Dokumentation zur betrieblichen Projektarbeit

Umzug einer Webseite auf einen schnelleren Server mit Datenbankanpassung

Name:	Ralf Mengel
Ausbildungsberuf:	Fachinformatiker / Systemintegration
Ausbildungsbetrieb:	SPE Siemens AG
Prüflingsnummer:	133 90118
Prüfungsausschuss:	FISY Han 10 110

Inhaltsverzeichnis

Abkürzungsverzeichnis

ca.	circa
ggf.	gegeben falls
z.B.	zum Beispiel

1 Einleitung

1.1 Unternehmen

Das Unternehmen „ABACOM Datensysteme GmbH“ wurde im Jahre 1990 von Ulrich Abel gegründet. Der Schwerpunkt des Unternehmens lag in der kaufmännischen und Produktionsorientierten Softwareprogrammierung. Davor war er sieben Jahre lang als Softwareentwickler tätig.

Es sind sechs Vollzeitmitarbeiter angestellt und mehrere freiberufliche Kräfte. Im Laufe der Zeit hat sich das Aufgabengebiet des Unternehmens weiterentwickelt und deckt heute folgende Bereiche ab:

- Softwareentwicklung
- Installation und Verkauf von Software
- Einrichtung und Verkauf von PC-gestützten Netzwerken
- Netzwerkbetreuung

1.2 Webseite

Der Geschäftsführer spielt Tischtennis im Verein und hatte vor einigen Jahren eine Webseite in Auftrag gegeben, um Spiele zu verwalten. Eine Besonderheit der Webseite ist die Berechnung der Wertungszahlen. Diese ergeben sich aus den Spielen und deren Ergebnissen.

2 Ist-Analyse

2.1 Hardware

Der Webserver wird von dem Provider 1Blue bereitgestellt.
Es ist ein virtuelles System mit einem Speicherplatz von 18 GB und 400 MB RAM. Zur Administration des Servers ist die Software Plesk in der Version 8 installiert.

2.2 Software

Auf dem Server läuft die Linux Distribution SuSE 9.3. Weiterhin sind die Pakete Apache 2.0.53, PHP 4.3.10 und MySQL 4.1.10a installiert.

2.3 besondere Einstellungen

register_globals = on

Dadurch, dass die Webseite historisch gewachsen ist, sind einige sicherheitskritische Einstellungen vorgenommen worden. Daraufhin wurde anhand dieser Einstellung programmiert:

```php
<?php
// REGISTER_GLOBALS ON
// Beispiel Link: http://www.domain.de/index.php?uebergabe=1
echo $uebergabe; // Ausgabe: 1

// REGISTER_GLOBALS OFF
// Beispiel Link: http://www.domain.de/index.php?uebergabe=1
echo $_GET['uebergabe']; // Ausgabe: 1
?>
```

Jetzt ist es möglich, z.B. bei einem Formular, bei dem die Daten mit POST gesendet werden in der Adresszeile auch GET Parameter mitzugeben, um die POST Variablen zu überschreiben.

