTML-Quelltext an, wird der JavaScript-Teil so ähnlich wie hier aufgeführt aussehen. Strukturelle Abweichungen deuten dann auf Fehler in Ihrer php-Datei. Auch sind in diesem Beispiel die Konstanten nicht benannt worden.

```
001 <?php
002 require_once('./inc/header.inc.php');
003 ?>
004   <script type="text/javascript">
005 //<![CDATA[
006 function initialize() {
007   var latlng = new google.maps.LatLng(51.51611, 7.46805);
008   var myOptions = {
009      center: latlng,
010      mapTypeId: google.maps.MapTypeId.ROADMAP,
011      overviewMapControl: true,
012      overviewMapControlOptions: {
013         opened: true
014      },
015      scaleControl: true,
016      zoom: 12,
017      zoomControl: true,
018      zoomControlOptions: {
019         style: google.maps.ZoomControlStyle.LARGE
020      }
021   };
022   var map = new
google.maps.Map(document.getElementById("map_canvas"), myOptions);
023   var icon = new google.maps.MarkerImage("grafics/marker.png",
024      // This marker is 116 pixels wide and 116 pixels tall.
025      new google.maps.Size(116, 116),
026      // The origin for this icon is 0,0.
027      new google.maps.Point(0, 0),
028      // The anchor for this icon is the base of the flagpole at 25,25.
029      new google.maps.Point(25, 25),
030      // The scaled Size for this icon is 50,50.
031      new google.maps.Size(50, 50)
```

```
032   );
033   var shadow = new google.maps.MarkerImage("grafics/marker_shadow.png",
034      new google.maps.Size(116, 116),
035      new google.maps.Point(0, 0),
036      new google.maps.Point(25, 25),
037      new google.maps.Size(50, 50)
038   );
039 /* Define and locate markers */
040   var point = new Array();
041   var text = new Array();
042       point[0] = new
google.maps.LatLng(51.529670568937,7.4678919705624);
043       text[0] = "Text mit <b>Hervorhebung<\/b><br> in mehreren Zeilen";
044       point[1] = new
google.maps.LatLng(51.492303610236,7.4949491229473);
045       text[1] = "Text mit &Auml; und &auml; und &szlig;";
046       point[2] = new
google.maps.LatLng(51.512995359297,7.4643890005705);
047       text[2] = "Text mit Ä, ä, ß, ë, ô und ç";
048       point[3] = new
google.maps.LatLng(51.510555229501,7.4647591604423);
049       text[3] = "Text mit ', \", \\, &, < und >";
050   var marker = new Array();
051   for (i=0; i< point.length; i++) {
052      marker[i] = new google.maps.Marker({
053         position: point[i],
054         map: map,
055         icon: icon,
056         shadow: shadow,
057         title: text[i],
058         clickable: true
059      });
060      google.maps.event.addListener(marker[i], 'click', function () {
061         infowindow.setPosition(point[i]);
062         infowindow.setContent(this.title);
063         infowindow.setOptions({
064            pixelOffset: new google.maps.Size(-35, 25)
065         });
066         infowindow.open(map, this);
067      });
068   }
069   var contentString = 'Nach Auskunft des Dortmund-Portals ist hier das
<b>Zentrum<\/b> von Dortmund!';
070
071   var infowindow = new google.maps.InfoWindow({
072      content: contentString,
073      position: latlng
074   });
075   infowindow.open(map);
076 }
077 google.maps.event.addDomListener(window, "load", initialize);
078 //]]>
079   </script>
080 <?php
081 require_once('./inc/footer.inc.php');
082 ?>
```

Beispiel 2 (SampleDivInc+Google3/query22.php)