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>register globals on</title>
</head>
<body>
<form action="<?=$_SERVER['PHP_SELF']?>" method="POST">
<input type="text" name="uebergabe"><br>
<input type="submit"><br>
<?php
echo "POST: ".$_POST['uebergabe']."<br>";
echo "GET: ".$_GET['uebergabe']."<br>";
echo "OHNE: ".$uebergabe;
?>
</body>
</html>
```

Beim Aufrufen der Webseite wird nichts angezeigt (POST, GET, OHNE), nachdem das Formular ausgefüllt und auf Abschicken geklickt wurde, wird bei POST und OHNE die Eingabe angezeigt.

Ruft man die Webseite mit folgenden Parametern auf werden GET und OHNE angezeigt
`[...]/register_globals_on.php?uebergabe=2`

max_execution_time = 14400

Die Zeit für die Ausführung jedes PHP-Scripts in Sekunden.
Einige Berechnungen dauern sehr lange, und damit das PHP-Script nicht abstürzt, wurde diese relativ hohe Zeit eingestellt.

2.4 manuell Ausgeführte wiederkehrende Ereignisse

2.4.1 wzcalc

Es sind PHP-Scripte vorhanden, die manuell ausgelöst werden, um die Berechnungen der Wertungszahlen durchzuführen. Auf dem aktuellen Server dauert die Berechnung ca. 6 Stunden.

2.4.2 addSaison

Für neue Saisoneinträge wird eine vorhandene Excel Tabelle mit Saisondaten benutzt, die manuell angepasst wird, und anschließend als CSV Dump in die Datenbank eingespielt wird. Diese Arbeit nimmt einmal im halben Jahr ca. 2 Stunden Zeit in Anspruch.

3 Soll-Konzeption

Der Umzug auf einen neuen Server hat das Ziel eine höhere Geschwindigkeit mittels neuerer und besserer Hardware zu erzielen. Der Einsatz von aktuellen Softwarepaketen, Anpassungen an der MySQL Datenbank und den PHP-Scripten soll ebenfalls Geschwindigkeitsvorteile erbringen.

3.1 Hardware

Der virtuelle Server von 1blu wurde abgeschafft und ein eigener Server benutzt, der schon vor dem Projektbeginn im Hause vorhanden war. Dieser Server hat einen Intel Pentium 4 mit 3.00GHz, 1024 MB SDRAM und eine 80 GB (ST380011A) Festplatte.

3.2 Software

Installation eines Debian 4.0 RC2 ohne grafischer Oberfläche mit den Paketen Apache 2.2.3, PHP 5.2.0-8+etch10 und MySQL 5.0.32.

3.3 besondere Einstellungen

register_globals = on

Eine Anpassung dieser Einstellung ist in einem anderen Projekt vorgesehen, dass dann eine ganzheitliche Überarbeitung der Webseite beinhalten wird.

max_execution_time = 14400

Die oben genannten Scripte werden weiterhin verwendet, sodass diese Einstellung weiterhin wichtig ist.

3.4 Cronjobs

3.4.1 wzcalc

Das noch manuelle angestoßene PHP-Script zur Berechnung der Wertungszahlen soll jede Nacht um 6 Uhr selbstständig durchlaufen und einen Log schreiben, der dann als Archiv gespeichert wird. Das Ergebnis der Berechnung wird in zwei Dateien gespeichert, einmal in die **wzcalc.access.log** und einmal in die **wzcalc.error.log**.
So sind die Erfolgsmeldungen von den Fehlermeldungen getrennt und man kann sich die Fehlermeldungen separat anschauen, um mögliche Fehler in der Berechnung zu beheben.
Die Erfolgsmeldungen werden für manuelle Überprüfungen benötigt, die aber Stichprobenartig durchgeführt werden.

3.4.2 addSaison

Ziel des Cronjobs ist es, alle halbe Jahre ein PHP-Script zu starten, dass die aktuelle Tischtennis-Saison in die Datenbank hinzufügt. Beim direkten Aufruf in einem Browser soll das PHP-Script ein Formular anzeigen, indem auch vergangene und fehlende Saisons eingetragen werden können.

4 Projektdurchführung

4.1 Planung

Durch Absprache mit dem Auftraggeber des Projektes wurden die wichtigen Aspekte hervorgehoben und ein Ablaufplan erstellt, der sich strukturell in der Gliederung dieses Abschnittes wiederspiegelt.

4.2 Installation des Betriebssystems

Die Installation des Betriebssystems Debian 4.0 RC2 wurde mit einer NetInstall CD (Link im Quellenverzeichnis) ohne grafische Oberfläche durchgeführt. Nach der erfolgreichen Installation des Betriebssystems wurden mit **aptitude** weitere Pakete nachinstalliert. Darunter waren Apache 2.2.3, PHP 5.2.0-8+etch10, MySQL 5.0.32 und der OpenSSH Server.

aptitude

Aptitude ist ein grafisches Frontend für APT (Advanced Packaging Tool) zum hinzufügen und entfernen von Softwarepaketen.

4.3 Konfiguration der Software

4.3.1 Apache

Der Apache wurde dahingehend konfiguriert, dass das Hauptverzeichnis (DocumentRoot) für die zu verarbeitende Dateien auf **/srv/www/htdocs** zeigt.

4.3.2 MySQL

Sämtliche Daten wurden auf dem alten Server in der Datenbank **tt_rangliste** gespeichert und der vom Provider vorgeschriebene Benutzername lautete **usr_tt_rangliste**. Da die PHP-Scripte diese Daten gespeichert haben, wurden auch diese Daten für den neuen Server benutzt.
Um SQL-Befehle abzusetzen, ist es erforderlich, sich an der Datenbank anzumelden.