Das Beispiel 2 enthält in Erweiterung des Beispiels 1 zusätzlich den PHP-Zugriff auf die Datenbank und die Erzeugung von JavaScript mit PHP. Das ist ein typisches Beispiel für "mash up"!

```
001 <?php
002 require_once('./inc/header.inc.php');
003 ?>
004 <?php //Start of PHP
005 $lat = '51.51350';
006 $lon = '7.46500';
007 $bezeichnung = 'Willkommen in Dortmund!';
008 $zoom = '12';
009 /* Build SQL query */
010 $query = /*Start of SQL*/"
011   SELECT x, y, name FROM pois;
012 "; /*End of SQL*/
013 /* Execute SQL query */
014 $result = mysql_query($query);
015 if ($result) {
016   $rows = 0;
017   while ($row = mysql_fetch_object($result)) {
018     $xobj[$rows] = $row->x;
019     $yobj[$rows] = $row->y;
020     $nobj[$rows] = $row->name;
021     $rows++;
022   }
023 } else {
024   echo 'Fehler';
025 }
026 /* $result is not used anymore */
027 mysql_free_result($result);
028 echo '<script type="text/javascript">//<![CDATA['; //Generate JS with
PHP
029 echo '/*'.$rows.' '.$xobj[0].' '.$yobj[0].'*/';
030 echo /*A snippet of pure JS*/ '
031   function initialize() {
032      var latlng = new google.maps.LatLng('.$lat.', '.$lon.');
033      var myOptions = {
034         center: latlng,
035         mapTypeId: google.maps.MapTypeId.ROADMAP,
036         overviewMapControl: true,
037         overviewMapControlOptions: {
038            opened: true
039         },
040         scaleControl: true,
041         zoom: '.$zoom.',
042         zoomControl: true,
043         zoomControlOptions: {
044            style: google.maps.ZoomControlStyle.LARGE
045         }
046      };
047      var map = new
google.maps.Map(document.getElementById("map_canvas"), myOptions);
048      var point = new Array();
049      var text = new Array();
050      var marker = new Array();
051 '; /*End of JS, $lat, $lon, $zoom are embedded PHP*/
052 for ($r=0; $r<$rows; $r++) {
053   echo /*Again start of JS*/'
054      point['.$r.'] = new
google.maps.LatLng('.$yobj[$r].','.$xobj[$r].');
055      text['.$r.'] = "'.$nobj[$r].'";
056   ';/*End of JS*/
```

```
057 }
058 echo /*Again start of JS*/'
059      var infowindow = new google.maps.InfoWindow();
060      for (i=0; i< point.length; i++) {
061         marker[i] = new google.maps.Marker({
062            position: point[i],
063            map: map,
064            title: text[i],
065            clickable: true,
066         });
067         google.maps.event.addListener(marker[i], "click", function () {
068            infowindow.setPosition(point[i]);
069            infowindow.setContent(this.title);
070            infowindow.open(map, this);
071         });
072      }
073   }
074   google.maps.event.addDomListener(window, "load", initialize);
075   //]]></script>
076 ';/*End of JS script creation*/
077 ?> <!--End of PHP-->
078 <?php
079 require_once('./inc/footer.inc.php');
080 ?>
```

Fehlersuche

Siehe auch oben PHP-Fehler finden.

Im Beispiel 2 sind folgende Sprachen miteinander "verwurschtelt" (mash up): PHP, XHTML, MySQL, JaveScript. Das macht die Erstellung ziemlich fehleranfällig und Sie sollten sich von Beginn an damit vertraut machen, wie Sie Fehler lokalisieren können. Als erstes muss PHP fehlerfrei sein. Das überprüfen Sie, indem Sie sich das Apache-Fehler-Log xampp/apache/logs/error.log ansehen. Sobald PHP fehlerfrei ist, muss als nächstes auch das erzeugte JavaScript fehlerfrei sein. Das JavaScript können Sie sich ansehen, indem Sie sich im Browser den Quelltext zeigen lassen. Er sollte die Struktur aus Beispiel 1 haben. Fehler im JavaScript werden im Browser in der Fehler-Konsole angezeigt. Firefox: Extras-->Fehler-Konsole; IE: im Frame rechtsklicken und "Quelltext anzeigen" wählen.