```
mysql -u root –p
```

Daraufhin wird man nach dem Passwort für den Benutzer root gefragt und kann dann die unten genannten SQL-Befehle ausführen.

Datenbank anlegen:

```
CREATE TABLE tt_rangliste;
```

Anlegen des Benutzers:

```
CREATE USER 'usr_tt_rangliste'@'localhost' IDENTIFIED BY '****';
GRANT USAGE ON * . * TO 'test'@'localhost' IDENTIFIED BY '****'
WITH
   MAX_QUERIES_PER_HOUR 0
   MAX_CONNECTIONS_PER_HOUR 0
   MAX_UPDATES_PER_HOUR 0
   MAX_USER_CONNECTIONS 0;
```

4.3.3 PHP

Einstellungen in der **/etc/php5/apache2/php.ini**:

register_globals = on
max_execution_time = 14400

4.4 Sichern und Wiederherstellen der Daten

4.4.1 MySQL

Einwahl auf den Server mittels **SSH** und erstellen eines MySQL-Dumps
```
mysqldump tt_rangliste –u USER –p > tt_rangliste.sql
```

Einspielen des Dumps auf dem neuen Server
```
mysql tt_rangliste -u USER -p < tt_rangliste.sql
```

4.4.2 PHP

Einwahl auf den Server mittels FTP-Programm und Download der PHP-Scripte in das DocumentRoot

4.5 Anpassung der PHP-Scripte

Bei dem ersten Aufruf der Webseite im Browser waren keine Konflikte festzustellen. Das einzige Problem, das bei der Umstellung von PHP4 auf PHP5 auftrat, war der Konflikt einer Funktion im Forum (Phorum) (Link im Quellenverzeichnis), die in PHP5 den gleichen Namen besitzt.
Das Auskommentieren der Funktion und Anpassen der Funktionsaufrufe war der einfachste Weg um das Problem zu beheben.
PHP bietet auch die Möglichkeit eine existierende Funktion zu überschreiben, aber dieser Weg würde die Laufzeitgeschwindigkeit mindern.

bool **override_function** (string $function_name , string $function_args , string $function_code)

Selbstprogrammierte Funktion aus dem Forum *Phorum*

```php
<?php
function date_format($datestamp){
    $tzoffset = 0;
    if ($datestamp == "0000-00-00") {
      $datestamp = "0000-00-00 00:00:00";
    }
    list($date,$time) = explode(" ",$datestamp);
    list($year,$month,$day) = explode("-",$date);
    list($hour,$minute,$second) = explode(":",$time);
    $hour = $hour + $tzoffset;
    $tstamp = mktime($hour,$minute,$second,$month,$day,$year);
    $sDate = date("d.m.y H:i",$tstamp);
    return $sDate;
}
?>
```

PHP 5 Funktion

string date_format (DateTime $object , string $format)

alter Aufruf der Funktion

```
<?php
$neue_variable=date_format($array["datum_als_String"]);
?>
```

neuer Aufruf der Funktion

```
<?php
$neue_variable=date_format($array["datum_als_String"], "d.m.Y");
?>
```

4.6 Anpassungen testen

Nach jeder Programmänderung wurde die Webseite einmal neugeladen, um die Auswirkungen der Veränderungen zu sehen. Danach wurde ggf. weiter an der Veränderung gearbeitet oder mit dem nächsten Punkt begonnen.

4.7 Programmierung des Spam-Schutzes

Ein Spamschutz ist heutzutage unumgänglich, da Bots („intelligente“ Programme) Formulare selbstständig ausfüllen können um auf diesem Wege, z.B. in Gästebücher oder Foren, Werbung für andere Webseiten einzutragen mit fragwürdigem Inhalt.

Auf der Webseite war schon ein Spam-Schutz vorhanden, dieser wurde jedoch von den Spambots ausgelesen und so in Forum und Gästebuch geschrieben.
Dieser Spam-Schutz zeigte eine Additionsaufgabe an und verlangte nach dem Ergebnis, somit sollte geprüft werden ob es sich um ein Spambot oder einen Menschen handelt.

Diese Aufgabe war aber im HTML Quelltext sichtbar, womit der von Spambots ausgelesen und das Ergebnis in das richtige Feld eingetragen werden konnte.

Somit wurde ein neuer Spam-Schutz benötigt, der dieses Auslesen verhindert. Dazu war es erforderlich im Forum und Gästebuch zwei Dateien zu bearbeiten und eine Datei hinzuzufügen.

Auszüge oder kompletter Quellcode der Dateien sind im Anhang A zu finden

create_img.php

erzeugt ein Bild mit einer vierstelligen Zahl die als GET-Parameter übergeben wird.

form.php (Forum)

Vor der Darstellung des Formulars zum Einsenden eines neuen Beitrages wird erst einmal eine vierstellige Zahl anhand eines Zufallsgenerators (Software-Algorithmus) ein Hash-Wert erzeugt. Dieser Hash-Wert wird als GET-Parameter der Zieladresse mit angehängt. Somit kann überprüft werden ob die Eingabe stimmt oder nicht.
Des Weiteren wird jetzt die **create_img.php** mit einem GET-Parameter innerhalb des HTML Tags „<img>“ aufgerufen, um das Bild im Formular anzuzeigen.

Post.php (Forum)

Hier wird jetzt überprüft, ob die Eingabe und die Zahl auf dem Bild übereinstimmt und dann erst der Beitrag in der Datenbank abgespeichert.

4.8 Spam-Schutz testen

Alle drei Optionen wurden einmal ausprobiert und überprüft, ob der Beitrag gespeichert wird oder im Fehlerfall eine Meldung erscheint.

Spam-Code wurde nicht eingegeben:
Anzeige einer Fehlermeldung

Spam-Code wurde falsch eingegeben:
Anzeige einer Fehlermeldung

Spam-Code wurde richtig eingegeben:
Speichern des Beitrages in der Datenbank und Weiterleitung zum Beitrag im Forum

4.9 Programmierung der Cronjobs

4.9.1 wzcalc

Die Ursprüngliche wzcalc.php wurde dahingehend angepasst, dass alle Meldungen, die im Browser angezeigt werden, auch in einer Datei abgespeichert werden.

```
<?php
define
("defined_log_file", "/srv/cronjobs/tt-rangliste/log/wzcalc.access.log");
define
("defined_error_file", "/srv/cronjobs/tt-rangliste/log/wzcalc.error.log");

function setLog($option, $value)
{
   if ($option == "log")   $file = defined_log_file;
   if ($option == "error") $file = defined_error_file;
   $datei = fopen($file,"a+");
   fwrite($datei, $value);
   fclose($datei);
   echo $value;
}
?>
```

Die Anzeige im Browser muss weiterhin bestehen bleiben, weil **wget** nach 15 Minuten abbricht wenn keine Daten empfangen werden. Die Berechnung auf den neuen Server dauert aber 2 Stunden.

crontab -e

```
0 6 * * * /srv/cronjobs/tt-rangliste/wzcalc.sh
```

wzcalc.sh

```
#!/bin/bash

# Aufrufen des PHP-Scriptes und alle Meldungen ins „nichts“ schreiben
wget http://localhost/cronjobs/wzcalc.php?commandline=true > /dev/null 2>&1

# Archiv erstellen
tar -czf archive/wzcalc.log.`date +%y.%m.%d`.tar.gz ./log/*

# löschen der Logdateien und der Datei die wget gespeichert hat
rm wzcalc.php*
rm log/*
```

4.9.2 addSaison

Der Source-Code des PHP-Scriptes addSaison.php ist im Anhang B zu finden

```
0 0 0 1,7 * /srv/cronjobs/tt-rangliste/addSaison.sh
```

addSaison.sh

```
#!/bin/bash

# Aufrufen des PHP-Scriptes und alle Meldungen ins „nichts“ schreiben
wget http://localhost/cronjobs/addSaison?op=addSaison > /dev/null 2>&1

# löschen der Logdateien und der Datei die wget gespeichert hat
rm addSaison.php*
```

wget

GNU wget ist ein freies Kommandozeilen-Programm zum Herunterladen von Ressourcen (Dateien, Webseiten, etc). Die unterstützten Protokolle sind ftp, http und https.

4.10 Cronjobs testen

4.10.1 wzcalc

Um den Cronjob zu testen, wurden einige neue Test-Spieler mit Test-Ergebnissen eingetragen. Zu diesem Zeitpunkt waren die Wertungszahlen dieser Spieler bei null. Daraufhin wurde ein Datenbank-Dump erstellt, falls der Cronjob falsch arbeiten sollte und die Wertungszahlen der „echten“ Spieler dadurch verfälscht werden sollten.

Jetzt wurde das Script **wzcalc.sh** einmal manuell ausgeführt und danach in der Datei **wzcalc.access.log** nach den Test-Spielern geschaut.
Durch eine manuelle Berechnung der Wertungszahlen konnte nachgewiesen werden, dass das Script ordnungsgemäß arbeitet.

4.10.2 addSaison

Die Tests wurden ausschließlich im Browser durchgeführt, da **wget** genau die Adresse aufruft, die auch benutzt wird wenn keine Daten ins Formular eingegeben werden.

Nachdem einmal das Script ausgeführt wurde, wurde einmal in die Datenbank geschaut, ob die Einträge hinzugefügt wurden, sie vollständig waren und ob sie auch korrekt eingetragen wurden.
Anschließend wurde noch überprüft, ob auch die Daten auf der Webseite angezeigt werden.
Jetzt wurden noch die Fehlermeldungen getestet, z.B. ob es die Saison schon gibt oder ob sie noch gar nicht begonnen hat.

4.10.3 Cronjobs starten

Um zu überprüfen, ob der Cronjob auch wirklich durchläuft, wurden die Shell-Scripte für die Cronjobs so angepasst das der Start überwacht werden konnte.

Bei wzcalc wurde die Startzeit auf 9 Uhr morgens und bei addSaison auf den nächsten Monatsanfang um 12 Uhr gestellt.

4.11 DNS-Einträge anpassen

Nachdem alles voll funktionstüchtig war, wurden nur noch die DNS-Einträge dahingehend bearbeitet, sodass unter der Adresse **tt-rangliste.de** der neue Server angesprochen wurde.

5 Fazit und Ausblick

Das Ziel des Projektes wurde den Vorgaben entsprechend erreicht.
Einen Geschwindigkeitszuwachs konnte vor allem bei dem Cronjob **wzcalc** verzeichnet werden. Auf dem alten Server betrug die Laufzeit ca. sechs Stunden, hingegen auf dem neuen Server nur noch zwei Stunden. Durch die Einrichtung der Cronjobs konnte auch Arbeitszeit eingespart werden.

Spams sind im Forum und Gästebuch durch den neuen Spam-Schutz nicht mehr zu verzeichnen.

Durch Anpassungen in den PHP-Scripten konnte ein Geschwindigkeitsvorteil erzielt werden. Bei dem Eintragen von neuen Spielständen wurden schon Wertungszahlen und die Tabellen-Positionen neu berechnet. Diese Aufgabe übernimmt aber schon der Cronjob wzcalc, der jeden Morgen um 6 Uhr läuft, und so konnte diese Funktion beim Anlegen von Spielständen entfernt werden.

Es sind noch einige sicherheitskritische Punkte offen die geschlossen werden müssen. Dazu zählen z.B. die register_globals, die noch angeschaltet sind und somit ein sehr hohes Sicherheitsrisiko darstellen. Die SQL-Statements in den PHP-Scripten sind auch nicht sicher programmiert und bieten eine Angriffsfläche für SQL-Injektion. Desweiteren wird ein Formular zum versenden von Spams missbraucht, das die Funktion hat Freunden die Webseite zu empfehlen.

Ralf Mengel

Digital unterschrieben von Ralf Mengel
DN: cn=Ralf Mengel, o, ou, email=ralf.mengel@anime-naruto.de, c=DE
Datum: 2008.05.02 14:36:30 +02'00'

Glossar

Apache:
Der Apache HTTP Server ist ein Produkt der **Apache Software Foundation** und der meistbenutzte Webserver im Internet. Eines der stärksten Konkurrenzprodukte ist der Internet Information Services (IIS) von Microsoft.

Cronjob:
Ein Cronjob startet zu der eingestellten Zeit das eingestellte Programm/Script, das dann automatisch abläuft.

CSV:
CSV (Comma-Separated Values) ist ein Dateiformat das nach einem bestimmtem Aufbau abgespeichert. Dieser Standard ist in RFC 4180 beschrieben.

Dump:
Im Bereich der Datenbanken ist damit eine Datei gemeint, die die SQL-Statements der gesamten Datenbank oder Teile davon beinhaltet.

GPL:
Die **GNU General Public License (GPL)** ist eine von der Free Software Foundation herausgegebene Lizenz mit Copyleft für die Lizenzierung freier Software, die zurzeit hauptsächlich in ihrer zweiten Version und seit dem 29. Juni 2007 auch in ihrer dritten Version Verwendung findet.
(http://de.wikipedia.org/wiki/GNU_General_Public_License)

MySQL:
MySQL ist ein Relationales Datenbankverwaltungssystem (RDBMS). Diese Datenbank ist eines der meistgenutzten Datenbanken für dynamische Webseiten, da diese Datenbank kostenlos ist (Community Server von Sun Microsystems)

PHP:
PHP steht für **P**HP **H**ypertext **P**reprocessor (rekursives Backronym) und ist eines der meistgenutzten Script-Sprachen für dynamische Webseiten. PHP steht unter der GPL und ist somit kostenlos.

Shell-Scripte:
Shell-Scripte sind in der Unix/Linux Welt im Einsatz und mit der Batch-Programmierung unter Windows zu vergleichen. Damit ist es z.B. möglich, mit den vom Betriebssystem zur Verfügung gestellten Befehlen, Aufgaben zu programmieren.

SSH:
Secure Shell ist der Nachfolger von Telnet. SSH ist verschlüsselt und kommt meist auf Unix/Linux Systemen zum Einsatz. Unter Windows kann man sich einen SSH-Server nachinstallieren um auch dort eine verschlüsselte Verbindung zu haben.

Quellenverzeichnis

Softwarequellen

Debian NetInstall Download	http://www.us.debian.org/distrib/netinst
Phorum	http://www.phorum.org/

Dokumentationen

Apache Dokumentation	http://httpd.apache.org/docs/2.2/
PHP-Dokumentation	http://www.php.net/download-docs.php
MySQL-Dokumentation	http://dev.mysql.com/doc/

Anlagen

A:

form.php

```
<?php
$code = rand(1000,9999);
$gfx_img  = $code;
$gfx_code = md5($code);
?>

<form action="<?PHP echo $post_page.".".$ext."?code=".$gfx_code.""; ?>" met
hod="post" enctype="<?PHP echo $enctype ?>">
[…]
<tr>
   <td <?PHP echo bgcolor($ForumTableBodyColor1); ?> nowrap>
      <font color="<?PHP echo $ForumTableBodyFontColor1; ?>">
          SPAM-Check:
      </font>
   </td>
   <td <?PHP echo bgcolor($ForumTableBodyColor1); ?>>
      <img src="../create_img.php?code=<?PHP echo $gfx_img;?>" alt="gfx">
      <input type="Text" name="gfx_check" size="5" maxlength="5" value="">
   </td>
</tr>
[…]
```

Post.php

```
<?php
if ($_POST['gfx_check'])
{
   if (md5($_POST['gfx_check']) != $_GET['code'])
   {
      $IsError = 'Die Sicherheitsabfrage wurde falsch eingegeben';
   }
} else {
   $IsError = 'Die Sicherheitsabfrage wurde nicht eingegeben';
}
?>
```

create_img.php

```
<?php
header ("Content-type: image/png");
$im = @imagecreatetruecolor(60, 14);
$text = imagecolorallocate($im,  20, 20,255);
$back = imagecolorallocate($im, 239,239,222);
imageFilledRectangle($im, 0, 0, 80,20, $back);
imagestring($im, 5, 10, 1, $_GET['code'], $text);
imagepng($im);
imagedestroy($im);
?>
```

B:

```
<?php

##########
# config #
##########

$dbhost = "localhost";
$dbuser = "usr_tt_rangliste";
$dbpass = "******";
$dbbase = "tt_rangliste";

$tabTurniere = 'ttturniere';
$tabTurniereSaison = 'ttturniere_halbserie';

#################
# db-connection #
#################

$conn = mysql_connect($dbhost, $dbuser, $dbpass)
   or die("Die Verbindung zum DB-Server nicht möglich");
        mysql_select_db($dbbase, $conn)
   or die ("Die Verbindung zur Datenbank nicht möglich");

##########
# script #
##########

function chooseMode()
{
   return '
   <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
   <html>
   <head>
   <title>TT-Rangliste | Saison hinzufügen</title>
   </head>
   <body>
   <form action="?op=addSaison" method="POST" style="display:inline;">
   Jahr: <input type="text" name="year" size="2" maxlength="4">
   Saision:
   <select name="saison">
      <option value="">---</option>
      <option value="12">Hin</option>
      <option value="06">R&uuml;ck</option>
   </select>
   <br>
   <input type="submit" value="Datensätze erstellen und einspielen">
   </form>
   <br>
   <br>
   <form action="?op=addSaison" method="POST" style="display:inline;">
   <in-
put type="submit" value="aktuelle Saison eintragen >>".getNewDateTT()."<<“>
   </form>
   <br>
   </body>
   </html>
   ';
}
function addSaison($option = "")
{
```

```
   if ($_POST['year'] AND $_POST['saison'])
   {
      $newDate = substr($_POST['year'],2,2)."-".$_POST['saison'];
   } else {
      $newDate = getNewDateTT();
   }
   echo SaisonData($newDate);
}
function SaisonData($date)
{
   global $tabTurniere, $tabTurniereSaison, $conn;

   $checkDateFuture = checkDateTT("future", $date);
   $checkDateExists = checkDateTT("exists", $date);

   if (!$checkDateFuture AND !$checkDateExists)
   {
      $sql = "SELECT * FROM ".$tabTurniereSaison;
      $result = mysql_query($sql);
      while ($row = mysql_fetch_assoc($result))
      {
         $sql_turniere_halbserie = "
         UPDATE `".$tabTurniereSaison."` SET
         `Datum`     = '".$date."'
         WHERE `Bez` = '".$row['Bez']."';
         ";
         mysql_query($sql_turniere_halbserie)
            or die ("Fehler im Statement: ". mysql_error());

         $sql_turniere = "
         INSERT INTO `".$tabTurniere."` SET
         `Bez`       = '".$row['Bez']."',
         `Kurz`      = '".$row['Kurz']."',
         `Ebene`     = '".$row['Ebene']."',
         `Geschl`    = '".$row['Geschl']."',
         `Datum`     = '".$date."',
         `Art`       =  ".$row['Art'].",
         `Quelle`    =  ".$row['Quelle'].",
         `Bemerkung` = '".$row['Bemerkung']."',
         `Staerke`   =  ".$row['Staerke'].";
         ";
         mysql_query($sql_turniere)
            or die ("Fehler im Statement: ". mysql_error());
      }
      echo "Die Datensätze wurden erfolgreich eingefügt";
   } else {
      if ($checkDateFuture) echo $checkDateFuture;
      if ($checkDateExists) echo $checkDateExists;
   }
}
function getDateTT()
{
   global $tabTurniereSaison;

   $sql = "SELECT Datum FROM ".$tabTurniereSaison." LIMIT 0,1;";
   $result = mysql_query($sql);
   $row = mysql_fetch_assoc($result);

   return $row['Datum'];
}
function getNewDateTT()
{
   $oldDate = getDateTT();
```

```
   $oldYear   = substr($oldDate,0,2);
   $oldSaison = substr($oldDate,3,2);

   if ($oldSaison == "06")
   {
      $newSaison = "12";
      $newYear   = $oldYear;
   }

   if ($oldSaison == "12")
   {
      $newSaison = "06";
      $newYear   = $oldYear+1;
      if (strlen($newYear) == 1)
      {
         $newYear = "0".$newYear;
      }
   }

   $newDate = $newYear."-".$newSaison;
   return $newDate;
}
function checkDateTT($option, $value)
{
   global $tabTurniereSaison;

   if ($option == "future")
   {
      if (getNewDateTT() < $value)
      {
         return "
         Das Datum liegt zuweit in der Zukunft<br>
         Nächstmögliche Auswahl: ".getNewDateTT()."<br>
         Ihre Auswahl: ".$value."<br>
         ";
      }
   }

   if ($option == "exists")
   {
      $sql = "SELECT Datum FROM ".$tabTurniereSaison." WHERE Datum = '".$va
lue."' LIMIT 0,1";
      $result = mysql_query($sql);
      if (mysql_num_rows($result) > 0) return "Das Datum ist schon vorhande
n<br>";
   }
}
switch ($_GET['op'])
{
   case "addSaison":
   addSaison();
   break;

   default:
   echo chooseMode();
   break;
}
?>
```